ST10163788 **PEACE SALOMY PHIRI**

SOEN6222 PART 2

SOFTWARE ENGINEERING

OCTOBER 25, 2024

IIEMSA RUIMSIG CAMPUS

POE PART 2

Please note that all corrected and new information will be written in red

Question 1 – Architecture Pattern

For the ***Helping Hands*** software solution, the most suitable architecture pattern is ***the Model-View-Controller (MVC)*** architecture.

How the MVC Architecture Works:

Model:

The ***Model*** represents the data and business logic of the application. It handles the data retrieval and updates from the database or external data sources, and processes it according to business rules. For the Helping Hands system, the Model would manage information such as users (volunteers, staff, beneficiaries), resource inventory, program outcomes, and donations (Kumar, 2021).

View:

The ***View*** is responsible for presenting the data to the user. It is the UI layer that displays the results processed by the Model. For Helping Hands, the View will include user-facing pages where volunteers can register, log in, and view program details, while staff members can track resource distribution and program outcomes.

Controller:

The ***Controller*** acts as the intermediary between the Model and the View. It handles user inputs, interacts with the Model to retrieve or manipulate data, and updates the View accordingly. In the Helping Hands system, the Controller would manage inputs like user registration, resource tracking, event scheduling, and the processing of donations (freeCodeCamp.org, 2021).

Why MVC is Chosen:

1. **Clear Separation of Concerns (GeeksforGeeks, 2022):**

MVC enforces a clean separation between data, user interface, and business logic, which is crucial for the ***Helping Hands system***. This separation makes it easier to maintain and extend the system over time, especially as the organization grows and new features or modules are added. Volunteers, staff, and beneficiaries all have distinct interactions with the system, and the separation provided by MVC ensures that changes to one part (e.g., the UI for volunteers) won't disrupt others.

2. **Scalability and Flexibility (Anushka, 2021):**

The MVC architecture allows the Helping Hands website to scale easily. The separation of the Model, View, and Controller layers makes it possible to modify or add new functionality to one layer without affecting others. For example, if the organization decides to add new data management processes , these can be added within the Model without requiring changes to the View or Controller.

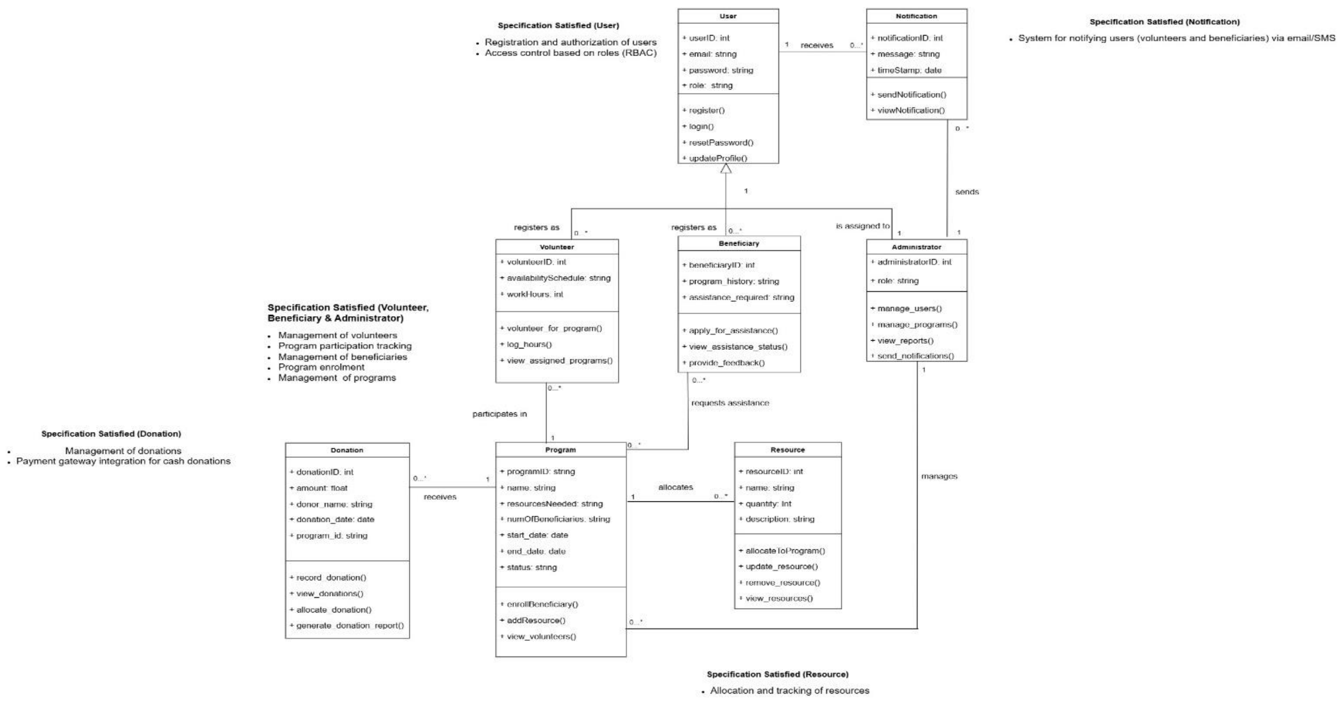
3. **Testability (Anushka, 2021):**

MVC enhances the testability of an application by allowing each component; Model, View, and Controller; to be tested separately. This facilitates complete testing of every function. For a system like Helping Hands, where maintaining data integrity is crucial for tasks such as tracking resources and monitoring program outcomes, this feature is essential.

4. **Reusability:**

MVC promotes code reusability. For instance, the Model components that handle donations or monitor resources can be utilized across various interfaces. This facilitates future developments, such as extending the Helping Hands platform to other devices, by allowing them to utilize existing business logic.

Question 2 UML Class Diagram



Requirements Satisfied by the Diagram

Registration and Authorization of Users:

The User class provides the operations register, login, and resetPassword, ensuring that users can create an account, log in, and reset passwords, while validation of email and password strength can be handled in the register method.

Access Control Based on Roles (RBAC):

Role-based inheritance is evident with Volunteer, Beneficiary, and Administrator inheriting from User. Each role has specific operations, ensuring that volunteers can register for programs, administrators can manage the system, and beneficiaries can request assistance.

Management of Programs:

The Administrator has an operation manageProgram. This satisfies the requirement to manage and create programs.

Management of Volunteers:

The Volunteer class contains the log_hours operation, allowing volunteers to track their work hours. While the Volunteer and Administrator classes are not directly connected, the Administrator class can manage volunteers via the shared User class, where the role-based access control (RBAC) ensures that administrators have the authority to view and manage volunteer activities. This satisfies the system's requirement for managing volunteers.

Management of Beneficiaries:

The Beneficiary class allows beneficiaries to enrol in programs and track their program history, ensuring the management of beneficiary information.

Management of Donations:

The Donation class includes the allocate_donation operation to manage donations, both cash and non-cash, satisfying the donation management requirement.

System for Notifying:

The Notification class handles sending email and SMS notifications to users for critical updates, such as program status or volunteer shifts.

Reporting Problems and Providing Feedback:

Feedback reporting can be represented as part of the User class operations, allowing volunteers and beneficiaries to submit issues or feedback.

Appendix: Feedback Addressed

No changes were made to the document because no feedback was provided. However, I carefully reviewed the document to make sure it meets all the requirements and is clear and complete. Here's how I ensured its quality:

Requirements Are Covered: The architecture (MVC) and UML class diagram were checked to confirm that they include everything needed especially the UML diagram, like user registration, role management, program tracking, donations, and feedback.

Clear Explanations: The explanations of the MVC pattern were reviewed to make sure they are easy to understand and show how it fits the Helping Hands system.

Neat and Well-organised: The document is well-organized and includes references to reliable sources to support the information provided.

Since no feedback was received, the document was resubmitted as it is.

References

GeeksforGeeks. (2022). *MVC Framework Introduction*. [online] Available at: https://www.geeksforgeeks.org/mvc-framework-introduction/.

Kumar, N. (2021). *How the Model View Controller Architecture Works – MVC Explained*. [online] freeCodeCamp.org. Available at: https://www.freecodecamp.org/news/model-view-architecture/.

freeCodeCamp.org. (2021). *MVC Architecture – What is a Model View Controller Framework?* [online] Available at: https://www.freecodecamp.org/news/mvc-architecture-what-is-a-model-view-controller-framework/.

Anushka, V. (2021). *Benefit of using MVC*. [online] GeeksforGeeks. Available at: https://www.geeksforgeeks.org/benefit-of-using-mvc/.