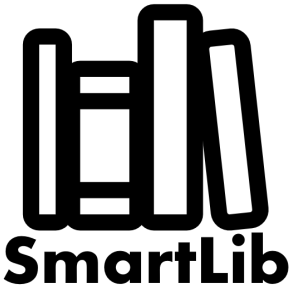


DESIGN REPORT ON SOFTWARE MAINTAINABILITY

Version *1.1*
3/4/2021

Team Xeon

Document Number: SL-07

REVISION HISTORY

Version	Date	Author	Description of Changes	Approved by
1.0	31/03/2021	Wilson Tai	Initial Template	Kenny Voo Tze Rung
1.1	03/04/2021	Wilson Tai	Add Design strategies Add Architectural design pattern Add Software Configuration tools management	Kenny Voo Tze Rung

TABLE OF CONTENTS

1 Design Strategies

1.1 Planning Phase Before Development

From the start, we would analyse and predict what kind of improvements we would be implementing in the future after the release of the application. Once the widespread usage of this application is used, it will be used in different environments. Hence, we targeted reusability as one of the factors that we would have to look into the future.

SmartLib uses many cameras for its detection system, so we decided to adapt the client and server architectural model as multiple cameras can connect to the same server. The client side is mainly used by students, therefore user experience is important. We will be adapting the Model-View-Control (MVC) model as our architecture, the MVC model allows for easy change to the UI, this is done by decoupling of components, making it easier to add functions or modify functions without affecting the software.

1.2 The process of Developing

We are testing out in a small, test driven development. We will focus development and testing in NTU library. Students and librarians will perform the role of tester and provide continuous feedback on the design and usability of the application. After testing, SmartLib will be pushed out to libraries all across Singapore.

1.3 Correction by Nature

We will correct our application while testing the application, this is what we will look out for:

1.3.1 Corrective Maintainability

Fault detection done through testing.

1.3.2 Preventive Maintainability

Features implemented in atomic manner, each feature, tested independently, error detected easily.

1.4 Enhancement by Nature

We will enhance our application while testing the application, And this is what we will look out for:

1.4.1 Adaptive Maintainability

Can be easily adapted to a new operational environment.

1.4.2 Perfective Maintainability

After product delivery, quickly detect an error and correct it, reducing maintenance costs and time required.

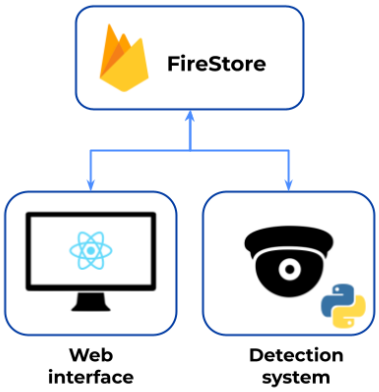
1.5 Maintainability Practices

To uphold quality in both process and product, we have implemented the following maintainability practices over the course of our project:

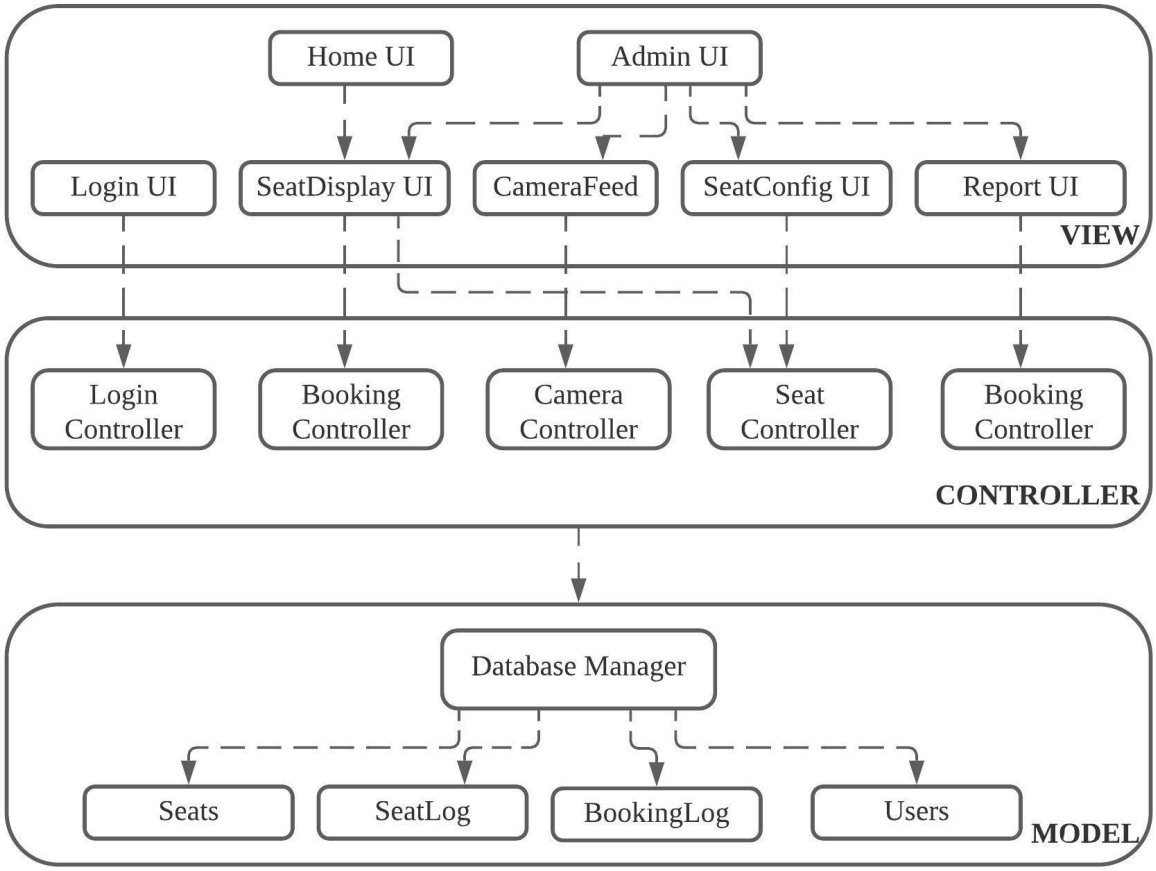
- Readable Code
- Version Control
- Standardized Documentation
- Modularity

2 Architectural Design Patterns

SmartLib uses a Client Server Architecture. The client consists of the camera detection system and the user interface. The client would connect to the firebase server where it holds our database. This allows the camera detection system to send post requests to the database to update and the client is able to send requests to retrieve information.



SmartLib user interface is using the Model-View-Controller (MVC) architectural design pattern. The Model layer is where it holds the user data and records for the usage from the user interface layer. The View layer is where it shows the components that display data from the Model layer to the user. Lastly, the Controller layer is where the components of the application that receive and carry out commands from the users to alter the View or Model layer. Following is the design:



3 Software Configuration Management Tools

This is where we will discuss version control management, and tracking on who made what changes and when.

3.1 Media Wiki

MediaWiki is a free and open-source application. This service is used as it is easy for beginners to pick up. There are many FAQs provided which can teach users the functions required by the users. There is a wide range of functions which allows users to create their information in different styles. It also allows users to concurrently edit the page at the same time. Hence, editing of the page will not result in a loss of information.

3.2 GitHub

GitHub is a source code hosting platform using the distributed version control and source code management Git. GitHub is chosen for its familiarity and support provided by various IDE applications. GitHub also supports issue tracking similar to a ticketing system. Whether it's a software bug, code enhancement or documentation, users can open an issue, label them appropriately and assign them for other team members to resolve. All users involved will receive timely updates on the progress of the issue.

3.3 Google Drive

Google Drive service is used as a file storage and for the backup of documents created. This service allows users to share and store files within the team. This service also allows users to edit documents concurrently. Changes made by different users will also be logged, allowing for version control.