Test Plan and Report

Product: RoadBuddy
Team RoadBuddy
Date: 03/06/25

System and Scenario Test Report:

Sprint 1:

User Story 1: I, as an unregistered user, want to register for the service so I don't have to enter my information every time I use the service
User Story 2: I, as a registered user, want the ability to login and logout as I wish

Scenario 1: User Signup

1. Open RoadBuddy app and navigate to the signup page
2. Enter the following details:
    a. Name: <Banana Slug>
    b. Email: <bananaslug@email.com>
    c. Password: <password123>
3. Click 'Log In'
4. Expected Outcome:
    a. Users should be redirected to the logged in homepage
    b. Users' information will be stored in FireStore

Scenario 2: User Login

1. Open RoadBuddy app and navigate to the login page
2. Enter email and password:
    a. Email: <bananaslug@email.com>
    b. Password: <password123>
3. Click 'Log In'
4. Expected Outcome:
    a. Users should be redirected to the logged in homepage
    b. Session should be active

c. User should be able to do the following:
    i. Post a ride
    ii. Book a ride
    iii. Add a vehicle
    iv. View upcoming rides
    v. View rides that are being offered by other users

Scenario 3: User Logout

1. While logged in, navigate to the homepage.
2. Click ‘Logout’
3. Expected Outcome:
    a. Users should be logged out.
    b. Session should be deactivated.
    c. Users should be redirected to the start page.

Sprint 2:

User Story 1: I, as a user, need to be able to make carpool offers and a request so that I can either get a ride or carpool buddy.
User Story 2: I, as a user, need to be able to see carpool offers and books so that I can either get a ride or carpool buddy.
User Story 3: I, as a driver, need to specify the amount of people I can take in my car so that other people can plan accordingly.
User Story 4: I, as a driver, need to be able to cancel my ride offer so that if any emergency happens, people know that I can no longer drive.

Scenario 1: Create a Carpool Offer

1. Log in to the RoadBuddy app.
2. Click on ‘Post Ride’
3. Enter ride details:
    a. Start Location: <Santa Cruz>
    b. Destination: <San Jose>
    c. Date: <03-06-2025>
    d. Time: <10:00 AM>
    e. Number of Passengers: <3>

    f. Vehicle: <2024 Blue Lexus RX 450H>
    g. Cost: <$10>
4. Click ‘Post Ride’
5. Expected Outcome:
    a. Users should be redirected back to the homepage.
    b. The ride should be added to the rides collection in Firestore.

Scenario 2: VIew Available Carpools

1. Log in to the RoadBuddy app.
2. Click on ‘Available Rides’
3. Expected Outcome:
    a. A list of available rides should be displayed

Scenario 3: Book a ride

1. Log in to the RoadBuddy app.
2. Click on ‘Available Rides’.
3. Click on one of the rides.
    a. Users should be redirected to the ride details page.
4. Click on ‘Book This Ride’.
5. Expected Outcome:
    a. Users should be redirected to the upcoming page.

Scenario 4: Specify Passenger Capacity

1. Log in to the RoadBuddy app.
2. Click on ‘Post Ride’
3. Enter ride details and specify passenger capacity
4. Click ‘Post Ride’
5. Expected Outcome:
    a. Users should be redirected back to the homepage.
    b. The ride should be added to the rides collection in Firestore.
    c. The number of passengers will not exceed this limit.

Scenario 5: DeleteRide Offer

1. Log in to the RoadBuddy app.

2. Click on ‘My Rides’.
    a. Users should be redirected to the upcoming rides page.
    b. A list of booked and posted rides should be displayed
3. Click on a ride that the user posted.
    a. Users should be directed to the ride details page
4. Click ‘Delete This Ride’.
    a. If the ride has no passenger.
        i. Users should be redirected to the upcoming page
    b. Otherwise,
        i. Users will be notified that canceling this will result in a refund to all the passengers and plus a 20% fee.
        ii. If the users proceed, users enter their card information.
        iii. On success, users will be redirected to the upcoming page.
        iv. Otherwise, an error message will be displayed

Sprint 3:

User Story 1: I, as a driver, want to charge passengers for my service so I can make some money.

User Story 2: I as a passenger, I should be able to sort the trips by price so I can pay

Scenario 1: Payment

1. Log in to the RoadBuddy app.
2. Click on ‘Available Rides’.
3. Click on one of the rides.
    a. Users should be redirected to the ride details page.
4. Click on ‘Book This Ride’.
5. Users will be asked to enter their credit card information for payment.
6. Expected Outcome:
    a. Users should be redirected to the upcoming page.

Scenario 2: Sorting

1. Log in to the RoadBuddy app.
2. Click on ‘Available Rides’.
3. Users can select one of the options to sort the rides being offered by.

4. Expected Outcome:
    a. The list of rides will be sorted based on the selected option.

Sprint 4:

User Story 1: I, as a driver and passenger, want to be notified when a passenger in my ride has canceled or the owner of a ride has deleted a ride that I am a passenger in.

Scenario 1: Notification of Ride Deletion

1. Driver cancels their ride offer.
2. Expected Outcome:
    a. All passengers in the ride should receive a notification that the ride has been deleted.

Scenario 2: Notification of Ride Cancellation

1. Passenger cancels their booking for a ride.
2. Expected Outcome:
    a. The driver should receive a notification that the passenger has canceled.

Unit Tests:

1. Static Code Analysis:
    - A Pylint Github Actions workflow has been set up to analyze the backend code.
    - The workflow is triggered on every push and ensures code quality standards
    - Pylint checks files in the *backend/* directory with specific disabled warnings for flexibility.
    - The configuration includes caching dependencies.
    - If critical issues arise, they are logged and reviewed in the CI pipeline.

Definition of Done:

1. Code is pushed into a branch (not main).
2. Code reviewed for styling guidelines automatically.
3. Code reviewed by team members, running their own tests.
4. Code merged into the main branch.