Design Report on Software Maintainability

Version 1.0

By: Mishra Apurva, Gambhir Dhruv, Nithya Hariharan, Mehta Viral Sujal, Samanatha Tan, Najah Ismail, Tan Jing Jie

Document Change Record

Revision	Description of Change	Approved by	Date
0.10	Initial Template	Mishra Apurva	12/10/2024
1.00	Revised Edition	Mishra Apurva	21/10/2024

Content

1. Design Strategies **4**
1.1. The Planning Phase Before Development 4
1.2. The Process of Developing 4
1.3. Correction by Nature 4
1.4. Enhancement by Nature 4
2. Architectural Design Pattern **6**
3. Software Configuration Management Tools **6**
3.1. Wiki 6
3.2. Git and GitHub 7
3.3. Onedrive 7

1. Design Strategies

1.1. The Planning Phase Before Development

PetPal's concept is unique compared to common websites. While brainstorming this new idea, the team decided which features were essential for the project's duration and which would be lower priority for future improvements. As a user networking platform, it is expected that existing users will invite more to join. Thus, in the production version of this web app, scalability becomes an important consideration.

Since PetPal prioritizes user experience, the client-server architecture is adopted, as it is widely used in web applications.
The following design features were integrated to ensure maintainability:

- The frontend is designed to allow the addition of future features, e.g., extra tabs in the navigation pane.
- The database is structured for easy modification as the application evolves.

1.2. The Process of Developing

- The team divided application features such as pet sitting, events, adoption, database setup, and data creation.
- Each feature was built in iterations and continuously integrated with local and cross-testing between team members.
- Similar to an Agile team, where every developer is expected to test, the PetPal team practices continuous testing.
- Issues are flagged through group communication or raised during team meetings to be addressed in a timely manner

1.3. Correction by Nature

We will correct our application while testing the application. And 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.

1.3.3. Cohesive Maintainability

Features are tested after integration to ensure smooth running.

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 new features and a variety of data.

1.4.2. Perfective Maintainability

After every feature is delivered, 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 with comments
- Version Control
- Meaningful names for commits and branches
- Standardized Documentation
    - Documentation created later is consistent with previous
- Modularity of different components

2. Architectural Design Pattern

PetPal is using the client-server architectural design pattern. Here is the breakdown:
Client (Next.js frontend with Firebase authentication):

- User interface for browsing pets, events, etc.
- Sends API requests to the server (Flask API).
- Handles user authentication using Firebase.

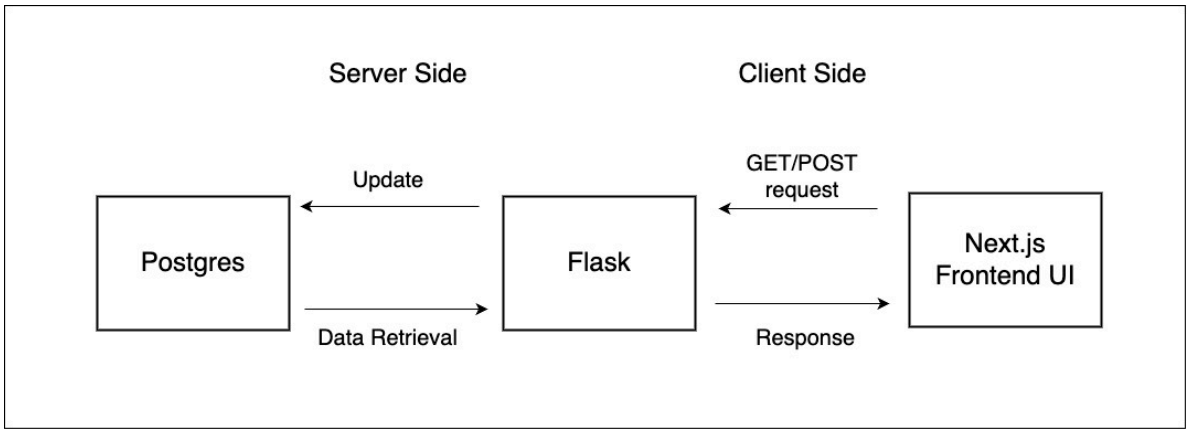
Server (Flask):

- Receives client requests (e.g., request for pet details).
- Executes business logic (e.g., filtering search results, user authorization).
- Interacts with the PostgreSQL database to fetch or update data.
- Sends the response back to the client.

Database (PostgreSQL):

- Stores user data, event details, pet adoption records, etc.
- Serves data to the Flask API based on client request

The architecture design is:



3. Software Configuration Management Tools

This section discusses version control management, which includes tracking of the who, what, and when of each change made.

3.1. Wiki

Wiki is used for the record keeping of documentation. It is beginner-friendly, and uses a similar approach as commercial software such as Confluence. It is intuitive to upload, update and see change history of documents. Formatting and customisation is also possible. Users may concurrently edit the page.

There is negligible risk of overwriting information. The same documentation is seen by the whole team, which allows the team to be on the same page. When someone is new to a part of the project component, they may view the history of documentation to gain a better understanding.

3.2. Git and GitHub

Git is the most familiar version control tool. GitHub is the cloud based hosting platform using Git. Git and GitHub are user-friendly, and can be readily integrated with other development tools such as Visual Studio Code. Git allows the project development progress and the different versions of the code to be tracked in a meaningful way. It supports the collaboration of PetPal team members, by allowing them to be working on the same code without the risk of overwriting.

3.3. OneDrive

OneDrive service is used to create and store documentation. This service allows users to share and store files within the group easily. Furthermore, OneDrive allows users to edit documents using the full functionality of Microsoft Office desktop applications. This service allows users to edit documents concurrently and allows the user to check the document history; this is helpful in case of unintended changes.