Design Report on Software Maintainability

Version 2.0

By: Anandarajan Sindini, Verma Nandini, Duong Ngoc Yen, Jim Sean, Kolady Anamika Martin

Document Change Record

Revision	Description of Change	Approved By	Date
1.0	Initial Version	Kolady Anamika Martin	18/10/2022
1.1	Add Design Strategies, Architectural Design Patterns, Software Configuration Management Tools	Kolady Anamika Martin	19/10/2022
2.0	Added System Architecture Design	Kolady Anamika Martin	26/10/2022

Contents

1 Design Strategies

1.1. The Planning Phase Before Development

Initiating the planning phase before the development, Foodify has been designed to be made extensible and scalable as the user base grows. Any improvements required in the future are anticipated and analysed earlier in the life cycle, to ensure a flexible design of the application. As the application traffic increases it will be scaled up to handle higher user requests.

Foodify is a web application for users to track their food items and encourage a reduction in food wastage by them. Our target audience are individuals in Singapore who have a difficult time keeping track of soon to be expired food items. Once the application becomes popular, we would be extending our application to users in other countries. This would cause our application to be scaled up to handle heavy user traffic. Hence, scalability is one of the factors we listed that needs to be investigated in the future.

We would also be adding new features such as a forum for users to share recipes made by them with their soon-to-be-expired food items. Hence we also considered Flexibility as one of the criteria while designing the system architecture. Flexibility and Scalability each promotes the other.

Considering the above constraints, we decided to develop Foodify based on Model-View-Controller Architecture. This system architecture allows the application to deploy and maintain components independently and separates user interface from business logic. This makes the application more flexible as any change to a component will not break the whole application.

1.2. The Process of Developing

The Foodify application is tested using a test-driven approach. To follow quality assurance, our team members have taken up the role of the tester. Continuous feedback is provided by them on the design and usability of the product which are considered for improving the software application.

During the development phase itself, the application continued to evolve, and the design went through several revisions throughout the software development lifecycle. Various kinds of

tests, reviews and usability analysis are used to ensure that the application has been designed to achieve high scalability, usability, and maintainability.

During the code phase, the various components of the application have been developed to be as generic and reusable as possible to ensure easy extensibility, and reusability. The use of single use application components allows for an easy maintenance of the application.

1.3. Correction by Nature

The feedback and errors received during testing needs to be corrected.

1.3.1. Corrective Maintainability

This approach is used to fix bugs reported from the users during testing.

1.3.2. Preventive Maintainability

Each feature is tested independently such that errors are detected easily to increase the life of the software application and make it more reliable.

1.4. Enhancement by Nature

The software needs to be updated based on the changes and new features that will be added from time to time. We will look out for:

1.4.1. Adaptive Maintainability

This allows the application to be flexible to update to these new changes.

1.4.2. Perfective Maintainability

The software maintenance that arises after the release of the product in the market will be taken care of by this approach.

1.5. Maintainability Practices

Software maintainability is defined as the ease with which a software system or component can be modified to correct faults, improve performance or other attributes, or adapt to a changed environment.

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

1.5.1 Readable Code

The requirements on which Foodify has been built is likely to have some modifications in the future. This is because Foodify will interact with the real world and new changes and requirements will be necessary. In these scenarios, having readable source code facilitates the understanding of the abstraction phases and hence eases the evolution of the codebase. Thus, code readability is a very significant part of Foodify to ensure high maintainability.

1.5.2 Version Control

Version Control allows to effectively track and control changes of a collection of related entities. It allows tracking and controlling changes to source code.

For the development of Foodify we have used the Git version control system, hosted on the web-based application GitHub. All the development code has been hosted on GitHub and all the changes, branches and various versions of the application are tracked, and controlled on the platform. For tracking and controlling all the documentation and various documents pertaining to various stages in the SDLC of Foodify, SVN has been used.

1.5.3 Standardised Documents

All documents for Foodify have been prepared using standardised Industry templates, guidelines and frameworks. This ensures that all prepared documentation items are standardised and can be tracked and maintained effectively through version control.

1.5.4 Modularity

NTUCollab has been developed using a Model-View-Controller (MVC) System Architecture. In MVC, different components of the application can be independently deployed and maintained which provides us with maximum modularity. This allows for easy reuse and replacement of components in the application. Changes and updates made in the future can be effectively maintained because of the modularity in the application design.

2 Architectural Design Patterns

Foodify is using the Model-View-Controller (MVC) System Architecture Design Pattern.
The MVC divides the application into three logical parts:

- **Model**: Component which is responsible for maintaining data. It handles data logically and is connected to the database.
- **View**: This component is used for data representation. It generates the user interface (UI) for the user.
- **Controller**: This component enables interconnection between the views and the model, so it acts as an intermediary. After receiving data from the model, it processes it and sends the information to the view.

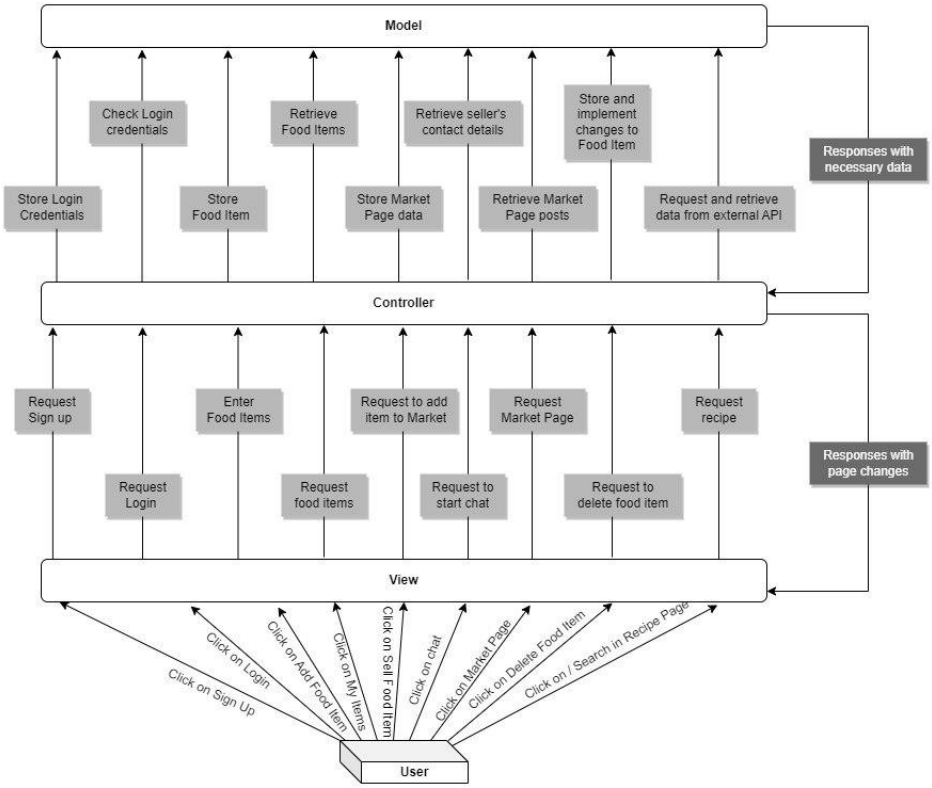

Figure 1: System Architecture Design

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. MediaWiki

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 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 concurrently edit documents. Logs are maintained for all the changes made by users for version control.