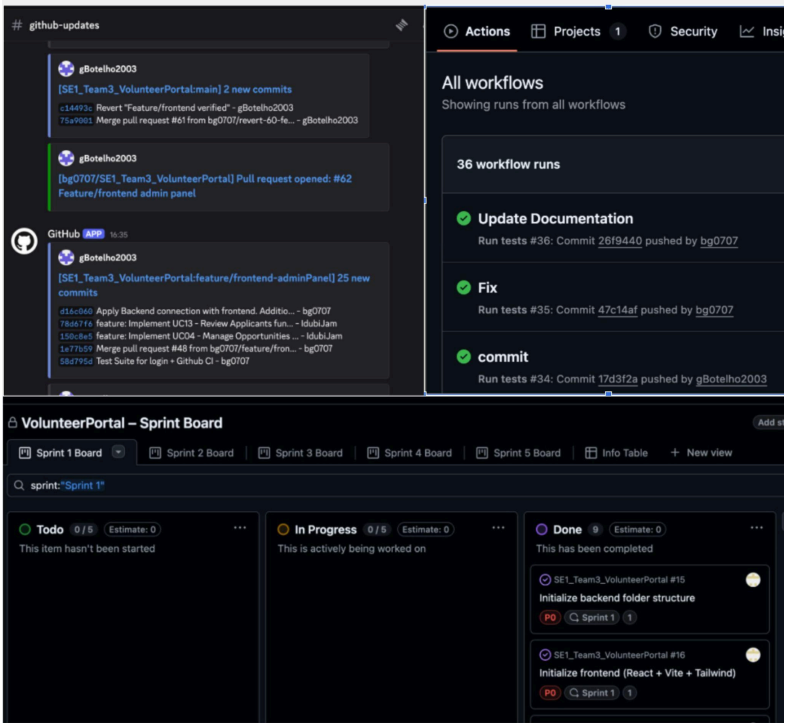
Project Workflow, Engineering Practices & Learning Reflection

Throughout the entire process of this project, we have practiced the key concepts of SE1 in action, including requirements prioritization, collaborative work in an agile fashion, domain modeling, architectural modularization, test-driven thinking, and CI-supported development. The final system not only achieves completeness in functionality but also reflects a process of software engineering in accordance with the underlying principles.

1. Development Approach

We utilized an Agile-inspired Kanban system with GitHub Projects as a toolset. We broke down development into small increments that allowed us to have continuous development, review, and integration. We implemented each feature or bug fix on a separate branch, merging it with our main branch via a pull request. This ensured code quality. We utilized Discord integration as a tool that allowed real-time notifications, enabling us to stay synchronized as a team.



2. Architecture & Project Structure

The system is designed using a client-server architecture, which is in line with the principles of modularity and separation of concerns taught in the course. The backend part of the system offers RESTful API services, while the frontend part consumes these services.This design enables scalability, maintainability, and separation of concerns between the system components.

3. Requirements Execution & Backlog Management

It is designed using the client-server architecture, which aligns with the course's principles of modularity and separation of concerns.The backend portion of the system provides RESTful API services, while the frontend portion consumes the services. This design allows for scalability, maintainability, and separation of concerns for the components of the system.

4. Testing & Quality Assurance

Our testing strategy applied concepts from **Lecture 10 (Testing I)** and **Lecture 11 (Testing II)**:

Unit Testing (Jest)

- Tests were structured using the **Arrange–Act–Assert** pattern.
- Core business logic was validated through isolated service tests.

API Testing (Postman)

- Endpoints, status codes, and error responses were verified manually and through scripted Postman tests.
- This complemented unit testing by validating integration behaviour.

Continuous Integration

- **GitHub Actions** automatically executed the full test suite on each commit and pull request, ensuring stable integration and preventing regressions.

5. Conclusion

Throughout the project, the key concepts of the SE1 course were put into practice, including agile task management, requirement prioritization, architectural modularization, version control,

testing methodologies, and CI-supported developments. The developed system is not only functionally complete, but it has also been developed through a structured engineering process in alignment with the concepts presented in the course.