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
1.1.0	April 16, 2022	Wynona, Devin, Asifur	Added acceptance tests and other features' tests

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: at least can handle 1000 requests/minute

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.

Acceptance Test

As a user, I want to be able to create my profile

Test Steps:

1. Click on "Sign in" link on the navigation bar
2. Click on "Sign Up" link on the Sign in page
3. Fill in the sign up form with the info below:

    - First Name : Jordan,
    - Last Name: Peterson,
      Email: jpeterson@gmail.com,
    - Password: 12341234,
    - Phone no- 431-399-5710
4. Click “Sign Up” button to create an account
5. Click on “Sign in” link again
6. Fill in the form with the above email and password

Expected Result:

- All the buttons that are mentioned should be on the application and navigate to the referred form/page when clicked
- Should be able login after signing up for an account
- Should not be able to login with an unregistered email and password

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

Acceptance Test

As a seller, I want to post an ad listing

Pre-conditions:

- Has a registered account

Test Steps:

1. Click the "Sign In" button on the top right of the navigation bar
2. Fill in your registered email and password
3. Click the "Login" button
4. You're now logged in with your account
5. Click "Post" link on the navigation bar
6. Fill in the Form as follows:
    - Type "New Couch" in the Title field
    - Select "New" in the Condition drop-down option
    - Select "Sofa" in the Category drop-down option
    - Type "Trendy, unique couch code:1234321" in the Description field
    - Type "Winnipeg" in the City field
    - Select "MB" in the Province drop-down option
    - Attach any file that ends in .png, .jpg as an image
    - Type 100 in the Price field
7. Click "Post" button

Expected Result:

- All the buttons that are mentioned should be on the application and navigate to the referred form/page when clicked
- A listing with the title "New Couch" and the description "Trendy, unique couch code:1234321" should be created
- Should see this listing on the Search page by clicking on the "Search" button on the top navigation bar.
- Should see this listing on the My Listings page by clicking on the "My Listings" button on the top navigation bar.

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.

Acceptance Test

As a user, I want to be able to filter and catalog ad listings

Pre-conditions:

- Multiple listings with Condition "Used" and Category "Bed" are already posted by Fujiji's users
- Multiple listings with Condition "New" are already posted by Fujiji's users
- Multiple listings with Category "Other" are already posted by Fujiji's users

Test Steps:

1. Click the "Search" button on the navigation bar
2. You can now see all the listings posted by Fujiji's users
3. On the search bar, click on the options icon beside the magnifying glass icon
4. Select "Used" in the Condition drop-down option
5. Select "Bed" in the Category drop-down option
6. Click the "Apply" button

Expected Result:

- All the buttons that are mentioned should be on the application and navigate to the referred form/page when clicked
- Only listings with Condition “Used” and Category “Bed” should be shown on the Search page
- Listings with Condition “New” should not be shown on the Search Page
- Listings with Category “Other” should not be shown on the Search Page

Core Feature 4: Boost an Ad

API:

```
POST /boost/:listing_id?packageID=package_id
```

Unit Test

- Ensure that we have the boost packages.
- Ensure that users always request for a valid boost package.
- Ensure that the listing owner can see a boost indicator on the listing that they boosted.
- Ensure that the listing owner can see the remaining days of their boost package on the listing that they boosted.
- Ensure that users can see a boost button on the listing if they haven’t boosted it.
- Ensure that users can’t see a boost button on the listing if they have boosted it

Integration Test

- Ensure that a user can boost 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 boost a listing when they are not signed in by successfully getting HTTP response 400 after sending the request.

Acceptance Test

As a seller, I want to boost my ad listing

Pre-conditions:

- Has a registered account
- Has multiple listings in “My Listings” page

Test Steps:

1. Click the “Sign In” button on the top right of the navigation bar
2. Fill in your registered email and password
3. Click the “Login” button
4. You’re now logged in with your account
5. Click “My Listings” link on the navigation bar
6. Click on a specific listing to see the listing info and boost info
7. Click “Boost” button on the listing info page
8. Select “30 Days Boost” from Boost package option
9. Click “Buy” button after selecting the package
10. Fill in the payment form with the info below:
    - Email: abc@gmail.com
    - Card Number : 4242 4242 4242 4242
    - Expiry Date: 05/25, CVC: 123
    - Postal Code: R3T 3C1

11. Click “Pay” button
12. Click “Search” link on the navigation bar
13. The boosted listing should be showed at the top
14. Click “My Listings” link on the navigation bar
15. Boost another listing by following above steps from 6-11 and choose “15 Days Boost” instead as boost package
16. Boost another listing by following above steps from 6-11 and choose “3 Days Boost” instead as boost package

Expected Result:

- All the buttons that are mentioned should be on the application and navigate to the referred form/page when clicked
- Should see these boosted listings are in the descending order based on the duration of the boost on the Search page by clicking on the “Search” button on the top navigation bar.
- Should see the listings which are not boosted comes at the end of the boosted listings based on creation date
- Should see the info that the ad is boosted when you click on an boosted ad from the “My Listings” page

Core Feature 5: Comments

API:

```
POST /comment/[listing_id]
PUT /comment/[comment_id]
DEL /comment/[comment_id]
PUT /comment/reply/[comment_id]
PUT /comment/delete-reply/[comment_id]
GET /comment/[listing_id]
```

Unit Test

- Ensure that comment can’t be empty.
- Ensure that only the commenter can edit their comment.
- Ensure that only the commenter can delete their comment.
- Ensure that only the listing owner can highlight a comment.
- Ensure that only the listing owner can reply to a comment.
- Ensure that users can see comments under the listing.
- Ensure that users that are not signed in can’t post a comment and shows a clickable link to the sign in/sign up page.
- Ensure that only the listing owner can edit a comment’s reply
- Ensure that only the listing owner can delete a comment’s reply

Integration Test

- Ensure that a user can post a comment on 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 comment on a listing when they are not signed in by successfully getting HTTP response 400 after sending the request.

Acceptance Test
As a Fujiji user, I want to leave comments on a listing
Pre-conditions:

- Has a registered account
- Multiple listings are already posted by Fujiji's users

Test steps:

1. Click the "Sign In" button on the top right of the navigation bar
2. Fill in your registered email and password
3. Click the "Login" button
4. You're now logged in with your account
5. Click the "Search" button on the top navigation bar
6. You can now see all the listings posted by Fujiji's users
7. Click on one of the listings
8. Fill out the comment form with "Is this limited edition?"
9. Click "Comment" button

Expected Result:

- All the buttons that are mentioned should be on the application and navigate to the referred form/page when clicked
- A comment "Is this limited edition?" with your name on it and the posted date should be created under the listing that you selected
- A notification that says "Successfully submit a comment!" should appear

*Note that the chat feature is just an additional feature

Regression Test

We run our regression tests on our CI pipeline (link to front-end and back-end).

We execute all the existing tests for our regression tests since our tests are very quick to run. Therefore, in every PR we try to make sure that all existing features do not break because of the new changes.

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
- Load testing: Artillery
- Security Analysis: CodeQL
- 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