TEST PLAN FOR

FUJIJI

ChangeLog

Version	Change Date	By	Description
1.0.0	March 6, 2022	Wynona, Devin, Jishan, Asifur	Initial version for Sprint 2

1 Introduction

1.1 Scope

By the Sprint 2, the following features are going to be tested:
Functional requirements:

1. User profile/account creation:
    - New users should be able to create an account with an unregistered email.
    - Users should be able to sign in with their registered email and password. Otherwise, their sign-in would fail.
    - Users should be able to edit their profiles.
2. Posting ad listings
    - Only signed-in users can post a listing.

    - Users (either signed-in or not) can view posted listings, in which they must specify their location by city or province.
3. Listings filter/catalog/sort
    - Users should be able to filter listings by condition (i.e., used, new, or refurbished).
    - Users should be able to filter listings by category (i.e., Tables, Dressers, Chairs, etc).
    - Users should be able to filter listings by a price range.
    - Users should be able to sort listings when viewing them.

Non-functional requirements:

- Error messages should be clear and customized to indicate issues that a user or the server encountered.
- Load testing (Milestone 4)

1.2 Roles and Responsibilities

Name	Net ID	GitHub username	Role
Jishan Arora	ARORAJ	jishanarora	DBA, Developer
Asifur Rahman	RAHMANA2	asifrahman57	Developer
Devin Efendy	EFENDYD	devin-efendy	QA Analyst, Developer
Wynona Goldiella	GOLDIELW	wynonagcc	Developer

Every member is a developer because everyone is responsible to write code for their assigned tasks and create tests for their code. When there is a Pull Request, the developers other than the assignee are responsible for reviewing it before it is being approved to merge to the main branch. On top of that, Jishan acts as a DBA because he is the authorized admin for the tables in the production. Devin also acts as a QA Analyst to maintain the consistency and the quality of our code and software.

2 Test Methodology

2.1 Test Levels

Core Feature 1: Fujiji user profile/account

API:

```
POST /auth/signin
POST /auth/signup
```

Front-end:
`/signin`
`/signup`

Unit Test

- Ensure that a new user with a unique email can sign up by successfully getting HTTP response 200 after sending a request with a valid body.
- Ensure that a user with a registered email can't sign up again by expectedly getting HTTP response 400 after sending a request with an existing email in the test DB.
- Ensure that the server returns an error when the payload is empty by expectedly getting HTTP response 400 after sending such request.
- Ensure that the server returns an error when the payload is invalid by expectedly getting HTTP response 400 after sending such request.
- Ensure that a user with a registered email and password can sign in by successfully getting HTTP response 200 after sending a request with existing user information in the test DB.
- Ensure that a user with a registered email and wrong password can't sign in by expectedly getting HTTP response 401 after sending a request with a different password than the one stored in the test DB.
- Ensure that a user with an unregistered email can't sign in by expectedly getting HTTP response 400 after sending a request with a non-existent email in the test DB.
- React re-usable component should show correct interfaces for sign in and sign up states
- Should show appropriate error messages in the user interface when name, email, password, phone-number fields are empty
- Should show appropriate error messages in the user interface when user submit invalid email, password, and phone-number format

Integration Test

- Sign in by expectedly getting HTTP response 401 after sending a request with a different password than the one stored in the test DB.
- Ensure that a user with an unregistered email can't sign in by expectedly getting HTTP response 400 after sending a request with a non-existent email in the test DB.
- Successful request to /auth/signup/ or /auth/signin will store user data in web localstorage and be redirected to homepage.
- Should show appropriate error messages in the user interface after fujiji API return an error because existing email used for signup.
- Should show appropriate error messages in the user interface after fujiji API return an error because invalid password submitted for an existing user.
- If a signed user visit the signin/signup page they should be redirected to homepage.

Core Feature 2: Posting ad listings

API:
`POST /listing/`
`GET /listing/[id]`

Front-end:

```
/listing/create
/listing/[id]
```

Unit Test

- Ensure that users can get all listings in the specified city by successfully getting HTTP response 200 after sending the request.
- Ensure that users can get all listings in the specified province code by successfully getting HTTP response 200 after sending the request.
- Ensure that the server returns an error when a user didn't specify a city or a province code by successfully getting HTTP response 400 after sending the request.
- Ensure that the server returns an HTTP response 404 when the specified city or province code doesn't have any listing.
- React re-usable component should show correct form for create listing and update listing state
- Should show appropriate error messages in the user interface when one of the field in the forms are empty
- Should show appropriate error messages in the user interface when user uploaded incorrect file format for listing image
- Should show appropriate confirmation dialog when discarding form input changes
- Should show appropriate confirmation dialog when deleting a posting (in update form)
- Should successfully submit the form if all input fields are valid

Integration Test

- Ensure that a user can post a listing when they are signed in (i.e., have a valid token and user ID) by successfully getting HTTP response 200 after sending the request.
- Ensure that a user can't post a listing when they are not signed in by successfully getting HTTP response 400 after sending the request.
- Ensure that a user can edit a listing when they are signed in (i.e., have a valid token and user ID) by successfully getting HTTP response 200 after sending the request.
- Ensure that a user can't edit a listing when they are not signed in by successfully getting HTTP response 400 after sending the request.
- Ensure that a user can delete a listing when they are signed in (i.e., have a valid token and user ID) by successfully getting HTTP response 200 after sending the request.
- Ensure that a user can't delete a listing when they are not signed in by successfully getting HTTP response 400 after sending the request.
- If a user is not signed they can't visit /listing/create to create an ad listing and will be shown Unauthorized page and presented with a link to Sign In
- If a user is not signed they can't visit /listing/[id] page to edit an ad listing and will be shown Unauthorized page and presented with a link to Sign In

Core Feature 3: Listings filter/catalog/sort

API:

```
GET /listing/[id]
```

Unit Test

- Ensure that users can get all listings in the specified city and specified category by successfully getting HTTP response 200 after sending the request.

- Ensure that users can get all listings in the specified city and specified condition by successfully getting HTTP response 200 after sending the request.
- Ensure that users can get all listings in the specified city and specified price range by successfully getting HTTP response 200 after sending the request.
- Ensure that users can get all listings in the specified province code and specified category by successfully getting HTTP response 200 after sending the request.
- Ensure that users can get all listings in the specified province code and specified condition by successfully getting HTTP response 200 after sending the request.
- Ensure that users can get all listings in the specified province code and specified price range by successfully getting HTTP response 200 after sending the request.
- Ensure that the server returns an error when the specified price range is invalid by successfully getting HTTP response 400 after sending the request.
- Ensure that the server returns an HTTP response 404 when the specified location and the specified category don't have any listing.
- Ensure that the server returns an HTTP response 404 when the specified location and the specified condition don't have any listing.
- Ensure that the server returns an HTTP response 404 when the specified location and the specified price range don't have any listing.

Integration Test

- Ensure that users can get all listings in the specified city and specified category by successfully getting HTTP response 200 after sending the request.

2.2 Test Completeness

Criteria that will deem our testing complete:

- At least 80% test coverage.
- All Manual & Automated Test cases executed.
- All open bugs are fixed or will be fixed in next release.
- Successful tests on merging to the main branch.

3 Resource & Environment Needs

3.1 Testing Tools

- Requirements Tracking Tool: GitHub
- Bug Tracking Tool: GitHub
- Automation Tools: Azure, Husky
- Backend Testing Tools: Jest, Supertest
- Frontend Testing Tools: Jest, React Testing Library
- Other: Postman

3.2 Test Environment

The minimum **hardware** requirements that will be used to test the Application:

- Laptop with Windows 10 or above. MacOS 11.6 or above
- At least 4 GB of RAM
- At least dual-core CPUs

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

- Node version 16.0.0
- Docker
- Modern web browser supporting ES6

4 Terms/Acronyms

TERM/ACRONYM	DEFINITION
API	Application Program Interface
AUT	Application Under Test
QA	Quality Assurance
DBA	Database Administrator
DB	Database
HTTP	Hypertext Transfer Protocol
GB	Gigabyte
RAM	Random-access Memory
CPU	Central Processing Unit