Vision Management System

Chad Piha
Computer Science Department
University of Cape Town
Cape Town, South Africa
phxcha001@myuct.ac.za

Justin Dorman
Computer Science Department
University of Cape Town
Cape Town, South Africa
drmjus001@myuct.ac.za

Zachary Bresler
Computer Science Department
University of Cape Town
Cape Town, South Africa
brszac002@myuct.ac.za

1 PROJECT DESCRIPTION

Vision Medical Suite (VMS) is a Non-Profit Company that provides free medical and dental care to orphanages, frail care centres and homes for the mentally and physically challenged [3]. Vision offers a range of services through volunteers at individual practices. Additionally, they offer a once a month mass dental and medical sedation clinic. Vision currently manages all their operations manually, which hinders their performance and efficiency. Hence, Vision is in dire need of an automated system in order to efficiently and effectively keep track of patients' visits and manage operations. This system will hence include a component for both the sedation clinic and individual practices.

2 PROBLEM STATEMENT

Vision is an NPO and hence has a limited budget. They cannot afford to outsource the development of this application and have therefore liaised with the University of Cape Town to assist. Thus, for our honours project, we have been presented with the following task: To create an online management system that streamlines the management of staff, patients and the various processes executed in Vision's operations. Vision has not proposed a budget for this project so we must ensure minimal cost throughout the development process. We are thus limited with respect to the tools and software we can use and can only make use of free software.

Vision requires a patient-centric system that can register or check in patients for a visit, capture their records and manage their transition throughout the visit. The application should be made easy to use in a fast-paced environment. The application will contain two levels of users, namely staff and admin. The staff will be the workers who are actively involved with managing the operations at the clinics/general practices and will control the patients' movements and capture their information. The admin will be the higher organisational level users and will be able to register staff, add/edit/delete information and manage the roles and permissions of the staff.

We aim to develop a fully functional, full stack application that will be used by Vision to ultimately increase their efficiency, better manage their operations and treat more patients.

3 PROCEDURES AND METHODS

In order to achieve our aims, we will go through the software engineering process which encompasses the entire range of associated activities, from project specification to software production.
The process begins with the requirements analysis phase where the stakeholders provide us with a detailed scope of the application. This will be conducted over a video call where the project proposer will outline both the functional and non-functional requirements. This allows us to move forward with the development process and gain an understanding of what needs to be incorporated in the application in order to ensure stakeholder value. Once the analysis phase is over and we have a good understanding of the requirements, we will proceed to create a high-level design of the application. This phase will include both a design of the interface and the database. A prototype will be created which will give a visual depiction of what the application will look like and what functionality will be included. The database will be designed to ensure all important patient visit information gets stored, and all functions will be able to be implemented. Best practices will be utilised to ensure a high performance, scalable application. Following the design phase, the implementation process will begin, which incorporates the development of the frontend and backend systems, to ultimately create the application. We will utilise the agile development methodology and develop the software in iterations. The team will work together, simultaneously working on the frontend and backend to produce mini increments of new functionality. In the first iteration, the team will concentrate on developing the foundation of the system, with the core functionality. Further iterations will build on the foundation, by producing batches of features. Iterative releases will improve efficiency and reduce risks by allowing us to find and fix defects and align expectations early on. [4]

3.1 Design Explanation

A prototype will be required in order to outline the design and ensure that the exact needs and expectations of the system are met. This will be achieved by receiving early feedback which will be used to make changes to the design. It will also allow us to save time and resources as changes made earlier in the development process are easier to handle. [20]

A low-fidelity, horizontal prototype will be developed to illustrate the user interface of the product. This will include a broader view of the entire system which will include the layouts of all the windows, menus and transitions.
The prototype will be evaluated by the stakeholders involved, as well as the Vision staff who will provide qualitative feedback to further improve the application. The main aim of this prototype is to identify whether the application captures all the requirements set by Vision, and if they are presented in an adequate manner. The prototype will be developed using Adobe XD, a specialized prototyping tool for mobile and web applications.

A minimalistic design will be implemented throughout the web application. It will be designed such that the speed of usage of application will be maximized, with responsiveness as a focus point. This is important for the proposed system as clinicians may often find themselves in time-critical situations. Simplicity and convenience will also be prioritized as it will allow staff members to complete their administrative tasks at a faster rate. Furthermore, the navigation throughout the system will be clear and user-friendly, a consistent colour palette and theme will be displayed throughout the application and user inputs will be minimized to prevent errors.

The expected challenge will be the compatibility of the application across various devices and browsers. Each of these provide different screen sizes and resolutions which our application will need to adjust to, in order to provide the same user experience throughout.

3.2 System Explanation

The architecture of the system is separated into layers relating to its functionality. Our system is split up into three separate parts, namely the database, application server and the frontend (React application). The application server is the middleman of the system and is responsible for processing the requests that are made by the frontend of the application (the client). Once these requests are processed by the server, they will be routed to the database where relevant data will be gathered and sent back as a response to the server. This data will then be returned to the frontend of the application where it will be managed and dealt with to present to the client. JSON will be used as the standardised format to interchange data between the client and the server.

3.2.1 Database

Due to the functional capacity of the system to be developed, a dynamic and interactive web application is required, along with a database.

A database acts as an incredible tool to record, store, retrieve and compare data. It is important that the database is designed using best practices to ensure a scalable, high performance application [10]. The relational database will be run on a MySQL server. In the system, lots of important data needs to be stored and retrieved at a fast rate. It is thus important to apply best practices to reduce redundancy and duplication, enhance the quality of information, and increase overall data integrity, to ultimately result in quick execution of queries and better performance [10,11].
An SQL will be most appropriate in the case of this system, as it relies heavily on complex relationships. For each patient visit, a visit instance will be created, which has a many to many relationship with the patient table. Any other information to be captured during the visit, will be contained in its appropriate table, which has a one to one relationship with the visit table.

3.2.2 Application server

The application server, which handles the business logic, sits between the web server and the database. The web server is there to process HTTP (hypertext transfer protocol) requests from the frontend (client). It then sends these requests to the application server. The logic developed in this server then determines what information it should request from the database [6, 7]. This logic is the API (Application Programming Interface). The API defines different routes that will be called from the frontend. These routes each represent a different action that makes requests to the database to create, remove, update, or to delete information. This API will follow a REST architecture which means that any interaction with the API requires that all information needed to perform a specific task must be provided. This increases performance and scalability of the server as it allows the server to remain stateless. [8]
The application server will be developed in node.js with the use of express.js as a framework, to enable fast development and increased scalability and performance. [9]

3.2.3 Frontend application

In order to build the frontend application, we will use React.js. React.js is an open-source JavaScript library used to build single-page applications (SPAs). A SPA sends only a single request to store all the data, which is transmitted back and forth to the server, allowing for high access speeds and offline capabilities.
The user interface of the application will be broken down into many smaller components which allows for easier configuration and tuning. Furthermore, React.js provides a Virtual DOM where all changes are stored. Therefore, once a change is committed, the Virtual DOM is compared against the real DOM, and only the related component is updated. This results in less page re-renders, thus improving application performance. [12,13]

3.3 System Evaluation

Throughout the development of the application, user testing and performance measurements will be conducted on each iterative increment of the system. Dummy data will be used to test the software during its development. Each increment of the software will be evaluated and tested by the stakeholders involved, to ensure we remain in line with their requirements and that the conceptual model remains intact. Feedback will be received, and change will be embraced.

Once the application is complete, it will be thoroughly tested (with the dummy data). To test quality and functionality, the team will conduct automated unit testing. We aim to further run a number of evaluations to test the usability and logical flow of the application. Initially, the selected participants will be the users of the system within the organisation and will therefore be able to provide the most meaningful conceptual insight. We aim to further test the application on users with a computer science background, as they will have the means to evaluate the system on a more technical basis.

We plan to use cognitive walkthroughs as a method for our usability evaluation. Participants will be asked to complete a series of tasks, while the evaluators (our team) observes and takes notes. Direct observations will be used such that interactions, processes and behaviours will be monitored as the users interact with the application [5]. The participants will be asked to verbalize their thoughts when using the application. This allows us to get the participants' perspective with minimal influence from the group members. Conducting the observations will provide valuable insight into the UI/UX and the functionality of the application, and subsequently allow for possible improvements based on the results. Once the evaluations and testing are complete, the application will be deployed, and made accessible only to authorised users. Upon deployment, the database will be re-created, and the system will start from scratch.

4 ETHICAL, PROFESSIONAL AND LEGAL ISSUES

When the system is completed and deployed, it will be owned by UCT and Vision Medical Suite. It will become Vision's responsibility to manage the long term maintenance of the application.

As we will be conducting testing of our application with real users, ethical clearance (expedited) will be required, in the form of signed or oral consent from the participating users. There will be no risks during the testing phase as the usability testing will be done over a video conference call in order to avoid the risk of exposure to Covid-19. However, the participants of the evaluations will likely be health workers. There is thus a potential ethical issue of taking the time of health workers during a pandemic. To combat this, we will make sure the evaluations are conducted efficiently and at the convenience of the participants. We will be testing the application using fake data, and thus won't require clearance for the use of data. Since the data will be fake, there is no danger of being able to identify a person.

The deployed application will be dealing with real, sensitive patient data. The application will thus need to be designed with the assurance of security of information and the appropriate role-based access configuration, in order to comply with Department of Health's requirements around patient data [2]. To do this, we will design the application with best security practices in place. Authentication, authorisation and encryption will be implemented, to ensure access control and confidentiality. During the testing phase, we will conduct penetration testing to find vulnerabilities. If a vulnerability is found, it will be fixed.

The system also needs to comply with the POPI act which states how institutions should act when collecting, processing, storing and sharing an entity's personal information, to ensure accountability. [1]

5 RELATED WORK

We analysed the following paid hospital management systems: eHospital systems, Sapphire, Medstar HIS, Athenahealth, TeamDesk and Health Information Management Solutions (HIMS); As well as the following open-source systems: OpenMRS and OpenVista. These management systems provide a large range of services, many of which fall outside of Vision's scope. The functionalities present across these systems include staff, patient, financial, inventory, bed and claims management systems, as well as online appointment scheduling, HR management, dedicated patient portals and native applications in conjunction with web applications.

These services are on a more general management scale and are catered for much larger organizations who have many departments requiring various functionalities. Conversely, Vision manages services on a much smaller scale and has a more patient-centric scope. Vision requires a fine-grained application to manage their unique services which target specific beneficiaries. Therefore, having a big system for Vision, with a large range of non-essential features may result in a degree of impracticality and may even reduce user efficiency. [14,15,16,17,18,19,21,22]

Vision staff members will manage the booking of appointments on behalf of the beneficiaries, leaving the self-service online booking scheduling system unnecessary. Vision also requires specialized functionalities catered for their

monthly sedation clinics, which no other hospital management systems provide. The clinics are located at different venues, making the bed management unnecessary. Additionally, finance, inventory and HR management are not required for Vision's application.
As an NPO, Vision lacks the budget required for most of the systems currently on the market and they also lack the budget to outsource developers to build a customized system for them. This has resulted in the need for us to create a system specifically tailored for them at no cost, where the monthly sedation clinics and daily general practices are considered.

We investigated the possibility of using the open-source systems and adding the various specific functionalities required by Vision, however we found this to be inefficient. Doing so will take up a lot of time trying to understand the system and integrate the functionalities. Additionally, there are more security risks as the code can be seen, making it is easier for hackers to find exploits. Moreover, developing a system from scratch would allow for greater customization and control over the system, and the system can be more efficient as we will provide the exact requirements.

6 ANTICIPATED OUTCOMES

6.1 System

6.1.1 Software

The system to be created will be a web application, compatible with both mobile and desktop devices. The system will be an interactive application where the users will be able to manage the transition of patients throughout their visits and capture necessary information inherent to said visits. The application will consist of components to manage both general practice and sedation clinic operations. In each component, users will be able to view and manage patient appointments on a specific day and select a patient from the appointment list to start their visit. Thereafter, the visit journey will begin. For the general practice component, patients will be managed individually, with necessary information (relevant to the service being rendered) being captured relating to their visit. The sedation clinic component, to the contrary, is a mass clinic which will require multiple patients to be managed at once. The visits will thus be split into various phases. In each phase, there will be a list of patients waiting for the respective activity. Patients will be ordered by their urgency and waiting time. The users will be able to select patients from these lists and capture information relevant to the phase they are in. Once the required information is captured, users will then be able to move the patients to the next phase, until the visit is complete, and the patient exits the clinic.

Included in the system will also be an admin panel, where specific admin users will be able to manage high level operations relating to the system. These include registering general users, adding, editing, and deleting information, as well as viewing reports based on data collected from visits.

6.1.2 Key features

- Sign into the system, and choose general practice or sedation clinic component (if sedation clinic is running on that day)
- View list of active visits and all patients in the system. Selecting a patient should display their profile, including information on their past visits (if permitted access).
- Register patient, book patient appointments, cancel bookings and push bookings forward. There should be a separate management interface which allows for flexibility when managing appointments.
- [General practice]: Select patients and capture information relevant to the service being rendered.
- [Sedation clinic]: Select patients and move them to different stages of the clinic. These stages include waiting for triage, waiting for theatre, in theatre, waiting for disposition and waiting for exit.
- Capture and edit patient information relevant to the stage they are in.
- Admin panel
    - Register staff. Email will be sent to staff members with a link with an expiration time. The link signs the staff in and requests that they change their password
    - Manage user roles and permissions
    - Add/edit/delete information
    - View automatically generated reports

6.1.3 Design Challenges

A major challenge anticipated in the creation of the system would be the inability to perform live tests and direct observations at the clinics, due to Covid-19. This prevents us from getting first-hand experience to observe the operations of the clinics. This may limit our understanding of the requirements and force us to rely solely on the information provided by the project proposer.
There is a further design challenge to manage the balance between the complexity and simplicity of the system. The system has a range of complex functionalities that need to be designed in a way to ensure optimal usability and user friendliness. This needs to be the case as the application should be easy to use in a fast-paced environment.

6.2 Expected Impact of the project

Implementing a healthcare management system will help Vision Medical Suite effectively and efficiently manage their

clinics and associated general practices. Processes will become seamless, removing the effort of manually recording and managing operations. It will enhance the staff's ability to coordinate care, streamline the search of patient files, and increase data security. This system could potentially enable Vision to accommodate more patients in their once a month clinic, as well as allow general practices to see more patients from Vision's affiliated beneficiaries.

People in the beneficiaries who are unable, or find it difficult, to receive healthcare will have access to well run, organised clinics and general practices.

6.3 Key Success Factors

The success of the system is based on three key factors, including performance, usability and Quality Assurance (QA) and testing results.

In terms of performance, measurements will be conducted to determine if there is a general improvement in efficiency and effectiveness of operations, after the system gets introduced.
After deployment of the system, the users will provide feedback on how their experience was while using the system. If the feedback is positive, and the system effectively performs its fundamental job of managing operations with minimal issues, then the project will be regarded as a success. If the feedback is negative, and the system fails to provide its fundamental functionality and has many issues, it will be regarded as unsuccessful.
Furthermore, to fully ensure that the systems functionality is working successfully, we will conduct QA and testing on the system. If the system passes all unit tests – test cases written to ensure specific functionality is working as desired – the system will be regarded as successful. Otherwise, we cannot deem the project as successful.

7 PROJECT PLAN

7.1 Risks and Risk Management Strategies

The risk and risk management strategies can be seen in our Risk
Matrix table (see Appendix A).

7.2 Timeline

This project runs from 30 March 2020 to 19 October 2020, and the timeline can be seen in our Gantt chart (see Appendix B).

7.3 Resources Required

- Three computers with at least 4 GB of RAM, intel i5 processor and minimum 128GB SSD
- Express.js
- Node.js
- JavaScript
- CSS (Cascading Style Sheets)
- HTML (Hyper Text Mark-up Language)
- JSX
- JOI
- MySQL
- SQL
- RESTful APIs
- Slack
- GitHub
- React.js
- Redux
- Servers for hosting
- Jira
- Visual Studio Code
- NPM package manager
- Adobe XD

7.4 Deliverables

Gather project requirements and features	Fri 17-April
Define functional requirements	Fri 1-May
Define non-functional requirements	Fri 1-May
Basic break down of team member roles	Thu 30-April
Database tables and basic database design	Sun 17-May
Use case diagram	Fri 1-May
Literature Review	Mon 11-May
Low-fidelity prototype (desktop)	Tue 19-May
Low-fidelity prototype (mobile)	Wed 3-Jun
Database (MySQL) design and set up	Mon 18-May
Configure application server	Wed 13-May
Complete RESTful API endpoints	Sun 19-July
Complete Authentication on the backend using JWT	Sun 24-May
Complete authorisation of API endpoints	Fri 10-Jul
Connect the database and the	Wed 20-May

server	
Create a React.js project using create-react-app	Mon 18-May
Complete structuring of frontend application pages	Fri 31-Jul
Complete frontend application components	Fri 31-Jul
Complete frontend functionality	Wed 12-Aug
Create an Admin Panel	Fri 10-Jul
Weekly reports	Sun 9-Aug
QA and testing	Fri 31-Aug

7.5 Milestones

Table 1 shows the major project milestones to diarise.

Table 1: Table showing the project milestones

Literature Review	Tue 12-May
Completion of analysis	Tue 12 May
Completion of database	Tue 26-May
Project Proposal due, including project plan	Tue 2-Jun
Completion of interface design	Thu 4-June
Completion of page structuring	Sat 6-June
Review of staff feedback on proposals	Mon 18-Jun
Revised Proposal Finalized and uploaded to Vula	Mon 29-Jun
Completion of API	Sunday 19 July
Initial Software Feasibility Demonstration	Mon 3 Aug – Tue 11 Aug
Completion of frontend functionality	Wed 12-Aug
Weighting for project marking decided	Mon 17-Aug
Completion of QA and testing	Sat 31-Aug
Final Complete Draft of paper	Fri 11-Sept
Project Paper Final Submission	Mon 21-Sept
Project Code Final Submission	Fri 25-Sept
Final Project Demonstration	Mon 5 Oct - Fri 9 Oct
Poster Due	Mon 12-Oct
Web Page	Mon 19-Oct

7.6 Work Allocation

We have ensured that the work allocation has been equally distributed amongst the group members. The split up of the tasks ensure that each member is working on separate components of the system. Although there is interleaving between some features, the features will still be able to work independently based on the modularity of the tasks.

Chad Piha:

- Information Capture & posting to database for general practice (GP) and sedation clinic
- UI/UX, Structure frontend containers and components and Prototyping
- Device Compatibility
- Editing staff/patient/screening/procedure/institution/ appointment information for GP and sedation clinic
- Visualizing weekly reports
- Voice-to-text, Printing option API
- Validation and error handling
- Screen responsiveness
- Documentation
- Appointments calendar
- QA and testing

Justin Dorman:

- Token authentication and management
- Ensure privacy and confidentiality - Authorisation
- Database design and queries
- Admin panel: Functionality

[Server side functionality]:

- Register patient and staff
- Get lists of patients and visits
- Add/edit/delete all visit related information

- Management of appointment system (check-in patient, schedule booking, cancel booking, push booking forward)
- Validation and error handling [server side]
- Generate reports
- QA and testing

Zachary Bresler:

- Frontend state management
    - Which additionally includes creating actions to fetch, post, delete and update data in the database
- Structure frontend containers and components
- Security and encryption
- Server configuration and basic API
- Connect server to database
- Performance enhancements techniques
- Generating reports
- Admin panel: Set up and structure
- Search (for ICD10 code and medication) API's
- Waiting lists management
- Appointment frontend management
- Block saved credentials on browser
- QA and testing
- Email functionality

REFERENCES

[1] Workpool. 2016. What is POPI? The Protection of Personal Information (PoPI) Act explained. Retrieved May 29, 2020 from https://www.workpool.co/featured/popi

[2] Health Professions Council of South Africa. 2016. Confidentiality: Protecting and Providing Information. ACM 16, 1-5

[3] Vision Medical Suite. 2016. About us. Retrieved June 2, 2020 from http://www.visionmedicalsuite.co.za/about-us/

[4] Gaurav Kumar, Pradeep Kumar Bhatia. 2012. Impact of Agile Methodology on Software Development Process. International Journal of Computer Technology and Electronics Engineering (IJCTEE) 2, 4 (August 2012), 46-49.

[5] CDC. Data collection methods for program evaluation: Observation No. 16. Retrieved May 9, 2020 from https://www.cdc.gov/healthyYouth/evaluation/pdf/brief16.pdf

[6] IBM. 2020. Introduction: Application servers. Retrieved April 24, 2020 from https://www.ibm.com/support/knowledgecenter/SSEQTP_8.5.5/com.ibm.websphere.base.doc/ae/welc_servers.html

[7] NGINX. n.d. NGINX: What Is an Application Server vs. a Web Server? Retrieved April 25, 2020 from https://www.nginx.com/resources/glossary/application-server-vs-web-server/

[8] Roy Fielding. 2000. Architectural Styles and the Design of Network-based Software Architectures. Ph.D. Dissertation. University of California, Irvine.

[9] Express.js. 2019. Express.js. Retrieved April 27, 2020 from https://expressjs.com/

[10] Leslie Bloom. 2019. Benefits of Using a Database. (June 2019). Retrieved April 22, 2020 from https://bizfluent.com/facts-4924693-benefits-using-database.html

[11] Kristi Castro. 2018. The benefits of good Database Design. (July 2018). Retrieved April 22, 2020 from https://www.tutorialspoint.com/The-benefits-of-good-Database-Design

[12] Reactjs. N.d. React. Retrieved June 2, 2020 from https://reactjs.org/

[13] Hima Chitalia. 2017. Hey React, What is the Virtual DOM?. (October 2017) Retrieved June 2, 2020 from https://medium.com/coffee-and-codes/hey-react-what-is-the-virtual-dom-466ec333bf9a#:~:text=%E2%80%9CThe%20Document%20Object%20Model%20(DOM,can%20connect%20to%20the%20page.

[14] Adroit Infosystems. eHospital Systems. Retrieved from https://www.adroitinfosystems.com/products/ehospital-systems

[15] Sapphire. Sapphire Hospital Management System. Retrieved from https://www.sapphirehms.com/

[16] Pinaacle, "MedStar Hospital Management and Information System," Available: http://medstarhis.com/docs/Medstar-Brochure.pdf. [Accessed 6 May 2020].

[17] AthenaHealth, "Home Page," 2020. Available: https://www.athenahealth.com/. [Accessed 6 May 2020].

[18] TeamDesk. 2020. Medical Practice Manager database. Retrieved May 7, 2020 from https://www.teamdesk.net/

[19] NBS Digital Technologies. 2020. Hospital Information Management Solution (HIMS). Retrieved May 7, 2020 from https://nbsdt.com/hims/

[20] Shanuj Mishra. 2019. The Importance Of Prototyping In Designing. Retrieved June 2, 2020 from https://uxdesign.cc/importance-of-prototyping-in-designing-7287c7035a0d#:~:text=Following%20are%20the%20fundamental%20reasons,focusing%20on%20important%20interface%20elements.

[21] OpenMRS. 2020. OpenMRS. Retrieved July 11, 2020 from https://github.com/openmrs

[22] Andy Pardue, Derek Veit, Pete Johanson, Rob Kilian. 2020. OpenVista®. Retrieved July 11 from https://sourceforge.net/projects/openvista/

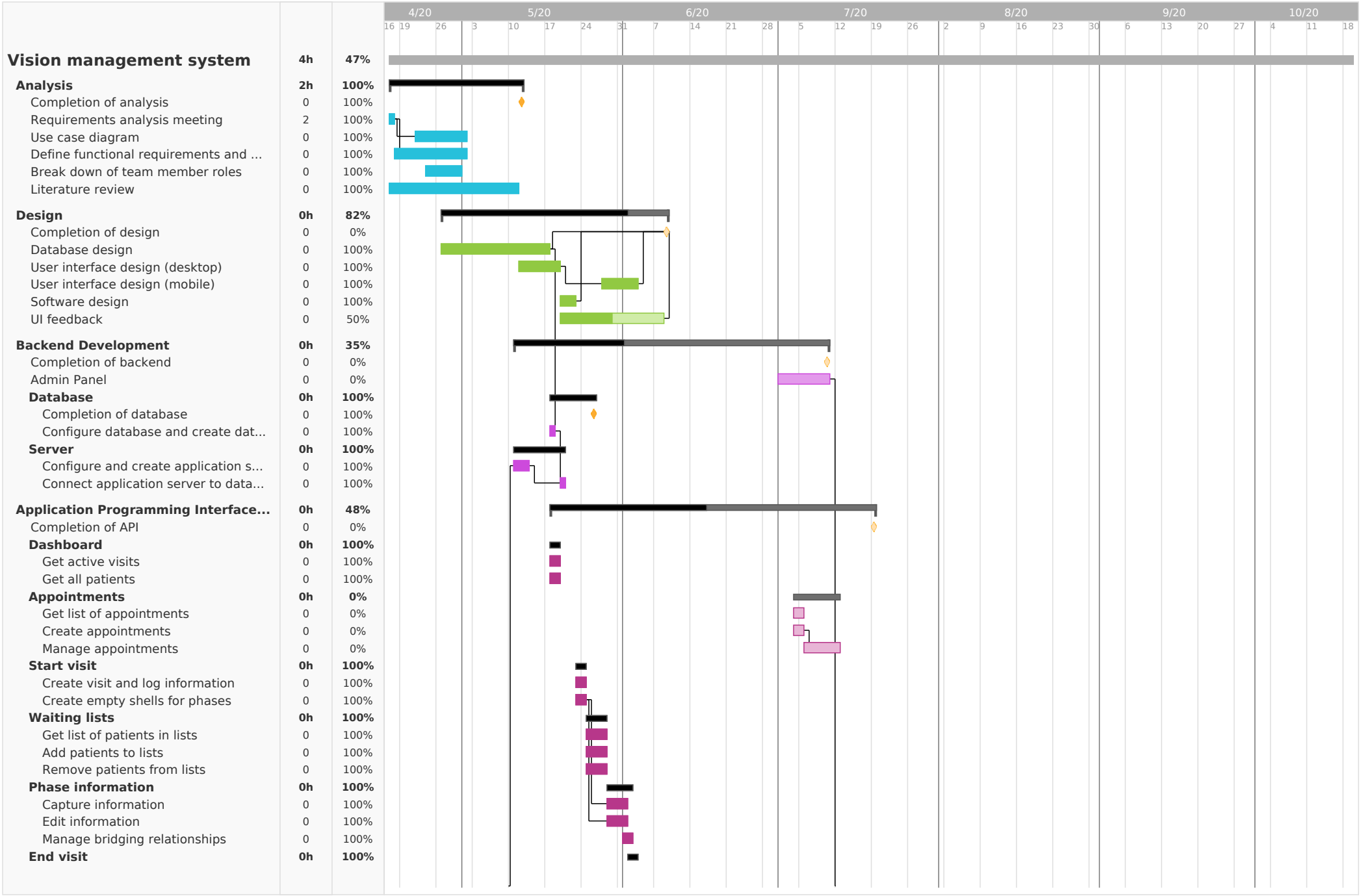
Appendix

A. Risk and Risk Management

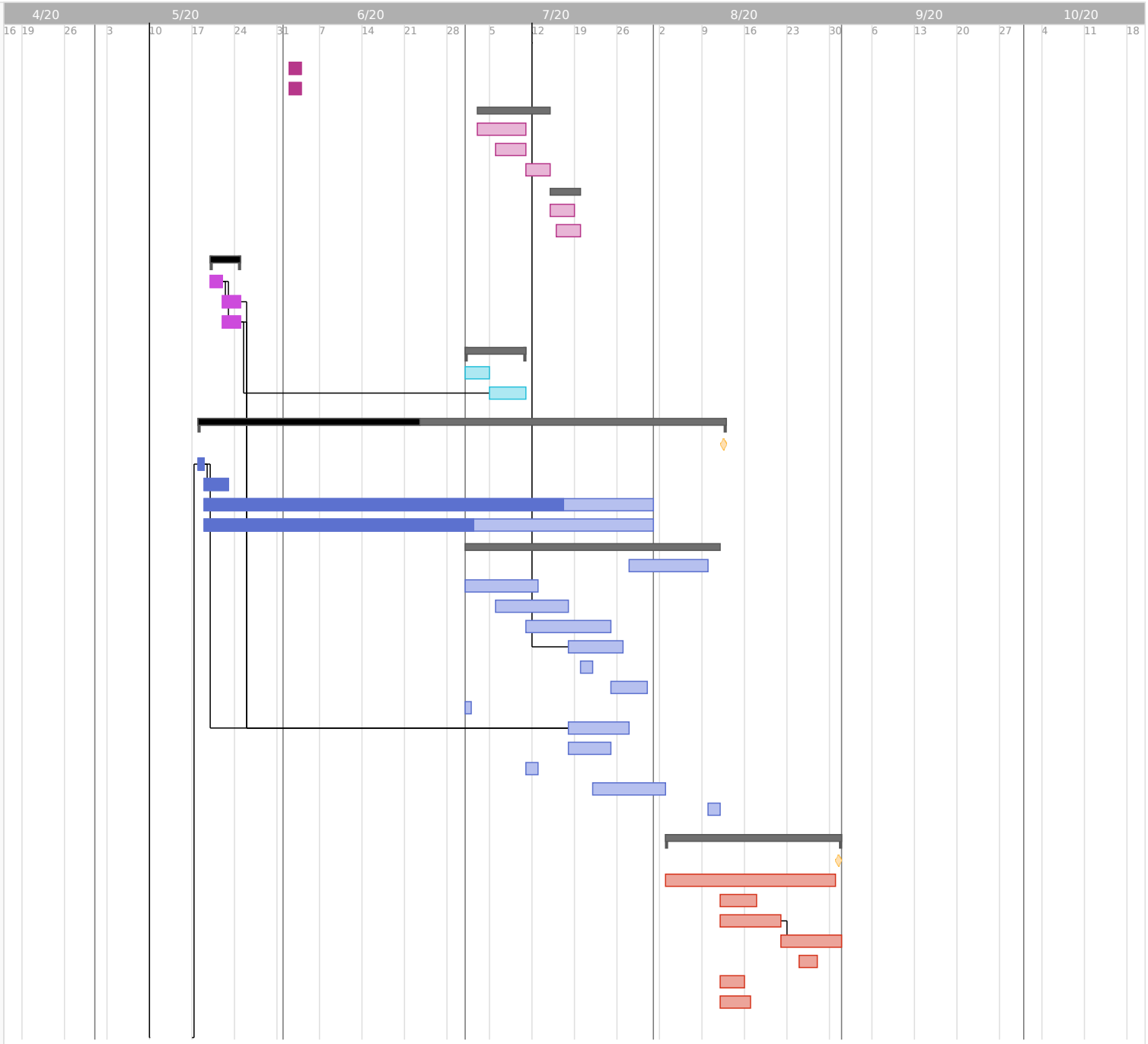
Risk Number	Risk	Impact	Probability	Management/ Mitigation
1	**Requirements inflation** - Features that were not identified at the beginning of the project emerge that threaten estimates and timelines.	7	7	Account for additional time for implementation in the project planning in case the risk becomes a reality.
2	**Inherent schedule flaws** - Incorrect scheduling of tasks resulting in the project not being completed.	10	6	Ensure core implementation is completed first. Concentrate on the main functionality, and once that is completed, focus on the extra features.
3	**Poor productivity** - Lack of urgency results in loss of time in early project stages that can never be regained, subsequently resulting in a low standard of work.	8	4	Making use of management tools and planning in advance, setting deliverables and timelines. Possible penalties can be introduced if a team member is not putting in the required amount of work.
4	**Imperfect knowledge between group members** - Lack of intragroup communication resulting in members not being aware of what other group members are doing which can subsequently result in time being wasted, delaying the overall progress.	8	5	Using software communication tools such as Slack to ensure the group aligned at all times during development. Additionally, scrum methodology will be enforced, with a daily stand up meeting.
5	**Absence of group members** - Group members not showing up for group meetings, or getting ill and not being able to work, or not communicating on the group communication channels resulting in delays and errors.	9	6	Ensuring that if a group member is to be absent then they should inform the other group members. The other members can therefore put in the necessary time and effort to cover the potential loss in progress.
6	**Unclear specifications (unclear breakdown of requirements)** - Conflicts may arise between requirements, resulting in either insufficient or unclear specifications. This could be a result of the project specifications not being properly broken down	9	7	Before development commences, ensure that all specifications and requirements are understood by all involved, to lessen the impact of this risk occurring. If any requirement is unclear, it should be clarified with the stakeholders involved.
7	**Gold plating** - Adding features that are out of the scope of the project's requirements resulting in wasted time, and thus delaying the project.	5	6	Ensure core implementation is completed first. Concentrate on the main functionality and features that match the requirements, and once that is completed, focus on the extra features.
8	**Software errors** - Issues with the chosen software due to conflicts with other software or the software's inability to achieve a specific requirement.	7	6	All software should be researched indepth to ensure its compatibility with other software. Additionally, have backup software that can migrate into the tech stack with little effort.

9	**Losing or accidentally deleting code -** Not saving work or accidentally deleting files or code snippets resulting in lost progress and errors in the code.	10	5	Using a version control service such as GitHub to keep track of different versions of the code. Code should be regularly backed up to an online repository. If any issues arise, an earlier version of the code can always be recovered.

B. Gantt Chart



Remove waiting lists	0	100%
End visit session	0	100%
General Practice	**0h**	**0%**
Manage visits	0	0%
Capture/edit information	0	0%
Manage appointments	0	0%
Admin	**0h**	**0%**
Manage roles and permissions	0	0%
Add/edit/delete information	0	0%
Authentication	**0h**	**100%**
Configure JSON Web Token	0	100%
Create API endpoint for sign in	0	100%
Create middleware	0	100%
Authorisation	**0h**	**0%**
Manage role-based access configurat...	0	0%
Control access for each endpoint	0	0%
Frontend Development	**1h**	**42%**
Completion of frontend functionality	0	0%
Create react app	1	100%
Set up Redux	0	100%
UI components	0	80%
UI Containers	0	60%
Functionality	**0h**	**0%**
View reports	0	0%
Capture/Edit visit information	0	0%
Manage waiting lists	0	0%
Manage appointment	0	0%
Register staff	0	0%
Switch between general practice a...	0	0%
View patients' past visits	0	0%
View user profile	0	0%
Authentication	0	0%
Register patient	0	0%
Manage user roles and permissions	0	0%
Add/edit/delete information	0	0%
Additional functionality	0	0%
QA and testing	**0h**	**0%**
Completion QA and testing	0	0%
Refactoring	0	0%
Review all requirements	0	0%
Design test cases	0	0%
Write unit tests	0	0%
Integration testing	0	0%
Performance testing	0	0%
Security testing	0	0%



Cross-platform testing	0	0%
Version Control	**1h**	**100%**
Set up Git repository (server)	1	100%
Set up Git repository (web app)	0	100%
Honours Project Deliverables	**0h**	**13%**
Project proposal and project plan	0	100%
Initial software feasibility demonstrat...	0	0%
Final complete draft of paper	0	0%
Final Project paper	0	0%
Project code - final submission	0	0%
Final project demonstration	0	0%
Poster due	0	0%
Web page	0	0%