Note that you can refine your testing plan as the project development goes on. Keep the change log as follow:

Change Log

Version	Change Date	By	Description
1	February 27, 2024	Alborz, Bricz, Demessie, Gurman, Safran	Added features for creating different types of posts within the community, seeing posts, making friends within the community, creating/logging into accounts, editing login information and tests for each feature.

Introduction

Scope

Scope defines the features, functional or non-functional requirements of the software that **will be** tested.

Roles and Responsibilities

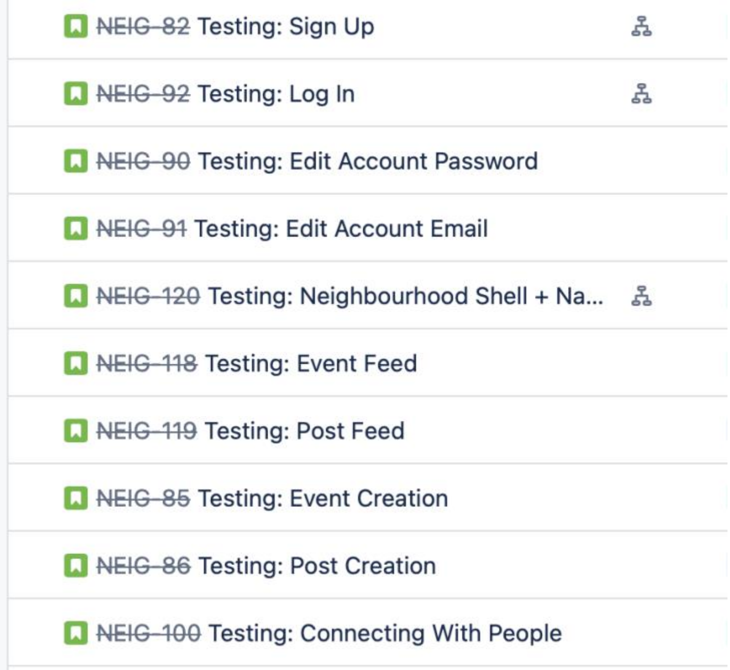
Name	Net ID	GitHub username	Role
Alborz	7882706	alborzk	Developers/QA Analyst
Bricz	7909956	br-cz	Developers/Configuration Manager
Demessie	7842385	DemISee1	QA Analyst
Gurman	7891379	GurmanToor	Developers/QA Analyst

Safran	7888632	sbinkader	QA Analyst

Test Methodology

Test Levels

The features/user stories, and the associated classes/methods contained within, we plan to test are:



~~NEIG-82~~ Testing: Sign Up

~~NEIG-92~~ Testing: Log In

~~NEIG-90~~ Testing: Edit Account Password

~~NEIG-91~~ Testing: Edit Account Email

~~NEIG-120~~ Testing: Neighbourhood Shell + Na...

~~NEIG-118~~ Testing: Event Feed

~~NEIG-119~~ Testing: Post Feed

~~NEIG-85~~ Testing: Event Creation

~~NEIG-86~~ Testing: Post Creation

~~NEIG-100~~ Testing: Connecting With People

We plan on executing various happy path testing (the feature works as expected under expected conditions), boundary conditions (testing the minimum or maximum inputs), error handling (the feature handles errors or unexpected inputs) and null and empty cases.

The acceptance criteria for all the tests is that each test must pass, independent of each other.

Unit Testing:

- We will use the Jest framework to do unit testing. Jest looks for a __tests__ directory in all folders, it then runs all the tests contained within that directory. Therefore, we plan to place the __test__ folder in the root directory and then running the *jest* command to run the unit tests for the specific component we wish to test.

Acceptance Testing:

- We will do end-user testing for these tests. We will have an arbitrary member go through each user story and see if the requirements are met for each feature.

Regression Testing:

- We will use jest to run all unit and integration tests contained within the repo by simply running *jest __tests__/testTypeFolder*, which looks for all __tests__ folders within the repository. Since we do not yet need integration tests, there exists no subfolder yet and are purely unit tests.

Test Completeness

The criteria that will deem your testing complete would be:

- 100% back-end code coverage (mandatory for this project), all the back-end source code should be covered by test cases.
- 100% front-end code coverage, where applicable

Resource & Environment Needs

Testing Tools

Make a list of Tools like

- Requirements Tracking Tool via **Jira**
- Bug Tracking Tool via **Jira**
- Automation Tool and Testing framework via **Jest**

Test Environment

It mentions the minimum **hardware** requirements that will be used to test the Application.

Example, following **software's** are required in addition to client-specific software.

- Windows 8 and above
- Travis-CI
- Jenkins
- Jmeter
- …

Hardware Requirements for running application under test:

CPU: Intel Core i5 or equivalent AMD processor, with at least 4 cores for efficient compilation and local testing.

RAM: Minimum of 8 GB, running development tools and testing environments smoothly.

Software Requirement:

- Modern operating systems, such as Windows 8
- Jest

- Some command line or equivalent application capable of running the tests

Terms/Acronyms

Make a mention of any terms or acronyms used in the project

TERM/ACRONYM	DEFINITION
API	Application Program Interface
AUT	Application Under Test
AWS	Amazon Web Services
CI	Continuous Integration
JS/TS	JavaScript/TypeScript
OIDC	OpenID Connect
IAM	Identity and Access Management