Detailed Test Plan

- How test cases of different levels (frontend, backend units, integration) are organized.

For the design of a test case, a set of test cases for each item will be created for each level. Each Test case specifies how the implementation is tested, how we know it is successful and the expected response. The backend will be tested first using unit testing, then the front end will be tested using mocking and selenium. The backend interacts with sqlite. Inside CI Python there is a directory called qa327_test which hold the various testing files. It's important to test the various aspects to the program such as fronted, backend, units, integration, system and acceptance. Each test will describe the following Test no, test type , test name, test purpose, test situation , expected output, actual output , outcome and actions required

- The order to the test cases (which level first which level second).

The frontend will be tested first separately from the back end. This will be one using mocking. We will test the connection to the we page, and then test the various different pages. It's imperative to test all the pages , /register, /login, /lougout , / and to test for functionality of the webpage. Next the backend will be tested for the functionality and the business logics. We will test all associated actions to be done to fcomplete a transaction, and the interaction of data models. Next we will test integration and that the back-end and front end work together seamlessly. Next we will test system and acceptance to check that we have all the specifications and that

- Techniques and tools used for testing.

  Selenium and pytest will be used to help automate testing. Testing techniques such as black box along with regression and failure test is appropriate here. In addition input coverage tests, output coverage test and functionality coverage test will be implemented. A hybrid method combining input partition and shotgun testing is a useful tool that will be implemented.

Environemnts (all the local environment and the cloud environment) for the testing.

Flask provides a way to test your application by exposing the Werkzeug test Client and handling the context locals for you. You can then use that with your favourite testing solution. Thus pytest will be used to test the backend. For the frontend selenium will be used to test the functionality of the web application.

- Responsibility (who is responsible for which test case, and in case of failure, who should you contact)

  The test cases will be split up among team members evenly. Each team member will be responsible to test and evaluate a certain test level or unit of code. In case of failure they would contact the team member responsible for developing the portion of code that's broken. If there is a major issue contact the group as whole to resolve it.

- Budget Management (you have limited CI action minutes, how to monitor, keep track and minimize unncessary cost)

  Due to limited CI actions minutes it's important to be well organized before testing. Organization will help illuminate unnecessary tests. Regression testing can be costly. Thus an approach to regression testing where one retest sections of code instead or prioritizes certain test is important. This would help minimize unnecessary costs and is more efficient than retesting all when the codebase has changed.