

KINGSTON UNIVERSITY
Faculty of Science, Engineering and Computing

Artificially Intelligent Virtual Assistant for Medical Diagnosis

Arturas Bulavko, MSc Software Engineering

Supervised by Prof. Souheil Khaddaj

September 2018

Table of Contents

Table of Figures	3
Table of Tables	5
Abstract	6
Acknowledgements	7
1 Introduction	8
1.1 Background and Problem	8
1.2 Proposed Solution	10
1.3 Aims and Objectives	10
1.4 Thesis Contribution	11
1.5 Thesis Summary	11
1.6 Project Management	12
2 Existing Solutions Review	14
2.1 Introduction	14
2.2 Reviewing Existing Systems	14
2.3 Individual Component Study	18
2.4 Summary	27
3 High Level Analysis	28
3.1 Introduction	28
3.2 Use Case Diagram	28
3.3 Functional Requirements with MoSCoW Prioritisation	29
3.4 Non-Functional Requirements	29
3.5 Summary	30
4 High Level Design	31
4.1 Introduction	31
4.2 System Structure	31
4.3 Database Design	35
4.4 Front-end Design	36
4.5 Summary	38
5 System Implementation	39
5.1 Introduction	39
5.2 Tools and Technology Selection	39
5.3 Back-End Development	42
5.4 Front-End Development	46
5.5 Icon Development	48
5.6 Summary	48

6 Software Quality Assurance	49
6.1 Introduction	49
6.2 Software Verification	49
6.3 Software Validation.....	50
6.4 Summary	51
7 Conclusion	52
7.1 Project Evaluation	52
7.2 Ethics Discussion	55
7.3 Future Work	56
References	57
Appendix A – Project Schedule	63
Appendix B – Project Risk Management.....	64
Appendix C – Requirements Catalogue	65
Appendix D – Application Icons	67
Appendix E – Analysis Phase SQA	68
Appendix F – Design Phase SQA	70
Appendix G – Implementation Phase SQA.....	72
Appendix H – Testing Phase SQA	75
Appendix I – Functional Testing.....	78
Appendix J – Non-Functional Testing.....	81
Appendix K – Usability and Field Testing	85
Appendix L – Evaluation Against the Existing Systems	87

Table of Figures

Figure 1 – NHS 111 Online Home Page	8
Figure 2 – NHS 111 Online Consent	9
Figure 3 – NHS 111 Online Initial Questionnaire	9
Figure 4 – Stakeholder Power-Influence Grid	13
Figure 5 – SWOT and TOWS Analysis	13
Figure 6 – Sensely Ask NHS – Virtual Assistant Application Example (Sense.ly Corporation, 2018)	14
Figure 7 – Your.MD Application Example (Your.MD 2018).....	15
Figure 8 – Buoy Health Application Example (Buoy Health, 2018)	16
Figure 9 – ASR Activity Diagram	19
Figure 10 – NLP Stages (Cawsey, 1998, p.100)	20
Figure 11 – Knowledge Discovery Process (Negnevitsky, 2011, p.367).....	22
Figure 12 – Decision Tree for Diagnosing Heart Attack (Cawsey, 1998, p.151).....	23
Figure 13 – ANN Architecture (Negnevitsky, 2011, p.167).....	24
Figure 14 – Generic ANN Medical Diagnosis Algorithm (Amato et al., 2013).....	24
Figure 15 – TTS Activity Diagram	25
Figure 16 – Use Case Diagram.....	28
Figure 17 – List of Functional Requirements with MoSCoW Prioritisation	29
Figure 18 – List of Non-Functional Requirements	30
Figure 19 – Service-oriented architectural style (Sommerville, 2010, p.510)	31
Figure 20 – Component Diagram	32
Figure 21 – Sequence Diagram	33
Figure 22 – Enter Symptom Activity Diagram	34
Figure 23 – Obtain Diagnosis Activity Diagram	34
Figure 24 – ER Diagram	35
Figure 25 – Data Dictionary.....	35
Figure 26 – Process of Medical Knowledge Capture (Infermedica-Developer, 2018).....	40
Figure 27 – Form Front-End	42
Figure 28 – Linking ASR Feature to the Buttons	42
Figure 29 – ASR Function	43
Figure 30 – Initiating the Diagnosis Process	43
Figure 31 – Capturing Form Data and Passing Values	43
Figure 32 – Sending Requests to the API	44
Figure 33 – Obtaining Treatment Advice from the Database.....	44

Figure 34 – Obtaining and Processing Triage Level	45
Figure 35 – Assembling the Diagnosis String	45
Figure 36 – Presenting Diagnosis to the User using Text and Voice	45
Figure 37 – Input Validation Example	47
Figure 38 – Symptom Suggestion Generation Example.....	47
Figure 39 – Example Symptom shown on Unsuccessful Try.....	47
Figure 40 – Project Schedule Gantt Chart.....	63
Figure 41 – Project Deliverables	63
Figure 42 – Preview of the Icon	67

Table of Tables

Table 1 – Existing System Feature Comparison	17
Table 2 – System Medium Fidelity Wireframes	36
Table 3 – System Prototype	37
Table 4 – System Screenshots.....	46
Table 5 – Application Icons	48
Table 6 – Verification Process Techniques.....	49
Table 7 - Project Testing.....	50
Table 8 – Risk Analysis and Contingency Plan.....	64
Table 9 – System Feature Comparison	87
Table 10 – System Diagnosis Accuracy Comparison	87

Abstract

The recent technological advancements have enabled to combine several components to provide innovative solutions to the daily problems. NHS currently uses a phone-based system to provide medical advice on urgent, non-emergency concerns. Such approach proves to be inefficient, inaccurate and unreliable due to insufficient amount of advisers who are prone to human error, have poor medical knowledge and do not take their jobs seriously due to low salary. This project is aimed at resolving these issues by implementing a highly dependable, web-based virtual assistant which will provide healthcare diagnosis and treatment advice using artificial intelligence.

Acknowledgements

I would like to thank Professor Souheil Khaddaj who was an exceptional project supervisor. His guidance and advice has constantly motivated me in achieving results beyond my limits, allowing to discover countless innovative approaches and technologies. He has always shared his extensive expertise, making this project a remarkable learning opportunity during which I have improved my academic writing skills and thoroughly explored the artificial intelligence field.

1 Introduction

1.1 Background and Problem

National Health Service (NHS) is a collective description of the medical services provided in the United Kingdom. Due to the high demand and popularity, NHS has developed several systems to effectively deal with all types of medical concerns. An example of such system is NHS 111 (2017) which provides medical advice over the phone across the United Kingdom (UK). This service is available 24/7 and delivers guidance for all urgent, non-emergency medical conditions. Depending on the situation, the staff are able to give self-care information, book face-to-face appointment, send ambulance or direct the patient to an emergency dentist, nurse or a General Practitioner (GP) (NHS 111, 2017).

The major problem of NHS 111 is the lack of qualified doctors who are able to assist the callers. The service instead employs “fully trained advisers” who handle the calls on daily basis (NHS 111, 2017). The advisers enter all symptoms into the internal system and based on the system’s suggestion, advise the caller about their condition, suggesting further steps (Amery, 2016). Such approach is time consuming for the caller because irrelevant questions must be asked in order to bypass the system rules. It is also costly for the NHS because the adviser must be appropriately trained and receive an annual salary. Additionally, due to the strong human factor, deaths arise as a result of the advisers entering the wrong symptoms into the system and ultimately providing incorrect treatment recommendations (ITV News, 2016). Such problems also emerge from low annual salaries, preventing staff from taking their job seriously and providing consistent correct advice (Walker, 2013). Being an urgent medical helpline, NHS 111 struggles to handle all phone calls during the peak times (Quinton, 2017), placing vast number of callers in danger and increasing the risk of death.

In an attempt to solve these problems, NHS is trialling a web-based version of the NHS 111 Online (2018) service. It allows patients entering their symptoms manually via a step-by-step guide online. On launch, the users are presented with a top-level menu demonstrated in *Figure 1*. This enables the user to select an appropriate service, allowing the system to deal with the query effectively.

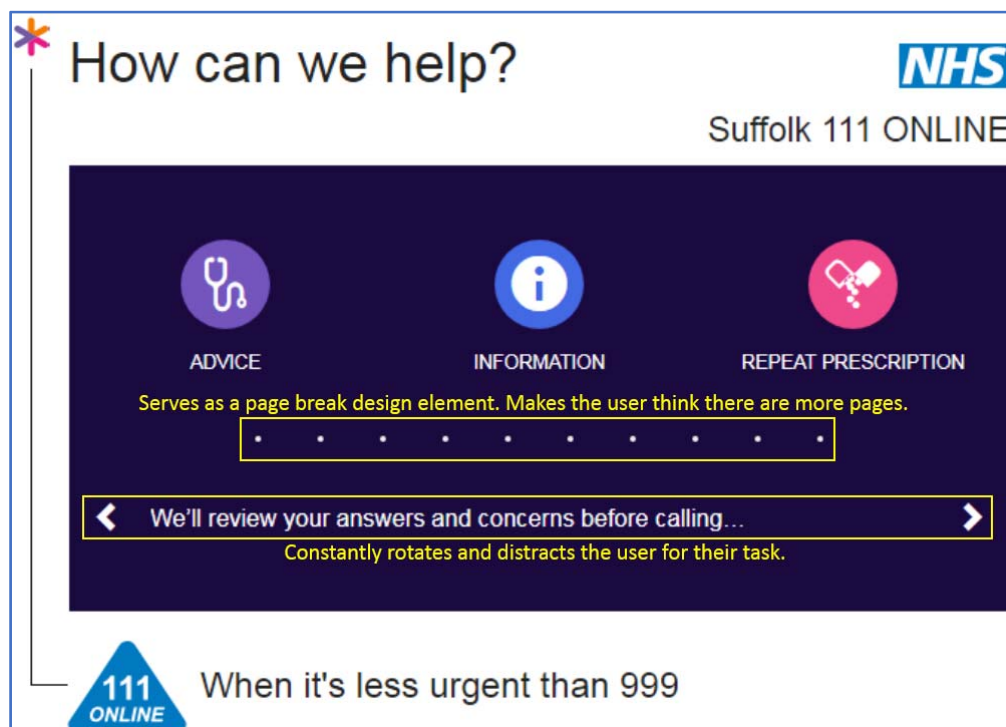


Figure 1 – NHS 111 Online Home Page

Having selected the *Advice* menu item, the users are asked for a consent, presented in *Figure 2*. The page provides clear explanation to the user, allowing them to make an informed decision about the use of the application. This is a crucial step in m-health applications which ensures that the users are aware of the service and agree to securely share personal information with the system. The application developed throughout this project must include similar page to reflect the regulations.

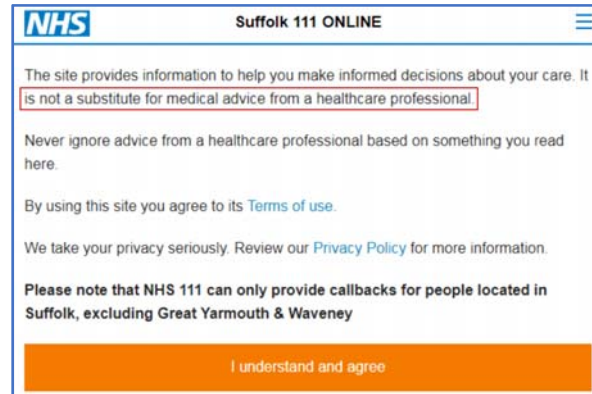


Figure 2 – NHS 111 Online Consent

Once the user agrees, they are automatically redirected onto a number of multiple choice questions which must be answered to proceed with the system. The questions enable the system to establish a correct symptom and generate an appropriate treatment suggestion. It is currently a manual and lengthy process, as shown by the *Questionnaire Progress* bar in *Figure 3*. It is clear that such process is lengthy, implying that the query velocity is low and the system remains loaded for a longer period of time, which in turn impacts the speed of the service for all users using the system concurrently.

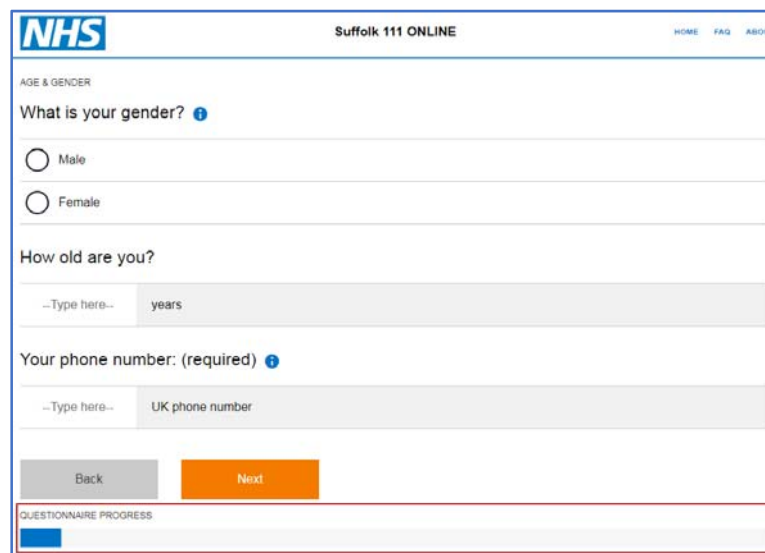


Figure 3 – NHS 111 Online Initial Questionnaire

Apart from being available in few areas, NHS 111 Online (2018) follows a defective UX (User Experience) practice which makes it hard to use by the average user. This includes the need to enter irrelevant information, lack of instructions/guidance, confusing design elements, scaling issues when used on varying-screen devices and carousel of quotes on the home page, which only distracts the users. As a result of this, the system may simply be abandoned in an urgent medical condition, placing the users in further risks due to the symptom becoming significantly severe. The service may also be problematic to use by the elderly people, individuals with visual impairment, hand tremor or injuries.

Another NHS Online: 111 (Babylon Health, 2018) application undergoing trials allows users to enter their symptom manually, read symptom facts, get nearby healthcare establishment directions and initiate a live chat with a chatbot. The chatbot is unable to recognise or output speech and follows a mundane algorithm to provide diagnosis to all users. Furthermore, the chatbot is developed using countless IF statements which lack performance and often unable to recognise the user questions. The users are once again required to answer irrelevant questions to bypass the system rules, preventing this to be an effective solution for an urgent service. These trials have however proven that online services are safe to use and about 6% of users are happy switching to an online version of the NHS 111 (Armstrong, 2018). Therefore, this project an ideal opportunity to explore and adapt novel technological advancements in order to eliminate issues discussed earlier.

1.2 Proposed Solution

The aim of this project is the development of a dependable, web-based application, utilising novel technologies in order to solve the earlier discovered problems of the NHS 111. This project is important because it will eliminate the human factor by developing an Artificially Intelligent (AI) system which will be used to analyse the symptoms and provide an accurate diagnosis as well as suggesting the treatment process. AI is less bias and significantly more effective at analysing large volumes of data when compared to the trained advisers. The application will also benefit from the computational linguistic technologies to significantly improve the UX by introducing verbal interaction feature within the system. Apart from being platform independent, the web-based application will be able to handle large volumes of people during peak times, accurately capture and process the patient information, improve the query velocity, reduce costs and provide a more convenient User Interface (UI) for people with various disabilities. These benefits will significantly improve the existing NHS 111 service due to simplistic and efficient design to target non-medical concerns.

1.3 Aims and Objectives

The goal of the project is to deliver a web-based application providing medical assistance using artificial intelligence through a verbal interaction. This goal will be achieved through the following aims and objectives.

1.3.1 Aims

1. Compatible on most modern smartphones, tablets, PCs and browsers.
2. Secure access to the web-based application.
3. Provide accurate and dependable medical advice and diagnosis.
4. Provide voice input and output facilities within the application.
5. Utilise Artificial Intelligence technologies to correctly generate answers.

1.3.2 Objectives

1. Each interaction must provide a textual transcript (chat) within the application.
2. Responsive layout on the application pages to make it interoperable.
3. All application pages must use HTTPS.
4. Correctly answer 60% of the user healthcare concerns.
5. Each interaction must have action feedback within 5 seconds across all internet connections.

1.4 Thesis Contribution

Currently there are no medical applications utilising verbal interaction which would correctly establish the user's diagnosis without relying on the human factor. Several m-health applications have been previously developed with an aim of providing accurate diagnosis established via a lengthy questionnaire which the user had to complete. Such approach was proven effective but introduced usability (defective UX) and interaction problems as well as the consumption of the user's time, completing the surveys. These issues were addressed with the introduction of virtual assistants which followed all UX design recommendations and enabled verbal interaction. However, such systems are unable to establish medical diagnosis within the application and thus redirect the user to an online search engine. The user is presented with countless pages of search results which do not answer the query directly, making the user manually search for the right diagnosis.

This thesis proposes to combine m-health applications and AI virtual assistant technologies to develop a stand-alone web-based application to establish medical diagnosis. Such innovative approach will benefit from the novel UX and Artificial Intelligence advancements to develop a dependable and ethical m-health application in order to eliminate the human-factor from medical applications and significantly enhance the UX. This will target more users, reduce the annual staff expenses, improve the query velocity and process large volumes of people concurrently which are the current problems.

1.5 Thesis Summary

The thesis will outline the software development cycle by creating a chapter for each phase. It will begin by completing SWOT/TOWS and stakeholder analysis as well as selecting an appropriate methodology to manage the project. It will then proceed onto reviewing the existing solutions and decomposing them into individual components. Each component will be examined and recommendations for the novel application will be made. The thesis will continue by analysing the innovative system by creating a use case diagram and establishing a range of functional and non-functional requirements which will be used to assess the system towards the end of the project. The design chapter will proceed onto selecting an appropriate system architecture and modelling the system using activity, sequence and component UML diagrams. An ER diagram and data dictionary will be created to reflect the database design. The design chapter will conclude by designing the application wireframes and prototypes by taking into the account all the earlier recommendations and discoveries. An implementation chapter will be initiated by reviewing and selecting the tools and technologies for the application's development. It will then discuss the back-end, front-end and the application icon development approaches to highlight how the recommendations affected the end-product. An SQA chapter will present the product validation and verification techniques which are used to ensure the system achieves the correct degree of quality and is compliant with the requirements. The thesis will conclude by evaluating the project and suggesting future work directives.

1.6 Project Management

1.6.1 Methodology

The project will be managed using a DSDM Atern Agile methodology. The fundamental principles of this methodology will enable the project to be completed on time and avoid compromising quality as these are the main project constraints. The incremental product development and continuous stakeholder feedback will ensure that all critical features are present and are implemented correctly. To achieve this, several DSDM Atern core techniques will be adopted throughout the software development lifecycle. These include; modelling, requirement prioritisation, prototyping and testing. Additionally, this methodology was chosen because the project will be completed by one person, allowing them to utilise a single methodology towards project management and software development. This will consent them to spend more time on the product, rather than paperwork.

Other methodologies such as Waterfall and PRINCE2 were neglected due to their inflexible nature. Both methodologies follow a strict schedule and restrict product changes, leaving no room for improvement once the phase is completed. This prevents the product from achieving a correct degree of quality as well as introducing the risk of incorrectly solving the stated problem. The dynamic style of the project also suggests that these methodologies are unsuitable for the project. Similarly, Agile SCRUM and Extreme Programming (XP) were disregarded due to lack of project team members. Daily SCRUM meetings and pair-programming principles would not be possible in this project.

1.6.2 Stakeholder Analysis

The internal and external stakeholders have a direct interest in the project and thus influence the project which in turn has an effect on the product. To ensure that all stakeholders are correctly managed throughout the project, it is vital to identify all stakeholders in order to discover their power and influence towards the project. The stakeholders of this project will be discussed next.

1.6.2.1 Doctors

Qualified individuals who treat ill people and provide medical advice. The project will benefit this stakeholder by decreasing the amount of daily urgent appointments required, allowing them to reduce queues in the healthcare establishments and the amount of paperwork each patient requires.

1.6.2.2 Patients/Application Users

Members of the public who will use the application for urgent, non-emergency medical concerns. The project will benefit this stakeholder by providing consistently accurate diagnosis and treatment suggestions, eliminating the waiting time and reduce the risk of human-error when entering symptoms into the internal system.

1.6.2.3 NHS

Organisation with a focus on improving the nation's health by setting priorities and directions of the service. Currently in charge of NHS 111. The project will benefit this stakeholder by reducing the annual cost of running the NHS 111 due to the reduction of staff required to answer queries.

1.6.2.4 Governing Bodies

Governing bodies such as The Secretary of State for Health and The Department of Health who are responsible for correct operation of NHS services and their funding (NHS, 2016). The project will benefit this stakeholder by reducing the required funding and improving the nation's health.

1.6.2.5 Software Developer & Project Manager

Individual in charge of the project and the end-product. The project will benefit this stakeholder by enabling them to develop new skills, explore new tools and learn new technologies.

1.6.2.6 Power-Influence Grid

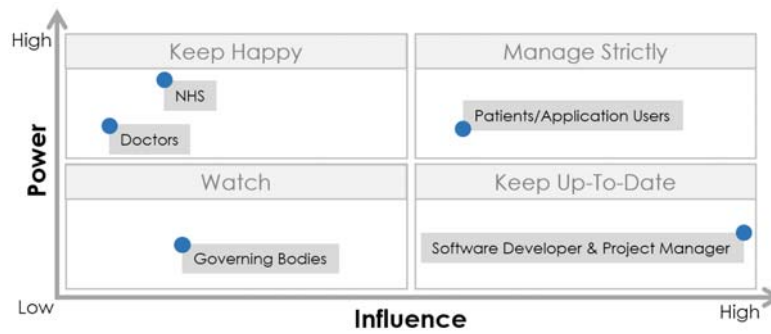


Figure 4 – Stakeholder Power-Influence Grid

As illustrated in *Figure 4*, the stakeholders previously identified were categorised based on their power and interest towards the project. Such strategy enables to establish correct communication protocols and rank stakeholder decisions which has a positive effect on the project management. Using the grid, the project manager can correctly operate the project by conducting different types of meetings aimed at different stakeholder groups and produce appropriate progress reports.

1.7 SWOT and TOWS Analysis

SWOT and TOWS analysis provides a comprehensive summary of the current business position in relation to the market and is frequently used to judge whether the project is worth undertaking. It also enables to assess threats and weaknesses at the start of the project, allowing to translate them into strengths and opportunities, ensuring the project achieves its aims.

	POSITIVES	NEGATIVES
INTERNAL CURRENT	<u>Strengths</u> <ul style="list-style-type: none"> ✓ Access to various resources. ✓ Loyal users, always able to provide feedback. ✓ High level of awareness. ✓ Management expertise. ✓ All required tools present. 	<u>Weaknesses</u> <ul style="list-style-type: none"> ✓ Lack of expertise in some areas. ✓ Strict deadlines and pressures. ✓ Costly tools. ✓ Limited research in this field.
EXTERNAL FUTURE	<u>Opportunities</u> <ul style="list-style-type: none"> ✓ Provision of new services. ✓ Deliver new system functionality effortlessly to always remain ahead of the competitors. ✓ Reduced staff due to automated online system. ✓ Increased diagnosis accuracy, leading to better healthcare. 	<u>Threats</u> <ul style="list-style-type: none"> ✓ Legislation change affecting the operation of the new system. ✓ Competitors divert customer base. ✓ High system maintenance cost. ✓ Providing incorrect diagnosis, resulting in health risks.

Figure 5 – SWOT and TOWS Analysis

Figure 5 demonstrates the SWOT and TOWS analysis which was used to discover the project positives and negatives. Similarly, PESTLE analysis could be used to assess the situation, however, the absence of internal evaluation made such analysis inappropriate for this project. Internal aspects are important in this project due to various project constraints, such as time, developer experience and budget which may affect the end-product. The analysis has shown that the project can be completed.

A detailed project schedule and the project risk management can be found in the *Appendix A* and *Appendix B* respectively. Since there are no critical risks, all project risks identified and strategies found to tackle them, it is considered safe to undertake this project. The thesis will now proceed onto the literature review where it will examine existing systems and individual components.

2 Existing Solutions Review

2.1 Introduction

The literature review chapter will examine the existing systems from two perspectives; the medical applications and the universal virtual assistants. Such decomposition will reveal the underlying architecture of each system as well as expose all components which deliver specific functionality. The process of each component will be examined and UML activity diagrams will be used to represent the model of each component, enabling to visualise the operating algorithm of each module. This will enable to assess the present approaches, discover the current limitations and thus enable to propose the new system requirements and quality metrics. By basing the new system on existing knowledge, common issues can be eliminated early in the project and recommendations will be used to achieve enhanced system quality and features. An emphasis will be made towards Artificial Intelligence in order to study the mapping of human behaviour on computers.

2.2 Reviewing Existing Systems

2.2.1 Medical Applications

There are several medical applications with the similar features which individually solve specific problems on NHS 111 service. Sensely (2018) is an organisation focused on developing medical chatbots which provide digital health solutions to various industries. An example of their system is Ask NHS – Virtual Assistant (Sense.ly Corporation, 2018) which follows a generic *IF* statement algorithm to assist their users, as oppose to AI-driven back-end. The users are able to interact with the application using keyboard and the presented menu option buttons, which help the system determine a diagnosis. As demonstrated in *Figure 6*, the users may be asked countless questions which leads back to the original issue of NHS 111 Online (2018). Additionally, the system occasionally redirects their users to the official NHS website in order to provide guidance, thus the application does not provide direct answer to the user queries. Such approach worsens the UX which is reflected by several negative reviews on the iTunes App Store and Google Play. However, research proves the application to be more convenient and engaging than NHS 111 service and can potentially save 14% of annual NHS expenditure (Odessky and Tamler, 2018). This shows that the application developed throughout this project will have a direct positive impact on the healthcare field.

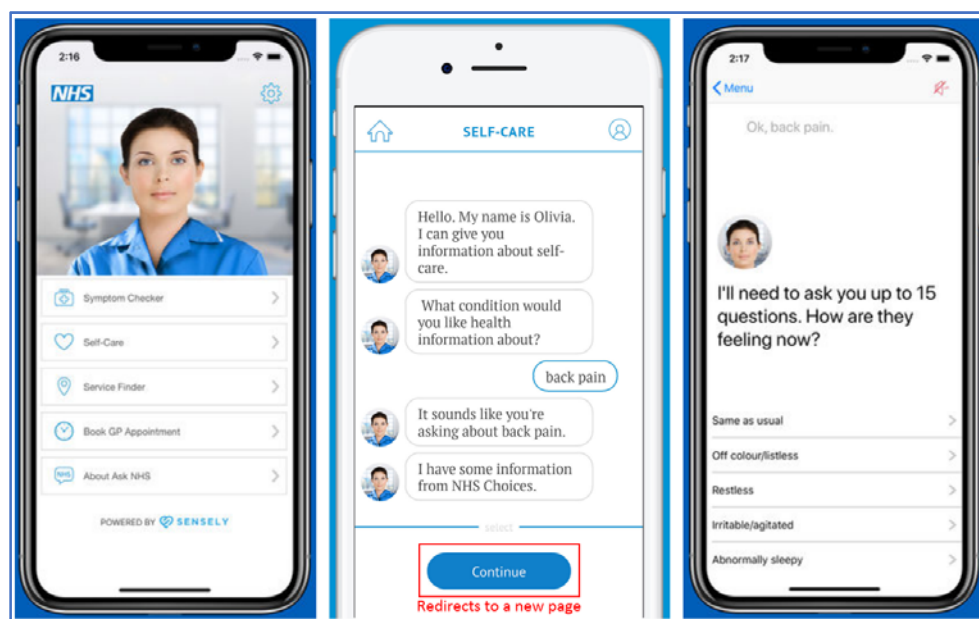


Figure 6 – Sensely Ask NHS – Virtual Assistant Application Example (Sense.ly Corporation, 2018)

Your.MD (2018) is a free AI-based service providing health information and diagnosis based on the supplied symptoms. The service is available to download on the iTunes App Store and Google Play (Your.MD AS, 2017) as well as providing a web-based client. Being chat-based, the application offers free integration functionality with Skype, Telegram, Kik and Slack platforms which enables their users to get instant medical advice. The application excels in UX aspects by providing an intuitive menu, clear instructions, easily-readable text and appropriate colour scheme, as shown in *Figure 7*. An important advantage of this system is the provision of example question which the user can potentially ask if the system did not understand the query from the first try. Such guidance not only enhances the overall UX but assists the user in formulating an appropriate question. It could be argued that the system provides incomplete or inaccurate advice because the only provided suggestion is to call 999 if it is not possible to walk. It would be more appropriate asking the user to visit the nearest Accident and Emergency (A&E) department if somebody is able to assist them. Such strategy would definitely decrease the 999 calls frequency and the user would still see a qualified doctor.

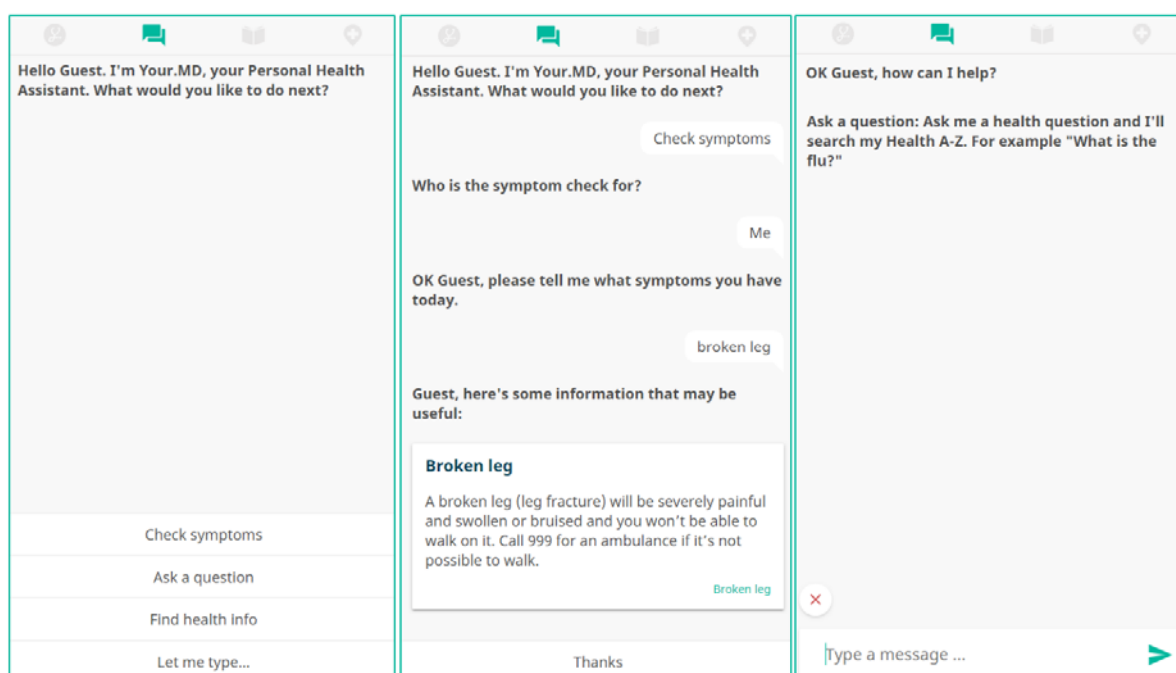


Figure 7 – Your.MD Application Example (Your.MD 2018)

Buoy Health (2018) is another example of a free web-based chatbot with Artificial Intelligence. It begins by asking three simple questions to establish the user details. The system then proceeds onto asking the user for initial symptom. Once the user starts typing, the system suggests *closest searches* to help the algorithm deal with the question effectively by proposing a recognised phrase. Such aspect greatly improves the query velocity and the overall UX of the application since the system will be able to process the query faster. However, as seen from *Figure 8*, the system may not accept some basic, yet accurate descriptions of the symptoms. The user is forced to select one of the generated matches and the system begins asking countless questions to determine the diagnosis. While the questions provide relevant and descriptive answers, the process of establishing a diagnosis increases significantly. Although the algorithm may arrive at the correct diagnosis, the time taken to complete this task could worsen the symptoms since NHS 111 is aimed at urgent medical concerns. On the other hand, the selected blue colour scheme calms and reassures the users which directly impacts the UX (O'Connor, 2011). It was proven that the colour scheme has a direct impact on the user's emotional and psychological state, allowing to change it by using appropriate colours within the application. O'Connor (2011) suggests using white, green and blue colours for medical applications.

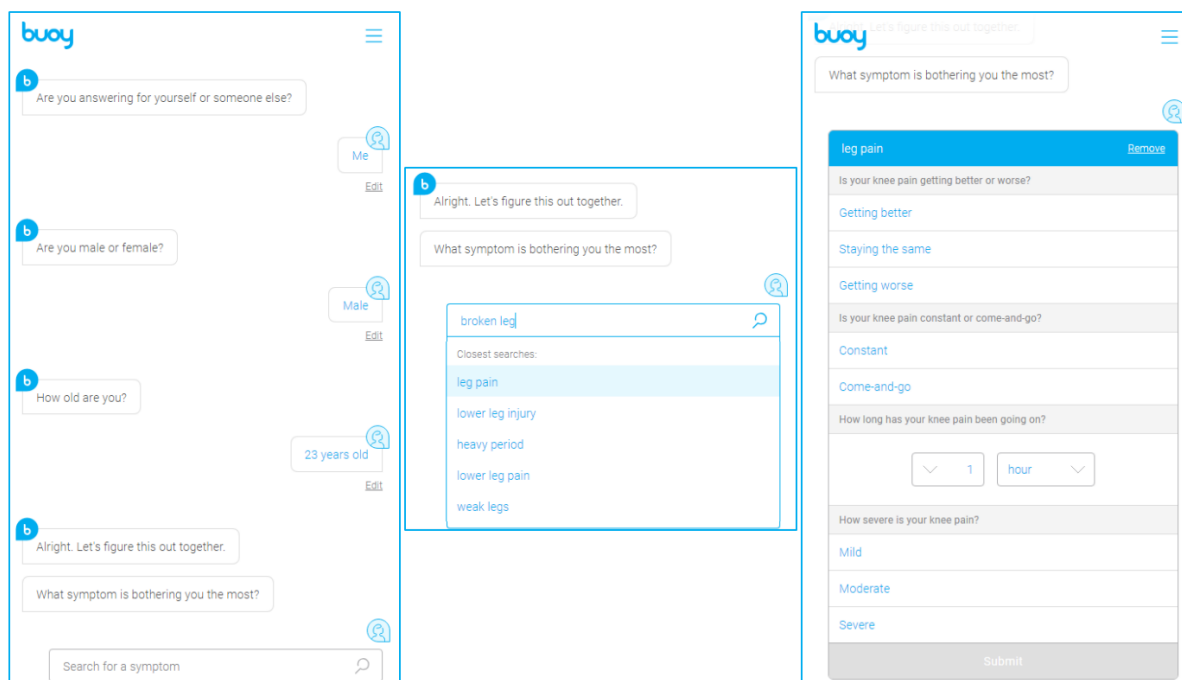


Figure 8 – Buoy Health Application Example (Buoy Health, 2018)

There are also MedWhat (2018) and Infermedica (2018) Artificially Intelligent systems which currently only provide API (Application Programming Interface) for the developers. Since they do not provide a client, these systems were not reviewed, however, they will be examined during the implementation.

2.2.2 Universal Virtual Assistants

Since the developers of the reviewed medical applications do not disclose the underlying system architecture, the universal virtual assistants will be examined next. They provide similar features to the proposed solution of this thesis and thus will be the perfect starting ground. Virtual assistants are an emerging trend with an aim of performing tasks or services for the individuals. Popular examples include Siri (Apple, 2018), Google Assistant (Google, 2018) and Cortana (Microsoft, 2018). These systems aid users by answering their questions, setting reminders/alarms and operating the device by initiating various functions automatically. The applications mainly operate via verbal interaction with the user, however, all three also provide the functionality to type the question using a keyboard if the device supports it. The assistants also support several interaction languages, allowing the developers to constantly add new languages due to the architecture. These approaches enhance the accessibility of the system, allowing users selecting a suitable interaction method in a language of their choice.

The primary drawback of such applications is the lack of direct answers to the medical questions. Instead, the systems redirect users to the popular search engines in order to display all relevant search results within the browser. While the results may be relevant to the question, the overall UX of the application suffers. The user is tasked with finding a suitable link and reading vast amount of information to familiarise themselves. This approach introduces additional complication of various medical advice supplied by non-qualified people. Examples include; treatment suggestions, provision of inaccurate diagnosis, home remedy ideas and general medical advice based on opinions. Following such advice from general internet users may lead to further health risks (Miller, 2015). Another issue of voice-assistants is the presence of listening and understanding problems. Such systems are frequently operated in noisy/loud environments or the users may have dialect/accent unrecognised by the system (Herff and Schultz, 2016). This crucial aspect of virtual assistants will be discussed in more depth further in the thesis and appropriate mitigation strategies will be proposed.

2.2.3 System Comparison

It is important to compare the reviewed systems in order to learn the negative aspects which must be avoided in the current project as well as the positive aspects which could compliment the new system. As shown in *Table 1*, the feature pattern is similar in medical application and universal virtual assistant categories. This is further contrasted against the proposed application as part of this thesis.

	Medical Applications			Universal Virtual Assistants			New
Features	Sensely	Your.MD	Buoy Health	Siri	Google Assistant	Cortana	Thesis App
Text Input	✓	✓	✓	✓	✓	✓	✓
Text Output	✓	✓	✓	✓	✓	✓	✓
Voice Input	✓			✓	✓	✓	✓
Voice Output				✓	✓	✓	✓
AI Platform		✓	✓	✓	✓	✓	✓
Web-Based		✓	✓				✓
Native Application	✓	✓		✓	✓	✓	
Direct Answer	✓	✓	✓				✓
Multilingual				✓	✓	✓	

Table 1 – Existing System Feature Comparison

Comparing Your.MD (2018) with Buoy Health (2018) in terms of AI module, it is evident that the first completes the task much faster and provides a more accurate diagnosis, simply because it does not restrict the user in symptom input. It is important to learn that while the new system may provide symptom suggestions, it must not force the user to select an option from the list. Sensely (2018) can be compared to Buoy Health (2018) in terms of operation due to the vast amount of questions asked by both systems. However, the first system relies on an algorithm, whereas the second applies AI module, allowing the system to enhance efficiency over time. Overall, Your.MD (2018) provides more features and higher product quality when compared with similar medical applications.

Universal Virtual Assistants offer similar functionality; therefore, it is important to understand how consumers make their choice. Research has shown that Siri leads the mobile voice assistants market share at 45.6%, leaving Google Assistant and Cortana at 28.7% and 4.8% respectively (Wagner, 2018). Based on the tests run by Bushnell (2018), Cortana lacks in voice recognition, resulting in an inability to process queries correctly, justifying the last place. Siri and Google Assistant on the other hand, are easy to setup, correctly recognise voice and understand the query context appropriately (Bushnell, 2018). Selection between the two relies on the hardware and the convenience since Siri is aimed at Apple users, whereas Google Assistant is targeted towards Android and Microsoft users. By making a web-based application throughout this thesis, users will not be restricted to a specific platform.

The only major difference between the reviewed medical applications and the universal virtual assistants is the ability to provide direct answers and voice recognition. It is evident that there are no applications presently which would combine these features together in a stand-alone system. Such application is proposed to be developed as part of this thesis. Sennaar (2018) suggests that all medical diagnosis systems can benefit from image and video processing facilities to enhance the symptom assessment process and increase the accuracy of diagnosis. It would also be beneficial to develop a system which can operate offline, implying that the internet connection is not required to check the symptoms. Such approach would be useful for users in rural areas and tourists. Although, this would require huge memory space on the user's device and would likely not be used due to this factor.

The existing systems will now be decomposed into individual components and engines which deliver specific functionality. Since the companies do not directly reveal the technologies behind their systems, thesis will examine the findings by Matyunina (2017), Jeegeek (2018) and Zubair et al. (2017).

2.3 Individual Component Study

2.3.1 Automatic Speech Recognition (ASR)

Automatic speech recognition is a fundamental AI component of the virtual assistants, enabling users to initiate an interaction process with the application. This computational linguistics technology operates by forming a transcription of an acoustic speech into a textual string which can be interpreted by the system. ASR component benefits countless sectors by providing an intuitive, accessible and easily usable platform for realistic dialogs between the users and their devices. Hempel (2008, pp.3-4) suggests that speech recognition enables to automate business processes, reducing the overall costs and increasing the company's productivity if implemented in the healthcare establishments. Poder et al. (2018) note that the turnaround time is significantly decreased in the systems with the speech recognition feature enabled. They also note that such systems significantly reduce the costs, because transcripts are automatically generated. Larson (2003, pp.2-5) also argues that speech recognition has a positive effect on UX, allowing users to input data more conveniently on the small devices, such as smartphones and iPads Mini. Due to the small screen sizes, elderly people might find it hard using a qwerty keyboard on their device to input data, suggesting that speech recognition would be more suitable for this age group. McAuliffe et al. (2012) indicate as part of their study that the age of the speaker does not influence the accuracy of speech recognition, making it an ideal input method for this application due to diverse target audience.

While speech recognition might introduce better UX practices into web-based applications, they are also dependent on variables which have a significant impact on quality, accuracy and performance. Herff and Schultz (2016) note that speech recognition might not operate correctly in the loud environments, near "bothering bystanders" and individual's inability to produce clear speech. Holmes and Holmes (2001, p.169) also reveal that poor-quality microphone can add unwanted noise and even prevent the speech from being understood by the system. In order to avoid these issues, it is advised that the users operate the application in quiet environments using modern microphones. Additionally, You and Ma (2017) indicate that speech recognition suffers in presence of noise and distortions. Such issue can be tackled by applying multi-conditional modelling which compares and trains the acoustic model with the noise database to eliminate them (You and Ma, 2017). While this sufficiently deals with noise and distortion elimination, the performance of the application is significantly reduced due to the lengthy data-driven algorithms which must process each word distinctly (Hardcastle et al., 2010, pp.805-807). Herff and Schultz (2016) also propose to use neural signals to deploy brain-to-text systems, allowing to remove the issues of background noise, distortions and inability to produce clear speech. However, such technology would require additional IoT (Internet of Things) devices which are not yet available. It would be beneficial to explore this topic in the future work.

Goss et al. (2016) reveal that speech recognition cannot accurately convert medical terminology. Their study has shown that 15% of medical notes contained at least one critical error, indicating that there is a strong issue with the technology. This is further supported by Poder et al. (2018) who suggest that the speech recognition advantages are countered by the risks of additional errors and the increased time required for dictation and correction. Hempel (2008, p.20) also suggests that the system must take into account the user's linguistic background to ensure the system can be operated by a range of people, which is an important factor in a multi-nation country, such as UK. Virtanen et al. (2012, p.10) note that ASR is fundamentally a Bayesian classifier due to the minimisation of probability of misclassification, which is achieved as a result of substantial algorithm. The method applies a range of tools and technologies to ensure each voice input results in an accurate textual output. The process of ASR component varies in different systems; therefore, a generic and simplified ASR process is illustrated using an activity diagram in *Figure 9*.

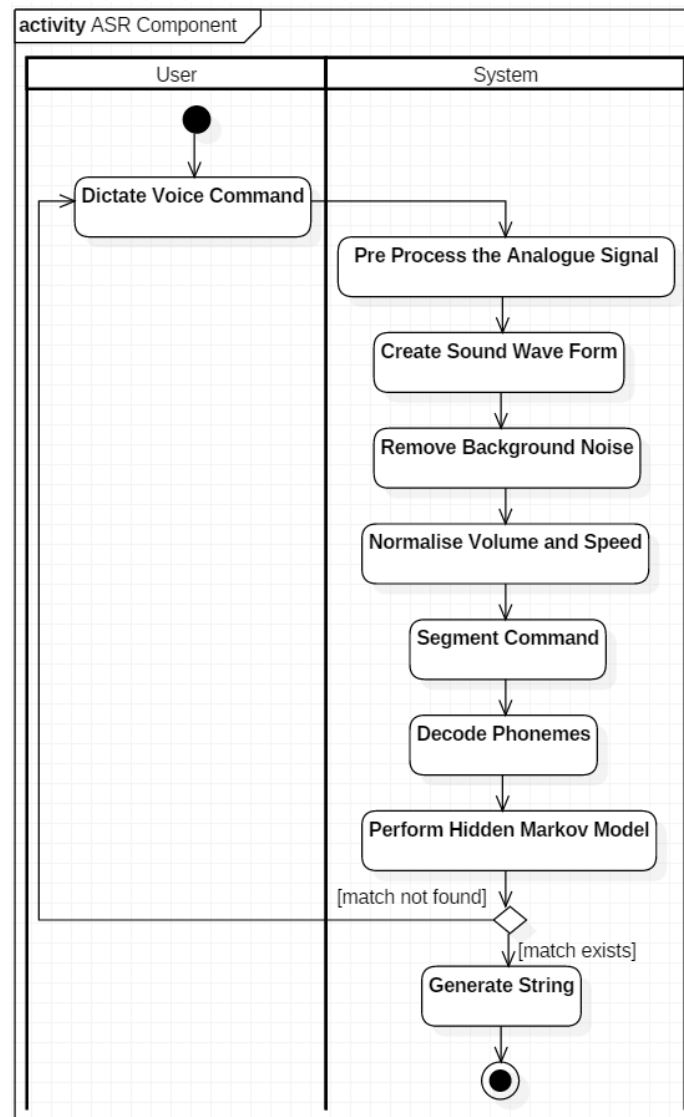


Figure 9 – ASR Activity Diagram

The process is started by the user dictating a necessary command which initiates the phase of acoustic processing in order to create a sound wave, remove background noise and normalise the sound parameters. This enables to achieve a higher success rate within the component as major imperfections are eliminated. The speech is then segmented into phonemes which enables the system to discover distinct sounds and apply Hidden Markov Model (HMM) for statistical analysis and pattern recognition which estimates the word probability based on the supplied phonemes (Gruhn et al., 2011, pp.6-8). In other words, HMM determines the probability of the output to be an accurate representation of the input. The HMM model is frequently trained on data, ensuring a high degree of accuracy is achieved under different conditions. If the command was correctly interpreted, the system generates and returns a textual string. Otherwise, the user is asked to repeat the command. In order to enhance the system's security, a voice biometrics unit may be used to ensure only the device owner is able to dictate the command. Additionally, speech compression element can potentially be used to compress the input voice data for faster interaction with the server, allowing to achieve low latency. This is necessary in the systems where the processing is completed on a centralised server, as oppose to the user's device. Matyunina (2017) proposes to use G711 algorithm which does not lose data during the compression, enabling to perform acoustic processing on an accurate data file.

2.3.2 Natural Language Processing (NLP)

Natural language processing is an AI component with the aim of processing a textual string generated by the ASR component in order to put structured context behind the input. This enables the system to correctly process the verbal input by putting meaning and action behind the words. Such approach allows to automate mundane procedures in various industries, provide accurate translations, process large amounts of structured data and thus reduce the staff required to complete these tasks. However, Callan (2003, p.348) states that such advancements are only possible due to vast amounts of domain knowledge presently available, suggesting that the systems can use large data sets to increase their efficiency and accuracy with each processed query. Fundamentally, NLP is constructed on syntactic, semantic and pragmatic analysis which are sequentially completed on each utterance to correctly compute the model (Cawsey, 1998, pp.98-121). The process is demonstrated in *Figure 10* below.

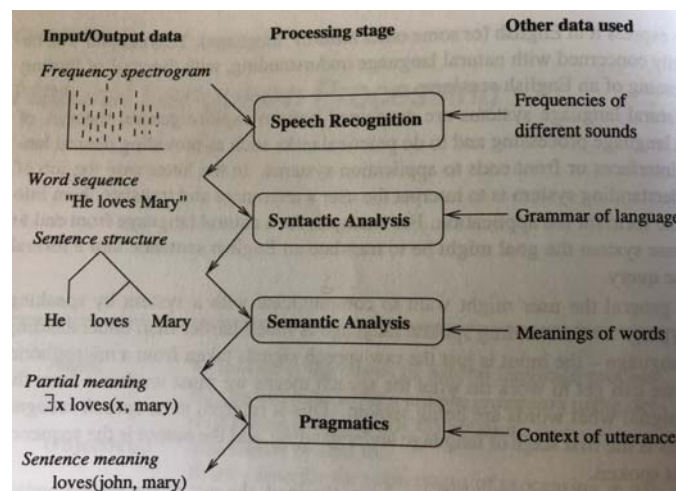


Figure 10 – NLP Stages (Cawsey, 1998, p.100)

Syntactic analysis is concerned with the structure of the complex sentences using language grammar combinations. The predefined syntax rules enable the system to form clauses of words which groups verbs, subjects and objects together to form phrases (Callan, 2003, p.349). Generally, an intelligent parser automatically checks the utterance for grammatic correctness and returns a parse tree reflecting the structure of a sentence according to the grammar. It searches for all possible combinations of the grammar rules which can be applied towards the sentence in order to form an accurate parse tree, allowing to perform further manipulation on the phrase. This method varies in length and difficulty which in turn affects the efficiency of the overall system (Cawsey, 1998, p.110). Additionally, due to the complexity of the language, the system may not be able to cover all possible grammatical sentences, making it difficult recognising advanced medical terms. Advanced systems may also benefit from entity and terminology extraction techniques which locate, extract and classify relevant terms of the text into predefined categories for easier manipulation. This enables to correctly recognise the intent of each domain-specific word and thus increase the accuracy of the component.

Semantic analysis utilises the knowledge of the word meaning in order to discover how their combination delivers meaning to the structured sentence representation (Callan, 2003, pp.349-350). The goal of semantic interpretation is the division of sentence into short and meaningful parts which represent the meaning of the sentence, commonly known as compositional semantics. It is concerned with the representation of each word in a sentence such that they can be systematically combined to reveal the meaning of the sentence. This is commonly represented using knowledge representation schemes which include logical, networked, procedural and structured schemes (Cawsey, 1998, p.112). The use of associations greatly enhances the accuracy of the algorithm because actions can be linked

to the objects, ensuring correct relationships are established and pronouns are correctly handled. This is made possible due to systematised lexical hierarchy and connotation concepts which ensure all words are correctly interpreted by the system in accordance with the intended meaning. Additionally, NLP can benefit from rich machine-readable sources of domain knowledge stored in databases and ontologies to correctly process domain-specific terminology, which includes extensive medical vocabulary (Cohen and Demner-Fushman, 2014, p.6). Lastly, machine translation techniques can be applied at this stage in order to translate the input to a different language.

Pragmatics, also known as context knowledge, examines the use of language and how the previous sentences impact on the interpretation of the current sentence. Fundamentally, this step supplies the sentence with an appropriate context to fill-out the meaning which is particularly useful when the verbal input has an action, adding deeper purpose into the sentence, allowing users achieving their goal. Such aspect is commonly known as speech act which includes informing, requesting and promising actions (Cawsey, 1998, p.116). In order to correctly understand the context, a distinction is made between pragmatic and discourse analysis (Callan, 2003, p.350). Pragmatic knowledge inspects the effect of current situation on the sentence, whereas discourse knowledge refers to a collection of sentences to establish correct associations and resolve pronoun references by converting them into objects. This process is commonly achieved by examining the previous sentence in order to understand the context and base the decision on that, however, such approach does not cover all the cases. For example, the pronoun may refer to an entity mentioned few sentences back. Such complication is frequently resolved by examining the entire text, which has a direct impact on the performance of the system. Modern techniques such as intelligence tagging, and lemmatisation can be used to enhance the algorithm productivity and accuracy of the context knowledge.

It is clear that such sequential algorithm is prone to error because if the syntactic analysis structures the sentence incorrectly, the meaning and context will be inaccurate (Callan, 2003, p.361). This leads to the main disadvantage of NLP which is ambiguity, meaning that there can be several interpretations of the same utterance (Jackson and Moulinier, 2007, p.14). This includes homophones, words which have more than one syntactic category, words having multiple meanings and referential ambiguity, whereby the system is unclear which object a pronoun refers to. These issues are resolved by supplying the systems with large datasets, world knowledge and general everyday information to aid the accuracy of NLP. Similarly, the system may analyse the entire sentence to select an appropriate synonym word based on the overall context to remove the ambiguity (Kocaleva et al., 2016). Jackson and Moulinier (2007, pp.20-21) also note that sentence delimiters are often ambiguous, suggesting that a comma can be used for conjunction and to specify medication dosage. This is commonly overcome by relying on regular expressions or exception rules defined within the system.

Another problematic area of NLP resides in handling of unknown words, suggesting that the system might not operate correctly if the word was not understood. The user may use slang words to get their message across. Cohen and Demner-Fushman (2014, p.4) also indicate that sentences in biomedical journals may begin with a lower-case letter of a gene which adds another layer of complication. Such cases are resolved by applying the following process (Jacobs, 1992, p.18):

1. Spelling Correction – A standard spelling algorithm is applied to the words.
2. Hispanic Name Recognition – Assigns the category of Last-Name to recognised names.
3. Morphological Category Assignment – Categorises words based on their ending.

Modern NLP systems are based on statistical machine learning which increases their accuracy with each processed string using pattern recognition (Kocaleva et al. 2016). This ensures that with time all words are correctly classified and thus the NLP accuracy is increased.

2.3.3 Machine Learning (ML)

Machine learning and knowledge discovery is arguably the most imperative AI component as it facilitates the decision-making process for the entire system. The world is filled with an enormous amount of raw information stored as facts, observations and measurements which constitutes valuable data (Amato et al., 2013). Presently, healthcare establishments generate enormous amount of complex data about patients which could be used for machine training purposes (Durairaj and Ranjani, 2013). This data however does not carry knowledge, which enables the systems to automatically make informed decisions. Data mining enables to extract knowledge and discover the hidden patterns within the large datasets through systematic analysis (Negnevitsky, 2011, p.366). Durairaj and Ranjani (2013) reveal that data mining can drastically increase the diagnostic accuracy and reduce the cost and time constraint in terms of human resources and expertise.

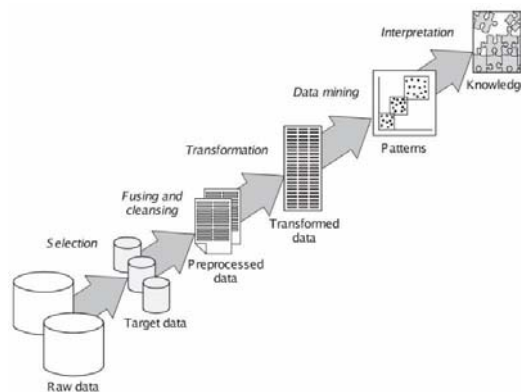


Figure 11 – Knowledge Discovery Process (Negnevitsky, 2011, p.367)

Fundamentally, data mining process consists of five steps which are illustrated in *Figure 11*. It begins by selection of relevant domain data by exploring various data sources and formats required. This is followed by retrieval of data from the target databases and stored in a common source, the collection is fused and cleansed to eliminate all redundant and inconsistent data. Third step performs data transformation ensuring it follows a suitable format for data mining. These three steps are frequently identified as pre-processing. The algorithm then proceeds onto data mining itself which extracts potentially valuable information using the following techniques (Negnevitsky, 2011, p.368):

- ✓ Statistical methods and data visualisation tools.
- ✓ Query tools.
- ✓ Online analytical processing (OLAP).
- ✓ Neural networks and neuro-fuzzy systems.

In general, data mining may utilise several other techniques which aids the system in extracting more knowledge from the data. The last step interprets the results in order to discover and document knowledge, allowing users to easily understand and visualise it. Such knowledge is ready to be used in decision-support systems in order to automate certain business processes.

The main advantage of data mining is the ability to describe structural data patterns in data, generalise and make accurate predictions based on automatic analysis of systematised data. The algorithm can easily develop models based on historical data to predict new tendencies, reveal new patterns aiding in new knowledge discovery and help with reliable decision-making process in numerous systems. However, data mining raises privacy and misuse issues due to acquisition of large datasets (Woodward, 2011). Additionally, the process can only collect accurate factual data, suggesting that the system must correctly differentiate opinions from facts.

Storing enormous amount systematised knowledge is worthless without being able to use it. ML is an AI component enabling the systems to automatically learn and improve from experience without being explicitly programmed. This is achieved by accessing the knowledge datasets discussed earlier. It is known that certain problems are easier to solve by looking at an example, therefore such systems can capture general rules and attempt problem solving without having to rely on a precise algorithm developed by a team of programmers. Machine learning also introduces the benefits of being flexible, adaptable, easily programmable and being able to provide real-time output due to fast processing. Such advancement enables to automate a range of mundane tasks. As the system operates, it becomes more evolved due to continuous improvement, ensuring an accurate output is generated more frequently. However, since the learning is based on knowledge, machine learning is heavily dependent on data which decreases the variability and makes it hard correcting errors. For example, machine learning may base its decision on defective data sample and thus operate incorrectly, leading to lengthy correction processes. Although these risks are minimal in modern systems.

Essentially, inductive learning is used to train a system for classification tasks. This method is perfectly suitable for medical diagnosis applications because the system is supplied with a number of symptoms which results in an output of possible diagnosis (Cawsey, 1998, p.114). By examining the knowledge sample, the system can easily generate rudimentary rules which will be used to classify and predict the output for the supplied input. However, the system is not limited to generating only one rule, suggesting that conflicts may arise when patterns are analysed differently due to large amount of variables. This is resolved by accepting the rule which makes the most predictions correctly. Alternatively, version space learning (VSL), genetic algorithms or decision tree technique demonstrated in *Figure 12* can be used in the ML module in order to educate the system.

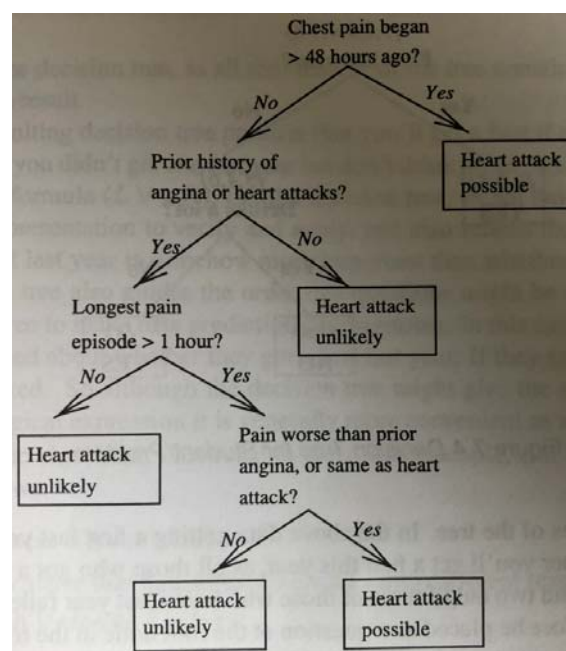


Figure 12 – Decision Tree for Diagnosing Heart Attack (Cawsey, 1998, p.151)

The inductive learning can be contrasted with artificial neural networks (ANN) which apply a significantly different approach to reasoning and learning. ANN is modelled after a human brain which constitutes enormous processing power with the aim of performing classification tasks based on samples. This is achieved by establishing a number of simple and highly interconnected processors called neurons which are linked via weighted links in order to transmit signals through the connections, illustrated in *Figure 13* (Negnevitsky, 2011, p.167). Each neuron accepts several inputs

but can only produce one output. Each link has an associated numerical weighting in order to express the importance/strength of each incoming neuron input. ANN learn by constantly adjusting weights on the links between the neurons until a correct output is generated for the supplied input based on the provided sample data/knowledge (Callan, 2003, pp.296-297). This process is commonly associated with deep learning which uses multiple layers of nonlinear processing units for feature extraction. The rule for updating the weights is defined by the backpropagation algorithm which examines the sample data and calculates the error vector to determine the weight which must change (Callan, 2003, p.301).

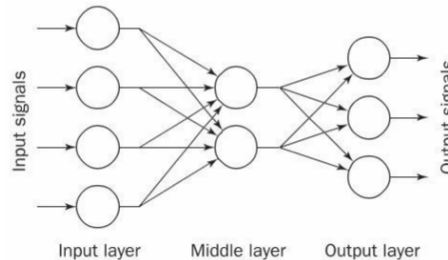


Figure 13 – ANN Architecture (Negnevitsky, 2011, p.167)

The main disadvantage of ANN is lack of versatility due to dependence on accurate data. The quality of data directly depends on training data, suggesting that defective training data will negatively affect the ‘learning’ process. Since the middle layer is generally hidden from the users and the developers, the system can be treated as a black-box, making it hard checking whether the learned knowledge is sensible (Cawsey, 1998, p.157). This leads to unexplained and unexpected system behaviour which can reduce the trust in such approach. However, ANNs are great for pattern recognition tasks which can work with incomplete data after training (Bishop, 1995, pp.301-302). The fault tolerance is also high because the algorithm can run even if one neuron is faulty. Lastly, ANN are capable of parallel processing, making them achieve high performance results. Although, such advantage makes the component hardware dependant since the device architecture must also support parallel processing.

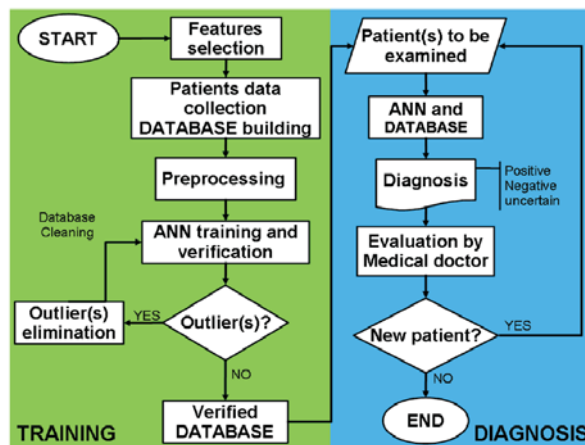


Figure 14 – Generic ANN Medical Diagnosis Algorithm (Amato et al., 2013)

Al-Shayea (2011) reveals that ANN can effectively deal with medical knowledge in order to provide a correct diagnosis. AI could be used to streamline the diagnosis process as demonstrated in *Figure 14* in order to avoid misdiagnosis (Amato et al., 2013). It can also be used in biochemical and image analysis as well as drug design to automate a range of processes and thus increase the diagnosis velocity. Data mining applications begin demonstrating exceptional accuracy by achieved over 97% accuracy predicting cancer in patients (Durairaj and Ranjani, 2013). Such results indicate that medical application developed as part of this project will have a direct positive effect on diagnosis automation.

2.3.4 Text-to-Speech (TTS)

Text-to-Speech or Natural Language Generation (NLG) is an essential AI component of the virtual assistants with the aim of speech output, operating in the reverse order to ASR. TTS provides two options for speech generation; synthesised speech or natural speech prompts recorded from human voice-over artist (Morton et al., 2010). A third option is available in specialised systems whereby every word that the system might verbally output is stored as a whole in the database, ensuring sentences are constructed correctly (Rajput and Nanavanti, 2012, p.80). While such approach pronounces all the words accurately, the versatility of the system is limited, implying that each new response must be recorded by the same voice-over artist. Synthesised speech first appeared in systems for accessibility purposes, allowing sight impaired individuals interacting with their device.

The technology soon evolved into natural speech prompts in order to introduce better UX by making the human-computer interaction feel realistic. This is achieved by recording phonetically rich texts with a number of different combinations of phonemes pronounced by the same voice-over artist (Tabini, 2013). The recordings are then sliced, analysed, catalogued and tagged in a dedicated database, allowing the system to assemble new words which the artist may not have uttered. Such progress introduced improved output quality but increased the cost due to the initial setup. Furthermore, the intonation of such speech generation suffers due to varying pronunciation of phonemes in different words (Cawsey, 1998, p.121). However, as the system formed new words over the time, the pattern recognition technologies enabled to improve the intonation accuracy of each generated word, using an appropriate phenome. Aside from introducing a higher degree of usability into the systems, TTS also enables to automatically translate text into another language by utilising an additional sub-component, making the system more diverse.

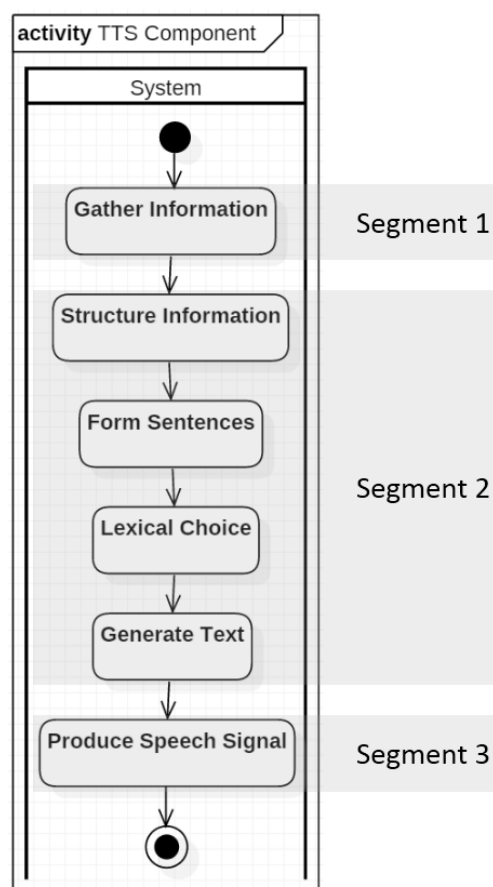


Figure 15 – TTS Activity Diagram

The process of NLG shown in *Figure 15* is fundamentally partitioned into three segments; the determination of what to say, how to say and the generation of speech signal (Cawsey, 1998, pp.120-122). The first is known as text planning which determines what information to say and how to organise this information. It generates the relevant information captured either in the knowledge base, logical form or the system's back-end which must be verbally output to the user. The second is concerned with the sequence of the required words to express the material. This process can be further partitioned into the following steps:

1. Data structuring to define the overall sentence sequence and organisation.
2. Aggregating similar data and splitting information into several descriptive sentences.
3. Using lexical choice produce grammatically correct sentences using appropriate words which the user is likely to understand correctly.
4. Generating the actual text in accordance with the syntax, morphology and orthography rules.

In special-purpose systems, the second step can be achieved by creating several fill-in-the-blanks templates which the system will automatically populate using the correct data (Rajput and Nanavanti, 2012, pp.86-90). This will enable the system to produce correct and easily understandable speech. The third step introduces a complication for a medical application because simplifying medical terminology can lead to inaccurate information being generated for the user. For example; simplifying the term *Migraine* down to *Headache* can potentially increase the health risks for the patient due to different treatment process. Lastly, the speech signal is generated using the process discussed earlier which is output to the user through the device, establishing a human-computer interaction.

2.3.5 Graphical User Interface (GUI) Client

A front-end is a crucial aspect of any modern application which initiates the interaction process with the system, allowing users completing their task. It is debated that a correctly-designed user interface helps users achieve their goal faster, significantly enhancing the overall UX. Johnson (2010, pp.1-10) notes that the humans commonly perceive what they expect, which is based on three factors; past experience, present context and future goals. This suggests that the front-end for the current system must follow predefined consistency conventions allowing users reaching their goal appropriately. These include intuitive menu/navigation, no information overload, unambiguous wording, distraction avoidance, use of appropriate graphics and language. Tidwell (2011, pp.444-445) recommends designing a GUI with the user's goal in mind, displaying only the necessary content which the user requires to view, without overcrowding the page. As previously discovered, an appropriate GUI colour scheme must be selected in order to positively impact the emotional and psychological state.

Another collection of issues arises from the interaction methods. Hempel (2008, p.4) notes that an incorrectly configured speech recognition system may fright customers who may never use the system again as a result of this. This suggests that if the users are unable to discover how to initiate a speech recognition function (commonly achieved by pressing a microphone icon or pronouncing a specific keyword) they might abandon the application. Moreover, Larson (2003, pp.6-9) points out that users are unable to edit text while they speak and might be afraid to use such applications in several situations. It is therefore vital not to restrict users to speech input, and instead allow them to enter data using a keyboard alternatively. Poder et al. (2018) suggest that a strong validation mechanism must be implemented either at the front-end or the back-end to limit the potential input errors. Larson (2003, pp.9-10) also notes that the users might not remember the conversation or hear the verbal response by the system, suggesting that the system must keep a transcript of the chat within the application. Such feature would also enable the user to obtain a copy of the conversation by requesting a transcript sent to their email address.

In addition to taking into consideration the above findings, an effective web-based application must adapt to a vast amount of devices with the wide-ranging screen sizes, ensuring that all content is scaled appropriately. This is commonly achieved using responsive design, which enables the developers to build an interface layout on a fluid grid which can “dynamically adapt to the diverse viewing environments” (Nebeling and Norrie, 2013). The main advantage of responsive design is maintenance because only one type of application requires bug fixing and introduction of new features. Moreover, users will experience consistent theme and features regardless of the device used, keeping the UX equivalent on all devices and preventing users from adapting to the system depending on the device used (Wisniewski, 2013). Such benefit also aids users completing their goal faster. Lastly, the adoption of responsive design makes the system future proof as the system will easily adapt to all screen sizes introduced in the nearby future.

Nonetheless, responsive design also introduces a number of significant drawbacks which must be considered during the analysis phase of the project. The significant disadvantage includes the initial setup costs because the developers must design the system with all screen sizes in mind, ensuring it scales correctly on a range of modern devices. This can be avoided by using a pre-made framework such as Skeleton (2018), Pure (2018) or Bootstrap (2018) which through the use of specialised CSS (Cascade Styling Sheet) classes ensure the content scales correctly on all devices. These tools come equipped with full documentation, relevant sub-components to complement the front-end and easy syntax, allowing different developers to code and maintain the system. Another drawback of responsive design is performance issues due to the fetching and scaling of all necessary components onto the user’s device (Wisniewski, 2013). The slow loading time of the application can be resolved by selecting short directory names, placing *script* tags towards the bottom of the page (Els, 2015, pp.13-14) and ensuring the code is clean without any syntax errors.

2.4 Summary

This chapter has reviewed the existing systems from two perspectives; the medical applications and universal virtual assistants. It then proceeded onto briefly comparing the reviewed systems in terms of features and operation to discover the main user interests of such systems. Existing systems were decomposed into components and engines to discover all relevant technology. Individual AI components were thoroughly inspected by discussing their advantages and disadvantages to reflect upon practicality towards the current project. Appropriate diagrams were presented in order to illustrate the algorithm of each component and sub-component, allowing to understand the main challenges. Throughout the literature review, the following recommendations were discovered which must be considered when implementing the new system.

1. Do not restrict users to selecting a specific symptom input.
2. Show input suggestions or most frequently asked questions.
3. Utilise responsive design frameworks to develop the front-end.
4. Avoid asking many questions to establish an accurate diagnosis.
5. Show example question if the user is unsuccessful on the first attempt.
6. Avoid simplifying diagnosis to basic words to eliminate misdiagnosis issue.
7. Generate the conversation transcript which the user can request if required.
8. Do not limit input fields to verbal input only, allow users to input textual data.
9. Image and video processing facilities within the system which can also operate offline.
10. Use blue or green colour-scheme to positively affect the emotional and psychological state.

Having proposed a number of useful recommendations, the thesis will now begin the system analysis.

3 High Level Analysis

3.1 Introduction

The analysis chapter will focus on capturing all actor goals in order to develop an accurate model of the system. The goals will be then translated into a number of requirements which will help evaluate the system and eliminate the requirement ambiguity at an early stage of the project. Such approach will ensure a correct system is delivered within the specified time and budget.

3.2 Use Case Diagram

The earlier literature review has provided insights into potential system users and their goals. Use Case diagram is frequently used to model the system which introduces a non-technical, easily understandable and maintainable representation of the computer system, clearly defining its boundary. The diagram is commonly created using UML (Unified Modelling Language) in StarUML (2018) software. It is a widely-used free tool allowing to develop a range of diagrams using standardised notation to graphically represent the system scope. A use case diagram to reflect the actors and their goals will now be developed.

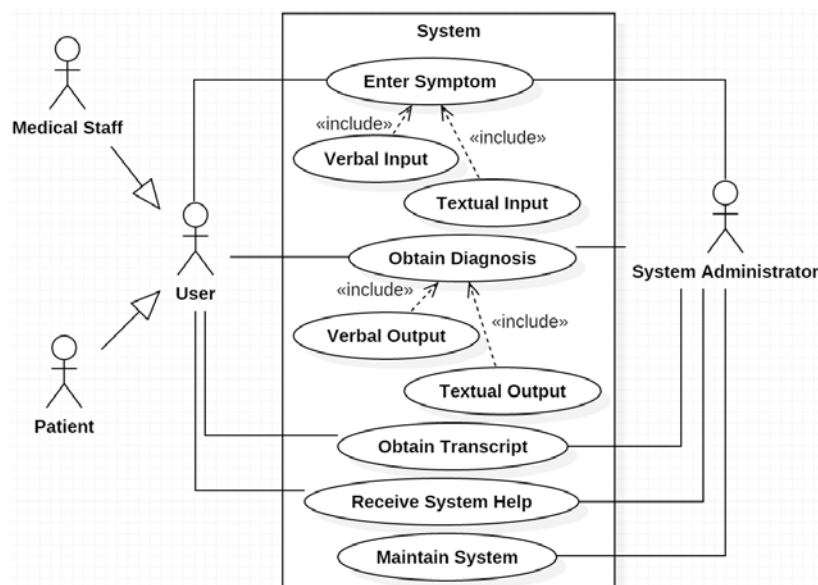


Figure 16 – Use Case Diagram

Figure 16 demonstrates an overview of the user goals when interacting with the system. There are limited use cases due to simplistic nature of the system, suggesting that the user wishes to receive an accurate diagnosis as soon as they enter relevant symptoms. Additionally, as a UX improvement to the current systems, the transcript of the chat feature was added as one of the user goals. It is believed that the users might want to obtain a copy of the diagnosis in order to show their doctor. Similarly, the system help feature is not a direct user goal, however, it is a relevant use case for this user group because the users must be shown how to use the system correctly. The *Maintain System* use case is the primary goal of the system administrator who will be in charge of ensuring the system is operating correctly and adding new features. This use case will be reflected as a non-functional requirement.

Use case descriptions are typically completed in conjunction with use case diagram to explore the user goal by introducing a detailed description of actions and alternate flows. While adding a number of benefits such as increased traceability and improved system understanding, use case descriptions for the entire system require a significant project time investment. Due to such limitation, use case descriptions are not developed in this project.

3.3 Functional Requirements with MoSCoW Prioritisation

The use case diagram will now be translated into system's functional requirements. These are used to signify the system's behaviour, limit the scope and provide testing fundamentals to measure the system quality. The requirements are typically prioritised using MoSCoW when following an Agile methodology, which divides the requirements into Must, Should, Could and Won't Have. Such strategy enables the developers to focus on important aspects of the system first, ensuring all critical functionality is ready to be delivered within the agreed time and budget. As a result, requirements with low ranking may not be implemented during the first revision of the system because they are not critical to its operation. These requirements will be considered during the later revisions of the system.

ID	REQUIREMENT	PRIORITISATION	USER GROUP
FR1	Enter Symptom	MUST	User
FR2	Obtain Diagnosis	MUST	User
FR3	Generate Transcript	COULD	User
FR4	Receive System Help	SHOULD	User
FR5	Generate Symptom Suggestions	SHOULD	System
FR6	Provide Verbal Interaction	MUST	System
FR7	Provide Textual Interaction	MUST	System
FR8	Provide Graphical Interaction	WON'T	System

Figure 17 – List of Functional Requirements with MoSCoW Prioritisation

As seen from *Figure 17*, the user goals were carried directly from the use case diagram to represent the major system functionality. Further requirements were added to support all intended system behaviour which will enhance the UX, as discovered earlier. FR6, FR7 and FR8 requirements denote the ability to use voice and text in order to interact with the system as well as the ability to share videos and images with the system to aid in diagnosis establishment. For example, the user may upload an image of a skin condition which the system shall process and generate a diagnosis. However, due to the complex nature of this feature, FR8 requirement was prioritised as Won't have and will not be developed throughout this project. Although, it can be considered in the future work.

The functional requirements are expanded into a Requirements Catalogue, located in *Appendix C*.

3.4 Non-Functional Requirements

Non-functional requirements are formulated from KQF (Key Quality Factors) to describe the system attributes, allowing to quantify and measure the quality using comprehensive metrics (Galin, 2004, pp.36-49). Several software quality models such as McCall's, Boehm's, Dromey's and ISO 9126 were established over the years to provide framework for measuring the system quality as a collection of the desired attributes (*Software Quality Management*, 2016). Individual models suggested specific KQF and a range of metrics which could be used to quantify and measure them (Vinayagasundaram and Srivatsa, 2007). It is suggested that a typical system has between three to eight KQF in order to limit conflicts, improve understanding and meaning of each KQF as well as focus on critical system characteristics to ensure the project is completed on time (Miyoshi and Azuma, 1993). By studying the existing systems, several KQF are defined for the new web-based application in order to ensure it achieves at least the same degree of quality in order to be accepted by the users. Below are the KQF which are defined for the system developed throughout the project.

- ✓ **Maintainability** – The system must achieve loose coupling and follow most programming conventions to ensure new system features or bug fixes can be easily completed.

- ✓ **Efficiency** – The throughput must be as high as possible while using minimal resources to generate an accurate diagnosis as fast as possible.
- ✓ **Reliability** – The system needs to provide reliable services at all times to build user trust. It should implement fault toleration mechanisms to avoid failures and recover automatically.
- ✓ **Usability** – Since the application is focused around customers, it is important to keep the system easy to use, otherwise the system may not be used even if other KQF are achieved.
- ✓ **Scalability** – The system should support the introduction of new medical domains in the future to ensure it can provide diagnosis for all types of medical concerns.
- ✓ **Portability** – The system users must always have access to the web-based application regardless of the device and internet connection.
- ✓ **Security** – Since the system will be processing (not storing) private medical data, it is vital that all connections are secure, and the data is fully encrypted.

The KQF are then translated into a number of SMART non-functional requirements demonstrated in *Figure 18*. It was made sure that there are no conflicts between the requirements, all requirements are realistic and measurable, allowing to assess the system correctly.

KQF	REQUIREMENT
Maintainability	NFR1. The application code shall be modular.
	NFR2. The application code shall be loosely coupled.
	NFR3. The application code shall follow most programming conventions.
Efficiency	NFR4. Latency shall be under 5 seconds across all internet connections.
	NFR5. The system shall be able to run smoothly with 1,000 concurrent users.
Reliability	NFR6. The system shall be able to process 10,000 requests within 12 hours.
	NFR7. The system shall have a ROCOF of 0.00001.
	NFR8. The system shall be available 99.8% of the time during the year.
	NFR9. The system shall be restored within 2 hours after a crash.
Usability	NFR10. The system shall use consistent layout on all pages.
	NFR11. The system shall provide guidance to the user completing a specific task.
	NFR12. The system shall provide meaningful error messages to assist the user.
	NFR13. Average user shall be able to complete their goal within 5 minutes.
Scalability	NFR14. New system functionality shall be added within current architecture/code.
Portability	NFR15. The system shall be accessible from desktops, laptops, tablets and smartphones.
	NFR16. The system shall be accessible from Apple iOS/macOS, Android and Windows OS.
	NFR17. The system shall be accessible at least from Google Chrome browser.
Security	NFR18. All system pages shall be HTTPS through the use of SSL.

Figure 18 – List of Non-Functional Requirements

By expanding each KQF into a number of non-functional requirements, the developer can accurately measure the quality of each system attribute to ensure the product is accepted by the users.

3.5 Summary

Throughout this chapter, the literature review findings were reflected in a use case diagram in order to represent all user goals for the system. Such strategy has enabled to establish several MoSCoW prioritised functional requirements which will be used to assess the system throughout the project. A detailed requirements catalogue was created to document all necessary information. Lastly, several KQF were defined for the current project, allowing to measure the system quality which were translated into a number of SMART non-functional requirements. In order to improve the analysis phase, use case descriptions could be created to outline each use case in more detail and eliminate ambiguity. However, due to time limit, they were not created. Since the analysis chapter is considered complete, the thesis will now proceed onto system design.

4 High Level Design

4.1 Introduction

The design chapter will utilise the previously discovered knowledge and recommendations in order to plan the novel web-based application. It will begin by examining the architectural styles used in the existing solutions and proposing an appropriate style for this project in order to complement the selected KQF. It will then proceed onto designing several UML diagrams to represent the system components and their interaction process. A suitable database design will be established to ensure all complications are resolved prior to the implementation. This chapter will then outline the optimal UI (User Interface) by considering UX complications and earlier gathered knowledge. Lastly, appropriate application icons will be developed to represent the system on all platforms.

4.2 System Structure

4.2.1 Software Architecture Selection

Software architecture has a direct impact on several KQF which include maintainability, scalability, efficiency and reusability (Vinayagasundaram and Srivatsa, 2007). Architecture helps the system achieve an adequate degree of quality, aside from allowing users completing their goal. Over the years, several architectural styles have been established, namely; client-server, n-tier, layered, component-based and service-oriented which aid in achieving KQF. Considering how novel virtual assistants are, the developers have adopted a modern and effective Service-Oriented Architectural (SOA) style. For example, Microsoft Cortana adopts cloud-based services (Kapko, 2018), whereas Apple Siri and Google Assistant utilises client-server approach through SOA. Google Assistant performs voice recognition on the client-side, whereas Apple Siri processes on the server-side (Kosner, 2012). Such strategy enables Google Assistant to achieve low latency and thus improve the UX. However, Siri allows server-side learning to improve accuracy of the system, making it smarter the more it is used.

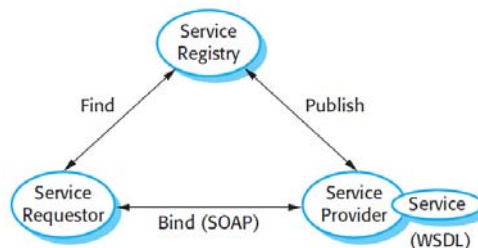


Figure 19 – Service-oriented architectural style (Sommerville, 2010, p.510)

SOA is a collection of independent components which communicate with each other in order to provide functionality to complex applications, demonstrated in *Figure 19*. It can be seen as a black-box solution enabling software communication using XML-based protocols, such as SOAP (Simple Object Access Protocol) which supports structural information exchange between the services and WSDL (Web Service Definition Language) which is a standard for defining service interfaces (Sommerville, 2010, p.509). REST (Representational State Transfer) is a simplified architectural style of SOAP and WSDL, allowing to implement web services. A service is software and hardware independent, stand-alone, scalable, reusable, self-contained, distributed, loosely coupled and standardised solution delivering a particular functionality (Vogel et al., 2011, p.206). It consists of service implementation and interface. The service providers develop a service along with full documentation and interface specification using WSDL. The service is published in the service registry to make it discoverable. The service requester locates the service specification on the service registry and finds the service provider. Once the provider is found, the communication between the application and the service can be established.

Earlier research has revealed that some components are platform-dependant, suggesting that SOA is a perfect solution in the current project. Such architectural style will enable to implement several hardware and software independent, scalable, reusable and self-contained services which will introduce a high degree of maintainability into the overall system. It will also enable to reuse ready-made services/components to deliver critical system features without having to spend project time developing same functionality from scratch. The only drawback of SOA is reliance on the service providers because by reusing their service, the product will depend on correct service provision by the vendors. Other architectural styles such as N-Tier and Layered were not the right choice for this project because they lack maintainability, efficiency and scalability KQF which SOA provides. Additionally, both architectural styles are restricted to language and hardware dependent tools/technologies which is a huge drawback in the current project.

4.2.2 Component Diagram

A component diagram is typically developed using UML to show the complex system distribution and relationships between each module. It is commonly used in Component-Based Development (CBD) to demonstrate the system distribution. The components are split into logical and physical classes. The logical component is independent of the physical and represents what each component does, whereas the physical component represents software and hardware technology, describing how the component does its job. The components communicate by providing or requiring an interface in order to deliver a service and thus build a complete system.

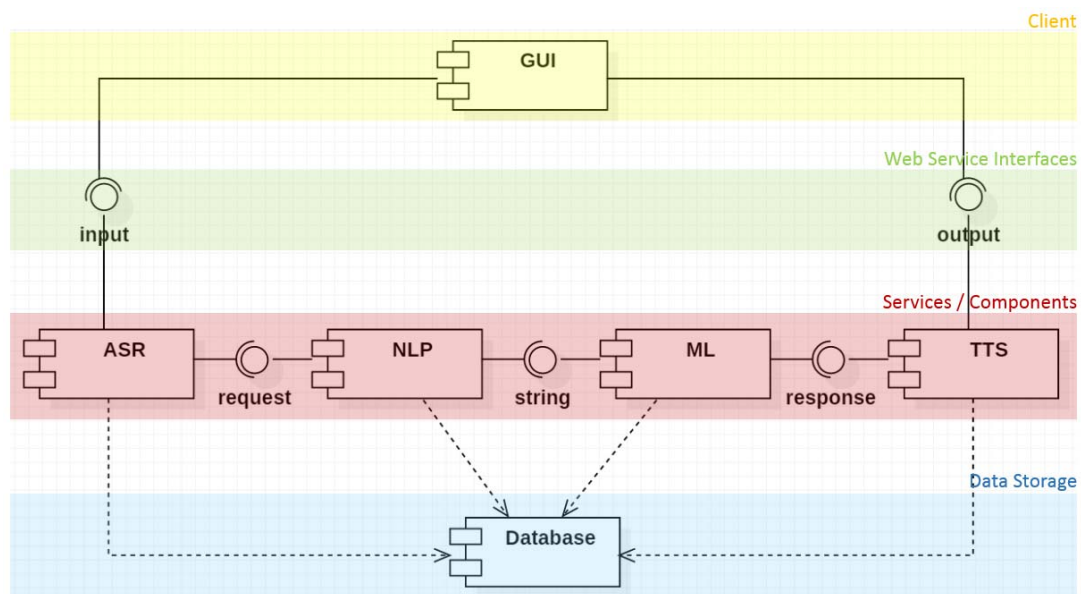


Figure 20 – Component Diagram

The SOA structure of the system for this project is illustrated in *Figure 20*. The web-based application will have one client which will utilise the services via web-service interfaces. Fundamentally, the GUI will pass an input from the system to the ASR component which will transcribe the utterance into a textual string. It will then be processed by the NLP component to introduce the meaning and context behind the request, allowing the ML component to correctly generate a textual output. The generated string will then proceed into TTS component which will output the response using speech. The services will also communicate with appropriate databases to perform processing and train the system further. Such modularised approach will ensure the system adheres to most non-functional requirements established earlier by implementing a range of independent services. It will also enable to change and introduce new components in the future in order to alter the system's functionality.

4.2.3 Top-Level System Sequence Diagram

The top-level sequence diagram is created to demonstrate the order of component interaction within the system. Essentially, such diagram enables to visualise the workflow of the entire system. Using UML enables to model the dynamic behavior of the system and the relationships between the components. Alternate and parallel system flows can be modeled to ensure the system operates correctly when implemented. A sequence diagram for this project will now be created.

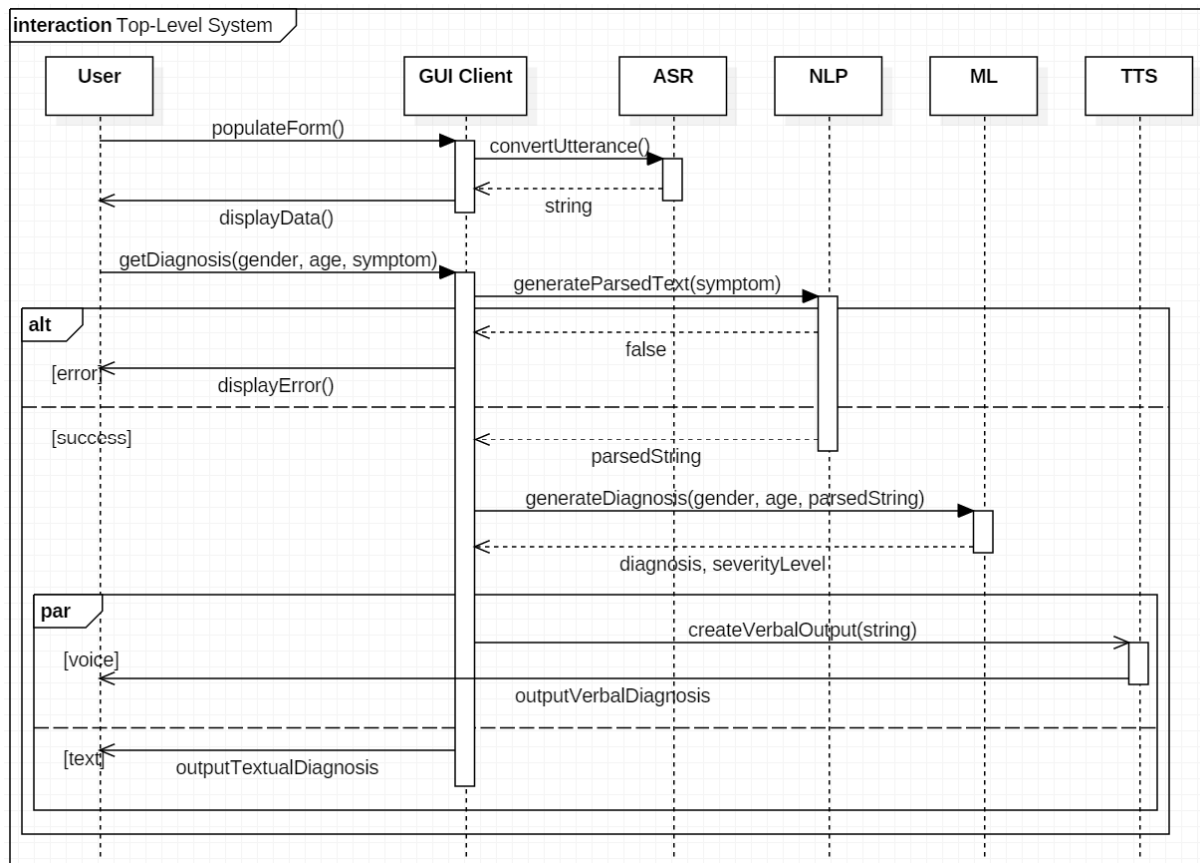


Figure 21 – Sequence Diagram

As seen from *Figure 21*, the sequence diagram demonstrates all the components from the component diagram created earlier. Such strategy introduces a more detailed visual model of the system interaction, allowing to eliminate major issues during the design phase. Fundamentally, the user will interact with the GUI client which will automatically send requests and process responses from a range of components. The diagram also demonstrates one alternative (`alt`) flow to accompany the NLP component which can return an error message if the NLP component is unable to process the supplied utterance. Similarly, the parallel (`par`) activities can happen when the diagnosis is output to the user. The system must use textual and verbal output methods concurrently in order to improve the UX.

4.2.4 Use Case-Level Activity Diagrams

Use-case level activity diagrams are mainly created to model a task and model the system's dynamic behavior of a specific task. The diagram consists of several linked actions which the users must complete in order to reach their goal. Such diagram enables to explore alternate flows by making a number of decisions and discover the parallel activities which the system can execute concurrently. Due to the time limit, it is not possible to implement activity diagrams for all use cases, therefore, two key activity diagrams were selected and will now be created.

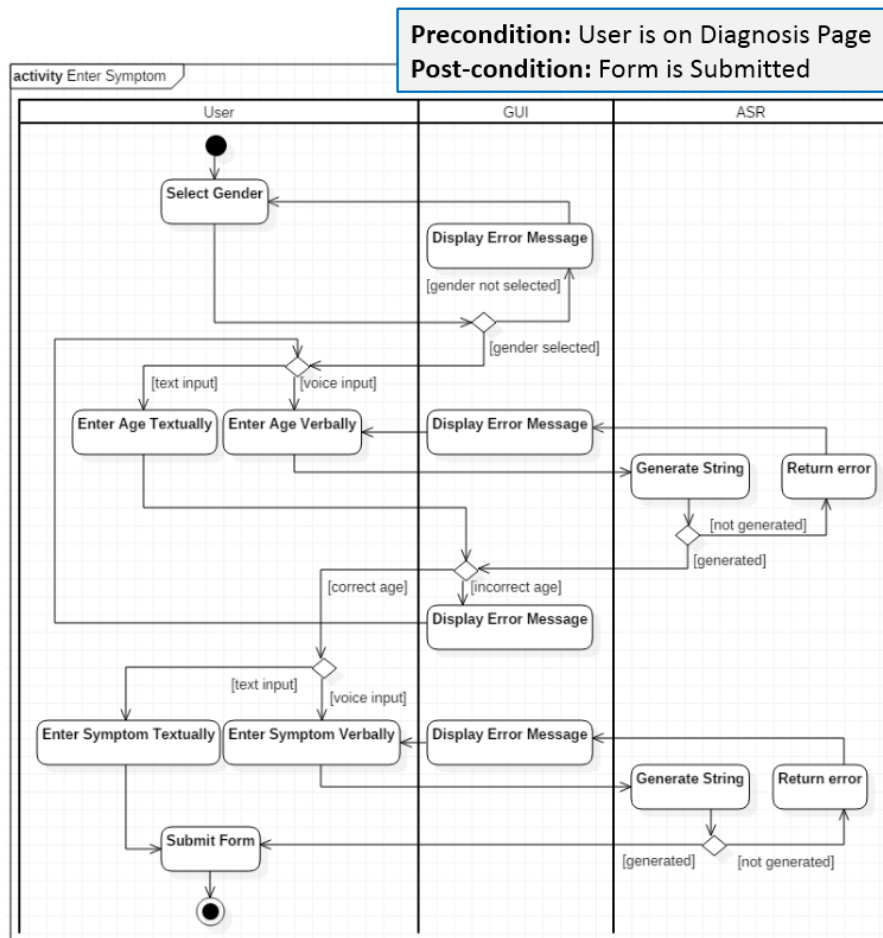


Figure 22 – Enter Symptom Activity Diagram

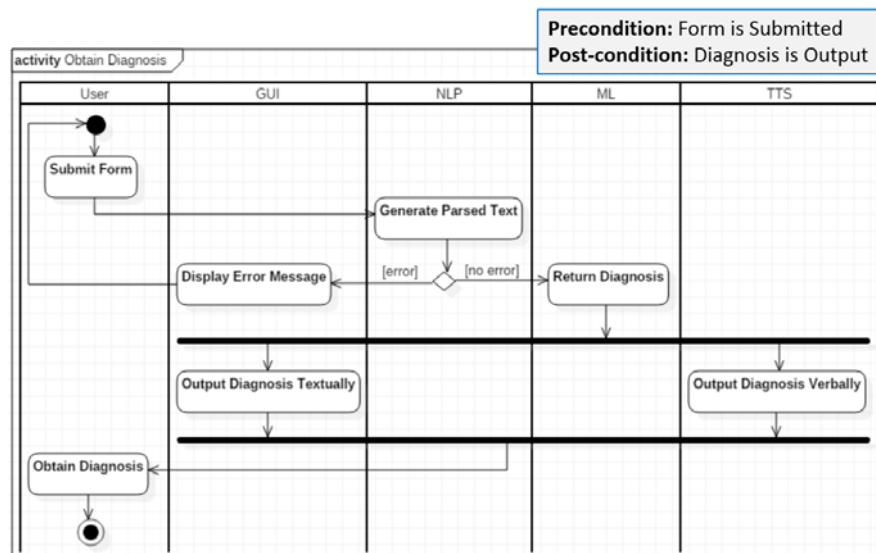


Figure 23 – Obtain Diagnosis Activity Diagram

As seen from *Figure 22* and *Figure 23*, each diagram consists of several intermediate actions with a number of decisions which the actors must complete in order to reach their goal. Each diagram is complimented with precondition which must be met to start the activity and post-condition which assesses whether the use case was achieved. It is evident that both diagrams correctly model the system behavior, allowing actors correctly achieving their goals.

4.3 Database Design

4.3.1 Entity-Relationship Diagram

Entity-Relationship (ER) diagram typically represents the data structure of the objects, allowing to design and implement a fully functional database from a UML model. Additionally, an ER diagram aids in multiplicity establishment by showing the number of occurrences which an entity can have with an instance of that type. An ER diagram for the novel system will now be designed.

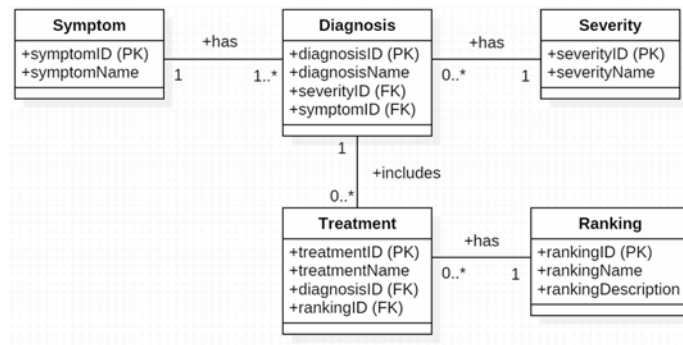


Figure 24 – ER Diagram

Figure 24 illustrates an ER diagram in 3rd Normal Form to reflect the database structure required for the current project. Each table consists of Primary Key (PK) and where relevant, a Foreign Key (FK) in order to show an interaction/relationship. It was assumed that the *Severity* table will hold values to reflect the severity of the diagnosis, allowing to suggest next steps to the users. Similarly, *Ranking* table is projected to store values to reflect the data source of the treatment. For example, the treatment suggestion could have been captured from a qualified doctor or a medical journal. It is evident that treatment suggestion from a doctor would be prioritised higher due to the increased accuracy. Lastly, it was assumed that the symptom may have many diagnoses because majority of symptoms overlap between the diagnoses and the user would have to supply additional information in order to help establish an accurate diagnosis based on the additional data.

4.3.2 Data Dictionary

Data dictionary outlines the database structure by showing format and relationships between tables.

TABLE NAME	ATTRIBUTE	DATA TYPE	LENGTH	PK/FK	NOT NULL
Symptom	symptomID	INT	10	PK	
	symptomName	VARCHAR2	100		•
Diagnosis	diagnosisID	INT	10	PK	
	diagnosisName	VARCHAR2	100		•
	severityID	INT	10	FK	
	symptomID	INT	10	FK	
Severity	severityID	INT	10	PK	
	severityName	VARCHAR2	20		•
Treatment	treatmentID	INT	10	PK	
	treatmentName	VARCHAR2	250		•
	diagnosisID	INT	10	FK	
	rankingID	INT	10	FK	
Ranking	rankingID	INT	10	PK	
	rankingName	INT	10		•
	rankingDescription	VARCHAR2	20		•

Figure 25 – Data Dictionary

As seen from Figure 25, the data dictionary for this project consists of integers (INT) and various character (VARCHAR2) to reflect the PKs, FKs and fields requiring strings. It was decided to limit the length of each field to keep the data consistent and prevent all fields from being null. Such feature is default for PK and FK, therefore, it is not needed to specify Not Null attribute for them.

4.4 Front-end Design

4.4.1 Medium Fidelity Wireframes

In order to solve the front-end problems at an early stage, medium fidelity wireframes are developed to plan the initial layout and the navigation structure of the web-based application. Such modelling allows to rapidly conceptualise the system by taking into the account all user goals and capturing all necessary feedback. Changes can be easily introduced throughout this stage without impacting the cost and the schedule of the project. While there are several tools for wireframe designs, pencil and paper was selected in this project due to simplicity, effectiveness and ease of creation.

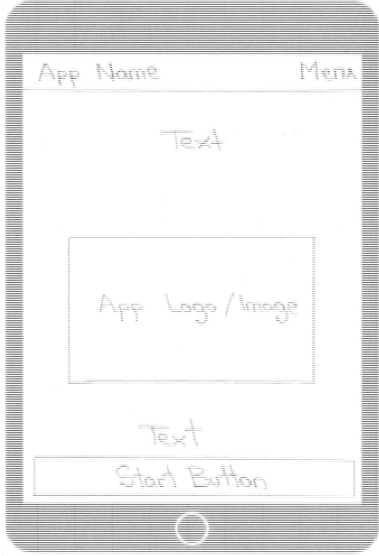



	
Home	Capturing User Consent
	
Diagnosis Establishment Page	Help Page

Table 2 – System Medium Fidelity Wireframes

Table 2 illustrates the four system pages which are critical to the users in achieving their goals. From the wireframes it was discovered that the menu elements must be hidden in the hamburger-style menu on all smartphones to prevent user distractions when displayed on the navigation bar. Additionally, it was revealed that *Help* page must adopt collapsible menu to hide question answers in order to prevent page clustering issues, as discovered during the literature review. Such approach will help users locate an appropriate question and by expanding it, view the corresponding answer.

4.4.2 Low Fidelity Prototype

After examining the wireframes, a low fidelity prototype of the system is developed to introduce further detail onto each screen. It enables to test different usage scenarios, visualise the end-product and make appropriate changes to ensure all conventions and best practices are taken into the account prior to the implementation. Mockingbird (2018) tool was used to develop the system prototypes because unlike similar products such as Axure (2018) and Mogups (2018) it provides a free and easy to use platform for interactive prototype designing with an ability to easily edit the content.

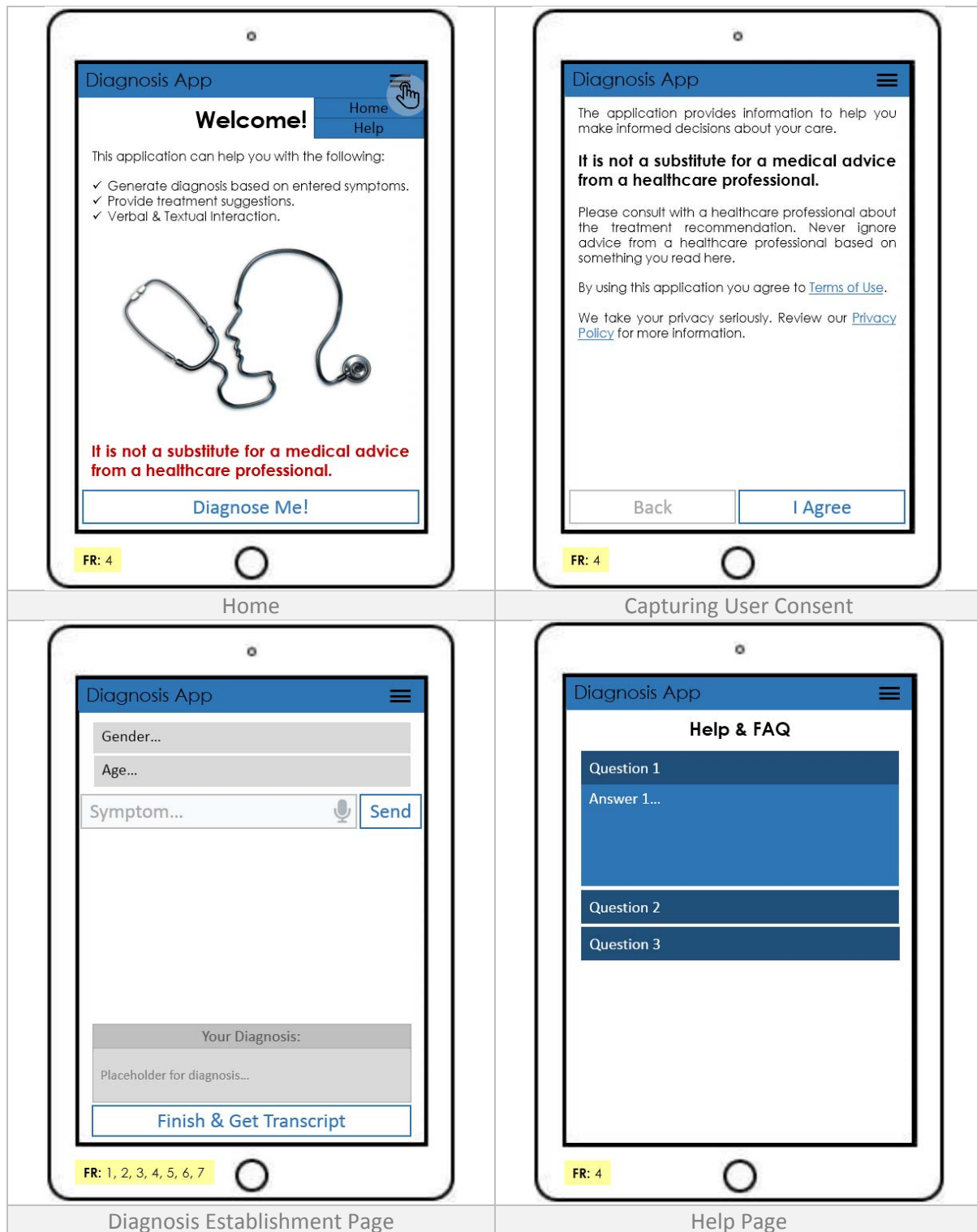


Table 3 – System Prototype

All earlier discoveries and recommendations were considered and applied towards the novel system prototype. To introduce traceability into the product, validate and verify all user goals, each page shown in *Table 3* displays the Functional Requirements which can be achieved on that application page by the users. It is also worth noting that while it is not possible to represent graphically, all textual output will be verbally produced by the system using TTS component. The page follows a standardised convention whereby the navigation bar is located at the top of the screen and critical system buttons are placed towards the bottom of the screen, allowing users reaching them easier. Appropriate colour scheme was selected on all front-end elements to ensure users are intuitively guided through the novel system. The prototype design has proven that the earlier completed analysis can be applied towards the new system in order to achieve all aims and goals. Lastly, it would be beneficial adding icons to the buttons to illustrate the action/functionality. This addition would make the system even easier to use from the UX point of view, as the users would not have to read the button name and instead look at the icon which accurately represent the action.

4.5 Summary

This chapter has selected a suitable architectural style for the novel web-based application. A component diagram was created to illustrate the component decomposition and their relationships. Furthermore, appropriate sequence and activity diagrams were designed to model critical system tasks prior to the implementation. The chapter then proceeded onto designing a suitable database by creating an ER diagram and a complimentary data dictionary. Based on literature review findings, medium-fidelity wireframes and low-fidelity prototype were designed to outline the front-end of the web-based application which the users will use to interact with the system.

It would be beneficial adding class and state-chart diagrams in order to show further system detail, however, since the system will adopt SOA, certain components can be reused from third-party vendors. Such approach would save valuable development time, but the service would come as a black-box, suggesting that the internal processing logic is hidden. As a result of this, the above-named diagrams are considered obsolete in the current project. The design chapter is considered complete, the thesis will now proceed onto the implementation chapter.

5 System Implementation

5.1 Introduction

The implementation chapter will begin by reviewing and selecting the suitable tools for the system's implementation in accordance with the earlier defined requirements and recommendations. It will then discuss the back-end and front-end development approaches in order to highlight how various complications were resolved throughout the implementation phase.

5.2 Tools and Technology Selection

5.2.1 System Development

It is important to choose the right technologies and tools which will support SOA, enabling to correctly implement the system conforming to all requirements. The web-based application will be developed using Atom (2018) IDE (Integrate Development Environment) because unlike competitors such as Notepad++ and NetBeans (2018) this software supports many languages and allows various customisation by installing relevant open-source packages. The use of Atom (2018) IDE will also positively impact the overall quality of the system due to intelligent code completion, incorrect syntax and logical error highlighting which provide comprehensive error reports.

The responsive front-end will be developed using HTML and CSS with the aid of Bootstrap (2018) library. Alternatively, it would be possible to use Zurb Foundation (2018), Skeleton (2018) and Pure (2018) libraries to implement the responsive front-end. Bootstrap (2018) was chosen because it is the most-widely used tool with countless additional easy-to-use frameworks and sub-components which are compatible with a range of other technologies. jQuery Form Validator (2018) will be used to perform client-side validation, ensuring the input matches the expected parameters. The closest validation alternative is jQuery Validation Plugin (2018) which was not chosen because unlike first, it does not work correctly with the Bootstrap (2018) framework. As discovered during the prototype designs, the buttons must have an icon to resemble the action. This will be achieved by using the Font Awesome (2018) library. Alternatively, Bootstrap glyphicons could be used to achieve the same effect, however, due to the limited amount of available glyphicons, these were neglected. The application icons and favicon will be developed using Adobe Fireworks because this software allows to easily create and edit *png* (Portable Network Graphics) files which introduce higher bit depth and thus result in broader colour variation. ConvertICO (2018) will be used to convert a *png* file into *iso* (icon) file, making it discoverable by the browsers. The icons will be validated using IconTester (2018) to ensure they are correctly displayed on all types of the required devices. This tool has no close alternatives because it is the only one which validates Apple and Android devices simultaneously.

The AI back-end components are best implemented using Python which is an interpreted high-level language, supporting multiple programming paradigms. Due to its flexibility and high performance, the language is widely selected for Artificial Intelligence (AI) implementation when compared to Java and Lisp alternatives. Additionally, Python offers a range of useful AI libraries to complement the system development. It helps write stand-alone service which can be used by the system. However, due to the limited time of this project, majority of the components will be reused from other vendors in the form of APIs. Such approach will enable to demonstrate a proof of concept within the provided time frame, allowing to discover new recommendations and thus add new knowledge into the field.

The ASR component will be implemented using Web Speech API (2018) which is an open-source API allowing to incorporate voice data into web applications. Alternatively, Google Cloud (2018) speech-to-text could be used to achieve similar result. Though, Google Cloud (2018) was not appropriate for this project because the API is only free for 60 minutes, after which the developer would have to pay

a small fee in order to use the service. Web Speech API (2018) on the other hand is always free, making it an ideal solution for the project. TTS component will be delivered using the ResponsiveVoice.JS (2018) JavaScript library which is fully suitable for this project. This is due compatibility with all modern devices, being free for non-commercial use, lack of dependencies and consuming little space on the server. Due to this, there are no close alternatives which could be used in this project.

To generate diagnosis based on the supplied symptoms, Infermedica (2018) API will be used. It is an advanced and secure AI symptom checker API which allows making 2000 free request calls, after which the developer is expected to pay a small fee. Similar APIs such as ApiMedic (2018) and MedWhat (2018) were disregarded because the first does not use AI to generate an accurate diagnosis and the second does not provide free API calls. There are presently no other close alternatives to Infermedica. The API operates by accepting cURL requests with the necessary data in the JSON format which it then processes using AI technologies and returns a JSON with the user's diagnosis. As seen from *Figure 26*, the API heavily relies on factual knowledge captured from the doctors which is manually tested and expertly reviewed, ensuring the system provides an accurate diagnosis each time.

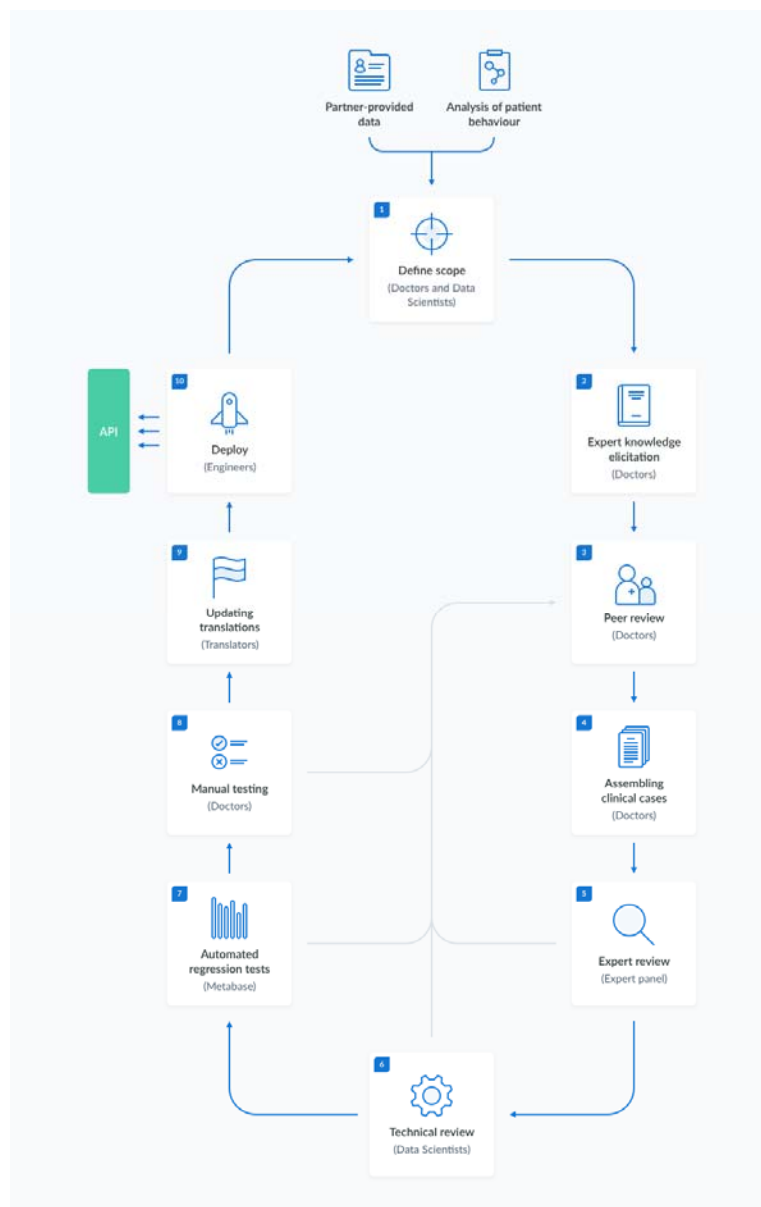


Figure 26 – Process of Medical Knowledge Capture (Infermedica-Developer, 2018)

Infermedica (2018) API also provides NLP facilities which correctly processes clinical concepts. As discovered during the literature review, medical terminology was a weak point of modern NLPs. The major advantage of this API is the ability to recognise the present and absent symptoms, ensuring FR1 is achieved. Alternatives such as NLTK (2018) and Wit (2018) were not used in this project because they are unable to process medical terminology correctly and thus critically affect the accuracy of the healthcare web-based application. Additionally, Infermedica (2018) API is able to process input strings which are input incorrectly due to typing error or misspelling. This helps improve the UX of the application because users will not be forced entering their symptoms correctly. Lastly, the API also returns a triage suggestion which can be used to suggest action steps to the user. It may output *emergency*, *consultation* and *self_care* string which will aid in determining the advice the system gives to the user, helping them decide whether to book a GP appointment or take a medication.

In order to establish a communication relationship between the client and the server (including APIs), PHP will be used to capture the input from the form and pass the data to the processing servers. The primary disadvantage of such approach is the need to reload the page each time the form is submitted. To overcome such issue, AJAX (Asynchronous JavaScript and XML) is used in conjunction with plain JavaScript to capture the form input and pass it to the back-end. JavaScript is also used to output the response onto the page without reloading the page. Such approach is highly beneficial in the current system because local storage or cookies do not have to be used in order to store the form data while the page is being reloaded. It is believed that by keeping the entered data within the form at all times, the user will have to alter less fields each time and be constantly aware which data the system is processing. The database does not have to be populated with synthetic data because it will not store any user data. Instead, the database will be used to store symptom treatment suggestions for each diagnose within the selected medical domain. This data will be extracted from the NHS Health A – Z page (NHS, 2018). In order to query the database, PDO (PHP Data Objects) will be used which is a much secure alternative to MySQLi and MySQL due to the use of parameterised queries. It will enable to execute queries which by default make it impossible to attack the system using SQL injections.

5.2.2 System Deployment

The application will be hosted on SiteGround (2018) because it provides a significant speed, cost and availability advantages compared to GoDaddy (2018) and 123 Reg (2018) competitors. FileZilla free, cross-platform software will be used to manage files on the server/host during and post deployment due to its simplistic layout and a range of useful features. Alternatively, files can be managed via the host's website using the provided GUI. Such approach is time consuming due to lengthy navigation and ineffective in terms of file transfers, since only one file can be transferred at a time. The database will be administrated through phpMyAdmin free software tool provided by the SiteGround (2018) host. Same task can also be completed using open-source MyWebSQL (2018) software, however, it was not selected due to lack of functions/features and the long setup process.

5.3 Back-End Development

This section will highlight the key implementation aspects when designing a proof of concept web-based application for the current project. The system uses a modularised approach through separation of concerns to improve the maintenance KQF. The code aims to follow all programming conventions which includes the use of appropriate variable/function name, meaningful comments in the code and appropriate code formatting/indentation. In order to obtain an accurate diagnosis and treatment suggestion, the user must first populate the form by supplying their gender, age and symptoms, shown in *Figure 27*. It is necessary to enter gender and age because the API can eliminate a number of gender-specific and age-related diagnosis, arriving to the correct diagnosis in the shortest amount of time. Such aspect has a direct benefit towards the overall UX of the application.

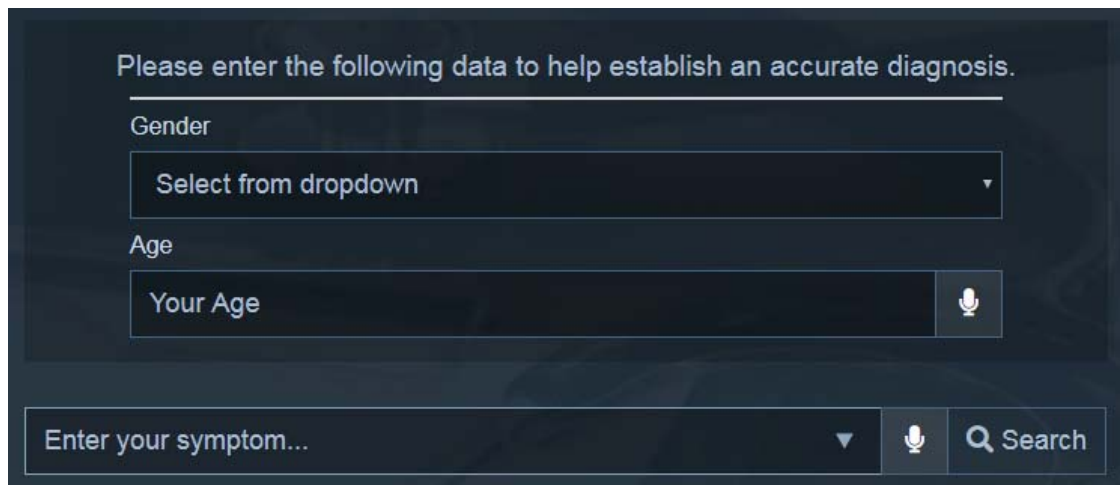


Figure 27 – Form Front-End

The input fields can be populated by typing the values using a keyboard or by pressing the microphone buttons to dictate the verbal input, demonstrated in *Figure 28*. It is presently not possible to verbally input a value into the gender drop-down input field due to the API's limitation.

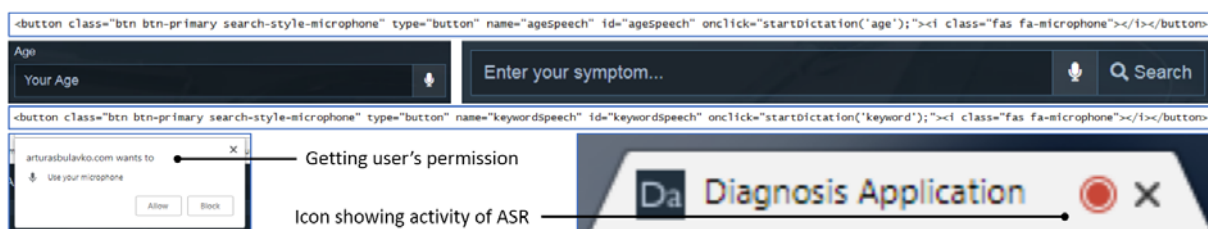


Figure 28 – Linking ASR Feature to the Buttons

When the microphone icon is clicked, the user is asked if the system can use their microphone. If the user blocks, they can only use keyboard to input data. If they allow, a red icon shown in *Figure 28* will appear in the tab to indicate that the system is listening to the user. The ASR is achieved using a function shown in *Figure 29*. The function accepts one parameter to indicate where the utterance will be transcribed into. The function is designed to listen to the user for as long they speak, suggesting that the speech recognition does not randomly end after 5 seconds. Once the user stops talking, the function processes the utterance via the API and returns a transcribed string. The string is then automatically inserted into the appropriate input field through the supplied parameter. Furthermore, given that the user correctly populated their age and gender, once they use ASR to input symptom, the function will automatically submit the form. It is believed that such feature positively affects the UX because less buttons must be pressed in order to interact with the application.

```

function startDictation(source)
{
    if (window.hasOwnProperty('webkitSpeechRecognition'))
    {
        var recognition = new webkitSpeechRecognition();

        recognition.continuous = false;
        recognition.interimResults = false;
        recognition.lang = "en-US";

        recognition.start();

        recognition.onresult = function(e)
        {
            document.getElementById(source).value = e.results[0][0].transcript;
            recognition.stop();
            if (source == 'keyword')
            {
                SubmitFormData();
            }
            else
            {}
        };

        recognition.onerror = function(e)
        {
            recognition.stop();
        }
    }
    else
    {
        window.alert('There was an error');
    }
}

```

Figure 29 – ASR Function

Once the user has correctly populated the three input fields within the form, they can click Search button within the web-based application in order to view their diagnosis, as shown in *Figure 30*.

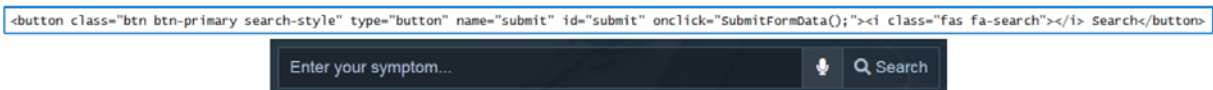


Figure 30 – Initiating the Diagnosis Process

The search button calls a function illustrated in *Figure 31*. The function is designed to capture all form input, perform additional validation and submit the form asynchronously using AJAX. The form is submitted using POST method to ensure the supplied data by the user does not appear in the URL bar. Alternatively, GET method could be used, however, it is not an appropriate solution in the current project because the data would be visible in the URL bar, introducing security risks.

```

function SubmitFormData()
{
    var gender = $("#gender").val();
    var age = $("#age").val();
    var keyword = $("#keyword").val();
    var errorMsg = 'Please populate the form fully to begin.';

    if ((gender=="" || gender==null) || (age=="" || age==null || age<18 || age>99) || (keyword=="" || keyword==null))
    {
        outputResponse(errorMsg)
    }
    else
    {
        $.post("../components/main.php", { gender: gender, age: age, keyword: keyword },
            function(data)
            {
                outputResponse(data)
            });
    }
}

```

Figure 31 – Capturing Form Data and Passing Values

Having submitted the form, the data is passed to the *main* PHP file which stores all the necessary variables demonstrated in partition 1 of *Figure 32* and calls functions stored in the external files, shown in partitions 2 and 3 of *Figure 32*. Fundamentally, the file controls the flow of the entire web-based application. In order to process the supplied symptom string, the code begins by performing NLP manipulation to capture and standardise the medical terms. Partition 2 of *Figure 32* demonstrates

that an API request is made to perform NLP. The API request initiates a function demonstrated in partition 4 of *Figure 32*. The function accepts two parameters, ensuring the code can be reused, allowing making multiple requests without having to rewrite function each time. The function begins by initialising cURL session with the API by passing the relevant data. It then passes a JSON string to the API for processing and captures a JSON response. Once the response is returned, JSON is passed back and decoded using *json_decode()* PHP inbuilt function, allowing to extract the required data. Having completed NLP and obtained the correct medical terms, another API request shown in partition 3 of *Figure 32* is made to obtain the diagnosis. The call follows the same procedure as before. Once a diagnosis is returned, JSON is decoded and the diagnosis string is stored in a variable.

<pre>\$gender = \$_POST['gender']; \$age = \$_POST['age']; \$keyword = \$_POST['keyword'];</pre>	<pre>function SendRequest(\$url, \$data) { \$ch = curl_init(); curl_setopt(\$ch, CURLOPT_URL, "https://api.infermedica.com/v2/\$url"); curl_setopt(\$ch, CURLOPT_RETURNTRANSFER, 1); curl_setopt(\$ch, CURLOPT_POSTFIELDS, \$data); curl_setopt(\$ch, CURLOPT_POST, 1); \$headers = array(); \$headers[] = "App-Id: "; \$headers[] = "App-Key: "; \$headers[] = "Content-Type: application/json"; curl_setopt(\$ch, CURLOPT_HTTPHEADER, \$headers); \$result = curl_exec(\$ch); if (curl_errno(\$ch)) { echo 'Error:' . curl_error(\$ch); } curl_close(\$ch); return \$result; }</pre>
<pre>\$result = SendRequest('parse', \$formattedkeyword); \$decodedsymptoms = json_decode(\$result);</pre>	
<pre>\$resultdiagnosis = SendRequest('diagnosis', \$datastring); \$decodeddiagnosis = json_decode(\$resultdiagnosis);</pre>	

Figure 32 – Sending Requests to the API

The Infermedica API does not provide diagnosis treatment recommendations and there are presently no APIs which would generate an appropriate treatment suggestion for a diagnosis. To overcome this, the developer has populated a database with several treatment options for a number of diagnosis within the selected domain, allowing to retrieve a relevant treatment suggestion when required. Within the *main* PHP file, a call to a function which fetches treatment suggestions from the database is made, demonstrated in *Figure 33*. The function accepts diagnosis parameter in order to correctly search for the treatment advice within the database. A simple parameterised query is run on the database in order to locate treatment suggestions for the required diagnosis. Once the treatment is found, it is returned back to the *main* PHP file for further processing.

```
foreach($treatment = obtainDiagnosisTreatment($diagnoseselected) as $nameTreatment)
{
    $treatmentstring = $nameTreatment->treatmentName;
}

function obtainDiagnosisTreatment($diagnosisName)
{
    global $con;
    $sql = $con->prepare("SELECT treatmentName FROM Treatment, Diagnosis WHERE Treatment.diagnosisID=Diagnosis.diagnosisID AND diagnosisName = ? ");
    $sql->execute([$diagnosisName]);
    $existence = $sql->fetchAll(PDO::FETCH_CLASS, 'treatment');
    if(count($existence) == 0)
    {
        return 0;
    }
    else{
        return $existence;
    }
}
```

Figure 33 – Obtaining Treatment Advice from the Database

In other to make the novel system more intelligent, triage is added which fundamentally decides how severe the user's diagnosis is. As shown in *Figure 34*, a previously discussed API request is made to capture the triage level of the diagnosis. The triage level is then passed to another function which turns triage level keyword into a human-readable sentence, demonstrated in partition 2 of *Figure 34*. A human readable sentence is then returned by the function, informing the user whether they need to call an ambulance, optionally visit a doctor or take no action. It is believed that such step will help users decide on next steps, preventing them from calling doctors as soon as they obtain a diagnosis.

```

$triage = $decodedTriage->triage_level;
$triageRecommendation = PerformTriage($triage);
1
function PerformTriage($passedTriage)
{
    if ($passedTriage == "emergency")
    {
        $returnString = "The reported symptoms may indicate a serious or a life-threatening condition. Please call 999 or 112 and seek immediate medical attention. Alternatively, please visit your nearest A&E department urgently.";
    }
    else if ($passedTriage == "consultation")
    {
        $returnString = "Your condition is not severe; however, you require a further medical consultation. Please book an appointment with your GP.";
    }
    else if ($passedTriage == "self_care")
    {
        $returnString = "A medical consultation is not strictly required. You must observe your symptoms and consult a doctor if symptoms worsen within 24 hours.";
    }
    else
    {
        $returnString = "error";
    }
    return $returnString;
}
2

```

Figure 34 – Obtaining and Processing Triage Level

Having obtained the diagnosis, treatment suggestion and the triage, a final output string is assembled using concatenation feature in PHP, demonstrated in *Figure 35*.

```

$diagnosisString = "Your diagnosis is ".$diagnoseSelected.". ".$treatmentString." ".$triageRecommendation;
echo $diagnosisString;

```

Figure 35 – Assembling the Diagnosis String

The final string is then passed to the output function shown in partition 1 of *Figure 36*. The function is responsible for calling TTS component in order to output the string using voice, demonstrated in partition 2 of *Figure 36*. Similarly, the function also dynamically inserts the string into the *paragraph* element of the page using *id*, illustrated in partition 3 of *Figure 36*. Such tactic positively affects the UX because the user can both, hear and read their diagnosis within the web-based application.

```

function outputResponse(response)
{
    generateVoiceOutput(response);
    document.getElementById("generatedDiagnosis").innerHTML = response;
    document.getElementById("generatedDiagnosis").classList.add('generatedDiagnosis');
}
1

function generateVoiceOutput(computedDiagnosis)
{
    if(responsivevoice.voicesSupport())
    {
        responsivevoice.speak(computedDiagnosis);
    }
}
2

<p id="generatedDiagnosis" name="generatedDiagnosis"></p>
3

```

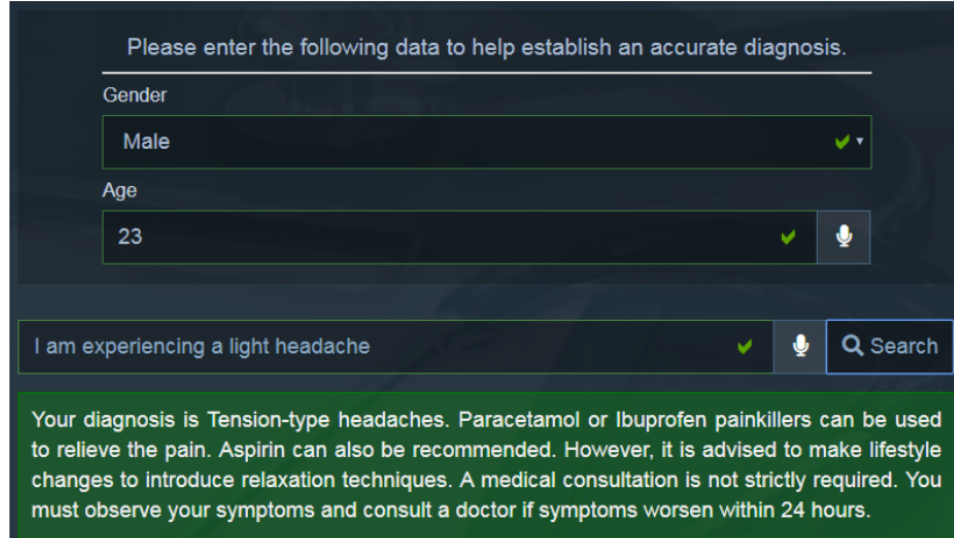


Figure 36 – Presenting Diagnosis to the User using Text and Voice

5.4 Front-End Development

As discovered during the literature review, front-end is one of the most critical system components, ensuring the system is usable by the target audience. This section will focus on discussing a number of GUI aspects which were considered during the implementation phase. *Table 4* below illustrates all system pages which were implemented throughout the project to deliver the web-based application.

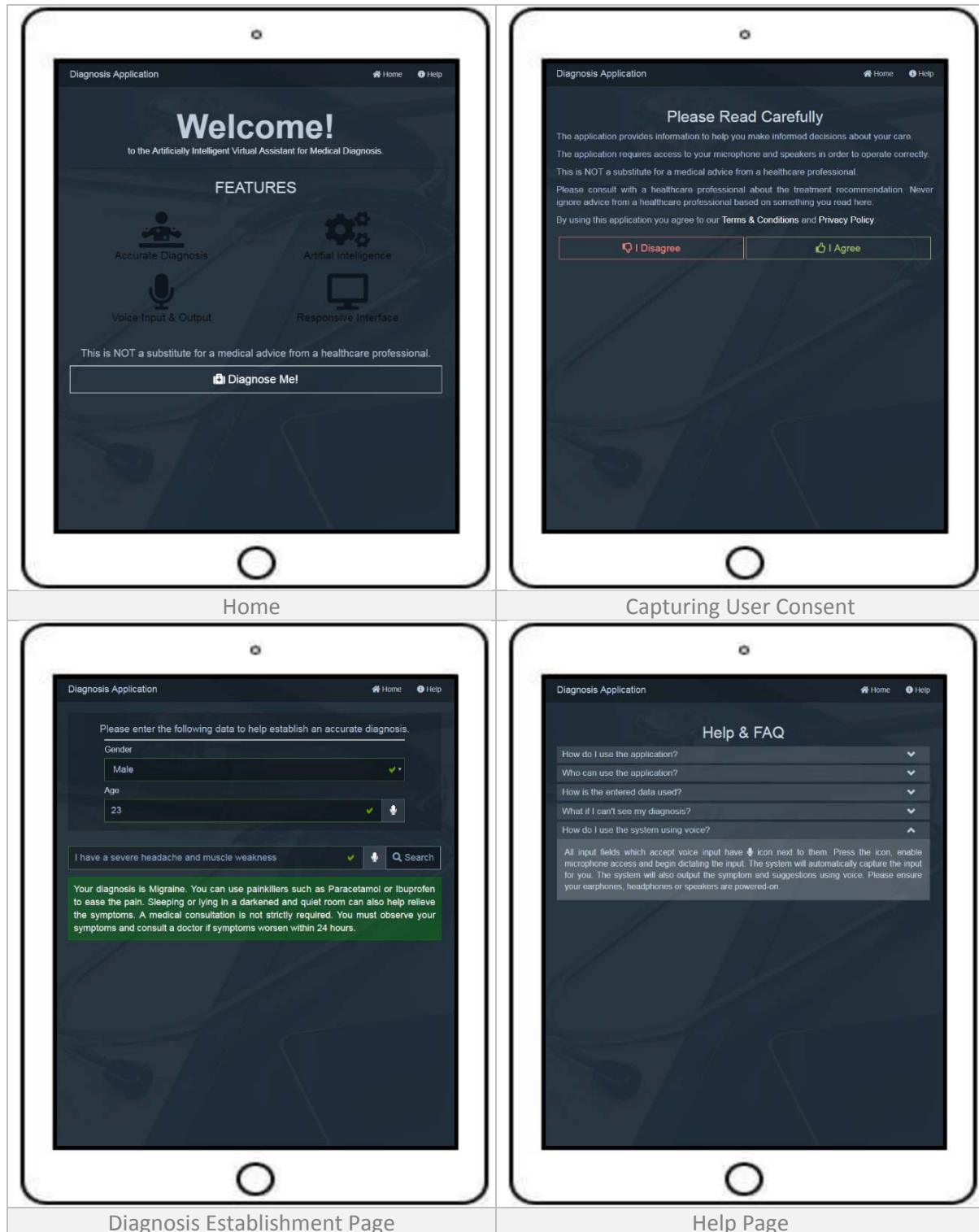


Table 4 – System Screenshots

When designing the front-end, all literature review findings and recommendations were considered. The application benefited from the responsive framework to ensure the content automatically scales to all screen sizes. Blue colour-scheme was used to influence the emotional and psychological state of the user, ensuring they are comfortable using the application. The system does not limit users to verbal input only, but also provides input fields which can be used to enter the data if the user does not wish to speak. Such facility added a complication of incorrect data input which was resolved by performing validation checks on all input fields, as demonstrated in *Figure 37*. Validation rules were added to all input fields to ensure only the allowed input can be inserted by the user.

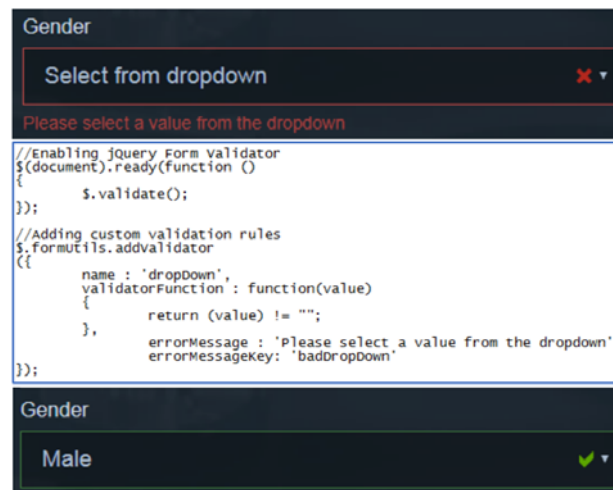


Figure 37 – Input Validation Example

Unlike reviewed existing solutions, current web-based application does not restrict users to selecting a predefined symptom. The users are free to type any symptoms they have. Instead, the application shows several input suggestions which the user can input into the system, illustrated in *Figure 38*. Such approach positively affects the UX because it indirectly shows how to use the system by an example.

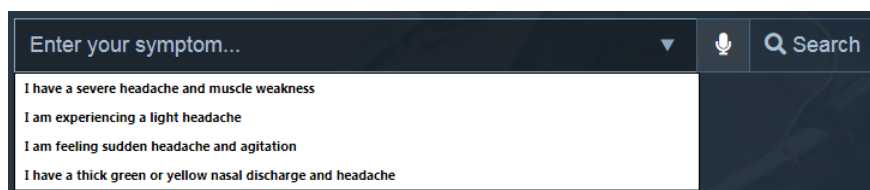


Figure 38 – Symptom Suggestion Generation Example

Additionally, if the user supplies an invalid symptom, the application will politely ask the user to try again and provide an example query to help them and thus enhance the UX, shown in *Figure 39*.

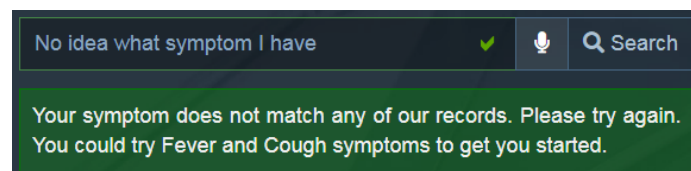


Figure 39 – Example Symptom shown on Unsuccessful Try

The implemented application also avoided simplifying the diagnosis as discussed in the literature review. Such measure ensures that all users receive an accurate diagnosis which they can forward to their doctors for an additional review. Lastly, the design phase prototype recommendation of adding icons to all buttons was implemented in the current application to make the navigation more intuitive.

5.5 Icon Development

Considering that the system created throughout this project is treated as a web-based application, two icons were created, shown in *Table 5*. The application icon will represent the application when the users add a shortcut onto their home screen or add a bookmark on Android or Apple devices. The favicon will represent the system within browser tabs or when the users bookmark the application. The process of the icon development can be seen in *Appendix D*.

	
Application Icon	Favicon

Table 5 – Application Icons

5.6 Summary

The implementation chapter has successfully provided insights into the application's development process. It begun by reviewing and selecting tools and technologies which were then applied in the current project in order to deliver a web-based application conforming to majority of the requirements established during the analysis phase. However, due to restricted project time, FR3 and FR8 were not implemented as they had *Could have* and *Won't have* MoSCoW priority. The chapter concluded by developing two application icons which will be used to represent the system. Overall, the implemented system used novel and futureproof tools/technology which forms a solid starting ground for the m-health application of the similar type.

It would be beneficial developing personal components from scratch instead of reusing APIs which are lacking certain functionality. However, it was not possible to develop components specifically for the current application due to the limited time. The thesis will now proceed onto SQA chapter where the product and process will be thoroughly tested and evaluated against the criteria.

6 Software Quality Assurance

6.1 Introduction

The SQA (Software Quality Assurance) chapter will apply numerous SQA techniques to validate and verify the artifacts developed throughout the project. Software quality can be examined from two distinct perspectives, first being the conformance to functional requirements and standards, and the second being the unquantifiable structure quality which reflects the non-functional requirements. As a result of this, several theories and techniques were established, enabling to define, quantify and measure the quality in relation to KQF characteristics. Numerous KQF and requirements for this project were defined during the analysis phase which will now be used to test and assess the quality of the developed product as well as the process applied to create it.

6.2 Software Verification

Software verification focuses on ensuring that the artifacts conform to the regulations, standards and the specifications. It is a manual process completed by a qualified person to discover and correct defects as soon as possible, ensuring the project costs remain low throughout the project because reworks towards the end of the project are less likely to occur. The fundamental benefit of verification allows to evaluate an unfinished artifact because the assessor can examine the structure of the document/code to locate potential defects (Manns and Coleman, 1996, p.126). Verification includes techniques such as reviews, inspections and audits to ensure best practices have been followed which affect the overall product quality. Review is typically completed to ensure the artifact fulfils the quality standards and specifications, inspection is used to identify defects and anomalies, and audit verifies the artifact in terms of readiness for the next phase. This project followed a software development lifecycle which consisted of four phases. As planned, each phase ended with an SQA milestone to ensure a correct degree of quality is achieved. *Table 6* below outlines the type of verification procedures which were completed in each phase of the project.

PHASE	TYPE	LOCATION
Analysis	Managerial Review, Phase-end Audit	Appendix E
Design	Managerial Review, Phase-end Audit	Appendix F
Implementation	Expert Review, Phase-end Audit, System Testing Readiness Inspection	Appendix G
Testing	Managerial Review, Phase-end Audit, Delivery Readiness Inspection	Appendix H

Table 6 – Verification Process Techniques

All SQA activities were completed in accordance with the project schedule found in *Appendix A*. Due to the word limit, please refer to the corresponding Appendix in order to locate the verification activities for each phase. Such verification strategy has enabled to increase productivity since errors were corrected early in the project. Similarly, KQF such as maintainability and scalability were indirectly improved because conventions and coding guidelines were enforced.

The analysis and design verification activities did not uncover any product defects, verifying that the documentation has been completed to a high-quality standard. They have also proven that both phases created the right amount of artifacts which are essential for the project. The implementation phase SQA activities have confirmed that FR3 and FR8 requirements were not implemented due to the time limit. Since these requirements had low priority, it was deemed acceptable to pass the review. As a result of this, requirements FR3 and FR8 will not be tested during the software validation procedure. Lastly, the testing phase verification techniques have confirmed the product to match the initial requirement specification, constituting the achievement of the right degree of quality. Verification has also demonstrated that all the necessary artifacts are ready for the delivery.

Ideally, the verification procedure should be completed by a different person to ensure they cannot make assumptions based on previous system experience. However, this project consists of one individual, making it challenging completing this sort of tasks. Nevertheless, completing SQA activities in this project was beneficial because they introduced more transparency into the product since all the artifacts were checked to ensure they exist and match the right degree of quality.

6.3 Software Validation

Software validation aims to test the artifacts against the specifications established at the start of the software development lifecycle. The technique can be divided into validation of software designs, validation of product specifications and validation of software product (Chemuturi, 2011, pp.132-135). Each process helps organisations prove that the right degree of quality was achieved in the developed product. This project will utilise validation of the software product technique to evaluate the product's correctness by applying several testing techniques. Such approach will introduce traceability into the project, aid in defect discovery and enable to form accurate test-cases which will be used to validate the product (Fournier, 2008, pp.102-103). Testing primary focuses on ensuring that the chosen input correctly matches the known output, making it a fundamental activity of any project (Chemuturi, 2011, p.134). A summary of the testing techniques used in this project is presented in *Table 7* below.

TEST TYPE	Functional Testing	TEST LOCATION	Appendix I
STRATEGY SUMMARY			
Functional testing is a type of black-box testing which ensures core and ancillary functions are working correctly. This testing will compare inputs with the outputs without examining the internal logic to ensure the product has achieved all user goals correctly.			
TEST TYPE	Non-Functional Testing	TEST LOCATION	Appendix J
STRATEGY SUMMARY			
Non-functional testing validates that all non-functional requirements have been achieved and thus the system conforms to a certain degree of quality. Such testing develops a number of black-box test cases which must pass in order to consider the system as complete.			
TEST TYPE	Usability and Field Testing	TEST LOCATION	Appendix K
STRATEGY SUMMARY			
Usability testing focuses on checking how easy the system is to use with the real users. Such testing attempts to examine different types of usage scenarios to ensure requirements have been fulfilled correctly. Field testing is used to validate that the application operates correctly under real working conditions (2003, pp.234-241). It is commonly performed in similar situations that the users would use the application in, to guarantee that it can operate correctly under common usage patterns. In order to save time, the two types of testing were combined to test the system with the real users.			
TEST TYPE	Evaluation Against the Existing Systems	TEST LOCATION	Appendix L
STRATEGY SUMMARY			
This test will evaluate the novel systems against the existing solutions in terms of features and accuracy. Such testing aims to demonstrate that the developed proof of concept is a suitable solution to the earlier established problems.			

Table 7 - Project Testing

Due to the word limit, please refer to the corresponding appendix in order to locate the test cases and the test results. Fundamentally, 81 test cases were written against which the product was tested, including 41 functional tests and 40 non-functional tests. Functional testing included techniques such as equivalence partitioning and decision table to validate all scenarios. Usability and field testing were also completed as part of software validation process to ensure the system can be used by the real people in varying situations. By evaluating the novel system against the existing solutions, the developer has proven that it can correctly solve the discovered problems which similar systems are

unable to address presently. Such testing has shown that the system accurately offers the intended functionality, which the existing applications fail to achieve. The web-based application was able to process all of the supplied symptoms in order to output an accurate diagnosis, treatment suggestion and triage level whereas the existing systems were unable to recognise several symptoms. Each type of the test has validated that all critical system features are working as intended and there are no serious defects within the application. Test cases SQANFT26 and SQANFT28 have revealed that the ASR feature does not work on the Apple devices. Essentially, the users are unable to input data using voice on any iOS device due to operating system restriction of the devices. It is believed that such defect is not critical to the current release of the product because the users can input the data using keyboards. In order to eliminate this defect, an Apple ASR component must be used in order to implement the voice input feature into the web-based application.

Usability and field testing were combined in order to save valuable project time. These tests have focused on validating the system's operation under the real working environment to check whether it can be used outside of the testing environment. Passing functional testing does not guarantee that the system can be used by the users, which is why usability testing was an important task in this project. Similarly, usability testing has indirectly tested several non-functional requirements, including: NFR10, NFR11, NFR12, NFR13, NFR15, NFR16 and NFR17 to validate their correctness. It is always more valuable capturing feedback from the real users as oppose to running a test. The only minor imperfection noticed during the usability and field testing was the inability for the ASR component to recognise the utterance due to the loud background noise. This is currently one of the project limitations because there are no known methods to overcome this.

Unit testing was not completed in this project because the main diagnosis AI component was reused, proving impossible to unit test a service. Additionally, NFR5, NFR6, NFR7 could not be tested because they require the application to be running and frequently used by the real users in order to monitor and analyse the required metrics. Vinayagasundaram and Srivatsa (2007) also propose to use metrics such as LOC (Number of Lines), cyclomatic complexity and vocabulary frequency in AI systems to ensure a correct degree of quality is achieved. Furthermore, sanity, load and stress testing would be beneficial to complete in the current project to examine how the system operates in specific conditions. However, due to limited time, such testing and metrics were not included in SQA.

6.4 Summary

The SQA chapter has verified the product and the process by applying a range of techniques to assess the quality during the different project stages. Similarly, it has validated the product by applying numerous testing methods to assess the system completeness and the overall quality. This chapter has proven that all the required artifacts match the quality standards and thus are ready for the delivery. As previously discussed, other testing methods and metrics could be used to evaluate the product further, however, due to strict time limit of this project it was not done. The thesis will continue by evaluating the project and concluding the findings.

7 Conclusion

7.1 Project Evaluation

The primary achievement of this project was the implementation of the planned web-based application within the provided time, considering that the chosen field was completely new to the developer. The established project goal was achieved and verified using the aims and objectives formed during the introduction chapter. The end-product was also validated using the extensive testing approach, proving the system does not have any critical defects and all planned features are operating as required. Since the key area of the project was AI virtual assistants, a detailed research was completed to study how similar systems are built. The significant issue encountered during the literature review was the lack of up-to-date research in the AI field. Majority of the research was old and focused on individual components, whereas very limited papers have discussed how AI virtual assistants are developed and assembled, proving the chosen thesis area-of-study innovative. This limitation was mitigated by studying the existing solutions in order to examine the appropriate architecture and processing logic. This has allowed to decompose existing systems into a number of components and sub-components, enabling to review them individually. Such approach has revealed all the details, allowing to make a list of the detailed recommendations for the project.

Majority of the discovered recommendations were adapted in the current project and reflected in each chapter. This has enabled to develop a dependable system within the agreed time and mitigate all serious problems. The analysis phase has begun forming the system requirements by consulting the existing systems in order to accurately capture the primary user goals. Such approach was proven effective because a list of SMART functional and non-functional requirements was created. Although, in future similar projects it would be advantageous capturing the feedback from the target users to ensure all major user goals were captured and documented correctly. This would add an additional degree of traceability into the project. The design phase modelled the system by means of UML diagrams, wireframes and prototype to reflect the earlier findings and the recommendations. This has enabled to plan the system and confirm that all of the requirements are acknowledged. While adding a degree of traceability into the project, it would be valuable capturing user feedback on the designs to ensure the end-product meets their expectations.

The implementation phase was the most challenging aspect of the project because the developer had no previous experience in this field. They started the phase by creating each component themselves, which was extremely time consuming. Various libraries were used to implement each component in order to achieve the right effect, however, none of them supported the medical terminology correctly. The ASR component was unable to transcribe most of the input due to inability to eliminate background noise, NLP could not correctly select the symptoms from the supplied string and ML lacked knowledge on which to base the decisions. Having previously selected SOA, the developer has decided to reuse existing components by integrating them into the current system. While it was easy integrating ASR and TTS into the current web-based application, finding the right AI API for symptom processing was challenging because not many exist. As discussed during the implementation chapter, only one suitable API was located, although it also had a number of drawback. They include: inability to process all symptoms (such as stroke or heart attack) and requiring initial two or three symptoms for the diagnosis to be accurate. During the project, it was discovered that the ASR component is unable to facilitate voice input for a drop-down element, which is a minor limitation. Similarly, TTS does not output the sentences correctly in certain situations. It may pause in the middle of the sentence and continue with a completely different dialect. One of the major TTS disadvantages is the pronunciation of punctuation marks under certain conditions. It tends to output the words *dot* or *comma* if the previous character was a bracket. This limitation cannot be presently resolved.

Reusing existing components where possible has significantly improved the pace of the project, considering how much time was spent initially trying to develop them from scratch. By saving the valuable project time, vendor dependency was introduced into the product because the developer has no control over the reused APIs. This was the only option to finish the project in time and implement a proof-of-concept. It is worth noting that the developer has improved their knowledge around APIs, in particular the establishment of cURL sessions and manipulation of JSON data. During the implementation chapter it was discovered that there are presently no components which would generate a treatment suggestion based on the diagnosis. To eliminate this problem, the developer has implemented a database with several diagnosis and corresponding treatments for them, based on the suggestions provided by the NHS (2018) website. Such approach has enabled to create a component which would accept a diagnosis and would return an appropriate treatment suggestion. In order to improve the component, it is vital to introduce ML with the aim of medical knowledge acquisition.

The front-end implementation was relatively simple due to previous experience. Several literature recommendations were reflected in the product to impact the UX. The system adopted blue colour scheme to calm and reassure the user via psychology which was complimented using intuitive menu elements with appropriate icons to resemble the action. One of the discoveries made during the front-end development was that the use of image backgrounds is not correctly supported by Android when adopting a responsive framework. This issue was resolved by writing custom CSS rules in order to overwrite the background formatting elements. Additionally, the collapsible menu elements on the *help.html* page provided by Bootstrap (2018) framework were negatively affecting the UX. The menu could only be expanded and collapsed by clicking the text, instead of being able to click anywhere on the box to achieve the same result. By rewriting the CSS rules for this element, the developer was able to achieve the desired effect which made the entire menu element clickable. Furthermore, arrow icons were added to show the state of the collapsible menu, clearly showing what is open and closed. Application icons and favicon were developed to represent the system on various platforms.

Verification SQA activities were completed at the end of each phase in accordance with the plan found in *Appendix A*. Reviews were primary used to check the artifacts against the quality standards, inspections were used to discover any defects and audits were used to verify the phase artifacts, ensuring the project is ready for the next phase. These activities were completed by the same person which is a major limitation because ideally, they must be done by the person who had no prior interaction with the artifacts they are checking. The use of verification activities has enforced better practices to be applied in the project which resulted in an improved quality. It is highly recommended that similar approach is applied in all projects. Validation SQA activities were focused on testing the product against the functional and non-functional requirements as well as the existing systems. It is a fundamental step of any SDL because it checks whether the system is ready for the deployment. Testing has discovered that that ASR component does not operate correctly on Apple products. This is a major limitation of the current project as it could not be easily resolved. This is because Apple has restricted all voice input components to be activated on their devices if they were not developed by Apple. Such problem can be potentially overcome by using their component, however, since this issue is not critical in the current project and the time is limited, it was not addressed. While this project has adopted five testing techniques, it can be argued that it is not enough for a system of this type. Stress and unit tests could be completed in the current project to validate the system further, however, due to time limit it was not possible to perform them.

Furthermore, as one of the testing limitations, it was not possible to thoroughly assess the system in terms of reliability and efficiency because the system must be heavily used first. This would enable to monitor and capture the system statistics when it is frequently used by the real users. Simulating the

usage patterns are not an effective approach because the data would be inaccurate, making the results irrelevant. The completion of usability and field testing was one of the beneficial tasks in this project because it not only assessed the functional requirements, but also validated the non-functional requirements indirectly. It has allowed to test the system with the real users in two different environments, confirming that all the functionality is implemented as expected and the system has the right degree of quality. By combining field and usability testing together, the project manager was able to save valuable project time because only two meetings were required, instead of four. It would be advantageous testing the system with the real doctors or perhaps integrating it into the NHS 111 service to capture the feedback from a significant number of users. Most importantly, testing has proven that the new system can accurately establish a diagnosis for the symptoms supplied by the user where existing systems fail. This is the biggest achievement in the current project because it demonstrates that the initial project goal was fulfilled correctly and the system conforms to the EU GDPR (2018) and M-health Applications (European Commission, 2017) Privacy Code of Conduct.

The selected Agile methodology has complimented the project by enforcing time restrictions, preventing the developer from spending vast amount of time on a single subject. This has helped complete the project on time. A primary example is the inability to implement requirements FR3 and FR8. These requirements were not implemented because by adapting MoSCoW prioritisation, they were ranked as *won't have* and *could have* respectively. This has enabled to postpone these requirements for the next revision of the system. By neglecting two requirements, the developer was able to correctly implement all other requirements, ensuring all critical system features are present. In future, it is advised to follow Agile methodology because unlike Waterfall and PRINCE2 it introduces more flexibility into the product, allowing making changes at all times.

This project was a remarkable learning opportunity for the developer, allowing them to expand their knowledge around the AI field. They had experience developing personal AI components which was not a correct approach to the current project and thus the developer gained valuable knowledge around component/service reuse. The application can be seen as a trivial everyday system, however, the amount of tools and technologies required to implement all of the intended functionality is immense. It was challenging finding the right tools, but it was even harder integrating them into the application, ensuring everything works correctly. The system consists of four front-end pages which are supported by an elegant back-end, supporting most programming conventions and standards. By following the recommendations and relying on personal expertise, a fully dependable and easily maintainable web-based application was developed within the specified time, being the greatest achievement. The novel web-based application was also evaluated against the existing solutions to compare how well it addresses the established problems. This has shown that unlike the existing systems, the new system can correctly solve the discovered NHS 111 problems. The discoveries made in this thesis will have a direct impact on the m-health field.

7.2 Ethics Discussion

7.2.1 Legal Aspects

The web-based application adheres to the newly established EU GDPR (2018) legislation. This means that the users are provided with a clear, accurate and concise Terms & Conditions as well as Privacy Policy at all times when using the application. On the first use of the application, the user is asked for a formal consent before they are able to interact with the system. This is further achieved by ensuring that all points stated in the Privacy Code of Conduct on M-health Applications (European Commission, 2017) are correctly implemented into the system. Additionally, all relevant data subject rights defined in the EU GDPR (2018) are made available to the users.

7.2.2 Ethical Aspects

The application abides by the medical ethics to ensure it is suitable for public use. Medical ethics is built upon four fundamental and non-hierarchical values, namely: autonomy, non-maleficence, beneficence and justice (Boylan, 2013, p.13). This ensures that all application users receive valuable information in an ethically-correct manner. Furthermore, the users are made aware that they are not interacting with a real person from an ethical point of view.

7.2.3 Social Aspects

There are no social issues because users are unable to socialise via the web-based application. The application only allows interaction between the user and the system. In order to combat the digital divide, the system has basic system and network requirements, allowing it to be used on various devices. Additionally, the intuitive UI allows all people with basic IT literacy skills usage of the provided services. The simple interaction nature of the application allows people with various disabilities to benefit from the application developed throughout this project.

7.2.4 Security Aspects

Apart from conforming to all legal security aspects discussed earlier, the application establishes a secure communication protocol between all system components. This ensures that all transferred data cannot be understood by the intruders in case of an attack. Since the user data will not be collected and stored by the system, it is not needed to introduce further security mechanisms.

7.2.5 Safety Aspects

While the system is safety-critical, it does not provide advice on critical health conditions and suggests the user to visit a qualified doctor. It therefore does not introduce any life-threatening risks to the user. The system informs the user that the provided advice is not a substitute to a real doctor.

7.3 Future Work

Due to various complications, it was not possible to explore and accomplish all the planned details throughout the project. Subsequently, the following recommendations in the order of difficulty are made for the future work within the field.

1. Develop personal AI components within the project in order to avoid reusing existing solutions because they are lacking in accuracy and functionality.
2. Create an AI component to make diagnosis treatment suggestions.
3. Introduce new medical domains into the system, allowing to process more symptoms.
4. Add the ability to upload and process the graphical content, enabling the system to diagnose various skin conditions and injuries.
5. Introduce new application languages, allowing the application to be usable abroad.
6. Convert the application into a chatbot with voice, enabling users to track their conversation with the application and requesting a transcript if required.
7. Implement the ability for the application to ask relevant questions and provide sample answers, increasing the diagnosis accuracy and improving the UX.
8. Introduce new component to enable ASR feature on Apple devices.
9. Add a map into the application, showing the nearest medical establishments.
10. Create speech/voice input for a drop-down element.

The recommendations 1-4 were not implemented in this project because they require a substantial time investment. It would be impossible to complete the project on time if these recommendations were considered. The suggestions 5-10 are relatively easy to accomplish by reusing external APIs and libraries, however, due to time constraints they were not achieved.

References

- 123 Reg (2018) *Web Hosting*. Available at: <https://www.123-reg.co.uk/web-hosting/>. (Accessed: 14 May 2018).
- Al-Shayea, Q. (2011) 'Artificial neural Networks in medical Diagnosis', *International Journal of Computer Science Issues (IJCSI)*, 8(2), pp.150-154.
- Amato, F., Lopez, A., Pena-Mendez, E., Vanhara, P., Hampl, A., Haval, J. (2013) 'Artificial Neural Networks in Medical Diagnosis', *Journal of Applied Biomedicine*, 11(2), pp.47-58. DOI: 10.2478/v10136-012-0031-x.
- Amery, R. (2016) 'If I had taken the advice of the NHS 111 helpline I would be blind', *The Guardian*. Available at: <https://www.theguardian.com/healthcare-network/2016/jan/28/advice-nhs-111-helpline-blind-wrong>. (Accessed: 14 May 2018).
- ApiMedic (2018) *Symptom Checker API | Integrate medical diagnosis into your app today*. Available at: <https://apimedic.com/>. (Accessed: 25 July 2018).
- Apple (2018) *iOS – Siri – Apple (UK)*. Available at: <https://www.apple.com/uk/ios/siri/>. (Accessed: 15 May 2018).
- Armstrong, S. (2018) 'The Apps Attempting to Transfer NHS 111 Online', *BMJ*. Available at: <https://www.bmj.com/content/360/bmj.k156>. (Accessed: 14 May 2018).
- Atom (2018) *Atom*. Available at: <https://atom.io>. (Accessed: 22 July 2018).
- Axure (2018) *Prototypes, Specifications, and Diagrams in One Tool | Axure Software*. Available at: <https://www.axure.com>. (Accessed: 23 July 2018).
- Babylon Health (2018) *NHS Online: 111*. (Version 2.0.1) [Mobile app]. Available at: iTunes App Store & Google Play (Downloaded: 14 May 2018).
- Bishop, C. (1995) *Neural networks for Pattern Recognition*. Oxford: Clarendon Press.
- Bootstrap (2018) *Bootstrap Documentation*. Available at: <http://getbootstrap.com/docs/4.1/getting-started/introduction/>. (Accessed: 23 July 2018).
- Boylan, M. (2013) *Medical Ethics*. 2nd edn. Chichester, West Sussex, U.K.: Wiley-Blackwell.
- Buoy Health (2018) *Buoy Health: Check Symptoms & Find the Right Care*. Available at: <https://www.buoyhealth.com/>. (Accessed: 25 June 2018).
- Bushnell, M. (2018) 'AI Faceoff: Siri vs. Cortana vs. Google Assistant vs. Alexa', *Business News Daily*. Available at: <https://www.businessnewsdaily.com/10315-siri-cortana-google-assistant-amazon-alexa-face-off.html>. (Accessed: 27 June 2018).
- Callan, R. (2003) *Artificial Intelligence*. Basingstoke: Palgrave Macmillan.
- Cawsey, A. (1998) *The Essence of Artificial Intelligence*. Harlow: Prentice Hall.
- Chemuturi, M. (2011) *Mastering Software Quality Assurance: Best Practices, Tools and Techniques for Software Developers*. Fort Lauderdale, Fla.: J. Ross Pub.
- Cohen, K., Demner-Fushman, D. (2014) *Biomedical Natural Language Processing*. Amsterdam: J. Benjamins Publishing Company.

ConvertICO (2018) *ConvertICO.com – Convert PNG to ICO*. Available at: <https://convertico.com>. (Accessed: 28 August 2018).

Durairaj, M., Ranjani, V. (2013) 'Data Mining Applications in Healthcare Sector: A Study', *International Journal of Scientific & Technology Research*, 2(10), pp.29-35.

Els, D. (2015) *Responsive Design High Performance*. United Kingdom: Packt Publishing.

EU GDPR (2018) *EU GDPR Information Portal*. Available at: <https://www.eugdpr.org/>. (Accessed: 16 May 2018).

European Commission (2017) *Privacy Code of Conduct on mobile health apps*. Available at: <https://ec.europa.eu/digital-single-market/en/privacy-code-conduct-mobile-health-apps>. (Accessed: 16 May 2018).

Font Awesome (2018) *Font Awesome*. Available at: <https://fontawesome.com>. (Accessed: 24 July 2018).

Fournier, G. (2008) *Essential Software Testing: A Use Case Approach*. Boca Raton, Fla.: Auerbach; London: Taylor & Francis distributor.

Galin, D. (2004) *Software Quality Assurance: From Theory to Implementation*. Harlow: Pearson Addison Wesley.

GoDaddy (2018) *Hosting | See our Web Hosting Solution – GoDaddy UK*. Available at: <https://uk.godaddy.com/hosting?isc=gcouuk53>. (Accessed: 14 May 2018).

Google (2018) *Google Assistant – Your Own Personal Google*. Available at: https://assistant.google.com/intl/en_uk/. (Accessed: 15 May 2018).

Google Cloud (2018) *Cloud Speech-to-Text - Speech Recognition | Cloud Speech-to-Text API | Google Cloud*. Available at: <https://cloud.google.com/speech-to-text>. (Accessed: 28 July 2018).

Goss, F., Zhou, L., Weiner, S. (2016) 'Incidence of Speech Recognition Errors in the Emergency Department', *International Journal of Medical Informatics*, vol.93, pp.70-73. DOI: 10.1016/j.ijmedinf.2016.05.005.

Gruhn, R., Minker, W., Nakamura, S. (2011) *Statistical Pronunciation Modelling for Non-Native Speech Processing*. Springer-Verlag Berlin Heidelberg.

Hardcastle, W., Laver, J., Gibbon, F. (2010) *The Handbook of Phonetic Sciences*. 2nd edn. Chichester, UK; Malden, MA: Wiley-Blackwell.

Hempel, T. (2008) *Usability of Speech Dialog Systems*. Berlin: Springer.

Herff, C., Schultz, T. (2016) 'Automatic Speech Recognition from Neural Signals: A Focused Review', *Frontiers in Neuroscience*, vol.10, p.429. DOI: 10.3389/fnins.2016.00429.

Holmes, J., Holmes, W. (2001) *Speech Synthesis and Recognition*. 2nd edn. London: Taylor & Francis.

IconTester (2018) *Test icon for free on Android & iOS | IconTester.com*. Available at: <http://www.icontester.com>. (Accessed: 24 August 2018).

Infermedica (2018) *Artificial Intelligence for Medical Diagnosis – Infermedica*. Available at: <https://infermedica.com/>. (Accessed: 25 June 2018).

Infermedica-Developer (2018) *Medical Content – Infermedica for Developers*. Available at: <https://developer.infermedica.com/docs/content>. (Accessed: 28 July 2018).

ITV News (2016) 'Five deaths in five months after 'problems' with NHS 111 helpline', *ITV News*. Available at: <http://www.itv.com/news/2016-04-23/five-deaths-in-five-months-after-problems-with-nhs-111-helpline/>. (Accessed: 14 May 2018).

Jackson, P., Moulinier, I. (2007) *Natural Language Processing for Online Applications: Text Retrieval, Extraction and Categorisation*. Amsterdam, NLD John Benjamins Publishing Company.

Jacobs, P. (1992) *Text-based Intelligent Systems: Current Research and Practice in Information Extraction and Retrieval*. Hillsdale, N.J.: L. Erlbaum Associates.

Jeegeek (2018) 'How Does Siri Work? The Science Behind Siri', *Magoosh Data Science Blog*. Available at: <https://magoosh.com/data-science/siri-work-science-behind-siri/>. (Accessed 29 June 2018).

Johnson, J. (2010) *Designing with the Mind in Mind: Simple Guide to Understanding User Interface Design Rules*. Burlington, Massachusetts: Morgan Kaufmann Publishers.

jQuery Form Validator (2018) *jQuery Form Validator – Home*. Available at: <http://www.formvalidator.net/index.html#complete-test-form>. (Accessed: 26 July 2018).

jQuery Validation Plugin (2018) *jQuery Validation Plugin | Form validation with jQuery*. Available at: <https://jqueryvalidation.org>. (Accessed: 28 July 2018).

Kapko, M. (2018) 'Cortana Explained: How to Use Microsoft's Virtual Assistant For Business', *Computer World*. Available at: <https://www.computerworld.com/article/3252218/collaboration/cortana-explained-why-microsofts-virtual-assistant-is-wired-for-business.html>. (Accessed: 30 July 2018).

Kocaleva, M., Stojanovik, I., Zdarev, Z. (2016) 'Pattern Recognition and Natural Language Processing: State of the Art', *TEM Journal*, 5(2), pp.236-240. DOI: 10.18421/TEM52-18.

Kosner, A. (2012) 'Client vs. Server Architecture: Why Google Voice Search Is Also Much Faster Than Siri', *Forbes*. Available at: <https://www.forbes.com/sites/anthonykosner/2012/10/31/client-vs-server-architecture-why-google-voice-search-is-also-much-faster-than-siri/#53b5d4472958>. (Accessed: 30 July 2018).

Larson, J. (2003) *VoiceXML: Introduction to Developing Speech Application*. Upper Saddle River, N.J.: Prentice Hall.

Manns, T., Coleman, M. (1996) *Software Quality Assurance*. 2nd edn. Basingstoke: Macmillan.

Matyunina, J. (2017) 'AI in Mobile Apps: How to Make an App Like Siri', *Codetiburion*. Available at: <https://codetiburion.com/ai-mobile-apps-make-app-like-siri/>. (Accessed: 29 June 2018).

McAuliffe, M., Wilding, P., Rickard, N., O'Beirne, G. (2012) 'Effect of Speaker Age on Speech Recognition and Perceived Listening Effort in Older Adults with Hearing Loss', *Journal of Speech, Language, and Hearing Research*, 55(3), pp.838-847. DOI: 10.1044/1092-4388(2011/11-0101).

MedWhat (2018) *MedWhat | Your virtual medical assistant*. Available at: <https://medwhat.com/>. (Accessed: 25 June 2018).

Microsoft (2018) *Cortana | Your Intelligent Virtual & Personal Assistant | Microsoft*. Available at: <https://www.microsoft.com/en-gb/windows/cortana>. (Accessed: 15 May 2018).

Miller, J. (2015) 'Self-Diagnosis on Internet not Always Good Practice', *The Harvard Gazette*. Available at: <https://news.harvard.edu/gazette/story/2015/07/self-diagnosis-on-internet-not-good-practice/>. (Accessed: 26 June 2018).

Miyoshi, T., Azuma, M. (1993) 'An empirical study of evaluating software development environment quality', *IEEE Transactions on Software Engineering*, 19(5), pp.425-435. DOI: 10.1109/32.232010.

Mockingbird (2018) *Website Wireframes: Mockingbird*. Available at: <https://www.gomockingbird.com/home>. (Accessed: 23 July 2018).

Mogups (2018) *Online Mockup, Wireframe, & UI Prototyping Tool | Mogups*. Available at: <https://moqups.com>. (Accessed: 23 July 2018).

Morton, H., Gunson, N., Marshall, D., McInnes, F., Ayres, A., Jack, M. (2010) 'Usability Assessment of Text-to-Speech Synthesis for Additional Detail in Automated Telephone Banking System', *Computer Speech & Language*, 25(2), pp.341-362. DOI: 10.1016/j.csl.2010.05.008.

MyWebSQL (2018) *MySQL, SQLite, PostgreSQL web client – MyWebSQL*. Available at: <http://mywebsql.net>. (Accessed: 2 August 2018).

Nebeling, M., Norrie, M. (2013) 'Responsive design and development: Methods, technologies and current issues', *Interface-Driven Web Engineering: Responsive Web Design*, 7977, pp.510-513. DOI: 10.1007/978-3-642-39200-9_47.

Negnevitsky, M. (2011) *Artificial Intelligence: A Guide to Intelligent Systems*. 3rd edn. United Kingdom: Pearson Education M.U.A.

NetBeans (2018) *Welcome to NetBeans*. Available at: <https://netbeans.org>. (Accessed: 26 July 2018).

NHS (2016) *The NHS in England – NHS Structure*. Available at: <https://www.nhs.uk/NHSEngland/thenhs/about/Pages/nhsstructure.aspx>. (Accessed: 14 May 2018).

NHS (2018) *Health A-Z – NHS*. Available at: <https://www.nhs.uk/conditions>. (Accessed: 27 July 2018).

NHS 111 (2017) *Urgent and Emergency Care Services in England*. Available at: <https://www.nhs.uk/NHSEngland/AboutNHSServices/Emergencyandurgentcareservices/Pages/NHS-111.aspx>. (Accessed: 14 May 2018).

NHS 111 Online (2018) *Get Medical Help Near You*. Available at: <https://111.nhs.uk/>. (Accessed: 14 May 2018).

NLTK (2018) *Natural language Toolkit – NLTK 3.3 documentation*. Available at: <https://www.nltk.org>. (Accessed: 28 July 2018).

O'Connor, Z. (2011) 'Colour psychology and colour therapy: Caveat emptor', *Color Research and Application*, 36(3), pp.229-234. DOI: 10.1002/col.20597.

Odessky, A., Tamler, D. (2018) 'Sensely's "Ask NHS" App Shows Evidence of Channel Shift and Patient Engagement', *PR Newswire*. Available at: <https://www.prnewswire.com/news-releases/senselys-ask->

nhs-app-shows-evidence-of-channel-shift-and-patient-engagement-300631219.html. (Accessed: 25 June 2018).

Pingdom (2018) *Website Speed Test*. Available at: <https://tools.pingdom.com>. (Accessed: 27 August 2018).

Poder, T., Fisette, J., Dery, V. (2018) 'Speech Recognition for Medical Dictation: Overview in Quebec and Systematic Review', *Journal of Medical Systems*, 42(5), pp.1-8. DOI: 10.1007/s10916-018-0947-0.

Pure (2018) *Pure.css*. Available at: <https://purecss.io>. (Accessed: 23 July 2018).

Quinton, M. (2017) 'Suicide Botch NHS 111 helpline workers leave suicidal patients on hold and take naps at their desks while their bosses play cards', *The Sun*. Available at: <https://www.thesun.co.uk/news/3238459/sun-investigation-reveals-scandal-of-nhs-111-helpline-where-workers-are-told-to-lie-leave-suicide-risk-patients-on-hold-and-fall-asleep-at-desks-while-managers-play-cards/>. (Accessed: 14 May 2018).

Rajput, N., Nanavanti, A. (2012) *Speech in Mobile and Pervasive Environments*. Chichester, West Sussex, UK: Wiley.

ResponsiveVoice.JS (2018) *ResponsiveVoice.JS – Home*. Available at: <https://responsivevoice.org>. (Accessed: 24 July 2018).

Sennaar, K. (2018) 'Chatbots for Healthcare – Comparing 5 Current Applications', *TechEmergence*. Available at: <https://www.techemergence.com/chatbots-for-healthcare-comparison/>. (Accessed: 25 June 2018).

Sensely (2018) *Sensely – How are you feeling today?*. Available at: <http://www.sensely.com/>. (Accessed: 25 June 2018).

Sense.ly Corporation (2018) *Ask NHS – Virtual Assistant*. (Version 3.0.2) [Mobile app]. Available at: iTunes Store & Google Play (Downloaded: 25 June 2018).

SiteGround (2018) *Web Hosting Crafted for Top Website Performance & Speed*. Available at: <https://www.siteground.com/web-hosting.htm>. (Accessed: 14 May 2018).

Skeleton (2018) *A dead simple, responsive boilerplate*. Available at: <http://getskeleton.com/#intro>. (Accessed: 23 July 2018).

Software Quality Management (2016) Available at: http://sce2.umkc.edu/BIT/burris/pl/software_quality_management. (Accessed: 29 July 2018).

Sommerville, I. (2010) *Software Engineering*. 9th edn. Harlow: Addison-Wesley.

StarUML (2018) *Star UML 2*. Available at: <http://staruml.io/>. (Accessed: 28 June 2018).

Tabini, M. (2013) 'Vox Technica: How Siri Gets its Voice', *Macworld*. Available at: <https://www.macworld.com/article/2056718/vox-technica-how-siri-gets-its-voice.html>. (Accessed: 27 July 2018).

Tidwell, J. (2011) *Designing Interfaces*. Farnham: O'Reilly.

Vinayagasundaram, B., Srivatsa, S. (2007) 'Software Quality in Artificial Intelligence Systems', *Information Technology Journal*, 6(6), pp.835-842. DOI: 10.3923/itj.2007.835.842.

Virtanen, T., Singh, R., Raj, B. (2012) *Techniques for Noise Robustness in Automatic Speech Recognition*. Chichester, West Sussex, UK; Wiley.

Vogel, O., Arnold, I., Chughtai, A., Kehrer, T. (2011) *Software Architecture*. Berlin, Heidelberg: Springer Berlin Heidelberg.

Wagner, P. (2018) 'Siri Remains the Most Used Mobile Voice Assistant', *Statista*. Available at: <https://www.statista.com/chart/14505/market-share-of-voice-assistants-in-the-us>. (Accessed: 27 June 2018).

Walker, P. (2013) 'What is NHS 111 and what problems has it faced?', *The Guardian*. Available at: <https://www.theguardian.com/society/2013/jul/29/nhs-direct-111-problems>. (Accessed: 14 May 2018).

Web Speech API (2018) *Web Speech API – Web API | MDN*. Available at: https://developer.mozilla.org/en-US/docs/Web/API/Web_Speech_API. (Accessed: 26 July 2018).

Wisniewski, J. (2013) 'Responsive Design. (control-shift) (web design)', *Online Searcher*, 37(1), pp.74(3).

Wit (2018) *Wit.ai*. Available at: <https://wit.ai>. (Accessed: 28 July 2018).

Woodward, C. (2011) 'Data-Mining Case Tests Boundaries of Medical Privacy', *CMAJ: Canadian Medical Association Journal*, 183(9), pp.509-10. DOI: 10.1504/cmaj.109-3853.

You, C., Ma, B. (2017) 'Spectral-Domain Speech Enhancement for Speech Recognition', *Speech Communication*, vol.94, pp.30-41. DOI: 10.1016/j.specom.2017.08.007.

Your.MD (2018) *Your.MD – Health Guide and Symptom Checker*. Available at: <https://www.your.md/>. (Accessed: 25 June 2018).

Your.MD AS (2017) *Your.MD – Health Guide*. (Version 2.8.4) [Mobile app]. Available at: iTunes App Store & Google Play (Downloaded: 25 June 2018).

Zubair, P., Bhat, H., Lone, T. (2017) 'Cortana – Intelligent Personal Digital Assistant: A Review', *International Journal of Advanced Research in Computer Science*, 8(7), pp.55-57.

Zurb Foundation (2018) *Foundation for Sites*. Available at: <http://foundation.zurb.com/sites/docs>. (Accessed: 14 May 2018).

Appendix A – Project Schedule

In order to follow the Agile methodology techniques and keep the project on schedule throughout the software development lifecycle, a Gantt Chart is created to illustrate the essential flow of the project.

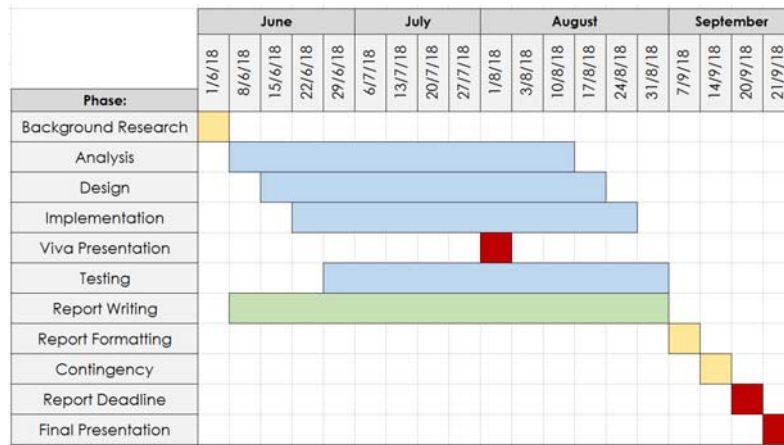


Figure 40 – Project Schedule Gantt Chart

The *Figure 40* above illustrates all major phases and deadlines of the project. As seen from the Gantt chart, the project will consist of four essential phases which will run simultaneously to reflect the Agile methodology. Additionally, the report will be written in concurrence with the product development to ensure both are completed on the stated deadline. It was decided to partition the project into weekly blocks which will be used to complete specific project deliverables.

DATE	DELIVERABLES
08/06/18	Complete Literature Review
15/06/18	SWOT and Stakeholder analysis, Use Case diagram & description
22/06/18	Prioritised Functional Requirements, Non-Functional Requirements
29/06/18	Perform SQA activities and finish Analysis chapter
06/07/18	Entity-Relationship diagram, Data Dictionary
13/07/18	Activity, Sequence, Class and Component diagram
20/07/18	Complete Wireframes and Prototypes
27/07/18	Perform SQA activities and finish Design chapter
03/08/18	Gather/Install necessary software and implement Database
10/08/18	Implement Back-end
17/08/18	Implement Back-end
24/08/18	Implement Back-end
31/08/18	Implement Front-end
07/09/18	Perform SQA activities and finish Implementation chapter
14/09/18	Document and perform testing
21/09/18	Document and perform testing
28/09/18	Perform SQA activities and finish Testing chapter
05/10/18	Complete Introduction, Conclusion and References chapters
12/10/18	Project Contingency Time
19/10/18	Report Deadline
26/10/18	Final Presentation

Figure 41 – Project Deliverables

Figure 41 demonstrates all critical project deliverables which must be achieved each week. Each phase will end with a number of Software Quality Assurance (SQA) activities to verify and validate that all necessary artefacts were correctly developed. These SQA activities will act as the minor project milestones and will be used to judge the velocity of the project. The major milestones are also demonstrated in the deliverables plan to highlight all critical project deadlines.

Appendix B – Project Risk Management

Table A1 captures and rates the risks associated with the project. Additionally, the matrix proposes mitigation and contingency plans to help deal with the risks during the SDL.

RISK	LIKELIHOOD	IMPACT	RATING	MITIGATION PLAN	CONTINGENCY PLAN
1. Not finishing project in time	Low	High	Medium	<u>Avoid:</u> Follow the plan	Implement what is finished
2. Skill shortage	Low	Medium	Low	<u>Control:</u> Learn from guides	Re-organise tasks
3. Natural disaster	Low	High	Medium	<u>Control:</u> Back-up in different locations	Recover from back-up
4. Failure to follow methodology	Low	Low	Low	<u>Control:</u> Read books and get help from supervisor	Follow methodology guide
5. No internet access	Low	High	Medium	<u>Transfer:</u> Users must deal with this	Display error message in the application
6. Change of technology	Medium	Low	Low	<u>Accept</u>	Adapt the application to new technology
7. User frequency cannot be supported by the system	Medium	Medium	Low	<u>Avoid:</u> Optimise the database	Invest in new hardware
8. Change of legislation	Medium	Medium	Medium	<u>Accept</u>	Adapt to the new legislation
9. Providing incorrect suggestion	Low	High	Medium	<u>Control:</u> Introduce checking mechanisms	Inform user that they must consult with doctor
10. Users reject the prototype	Low	Medium	Low	<u>Monitor:</u> Users sign off at the end of each prototype testing	Re-assess requirements of the project
11. Corrupt files	Low	High	Medium	<u>Control:</u> Routine back-ups	Recover from back-up
12. Database failure	Low	High	Medium	<u>Transfer:</u> Let the hosting company back-up	Recover from back-up
13. User amount cannot be supported by the system	Medium	Medium	Low	<u>Avoid:</u> Optimise the system	Invest in new hardware
14. Establishing wrong diagnosis	Medium	High	Medium	<u>Monitor:</u> Ensure only reliable sources are checked	Advise users to check the diagnosis with the real doctor
15. Component vendors discontinue their products	Low	High	Medium	<u>Control:</u> Constantly check for updates	Find substitute components
16. Rising hosting costs making it hard to maintain the app	Medium	High	Medium	<u>Accept</u>	Place verified medical ads to obtain revenue

Table 8 – Risk Analysis and Contingency Plan

Appendix C – Requirements Catalogue

Below is the requirements catalogue which describes each functional requirement in detail.

Source: User	Priority: MUST	Owner: Project Manager	ID: FR1
Functional Requirement	Enter Symptom		
Description	Users shall be able to add and edit their symptoms, age and gender.		
Benefits	Users will be able to obtain an accurate diagnosis by entering one or several symptoms which they are currently experiencing.		
Comments/Suggestions	User shall be able to enter data either using text or voice input. The system shall be able to correctly understand present and absent symptoms, for example; <i>I have a stomach pain, but no headache.</i>		
Related Documents	N/A		
Related Requirements ID	FR2, FR5, FR6, FR7		

Source: User	Priority: MUST	Owner: Project Manager	ID: FR2
Functional Requirement	Obtain Diagnosis		
Description	Users shall be able to obtain their diagnosis.		
Benefits	Users will be able to decide what to do next without having to wait in the queue to see or talk to a doctor.		
Comments/Suggestions	The diagnosis should be output using textual and voice technologies.		
Related Documents	N/A		
Related Requirements ID	FR1, FR6, FR7		

Source: User	Priority: COULD	Owner: Project Manager	ID: FR3
Functional Requirement	Generate Transcript		
Description	Users shall be able to request a conversation transcript after completion.		
Benefits	This will prevent users from having to remember their diagnosis.		
Comments/Suggestions	The transcript should be either sent to the provided email or a download link displayed to the user.		
Related Documents	N/A		
Related Requirements ID	FR1, FR2		

Source: User	Priority: SHOULD	Owner: Project Manager	ID: FR4
Functional Requirement	Receive System Help		
Description	Users shall be able to read system usage instructions and receive help if their action was completed incorrectly. For example, error messages.		
Benefits	Users will know how to use the system and be alerted if something is wrong.		
Comments/Suggestions	The text should be clearly visible on the page and the wording must be easily understood by users of all genders and ages.		
Related Documents	N/A		
Related Requirements ID	N/A		

Source: User -> System	Priority: SHOULD	Owner: Project Manager	ID: FR5
Functional Requirement	Generate Symptom Suggestions		
Description	The system shall show a number of possible symptoms which the users can enter if they are unsure how to begin or having trouble using the system.		
Benefits	Users will learn by example and thus become more comfortable using the novel system on their initial use.		
Comments/Suggestions	The suggestions should include popular and most frequent symptoms.		
Related Documents	N/A		
Related Requirements ID	FR1		

Source: User -> System	Priority: MUST	Owner: Project Manager	ID: FR6
Functional Requirement	Provide Verbal Interaction		
Description	Input fields shall provide verbal input and output facilities.		
Benefits	Users will be able to operate the application using voice.		
Comments/Suggestions	The component must correctly process the medical terminology.		
Related Documents	N/A		
Related Requirements ID	FR1, FR2		

Source: User -> System	Priority: MUST	Owner: Project Manager	ID: FR7
Functional Requirement	Provide Textual Interaction		
Description	All input fields shall provide textual input in case the verbal interaction component does not operate at a given time.		
Benefits	Users will be able to operate the application using keyboard.		
Comments/Suggestions	The input must be validated to check for incorrect input type and spelling mistakes. User should be shown success or error messages.		
Related Documents	N/A		
Related Requirements ID	FR1, FR2		

Source: User -> System	Priority: WON'T	Owner: Project Manager	ID: FR8
Functional Requirement	Provide Graphical Interaction		
Description	Users shall be able to upload images into the system for processing. For example; generating a diagnosis for a specific skin condition.		
Benefits	The system would be able to process more symptoms.		
Comments/Suggestions	Convenient uploading facilities.		
Related Documents	N/A		
Related Requirements ID	N/A		

Appendix D – Application Icons

The application icon was developed in Adobe Fireworks to achieve .png (Portable Network Graphics) image format. This format was selected because it introduces higher compression rate, allowing small files to download quicker. The bit depth of .png files also offers more colours as oppose to the alternatives. The favicon was initially designed in Adobe Fireworks, however, due to the requirement of the browsers, the file had to be converted into .ico (Icon) format using ConvertICO (2018) website. Considering the size of the favicon, the graphics were neglected and only two letters were used to represent the application. Making the favicon same as the application icon is meaningless because the favicon will be pixelated due to its small size and users would not be able to see anything.

Due to the varying device screen sizes, it was necessary to create several size application icons to ensure an appropriate one is used on each device. The following application icon sizes were created:

- ✓ 120x120px – Aimed at iPhones
- ✓ 180x180px – Aimed at iPads
- ✓ 128x128px – Aimed at Android smartphones
- ✓ 192x192px – Aimed at Android tablets

By supplied the system with a number of icons, the device will automatically select an appropriate size icon and use it throughout the device. The icons are attached via the *head* tag, as shown in Figure 42. The application icons were validated using IconTester (2018) website to ensure they are correctly displayed on the required devices. Favicon was validated by checking its presence within the browser.



Figure 42 – Preview of the Icon

The major limitation of developing icons for various devices is the lack of clear guidelines by the developers. While they provide size requirements and styling fundamentals, they do not specify how detailed the icon should ideally be. In order to overcome this issue, the developer has experimented with a range of different icons and selected the most appealing one.

Appendix E – Analysis Phase SQA

Below are the SQA Verification Activities which were completed during the Analysis phase.

Managerial Review

REVIEW REPORT		
NAME OF THE ARTIFACT	Analysis Phase Documentation	
REVIEWER NAME	Arturas Bulavko	
DATE OF THE REVIEW	22/06/2018	
TYPE OF REVIEW	Managerial Review	
UNCOVERED DEFECTS		
DEFECT ID	DEFECT DESCRIPTION	STATUS
No defects uncovered.		
CHECKLIST		
ITEM	ANSWER	RESULT
Are the requirements in compliance with the specification?	Yes	Pass
Have all requirements been listed?	Yes	Pass
Are there any ambiguous requirements?	No	Pass
Is each requirement described completely?	Yes	Pass
Can the requirements be verified/measured?	Yes	Pass
Has any functionality beyond the scope been included?	No	Pass
Is the rationale for any derived requirements satisfactory?	Yes	Pass
Do any requirements conflict or duplicate other requirements?	No	Pass
Are the maintainability requirements adequate and feasible?	Yes	Pass
Are the efficiency requirements adequate and feasible?	Yes	Pass
Are the reliability requirements adequate and feasible?	Yes	Pass
Are the usability requirements adequate and feasible?	Yes	Pass
Are the scalability requirements adequate and feasible?	Yes	Pass
Are the portability requirements adequate and feasible?	Yes	Pass
Have the security requirements been determined?	Yes	Pass
Are all security specifications properly specified?	Yes	Pass
Is each requirement written in clean, concise and unambiguous language?	Yes	Pass
Is each requirement free of content and grammatical errors?	Yes	Pass
Do the requirements provide an adequate basis for the software requirements specification?	Yes	Pass
Is each requirement in the scope of the project?	Yes	Pass
Have business rules been documented correctly?	Yes	Pass
Does use case diagram follow the correct notation?	Yes	Pass
Does use case diagram correctly present the system’s scope?	Yes	Pass
Were Functional Requirements prioritised correctly?	Yes	Pass
Were appropriate graphical illustrations used to present data?	Yes	Pass
Is completed literature review suitable for the subject?	Yes	Pass
Are all citations used correctly?	Yes	Pass
Do all citations have appropriate references?	Yes	Pass
Is reference table formatted correctly?	Yes	Pass
REVIEW RESULT		Pass

Phase-end Audit

AUDIT REPORT		
NAME OF THE ARTIFACT	Analysis Phase Documentation	
AUDITOR NAME	Arturas Bulavko	
DATE OF THE AUDIT	22/06/2018	
TYPE OF AUDIT	Phase-end Audit	
DISCOVERED NON-CONFORMANCES		
NC ID	NC DESCRIPTION	STATUS
None.		
OPPORTUNITIES FOR IMPROVEMENT		
DESCRIPTION	ARTIFACT	REFERENCE
None.		
CHECKLIST		
ITEM	ANSWER	RESULT
Are there any unresolved defects?	No	Pass
Is the Analysis chapter finished and approved?	Yes	Pass
Were all conventions and standards followed in the chapter?	Yes	Pass
Were all QA activities performed in Analysis phase?	Yes	Pass
Is the project on-time?	Yes	Pass
Are all relevant Analysis artifacts finished?	Yes	Pass
Did the Review Pass?	Yes	Pass
AUDIT RESULT		Pass

Appendix F – Design Phase SQA

Below are the SQA Verification Activities which were completed during the Design phase.

Managerial Review

REVIEW REPORT		
NAME OF THE ARTIFACT	Design Phase Documentation	
REVIEWER NAME	Arturas Bulavko	
DATE OF THE REVIEW	13/07/2018	
TYPE OF REVIEW	Managerial Review	
UNCOVERED DEFECTS		
DEFECT ID	DEFECT DESCRIPTION	STATUS
No defects uncovered.		
CHECKLIST		
ITEM	ANSWER	RESULT
Has the design been made flexible to meet future changes?	Yes	Pass
Were all requirements addressed?	Yes	Pass
Does the system architecture satisfy the requirements?	Yes	Pass
Can the program specification be easily coded?	Yes	Pass
Do all activity diagrams use correct notation?	Yes	Pass
Were conditions correctly shown in activity diagrams?	Yes	Pass
Are all paths correctly handled?	Yes	Pass
Were concurrent activities correctly documented?	Yes	Pass
Was the system architecture presented correctly?	Yes	Pass
Were wireframes created for all key screens?	Yes	Pass
Do wireframes reflect the requirements?	Yes	Pass
Do wireframes take into the account all standards and guidelines?	Yes	Pass
Are prototypes created for all key screens?	Yes	Pass
Do prototypes reflect the requirements?	Yes	Pass
Do prototypes take into the account all standards and guidelines?	Yes	Pass
Do the produced diagrams have any anti-patterns?	No	Pass
Were appropriate graphical illustrations used to present data?	Yes	Pass
Is completed literature review suitable for the subject?	Yes	Pass
Are all citations used correctly?	Yes	Pass
Do all citations have appropriate references?	Yes	Pass
Is reference table formatted correctly?	Yes	Pass
Does ERD use correct notation?	Yes	Pass
Were the relationships established correctly?	Yes	Pass
Were the multiplicities established correctly?	Yes	Pass
Does each table have a unique field (PK)?	Yes	Pass
Are all related tables linked correctly via FK?	Yes	Pass
Was an appropriate name given to all relationships?	Yes	Pass
Were all assumptions correctly documented?	Yes	Pass
Does data dictionary take into account all ERD tables?	Yes	Pass
Does data dictionary use correct data types?	Yes	Pass
Were field lengths correctly established?	Yes	Pass
Were all not null attributes correctly established?	Yes	Pass
REVIEW RESULT		Pass

Phase-end Audit

AUDIT REPORT			
NAME OF THE ARTIFACT	Design Phase Documentation		
AUDITOR NAME	Arturas Bulavko		
DATE OF THE AUDIT	13/07/2018		
TYPE OF AUDIT	Phase-end Audit		
DISCOVERED NON-CONFORMANCES			
NC ID	NC DESCRIPTION	STATUS	
None.			
OPPORTUNITIES FOR IMPROVEMENT			
DESCRIPTION	ARTIFACT	REFERENCE	
None.			
CHECKLIST			
ITEM	ANSWER	RESULT	
Are there any unresolved defects?	No	Pass	
Is the Design chapter finished and approved?	Yes	Pass	
Were all conventions and standards followed in the chapter?	Yes	Pass	
Were all QA activities performed in Design phase?	Yes	Pass	
Is the project on-time?	Yes	Pass	
Are all relevant Design artifacts finished?	Yes	Pass	
Did the Review Pass?	Yes	Pass	
AUDIT RESULT			Pass

Appendix G – Implementation Phase SQA

Below are the SQA Verification Activities which were completed during the Implementation phase.

Expert Review

REVIEW REPORT		
NAME OF THE ARTIFACT	Implementation Phase Documentation	
REVIEWER NAME	Arturas Bulavko	
DATE OF THE REVIEW	17/08/2018	
TYPE OF REVIEW	Expert Review	
UNCOVERED DEFECTS		
DEFECT ID	DEFECT DESCRIPTION	STATUS
1	NFR3 and NFR8 not implemented.	Resolved.
CHECKLIST		
ITEM	ANSWER	RESULT
Are all input fields correctly validated?	Yes	Pass
Are the error messages, warnings and information messages adequate?	Yes	Pass
Are the interfaces provided easy to use and consistent in format?	Yes	Pass
Were all functional requirements addressed?	No	Fail
Were all non-functional requirements addressed?	Yes	Pass
Were all programming standards and guidelines followed?	Yes	Pass
Is inline documentation adequate?	Yes	Pass
Has code been properly formatted?	Yes	Pass
Is there any redundant or trash code?	No	Pass
Are “if” statements correctly implemented?	Yes	Pass
Are there any unused variables declared?	No	Pass
Is there any security-related text left in comments?	No	Pass
Have all error conditions been trapped and handled?	Yes	Pass
Are the code files stored in the appropriate folders?	Yes	Pass
Do folders only contain relevant files?	Yes	Pass
Were all identified security features implemented?	Yes	Pass
Is all spelling within the system correct?	Yes	Pass
Is the text readable (font size/colour) on all intended devices?	Yes	Pass
Is the page layout consistent on all pages?	Yes	Pass
Do the icons correctly represent the action?	Yes	Pass
Is navigation consistent on all pages?	Yes	Pass
Is there a button to activate speech recognition?	Yes	Pass
Does speech recognition work correctly?	Yes	Pass
Does the system use TTS to output information?	Yes	Pass
Were appropriate graphical illustrations used to present data?	Yes	Pass
Was ERD correctly translated into database tables?	Yes	Pass
Does database follow standards defined in data dictionary?	Yes	Pass
Were relationships correctly established?	Yes	Pass
Was ON UPDATE selected as CASCADE on all relationships?	Yes	Pass
Was ON DELETE selected as CASCADE on all relationships?	Yes	Pass
Were PK and FK correctly indexed?	Yes	Pass
Is completed literature review suitable for the subject?	Yes	Pass

Are all citations used correctly?	Yes	Pass
Do all citations have appropriate references?	Yes	Pass
Is reference table formatted correctly?	Yes	Pass
REVIEW RESULT		Pass

Please note that due to limited time of this project it was not possible to implement all requirements. As a result, FR3 FR8 and FR9 were not implemented. However, they were prioritised as *Could have* and *Won't have* and thus do not directly affect the operation of the system. Such approach is perfectly acceptable when following an Agile software development methodology. The overall expert review is passed because the general implementation phase is completed correctly.

System Testing Readiness Inspection

INSPECTION REPORT		
NAME OF THE ARTIFACT	Implementation Phase Documentation	
INSPECTOR NAME	Arturas Bulavko	
DATE OF INSPECTION	17/08/2018	
TYPE OF INSPECTION	System Testing Readiness Inspection	
LIST OF INSPECTED COMPONENTS		
NAME OF COMPONENT	NATURE OF COMPONENT	TYPE OF INSPECTION DONE
Database Server	Hardware	Visual and power-on inspection
	Data	Checked for existence of data
	RDBMS	Checked for existence of the actual database and tables
Software Components	Web-based System	Verified all QA records and ensured all bugs/defects have been resolved
Web Server	Hardware	Visual and power-on inspection
	Web Server Software	Checked for existence of web-server as well as the website
Connection	Internet Connection	Navigated to the application
UNCOVERED DEFECTS		
DEFECT DESCRIPTION	CORRECTIVE ACTION	LOCATION OF DEFECT
No defects uncovered.		
INSPECTION RESULT		Pass

Phase-end Audit

AUDIT REPORT		
NAME OF THE ARTIFACT	Implementation Phase Documentation	
AUDITOR NAME	Arturas Bulavko	
DATE OF THE AUDIT	17/08/2018	
TYPE OF AUDIT	Phase-end Audit	
DISCOVERED NON-CONFORMANCES		
NC ID	NC DESCRIPTION	STATUS
None.		
OPPORTUNITIES FOR IMPROVEMENT		
DESCRIPTION	ARTIFACT	REFERENCE
None.		
CHECKLIST		
ITEM	ANSWER	RESULT
Are there any unresolved defects?	No	Pass
Is the Implementation chapter finished and approved?	Yes	Pass
Were all conventions and standards followed in the chapter?	Yes	Pass
Were all QA activities performed in Implementation phase?	Yes	Pass
Is the project on-time?	Yes	Pass
Are all relevant Implementation artifacts finished?	Yes	Pass
Did the Review Pass?	Yes	Pass
Did the Inspection Pass?	Yes	Pass
AUDIT RESULT		Pass

Appendix H – Testing Phase SQA

Below are the SQA Verification Activities which were completed during the Testing phase.

Managerial Review

REVIEW REPORT		
NAME OF THE ARTIFACT	Testing Phase Documentation	
REVIEWER NAME	Arturas Bulavko	
DATE OF THE REVIEW	31/08/2018	
TYPE OF REVIEW	Managerial Review	
UNCOVERED DEFECTS		
DEFECT ID	DEFECT DESCRIPTION	STATUS
No defects uncovered.		
CHECKLIST		
ITEM	ANSWER	RESULT
Does the plan reflect the requirements?	Yes	Pass
Have the acceptance criteria for acceptance test plans been specified?	Yes	Pass
Is the test strategy adequate and uncover all defects?	Yes	Pass
Does each test case specify the test condition, test procedure and expected results?	Yes	Pass
Have the results been recorded in adequate detail?	Yes	Pass
Have all requirements been tested?	Yes	Pass
Was functional testing performed?	Yes	Pass
Was non-functional testing performed?	Yes	Pass
Was usability testing performed?	Yes	Pass
Was field testing performed?	Yes	Pass
Is completed literature review suitable for the subject?	Yes	Pass
Are all citations used correctly?	Yes	Pass
Do all citations have appropriate references?	Yes	Pass
Is reference table formatted correctly?	Yes	Pass
REVIEW RESULT		Pass

Delivery Readiness Inspection

INSPECTION REPORT		
NAME OF THE ARTIFACT	Everything	
INSPECTOR NAME	Arturas Bulavko	
DATE OF INSPECTION	31/08/2018	
TYPE OF INSPECTION	Delivery Readiness Inspection	
LIST OF INSPECTED COMPONENTS		
NAME OF COMPONENT	NATURE OF COMPONENT	TYPE OF INSPECTION DONE
System	Web-based application	Checked all application
	Database	Checked for existence of database, tables and relevant data
	Code	Checked all project code files to ensure they are in the right folders and that the code is formatted correctly
Documentation	QA Activities	Verified that all planned activities were completed by checking documents
Defects	Defect Resolution	Checked all logs to ensure no defects are present there
UNCOVERED DEFECTS		
DEFECT DESCRIPTION	CORRECTIVE ACTION	LOCATION OF DEFECT
No defects uncovered.		
INSPECTION RESULT		Pass

Phase-end Audit

AUDIT REPORT			
NAME OF THE ARTIFACT	Testing Phase Documentation		
AUDITOR NAME	Arturas Bulavko		
DATE OF THE AUDIT	31/08/2018		
TYPE OF AUDIT	Phase-end Audit		
DISCOVERED NON-CONFORMANCES			
NC ID	NC DESCRIPTION	STATUS	
None.			
OPPORTUNITIES FOR IMPROVEMENT			
DESCRIPTION	ARTIFACT	REFERENCE	
None.			
CHECKLIST			
ITEM	ANSWER	RESULT	
Are there any unresolved defects?	No	Pass	
Is the Testing chapter finished and approved?	Yes	Pass	
Were all conventions and standards followed in the chapter?	Yes	Pass	
Were all QA activities performed in Testing phase?	Yes	Pass	
Is the project on-time?	Yes	Pass	
Are all relevant Testing artifacts finished?	Yes	Pass	
Did the Review Pass?	Yes	Pass	
Did the Inspection Pass?	Yes	Pass	
AUDIT RESULT			Pass

Appendix I – Functional Testing

Below are the test cases and test results concerned with Functional Testing.

ITEMS TO TEST		TEST DESCRIPTION		TEST DATE	RESPONSIBILITY	
FR1 – Enter Symptom		Users shall be able to add and edit their symptoms, age and gender.		24/08/2018	Arturas Bulavko	
TEST CASES						
ID	OBJECTIVE	INPUT VALUE	EXPECTED OUTCOME	ACTUAL OUTCOME	COMMENT	RESULT
SQAFT1	Check <i>gender</i> input field validation.	Select: Male	Accept	Accepted	N/A	PASS
SQAFT2	Check <i>gender</i> input field validation.	Select: Female	Accept	Accepted	N/A	PASS
SQAFT3	Check <i>gender</i> input field validation.	Do not select anything	Reject	Rejected	N/A	PASS
SQAFT4	Check <i>age</i> input field validation.	18	Accept	Accepted	N/A	PASS
SQAFT5	Check <i>age</i> input field validation.	99	Accept	Accepted	N/A	PASS
SQAFT6	Check <i>age</i> input field validation.	23	Accept	Accepted	N/A	PASS
SQAFT7	Check <i>age</i> input field validation.	17	Reject	Rejected	N/A	PASS
SQAFT8	Check <i>age</i> input field validation.	100	Reject	Rejected	N/A	PASS
SQAFT9	Check <i>age</i> input field validation.	Leave empty	Reject	Rejected	N/A	PASS
SQAFT10	Check <i>age</i> input field validation.	ostrich	Reject	Rejected	N/A	PASS
SQAFT11	Check <i>symptom</i> input field validation.	I have headache	Accept	Accepted	N/A	PASS
SQAFT12	Check <i>symptom</i> input field validation.	Leave empty	Reject	Rejected	N/A	PASS
SQAFT13	Check <i>symptom</i> input field validation.	a	Reject	Rejected	N/A	PASS
SQAFT14	Check <i>symptom</i> input field validation.	1	Reject	Rejected	N/A	PASS
SQAFT15	Check what happens if an incorrect symptom is input.	ostrich	Accept and inform user this symptom does not exist	As expected	Gave symptom suggestion to trv	PASS

ITEMS TO TEST		TEST DESCRIPTION		TEST DATE	RESPONSIBILITY	
FR2 – Obtain Diagnosis		Users shall be able to obtain their diagnosis.		24/08/2018	Arturas Bulavko	
TEST CASES						
ID	OBJECTIVE	INPUT VALUE	EXPECTED OUTCOME	ACTUAL OUTCOME	COMMENT	RESULT
SQAFT16	Check if entered symptoms result in correct diagnosis based on NHS (2018) website.	I have severe headache and muscle weakness	Migraine	Migraine	Correct treatment and triage given	PASS
SQAFT17	Check if entered symptoms result in correct diagnosis based on NHS (2018) website.	I am experiencing a light headache	Tension-type Headache	Tension-type Headache	Correct treatment and triage given	PASS
SQAFT18	Check if entered symptoms result in correct diagnosis based on NHS (2018) website.	I am feeling sudden headache and agitation	Cluster Headache (CH)	Cluster Headache (CH)	Correct treatment and triage given	PASS
SQAFT19	Check if entered symptoms result in correct diagnosis based on NHS (2018) website.	I have thick green or yellow nasal discharge and headache	Sinus Headache	Sinus Headache	Correct treatment and triage given	PASS
SQAFT20	Check if entered symptoms result in correct diagnosis based on NHS (2018) website. This also checks if absent symptoms are correctly processed.	I have high blood pressure but no headache	Hypertension	Hypertension	Correct treatment and triage given	PASS

SQAFT21	Check if entered symptoms result in correct diagnosis based on NHS (2018) website. This also check if the diagnosis is generated when symptoms are mistyped.	I am experiencing breath shortnes and frequent cogh	Asthma	Asthma	Correct treatment and triage given	PASS
---------	--	---	--------	--------	------------------------------------	------

ITEMS TO TEST		TEST DESCRIPTION		TEST DATE	RESPONSIBILITY	
FR4 – Receive System Help		Users shall be able to read system usage instructions and receive help if their action was completed incorrectly. For example, error messages.		24/08/2018	Arturas Bulavko	
TEST CASES						
ID	OBJECTIVE	INPUT VALUE	EXPECTED OUTCOME	ACTUAL OUTCOME	COMMENT	RESULT
SQAFT22	Check if the menu correctly expands when clicked.	Expand 1 st element	1 st element expands	As expected	Correct content appeared, and arrow changed appropriately	PASS
SQAFT23	Check if the menu correctly expands when clicked.	Expand 3 rd element	3 rd element expands	As expected	Correct content appeared, and arrow changed appropriately	PASS
SQAFT24	Check if menu correctly collapses when clicked.	Collapse 2 nd element.	2 nd element collapses	As expected	Correct content appeared, and arrow changed appropriately	PASS
SQAFT25	Check if first menu element correctly collapses when second is expanded.	Expand 3 rd and instantly Expand 2 nd	3 rd must collapse and 2 nd must expand	As expected	Correct content appeared, and arrow changed appropriately	PASS
SQAFT26	Input incorrect <i>age</i> and check if error message appears correctly.	150	Red error message must appear under the input field	As expected	Clear message instructing user what to do	PASS
SQAFT27	Do not select <i>gender</i> to check if error message appears correctly.	Do not select anything	Red error message must appear under the input field	As expected	Clear message instructing user what to do	PASS
SQAFT28	Input incorrect <i>symptom</i> to check if error message appears correctly.	ostrich	Error message saying such symptom is not supported	As expected	User was also shown an example symptom which they could try	PASS

ITEMS TO TEST		TEST DESCRIPTION		TEST DATE	RESPONSIBILITY	
FR5 – Generate Symptom Suggestions		The system shall show a number of possible symptoms which the users can enter if they are unsure how to begin or having trouble using the system.		24/08/2018	Arturas Bulavko	
TEST CASES						
ID	OBJECTIVE	INPUT VALUE	EXPECTED OUTCOME	ACTUAL OUTCOME	COMMENT	RESULT
SQAFT29	Check if the symptoms suggestions are shown.	Click drop-down in the symptoms input field	Show a list of suggested symptoms	As expected	N/A	PASS
SQAFT30	Check if the suggestions can be selected	Click 1 st symptom suggestion	Symptom input field must be automatically populated with the selected suggestion	As expected	N/A	PASS
SQAFT31	Check if suggestions are shown when a keyword is entered	Insert <i>headache</i> into input field and click suggestions	Suggestion list must show suggestions containing headache	As expected	N/A	PASS

ITEMS TO TEST		TEST DESCRIPTION		TEST DATE	RESPONSIBILITY	
FR6 – Provide Verbal Interaction		Input fields shall provide verbal input and output facilities.		24/08/2018	Arturas Bulavko	
TEST CASES						
ID	OBJECTIVE	INPUT VALUE	EXPECTED OUTCOME	ACTUAL OUTCOME	COMMENT	RESULT
SQAFT32	Check if clicking microphone button initiates the ASR.	Click microphone button in <i>age</i> field.	App should start recording the input	As expected	N/A	PASS
SQAFT33	Check if clicking microphone button initiates the ASR.	Click microphone button in <i>symptom</i> field.	App should start recording the input	As expected	N/A	PASS
SQAFT34	Check is ASR stops recording when user stops talking.	I have a headache	ASR should stock recording after headache is said	As expected	N/A	PASS
SQAFT35	Check if age ASR correctly populates the <i>age</i> input field.	Twenty-three	Age input field must contain 23	As expected	N/A	PASS
SQAFT36	Check if symptom ASR correctly populates the <i>symptom</i> input field.	I have a light headache	Symptom input field must contain I have a light headache	As expected	N/A	PASS
SQAFT37	Check if TTS is correctly initiated when diagnosis is returned.	Obtain diagnosis	App should verbally output the diagnosis as soon as it appears on the page	As expected	N/A	PASS
SQAFT38	Check if the speech correctly outputs the diagnosis	Obtain diagnosis	Written diagnosis must be identically verbally output	As expected	N/A	PASS

ITEMS TO TEST		TEST DESCRIPTION		TEST DATE	RESPONSIBILITY	
FR7 – Provide Textual Interaction		All input fields shall provide textual input in case the verbal interaction component does not operate at a given time.		24/08/2018	Arturas Bulavko	
TEST CASES						
ID	OBJECTIVE	INPUT VALUE	EXPECTED OUTCOME	ACTUAL OUTCOME	COMMENT	RESULT
SQAFT39	Check if <i>gender</i> input field can be selected using drop-down.	Select Male	Male selected	Male selected	N/A	PASS
SQAFT40	Check if <i>age</i> input field can be populated using keyboard.	23	Value should be typed using keyboard	As expected	N/A	PASS
SQAFT41	Check if <i>symptom</i> input field can be populated using keyboard.	I have severe headache	Value should be typed using keyboard	As expected	N/A	PASS

Appendix J – Non-Functional Testing

Below are the test cases and test results concerned with Non-Functional Testing.

ITEMS TO TEST		TEST DESCRIPTION		TEST DATE	RESPONSIBILITY	
NFR1		The application code shall be modular.		27/08/2018	Arturas Bulavko	
TEST CASES						
ID	OBJECTIVE	INPUT VALUE	EXPECTED OUTCOME	ACTUAL OUTCOME	COMMENT	RESULT
SQANFT1	Check if the code is modular.	Application Code	Stand-alone components and parameterised functions	As expected	N/A	PASS

ITEMS TO TEST		TEST DESCRIPTION		TEST DATE	RESPONSIBILITY	
NFR2		The application code shall be loosely coupled.		27/08/2018	Arturas Bulavko	
TEST CASES						
ID	OBJECTIVE	INPUT VALUE	EXPECTED OUTCOME	ACTUAL OUTCOME	COMMENT	RESULT
SQANFT2	Check if the code is loosely coupled.	Application Code	Parameterised functions which are independent	As expected	N/A	PASS

ITEMS TO TEST		TEST DESCRIPTION		TEST DATE	RESPONSIBILITY	
NFR3		The application code shall follow most programming conventions.		27/08/2018	Arturas Bulavko	
TEST CASES						
ID	OBJECTIVE	INPUT VALUE	EXPECTED OUTCOME	ACTUAL OUTCOME	COMMENT	RESULT
SQANFT3	Check if code has appropriate comments.	Application Code	Comments in each file of the code	As expected	N/A	PASS
SQANFT4	Check if code uses appropriate names for variables.	Application Code	All variable names must be sensible	As expected	N/A	PASS
SQANFT5	Check if code uses appropriate names for functions.	Application Code	All function names must be sensible	As expected	N/A	PASS
SQANFT6	Check if code is indented correctly.	Application Code	Code must be properly indented	As expected	N/A	PASS

ITEMS TO TEST		TEST DESCRIPTION		TEST DATE	RESPONSIBILITY	
NFR4		Latency shall be under 5 seconds across all internet connections.		27/08/2018	Arturas Bulavko	
TEST CASES						
ID	OBJECTIVE	INPUT VALUE	EXPECTED OUTCOME	ACTUAL OUTCOME	COMMENT	RESULT
SQANFT7	Test website latency using Pingdom (2018) using ethernet.	HTTPS request	< 5 seconds	1.2 seconds	N/A	PASS
SQANFT8	Test website latency using Pingdom (2018) using home WI-FI.	HTTPS request	< 5 seconds	1.5 seconds	N/A	PASS
SQANFT9	Test website latency using Pingdom (2018) using public WI-FI.	HTTPS request	< 5 seconds	2.3 seconds	N/A	PASS

ITEMS TO TEST		TEST DESCRIPTION			TEST DATE	RESPONSIBILITY
NFR8		The system shall be available 99.8% of the time during the year.			27/08/2018	Arturas Bulavko
TEST CASES						
ID	OBJECTIVE	INPUT VALUE	EXPECTED OUTCOME	ACTUAL OUTCOME	COMMENT	RESULT
SQANFT10	Test availability of the application.	N/A	99.8%	99.9%	Since the application is hosted on SiteGround (2018), the availability of this service provider was consulted.	PASS

ITEMS TO TEST		TEST DESCRIPTION			TEST DATE	RESPONSIBILITY
NFR9		The system shall be restored within 2 hours after a crash.			27/08/2018	Arturas Bulavko
TEST CASES						
ID	OBJECTIVE	INPUT VALUE	EXPECTED OUTCOME	ACTUAL OUTCOME	COMMENT	RESULT
SQANFT11	Test the recoverability of the system.	N/A	< 2 hours	< 30 minutes	Since the application is hosted on SiteGround (2018), the availability of this service provider was consulted.	PASS

ITEMS TO TEST		TEST DESCRIPTION		TEST DATE	RESPONSIBILITY	
NFR10		The system shall use consistent layout on all pages.		27/08/2018	Arturas Bulavko	
TEST CASES						
ID	OBJECTIVE	INPUT VALUE	EXPECTED OUTCOME	ACTUAL OUTCOME	COMMENT	RESULT
SQANFT12	Check layout on <i>index.html</i>	index.html	Navigation bar, background, font-size, font-colour, element position must be identical on all pages	As expected	N/A	PASS
SQANFT13	Check layout on <i>consent.html</i>	consent.html	Navigation bar, background, font-size, font-colour, element position must be identical on all pages	As expected	N/A	PASS
SQANFT14	Check layout on <i>diagnosis.html</i>	diagnosis.html	Navigation bar, background, font-size, font-colour, element position must be identical on all pages	As expected	N/A	PASS
SQANFT15	Check layout on <i>help.html</i>	help.html	Navigation bar, background, font-size, font-colour, element position must be identical on all pages	As expected	N/A	PASS

ITEMS TO TEST		TEST DESCRIPTION		TEST DATE	RESPONSIBILITY	
NFR11		The system shall provide guidance to the user completing a specific task.		27/08/2018	Arturas Bulavko	
TEST CASES						
ID	OBJECTIVE	INPUT VALUE	EXPECTED OUTCOME	ACTUAL OUTCOME	COMMENT	RESULT
SQANFT16	Check if <i>age</i> error message is correctly shown to the user.	Leave <i>age</i> empty	Display error message	As expected	N/A	PASS
SQANFT17	Check if <i>gender</i> error message is correctly shown to the user.	Leave <i>gender</i> empty	Display error message	As expected	N/A	PASS
SQANFT18	Check if symptom suggestion is shown if	ostrich	Display error and provide symptom suggestion	As expected	N/A	PASS

	initial symptom was incorrect.					
SQANFT19	Check if error message is displayed when the entered symptom is not supported by the system.	I have passion fruit	Display error message	As expected	N/A	PASS
SQANFT20	Check if form can be submitted with empty values in input fields.	Leave input fields empty	Highlight input fields which are empty	As expected	N/A	PASS

ITEMS TO TEST		TEST DESCRIPTION		TEST DATE	RESPONSIBILITY	
NFR12		The system shall provide meaningful error messages to assist the user.		27/08/2018	Arturas Bulavko	
TEST CASES						
ID	OBJECTIVE	INPUT VALUE	EXPECTED OUTCOME	ACTUAL OUTCOME	COMMENT	RESULT
SQANFT21	Check if <i>age</i> error message is correctly shown to the user.	Leave <i>age</i> empty	Display error message	As expected	N/A	PASS
SQANFT22	Check if <i>gender</i> error message is correctly shown to the user.	Leave <i>gender</i> empty	Display error message	As expected	N/A	PASS
SQANFT23	Check if symptom suggestion is shown if initial symptom was incorrect.	ostrich	Display error and provide symptom suggestion	As expected	N/A	PASS
SQANFT24	Check if error message is displayed when the entered symptom is not supported by the system.	I have passion fruit	Display error message	As expected	N/A	PASS

ITEMS TO TEST		TEST DESCRIPTION		TEST DATE	RESPONSIBILITY	
NFR14		New system functionality shall be added within current architecture/code.		27/08/2018	Arturas Bulavko	
TEST CASES						
ID	OBJECTIVE	INPUT VALUE	EXPECTED OUTCOME	ACTUAL OUTCOME	COMMENT	RESULT
SQANFT25	Check whether new components are easily integrated into existing architecture.	Application Code	SOA Architecture	SOA Architecture	The code is very modularised and follow SOA, making new additions easy	PASS

ITEMS TO TEST		TEST DESCRIPTION		TEST DATE	RESPONSIBILITY	
NFR15, NFR16 and NFR17		The system shall be accessible from desktops, laptops, tablets and smartphones. The system shall be accessible from Apple iOS/macOS, Android and Windows OS. The system shall be accessible at least from Google Chrome browser.		27/08/2018	Arturas Bulavko	
TEST CASES						
ID	OBJECTIVE	INPUT VALUE	EXPECTED OUTCOME	ACTUAL OUTCOME	COMMENT	RESULT
SQANFT26	Test the application on <i>iPhone 8</i> in Google Chrome browser.	Navigate to all application pages	All application pages must work	As expected	Voice Input (ASR) does not work	PASS
SQANFT27	Test the application on <i>Samsung Galaxy S6</i> in	Navigate to all application pages	All application pages must work	As expected	N/A	PASS

	Google Chrome browser.					
SQANFT28	Test the application on <i>iPad 2017</i> in Google Chrome browser.	Navigate to all application pages	All application pages must work	As expected	Voice Input (ASR) does not work	PASS
SQANFT30	Test the application on <i>Sony Vaio laptop</i> in Google Chrome browser.	Navigate to all application pages	All application pages must work	As expected	N/A	PASS
SQANFT31	Test the application on <i>desktop PC</i> in Google Chrome browser.	Navigate to all application pages	All application pages must work	As expected	N/A	PASS
SQANFT32	Check the application icon on <i>iPhone 8</i> .	Add application to home screen	Icon must be added to the home screen	As expected	N/A	PASS
SQANFT33	Check the application icon on <i>Samsung Galaxy S6</i> .	Add application to home screen	Icon must be added to the home screen	As expected	N/A	PASS
SQANFT34	Check the application icon on <i>iPad 2017</i> .	Add application to home screen	Icon must be added to the home screen	As expected	N/A	PASS
SQANFT35	Check the favicon on <i>Sony Vaio laptop</i> .	Navigate to the application home page	Favicon must be displayed in the browser	As expected	N/A	PASS
SQANFT36	Check the favicon on <i>desktop PC</i> .	Navigate to the application home page	Favicon must be displayed in the browser	As expected	N/A	PASS

ITEMS TO TEST		TEST DESCRIPTION		TEST DATE	RESPONSIBILITY	
NFR18		All system pages shall be HTTPS through the use of SSL.		27/08/2018	Arturas Bulavko	
TEST CASES						
ID	OBJECTIVE	INPUT VALUE	EXPECTED OUTCOME	ACTUAL OUTCOME	COMMENT	RESULT
SQANFT37	Check if <i>index.html</i> is HTTPS	index.html	HTTPS	HTTPS	N/A	PASS
SQANFT38	Check if <i>consent.html</i> is HTTPS	consent.html	HTTPS	HTTPS	N/A	PASS
SQANFT39	Check if <i>diagnosis.html</i> is HTTPS	diagnosis.html	HTTPS	HTTPS	N/A	PASS
SQANFT40	Check if <i>help.html</i> is HTTPS	help.html	HTTPS	HTTPS	N/A	PASS

Appendix K – Usability and Field Testing

Below are the test cases and test results concerned with Usability and Field Testing.

USABILITY TESTING			
ITEMS TO TEST	TEST DESCRIPTION	TEST DATE	RESPONSIBILITY
Whole System including NFR10, NFR11, NFR12, NFR13, NFR15, NFR16, NFR17	Give testees a set of tasks which they must perform within the specified time allowance. They will then be asked a set of questions to validate certain system aspects.	30/08/2018	Arturas Bulavko

Below are the testees which were recruited to perform the usability and field testing.

NAME/SURNAME	AGE	LOCATION	COMPUTER LITERACY	DEVICE
<removed>	23	Busy Street	Proficient	Samsung Galaxy S6
<removed>	52	House	Intermediate	Windows PC

The following tasks were given to both testees to perform. An observer was making notes about the different testing aspects.

TASKS	TIME ALLOWANCE
Locate and read system Usage Instructions .	5 minutes
Read the Privacy Policy .	5 minutes
Navigate to the Diagnosis Page , enter your Gender and Age .	3 minutes
Verbally enter Symptom "I have a light headache" and view your diagnosis.	4 minutes
Verbally enter your personal Symptoms and view the diagnosis.	3 minutes

Below are the usability and field test outcomes/results.

TESTEE	<removed>		
OBSERVER	Arturas Bulavko		
CHECKLIST			
CRITERIA		ANSWER	
Did the testee complete all tasks within specified time frame?		Yes	
Did the testee experience any problems?		No	
QUESTIONS ASKED & TESTEE ANSWERS			
QUESTION		ANSWER	
Is the layout and theme consistent on all pages?		Yes	
Was it difficult to complete each given task?		No	
Is the font size acceptable for your vision?		Yes	
Is the colour scheme acceptable for you?		Yes	
Were you satisfied with the help provided on wrong actions?		Yes	
Was it difficult to perform all tasks using your device?		Yes	
Were the navigation elements easy to interact with?		Yes	
Were you able to input data using voice?		Yes	
Were you able to hear verbal output by the system?		Yes	
NOTES			
No notes.			
RESULT		PASS	

TESTEE	<removed>	
OBSERVER	Arturas Bulavko	
CHECKLIST		
CRITERIA		ANSWER
Did the testee complete all tasks within specified time frame?		Yes
Did the testee experience any problems?		No
QUESTIONS ASKED & TESTEE ANSWERS		
QUESTION		ANSWER
Is the layout and theme consistent on all pages?		Yes
Was it difficult to complete each given task?		No
Is the font size acceptable for your vision?		Yes
Is the colour scheme acceptable for you?		Yes
Were you satisfied with the help provided on wrong actions?		Yes
Was it difficult to perform all tasks using your device?		Yes
Were the navigation elements easy to interact with?		Yes
Were you able to input data using voice?		Yes
Were you able to hear verbal output by the system?		Yes
NOTES		
The testee was unable to enter their symptom correctly from the first try due to background noise. On second time, they have succeeded.		
RESULT		PASS

Appendix L – Evaluation Against the Existing Systems

In order to confirm that the proof of concept offers novel functionality compared to the existing systems, both will be tested in terms of features and accuracy. This will also enable to assess the diagnosis degree of accuracy within the new system because a range of symptoms will be used to ensure a correct diagnosis is given, based on the data provided by NHS (2018). Thesis will now perform and document a number of tests, followed by a short discussion of results.

	Medical Applications			Universal Virtual Assistants			New
Features	Sensely	Your.MD	Buoy Health	Siri	Google Assistant	Cortana	Thesis App
Text Input	✓	✓	✓	✓	✓	✓	✓
Text Output	✓	✓	✓	✓	✓	✓	✓
Voice Input	✓			✓	✓	✓	✓
Voice Output				✓	✓	✓	✓
AI Platform		✓	✓	✓	✓	✓	✓
Web-Based		✓	✓				✓
Native Application	✓	✓		✓	✓	✓	
Direct Answer	✓	✓	✓				✓
Multilingual				✓	✓	✓	
Treatment Suggestion	✓						✓
Triage Level	✓	✓	✓				✓

Table 9 – System Feature Comparison

	Medical Applications			Universal Virtual Assistants			New
Symptoms/Diagnosis	Sensely	Your.MD	Buoy Health	Siri	Google Assistant	Cortana	Thesis App
Severe headache and muscle weakness. Migraine	✓	✓					✓
Light headache. Tension-type Headache	✓		✓				✓
Sudden headache and agitation. Cluster headaches	✓	✓	✓				✓
Thick green nasal discharge and headache. Sinus headache	✓						✓
High blood pressure and no diarrhoea. Hypertension	✓	✓					✓
Breath shortness and frequent cough. Asthma	✓	✓					✓
Sneezing and itchy throat. Allergic Rhinitis	✓						✓
Runny nose and fever. Common Cold	✓		✓				✓
Fever and cough. Acute Bronchitis		✓					✓
Sore throat and trouble swallowing. Viral Tonsillitis			✓				✓

Table 10 – System Diagnosis Accuracy Comparison

Table 9 demonstrates that all the planned features were correctly implemented. This is contrasted against the existing systems to prove that the functionality offered by the novel application excels compared to the existing approaches. *Table 10* proves that the novel system can correctly process all the symptoms within the selected domain. This was validated by supplying each system with the identical symptoms and checking whether it can output the knows/expected diagnosis. The universal virtual assistants did not correctly process any symptom because the only output received by these systems was the opening of a web-browser with search results of the symptoms. This is not a suitable approach to diagnosing the user.

Medical applications on the other hand have managed to correctly establish few diagnoses based on the supplied symptoms. As seen from *Table 10*, the existing systems have managed to correctly output the expected diagnosis in few situations. They followed a long and mundane process of asking the user several questions which sometimes were irrelevant to the symptoms, for example; have you had any recent blood tests. Most importantly, the existing systems output irrelevant diagnosis if they have failed to match the expected diagnosis. For example, supplying *High blood pressure and no diarrhoea* symptom resulted in *Food Poisoning* diagnosis within the Buoy Health application. While the current system is unable to ask relevant questions, it has correctly processed all the planned symptoms which was verified using the NHS (2018) website. This is achieved due to the sophisticated AI-driven API which takes into the account the user's age and gender to eliminate a number of irrelevant diagnosis. As a result of this, the novel application is able to establish an accurate diagnosis within the least amount of time, directly benefiting the overall UX.

Furthermore, the existing systems did not offer any treatment recommendations, despite correctly establishing the diagnosis. This is a major benefit of the novel application because it can suggest a number of recognised unprescribed treatment options which the user can adopt to relieve their symptoms. Such evaluation proves that the application is a suitable solution to the established problems because it can effectively recognise symptoms and provide an accurate diagnosis along with a suitable treatment suggestion and triage.