VOTAFUN

DESIGN REPORT ON SOFTWARE MAINTAINABILITY

Version 1.1
25/10/2023

Version History

Version #	Implemented By	Revision Date	Approved By	Approval Date	Reason
1.0	Ng Yue Jie Alphaeus	12/10/2023	Ng Yue Jie Alphaeus	12/10/2023	Initial Design report on software maintainability draft
1.1	Roy Lau Run-Xuan	13/10/2023	Ng Yue Jie Alphaeus	25/10/2023	Design strategies added Architectural design patterns added Software configuration management added

Table Of Contents

1. Design Strategies

In this section, we will discuss the design strategies used by our team when developing VotaFun.

1.1. The Planning Phase Before Development

Our team believes the planning stage is the most important and should be as detailed as possible before implementation. The project manager constantly engages with the client to ensure that all requirements are captured, and are as detailed as possible. The project manager will also raise any ambiguities and discrepancies immediately and clarify with the client.

Once the team has captured all requirements, the team will start designing the software system. Use case models, entity relational (ER) diagrams, system requirements specifications (SRS) and the system architecture are developed to show the different components of VotaFun and how they interact with each other. Non-functional requirements such as reliability, performance, maintainability, scalability and usability are kept in mind while the team designs the VotaFun.

Besides, the team performs wireframing to show the frontend mockup and how VotaFun user interface will look.

1.2. The Process of Developing

During the implementation phase, the developer team will reference the documentation in the planning phase to build VotaFun. Our team uses the Agile development methodology and has sprint cycles of one week. Developers are split to work on either the frontend or the backend components.

As the developers build VotaFun, feedback is continually gathered from the client and incorporated into VotaFun. Changes are reflected in the documentation, ensuring that it is always up-to-date.

1.3. Correction by Nature

The following elaborates on how the developer team prevents and corrects bugs and defects in the system while developing VotaFun.

1.3.1. Corrective Maintainability

The developer team will fix bugs found during testing or after product delivery. The testing can be unit, integration, system or user acceptance tests. Users can also raise bugs found while using VotaFun.

Once the developer has fixed the bugs, the developer will perform validation to ensure that the developer has fixed the bug and that the developer has not introduced new bugs.

1.3.2. Preventive Maintainability

Clear documentation, use of a version control system and following coding standards are ways the developer team works towards preventive maintainability.

All these allow the developer to understand the components built and can minimise the chances of developers introducing bugs into the system. If the clients or developers find bugs in VotaFun, the developer can easily track changes made and allow them to pinpoint the issue that led to the bug.

1.4. Enhancement by Nature

The following shows the avenues where the developer team works to ensure that VotaFun remains easy to maintain and adds enhancements.

1.4.1. Adaptive Maintainability

Due to the clients changing requirements, there may be requests for additional support for installation on various hardware and operating systems. The developer team will ensure portability while designing VotaFun.

Additionally, the developer team has designed VotaFun to ensure that new changes can be incorporated after product delivery and allow VotaFun to be up-to-date.

1.4.2. Perfective Maintainability

After the initial delivery of the product, the developer team will work towards maintaining and perfecting VotaFun based on feedback. If needed, the developer team will perform code refactoring to increase the performance of VotaFun or remove redundant features. Thus, this ensures that the system is efficient.

1.5. Maintainability Practices

Good maintainability practices are paramount to ensure that the software remains sustainable and maintainable for modification or extension. Our team has implemented these maintainability practices while building VotaFun.

1.5.1. Modular Design

The developer team has broken down VotaFun into smaller independent components like the sockets, ChatGPT, database, game lobby and game room. A modular design ensures that software components are loosely coupled, and the developer team can change individual software components without affecting the entire code base. Developers can also test individual components independently before any integration.

1.5.2. Clean Code and Coding Standards

Our team utilises pre-commit hooks that run before any code can be committed to Git. The frontend pre-commit hooks use lint-staged, a linter and Husky, a code formatter. The backend uses black, a code formatter and flake8 to enforce coding styles.

These pre-commit hooks ensure that all developers' code will be in a standardised format. These formats include enforcing indentation, spacing, and checking for unused imports. The code is more readable and maintains a certain quality and consistency.

1.5.3. Documentation

The developer team will draft all design plans before any implementation begins. Design plans include SRS, system architecture, and ER diagrams, which give the developer an overview of how the system should look. When changes are needed when implementing VotaFun, the developer team will update the corresponding documentation immediately to ensure consistency of information.

Maintaining up-to-date documentation will allow everyone in the team to understand the system that the team has built. Future modifications will reference these documents, and personnel can know what components might be affected if there are changes. Bugs are less likely to be introduced, and developers can understand the implications of any changes. Thus, all these contribute to a maintainable system.

1.5.4. Version Control

The team has chosen Git as the version control system for VotaFun. With Git, the developer team can create new features in different branches, test features, and submit pull requests to merge branches. In Git, all changes are recorded, such as developers creating branches or when they push code. If any issues arise, it is possible to revert to an earlier baseline version, maintaining the integrity and accountability of the project.

Thus, the use of Git contributes to ensuring the integrity of the code and ensures that the code base remains maintainable over time.

2. Architectural Design Patterns

Our team has chosen the model-view-controller (MVC) design pattern for VotaFun.

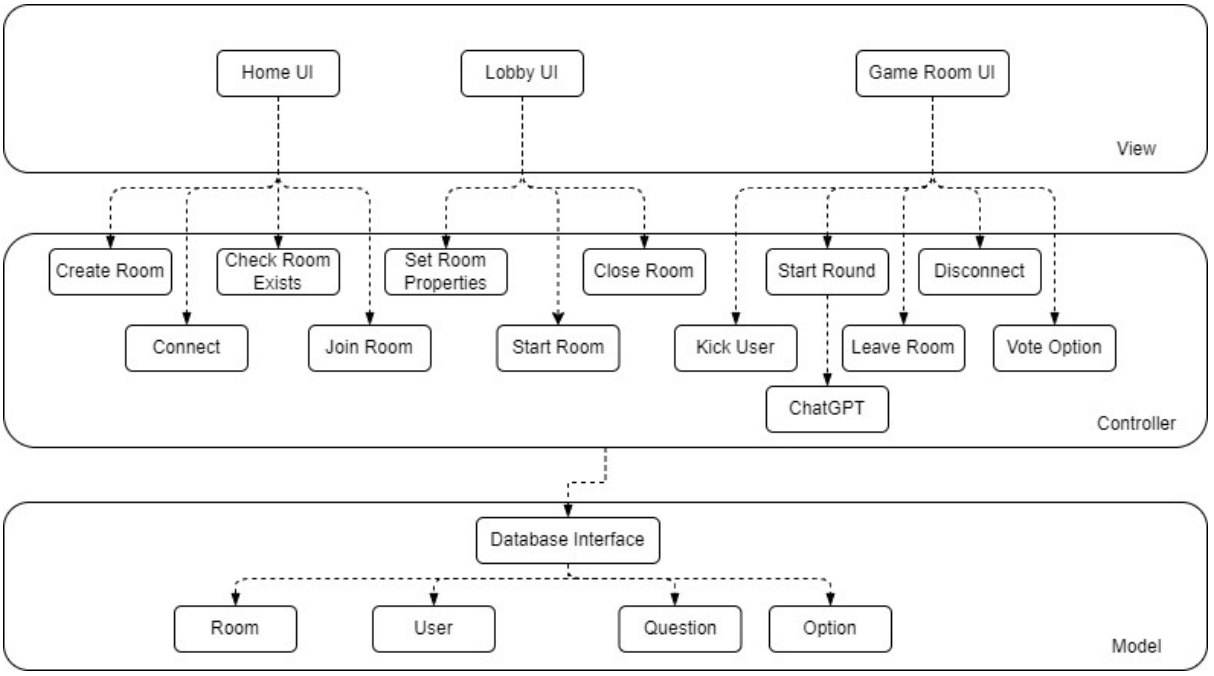
The database interface represents the model responsible for retrieving, storing and manipulating application data.

The user interface is the view responsible for presenting information to the user. The view renders data from the model and passes user interactions to the controller for processing.

Lastly, the sockets represent the controller and act as an intermediary between the model and the view. It holds the business logic and is responsible for processing any data received from either the view or the model.

The following image shows the system architecture of VotaFun.

Figure 1: System architecture design of Votafun



3. Software Configuration Management Tools

Our team will use the following tools for version control management.

3.1. MediaWiki

MediaWiki is a free and open-source, wiki software platform developed for Wikipedia. MediaWiki allows for collaborative work where users can create and edit content, track changes made in pages, roll back any changes made and allow configuration of access controls.

These key features allow the team to use MediaWiki as a repository for all documentation. Essential documents like the project plan, software requirements specifications (SRS), release plan, and so on are all checked into our team MediaWiki. Thus, MediaWiki allows the team to create a knowledge base for the entire project.

3.2. GitHub

GitHub is a web-based platform that allows version control during software development using Git. Git is a version control system that allows developers to collaborate on the same code base. GitHub also allows project tasks and issues to be tracked and assigned to different team members.

Using GitHub, the developer team can develop various features of VotaFun independently in distinct branches and test features before merging them into the main branch. Issues and bugs are tracked and assigned to members responsible for resolving them. Thus, GitHub provides a platform for our team to build VotaFun incrementally.

3.3. Google Drive

Google Drive is a cloud-based file storage and collaborative platform. Using Google Drive, users can store files such as Word documents, videos, and PowerPoint slides online and make them accessible on any device. Google Drive also features real-time collaboration, and multiple users can work on a document with changes synchronised between all users.

All these features of Google Drive allow the team to collaborate on different documentation in the project. Team members can access and work on the documentation at any time. Google Drive also tracks changes made, thus allowing for any rollback if needed.