<QMePls by Team Titans>

DESIGN REPORT ON SOFTWARE

MAINTAINABILITY

Version <1.3>

<5/10/2021>

Seow Jing Hng Aloysius
Jacob Law Zhen
Jolene Tan
Soh Qian Yi
Samuel Lee Si En
Zeta Chua Hui Shi

Team Titans
School of Computer Science and Engineering

VERSION HISTORY

Version #	Implemented By	Revision Date	Approved By	Approval Date	Reason
1.0	Jolene	1/10/2021	Aloysius	2/10/2021	Uploaded Initial Template
1.1	Aloysius	2/10/2021	Jolene	3/10/2021	Design Strategies Architectural Design Patterns
1.2	Aloysius	3/10/2021	Jolene	4/10/2021	Software Configuration Management Tools
1.3	Jolene	4/10/2021	Aloysius	5/10/2021	Architectural Design Patterns

TABLE OF CONTENTS

1 DESIGN STRATEGIES

1.1 PLANNING PHASE

During the Requirements Elicitation phase where we drafted out all the initial design plans for the project, we considered the possibility of expanding the functionalities and scale of the project after the release of the application.

Increasing the scale of the project would mean that we have to cater to a larger database size as well as higher traffic rates, thus we have focused on the Scalability of the project. In addition to that, we all focus on the Extensibility of the Project as we want to include new functions in the future. We thus chose the Model-View-Controller (MVC) model as our architectural model, since the different logic layers are separate, which means that the project will be easily scalable and extensible to future expansions without affecting the old functions.

1.2 PROCESS OF DEVELOPMENT

We will test the software in a small, test driven environment. Due to limitations arising from manpower costs as well as the Safe-Distancing measures put in place in Singapore, the testing role is being taken up by our own team members to conduct, where each member will be involved in the continuous feedback on the design and usability of the application.

1.3 CORRECTION BY NATURE

We will correct our application during testing of the application using:

1.3.1 Corrective Maintainability

Fault detection through conducting tests

1.3.2 Preventive Maintainability

Implement features in an atomic manner, where each feature is tested independently to allow easier isolation of errors.

1.4 ENHANCEMENT BY NATURE

We will enhance our application while testing the application using:

1.4.1 Adaptive Measure

Ensure software is easily adaptable to new operating environments.

1.4.2 Perfective Measure

After product release, we conduct constant and quick error detection and then rectify the errors immediately, reduce maintenance time and cost.

1.5 MAINTAINABILITY PRACTICES

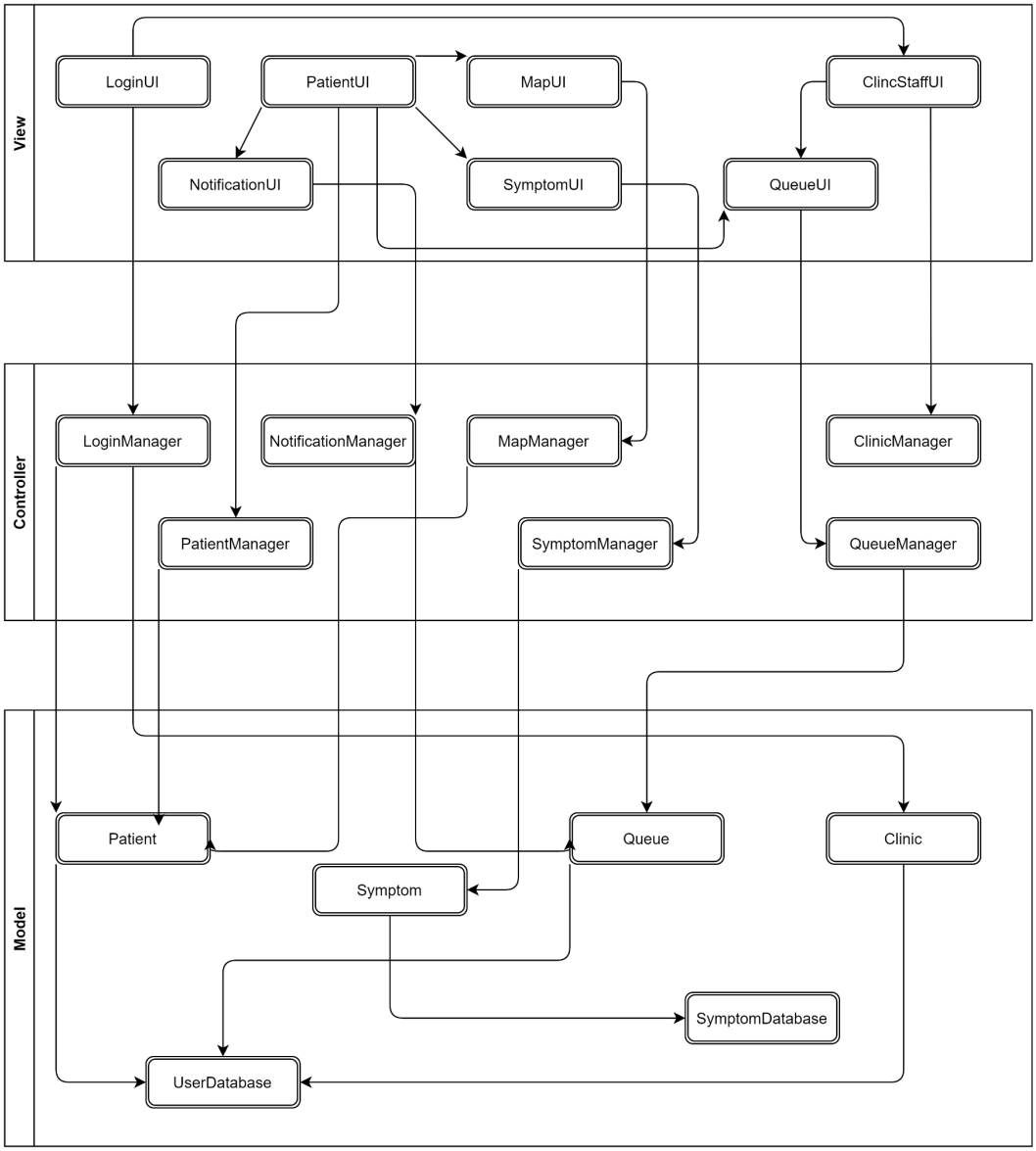
To ensure high maintainability in QMePls, we adopt the follow practices throughout our project:

- Codes are commented for better readability and ease of understanding
- Standardized documentation format
- Frequent and incremental changes to the code base
- Version control
- Loose coupling
- High cohesion
- Modularity

2 ARCHITECTURAL DESIGN PATTERNS

QMePls adopts the MVC (Model-View-Controller) architectural design pattern in a Layered System Architecture format. Classes are segregated into 3 distinct layers according to their functionalities, stacked on top of each other. Additionally, each layer is only allowed to interact with the layer directly below.

Firstly, the Model Layer is where persistent data logic and database records are being stored and retrieved by the Controller Layer. Next, the Controller Layer is where the application logic and processing of user inputs and data are being carried out. This is done by retrieving information from the model layer when requested, processing the information before returning the results to the View Later. Lastly, the View Layer is where user interaction with the application takes place. This is also where formatted information is being displayed to the user upon interaction.



3 SOFTWARE CONFIGURATION MANAGEMENT TOOLS

3.1 GITHUB

GitHub is a source code hosting platform using Git, a source code management and distributed version control software. We chose GitHub as most of the team members are familiar with it and it is also being supported by various IDE applications, making it more versatile. Our teammates can collaborate easily using the branch, pull and merge function to work on individual components of the project before merging to integrate the code. GitHub also allows opening of labels to assign specific tasks to team members, where the team members being assigned would receive updates on the progress for the issue, making collaboration more accountable and efficient.

3.2 MEDIAWIKI

MediaWiki is a free and open-source documentation and collaboration platform. There are many tutorials and Frequently Asked Questions (FAQs) available online which can help users to know the functions that are available. Since it allows live editing by multiple users at the same time, there will not be loss/overwriting during collaborative work.

3.3 GOOGLE DRIVE

Google Drive is a free online file storage system where teams can create a shared folder to store all relevant documents as backup. Our team chose this file management system as it supports the full functionality of Microsoft Office desktop applications, many of which are heavily used in our project. Google Drive also allows live collaboration and editing of documents which supports version control and will not lead to data loss/overwriting.