Software Implementation and Testing Document

- Increment 2 -
Group #2

Authors :

Daivion Brooks
Brooks Berry
Jason Lee
Nadav Peper
Nandagopal Nair

1. Programming Languages (5 points)

***TypeScript (Frontend**): Used with React to build the user interface. We chose TypeScript for its type safety and ability to reduce runtime errors, making development more manageable for a group project.*

***Python (Backend):** Used with Flask to build the API. Python was chosen because it is beginner-friendly, flexible, and integrates well with Flask for web application development.*

***SQL (Database):** Used mainly through SQLAlchemy to manage the Postgres database and handle queries. SQLAlchemy made it easier to work with structured data and store user, symptom, and treatment information.*

2. Platforms, APIs, Databases, and other technologies used (5 points)

***React** : JavaScript/TypeScript library used for building the frontend. Additional libraries like react-hot-toast are also used to aid in development.*

***Flask** : Lightweight Python framework used for the backend and API development.*

***PostgreSQL** : Relational database used for persistent storage of user accounts, symptoms, and treatments.*

***Docker** : Used for containerization to ensure consistent environments across team members' machines and deployment.*

***JWT (JSON Web Tokens)** : Used for secure authentication and session management.*

***Marshmallow:** Used for serialization and validation of data in the backend.*

***Postman** : Used for testing backend API routes during development.*

***GitHub** : Used for version control and collaboration between team members.*

3. Execution-based Functional Testing (10 points)

During this increment, limited functional testing was performed since most of our time was focused on preparation and smaller updates rather than major feature development. While we did not complete extensive new tests, we plan to continue using Postman to validate API endpoints such as registration, login, and symptom tracking in the next phase. Independent tests were done for each minor bug fix and feature.

4. Execution-based Non-Functional Testing (10 points)

Performance: *Measured response times using Postman and browser developer tools. All API responses returned within 1–2 seconds, meeting our performance goals.*

Security: *Verify that the encryption is implemented properly as intended in the project.*

5. Non-Execution-based Testing (10 points)

Code Reviews: *Team members reviewed pull requests on GitHub before merging to ensure code quality and reduce errors. All increment 1 documentation was reviewed, and bugs were resolved in increment 2 issues.*

Walkthroughs: *During meetings, we walked through key issues to address for increment 2 and 3*

Static Checking: *TypeScript's compiler caught many potential type errors in the frontend. Python linters and IDE warnings helped us maintain backend code quality.*

Manual inspection: *Team members periodically inspected the codebase to catch typos, syntax issues, and simple mistakes, and refactored parts of the code where improvements were needed.*