Design Report on Software Maintainability

Version 1.0

By: Royce Ang Jia Jie, Manav Arora, Jovan Huang Tian Chun, Clarence Hong Shi Man, Zhu Weiji, Tan Hui Zhan

Document Change Record

Revision	**Description of Change**	**Approved by**	**Date**
0.10	Initial Template	Clarence Hong	13/03/2022
0.20	Add Design Strategies, Architectural Design Patterns, Software Configuration Management Tools	Jovan Huang Tian Chun	14/03/2022
0.30	Review and check Design document	Clarence Hong	23/03/2022
1.0	Final review of Design document	Manav Arora	28/03/2022

Contents

1. Design Strategies

1.1. The Planning Phase Before Development

Prior to starting on the development phase, we planned thoroughly by analyzing and predicting the kind of improvements we would have for the application after its release. When there is extensive use of our application in the future, we will need to scale the application up to be able to handle higher traffic rates. We also plan ahead by identifying complex features which we might need to spend more time on.

Since we have multiple developers for this project and this project needs to be continuously maintained and updated, we decided to make use of the Model-View-Control (MVC) as our architectural model. With MVC, we can change, delete and add functions with great ease without breaking other functions. MVC model also helps to separate internal representation of information from how information is presented to end users. This makes MVC an appropriate choice as having a good user experience is pivotal to the success of Friendstagram. MVC also helps to decouple components, hence allowing us to reuse code efficiently.

1.2. The Process of Developing

For our development, we are using the Waterfall Software Development Life Cycle Model. Each subteam will be handling each phase of the life cycle. The development will be divided into subtasks and each subtask will be handled by a subteam. Our main goal is to develop a minimum viable product (MVP). Due to safe distancing measures by the government of Singapore in response to COVID-19, all our development will be done remotely. Continuous feedback on design and usability of the application, User Acceptance Testing and all kinds of discussions will also be done as remotely as possible over online platforms like Zoom. Integration between front-end and back-end systems, will take place face-to-face as far as possible to facilitate clearer communication and minimal misunderstandings.

1.3. Correction by Nature

We will correct our application while testing the application. Here are some factors we will look out for:

1.3.1. Corrective Maintainability

Detecting errors through testing and having our client look at the software to correct any changes we need.

1.3.2. Preventive Maintainability

Every feature is implemented in a modular manner which allows us to test them independently and detect errors with great ease. We also standardized all our codes and ensured that all style guides are being followed.

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. It can also be easily adapted to changing business needs such as catering to other target audiences like National University of Singapore students.

1.4.2. Perfective Maintainability

Once the product is completed and delivered, we will look through the software to detect potential errors and correct them as early as possible so as to reduce maintenance costs.

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
- Separate modules with good modularity

2. Architectural Design Patterns

Friendstagram uses Model-View-Controller (MVC) architectural design pattern.

The Model layer is where all user data and business objects are contained. Examples include user profiles, the history of matches between users and the interests that are used to match users. It provides a single source of truth for data to be queried by the View and Controller layers. In the current implementation, the React Redux store together with the Django Rest API implement this layer.

The View layer is where the user interfaces with the Model layer. Here we have the components that display and modify data. The user, who is a Nanyang Technological University student, will interact with the User Interface (UI) components in the View layer.

Lastly, the Controller layer holds the components of the web application that receive and carry out actions corresponding to the commands from the users to modify the View or Model layers. Business logic such as form validation is located here.

The diagram below illustrates the MVC architectural design for Friendstagram:

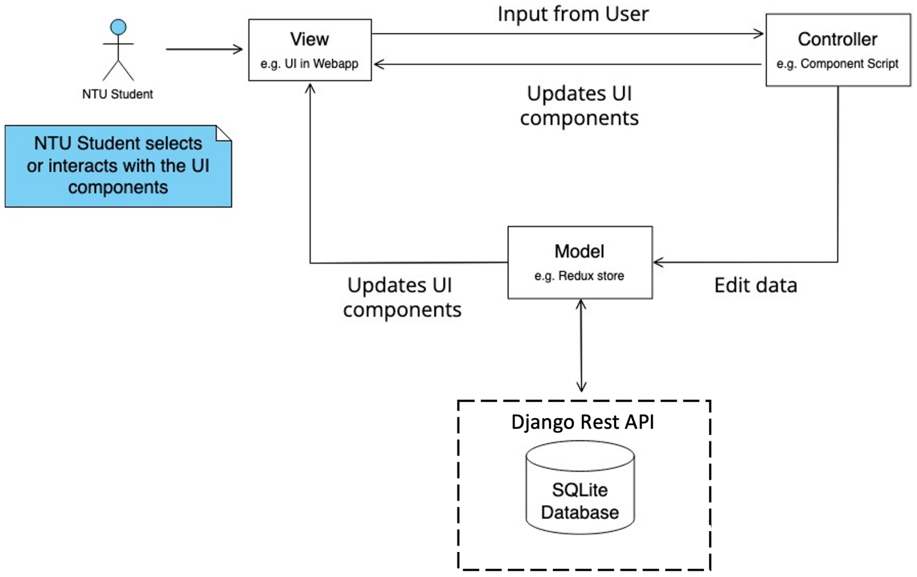

Fig 1. MVC Architectural Design of Frienstagram

The frontend and backend of the system follows a client-server architecture. The client and the server will be hosted on Netlify and PythonAnywhere. Communication between the client and server is made via HTTPS protocol using the RESTful API provided by the backend. Authentication and image storing will be done on Django.

The diagram below illustrates the Client-Server architectural design for Friendstagram's overall:

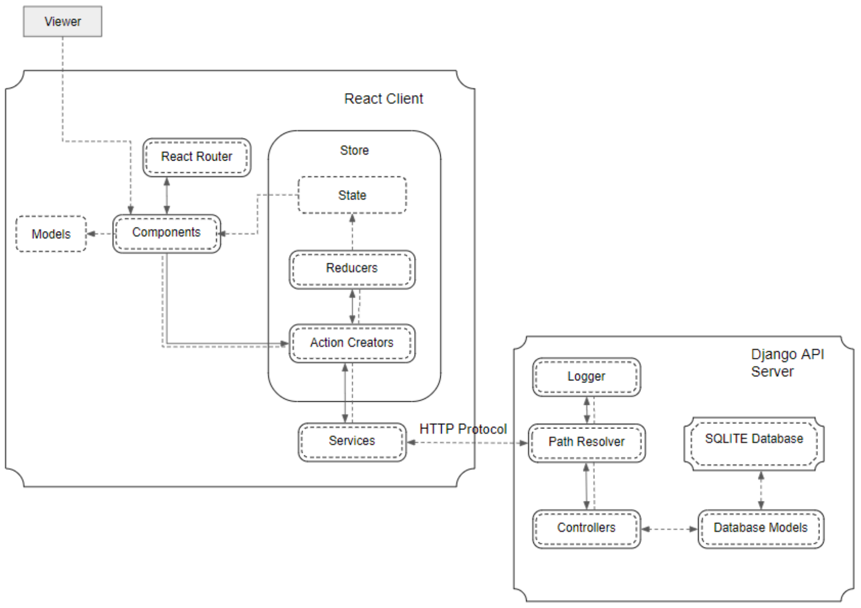

Fig 2. Client-Server Architectural Design of Friendstagram's overall system

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. GitHub

GitHub is a collaborative source code hosting platform that facilitates distributed version control and source management. We chose it as it is widely used and our development team is highly familiar with using it. In addition, GitHub is fully supported in the IDE (Visual Studio Code) used by our team. GitHub allows convenient and organized tracking of progress, issues and bugs. Everyone will also receive immediate updates about the progress of the issues / bugs.

3.2. Google Drive

Google Drive is a cloud-based storage solution that allows you to save files online. In our project, we choose to use Google Drive to store our documentation. This allows all documents stored in this platform to be accessed conveniently. It also allows us to have real time collaboration when making edits to our documents. Google Drive also supports version control, allowing us to keep track of our changes and revert changes when necessary.

3.3. MediaWiki

MediaWiki is a free and open-source application that allows us to present our documentation online. This application is well documented therefore allowing beginners to pick it up easily. A wide range of functions is provided in this application, allowing users to present our information in their ideal styling.

3.4 TortoiseSVN

TortoiseSVN is a Apache Subversion client, implemented as a Microsoft Windows shell extension. It is intuitive and easy to use and helps programmers to manage different versions of the source code for their programs. It is free software released under the GNU General Public License. It allows us to easily check out files and it provides the revision and author of the uploaded files. This provides traceability of documentations and uploaded files.