# Software Implementation and Testing Document

# For

# Group 1

Version 2.0

**Authors**:
Andrew Augustine
Anthony Ingle
Tristan Morris
Dawson Stoller
Cole Warren

## 1. Programming Languages (5 points)

- Python
    - Usage: Backend
    - Reasoning: Python (+ it's libraries) are easy for everyone to pick up, meaning less time spent training.
- Typescript
    - Usage: Frontend
    - Reasoning: Standard frontend designing language. Has type safety compared to Javascript.
- Bash
    - Usage: Inside Docker Images
    - Reasoning: Image bootstrapping.

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

- Flask - Backend HTTP Framework
- Pytest - Python testing library
- Solidjs - Frontend Javascript Framework
- Docker - Container runtime; used in development and production server
- MariaDB - Database of choice; Integrates with backend to store user data
- DigitalOcean - VPS that hosts our production server
- Doppler - Shared secrets manager
- Github - Shared Codebase
- Auth0 - Authentication service provider
- Nginx - Web server used in production to serve frontend
- Gunicorn - Web server used in production to serve backend
- Stripe - API for payment handling
- Figma - Mockup tool for designing frontend views

## 3. Execution-based Functional Testing (10 points)

We used testing libraries to make sure that a lot of our functional requirements gave the proper output or result that we expected.

For the backend, we used Pytest alongside Flask's testing suite to ensure that API responses were proper. We're also planning on expanding this to include authentication verification with Auth0.

For the frontend, we used Vitest alongside a custom testing library from SolidJS to ensure that the DOM responds properly when we click on certain buttons, and that the frontend can parse all kinds of responses from the API.

At the moment, we don't have any tools set up to automate integration testing, and we do the testing manually every release. It's a known flawed procedure. Call it a work in progress.

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

The only execution-based non-functional testing we do right now is UX testing. Every release, we'll spin up a mock version of the app and make sure the frontend is intuitive and responsive. We also test that the backend responds in a reasonable amount of time.

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

For this project, we've done lots of pair programming and all code that is contributed to the repo must be reviewed by at least one other person before being allowed to merge upstream.

We haven't done any walkthroughs yet, but those are planned.