# System and Unit Test Report for the Hitchhike platform by Uplyft

---

**System Test scenarios**

- Sprint 1
    - User story 1: I am a rider and I want to sign up with my school email because I want to make an account.
    - User story 2: I am a driver and I want to sign up with my school email because I want to make an account.
    - User story 3: I am a rider and I want to be able to sign in because I want to request a ride.
    - User story 4: I am a driver and I want to be able to sign in to offer rides.
    - User story 5: I am a user and I want to be able to verify my email in order to make the app more secure.

      Scenario:
        1. Start the hitchhike app and select *sign up*. Then fill in the fields first name, last name, username, password, email, and date of birth. Select the button that says sign up to complete.
        2. Signup is the same process for riders and drivers.
        3. Start the hitchhike app and select *sign in*. Then fill in the fields username and password and select the sign-in button.
        4. Sign-in is the same process for riders and drivers.
        5. Upon signup, users are notified that they were sent a verification email. Check the email used in signup and select the link provided in the email from hitchhike, you will then be notified that you are varified.

- Sprint 2
    - User story 1: I am a rider and I want to be able to view rides so that I can see what riders are available.
    - User story 2: I am a driver and I want to be able to post rides so that I can get other people to ride with me.
    - User story 3: I am a rider and I want to be able to request a ride because I want to get from one place to another.
    - User story 4: I am a driver and I want to be assured that my ride won't overfill because I want to obey the law.

      Scenario:

1. After signing into the app with username and password you will be on the view rides page.
2. In the homepage select post-ride and fill in the shown parameters.
3. In the homepage select request-ride and fill in the origin parameter.
4. Every time a ride is requested, the seatsLeft attribute is decremented.

- Sprint 3
    - User story 1: I am a rider and I want to be able to see my driver's contact information.
    - User story 2: I am a rider and I want to be able to see my driver's rating so that I can feel safe.
    - User story 3: I am a driver and I want to be able to see my rider's contact info so that I know who to pick up.

      Scenario:
        1. Sign in to the app and request a ride. Once the ride is booked, the driver's contact information is displayed.
        2. Not functional yet.
        3. Sign in to the app and select my rides. The rider information will be displayed.
- Sprint 4
    - User story 1: I am a user and I want to be able to post a picture of myself so my riders'/driver can recognize me.
    - User story 2: I am a user and I want to be able to rate my drivers so other people know about my experience.
    - User story 3: I am a user and I want to be able to navigate through the app easily so it acts as a good replacement to the Facebook group.
    - User story 4: I am a user and I want to be able to download the app to use it.

      Scenario:
        1. Not functional yet.
        2. Not functional yet.
        3. Sign in to the app and select desire pages and move back to previous pages.
        4. Not functional

**Unit tests**

- Unit tests were built in the Jest framework for comprehensive javascript jesting. We created two test suits with corresponding modules for our API and database. The two modules have similar acceptance criteria but since our API and database were developed

in parallel, we wanted to have tests for both. These modules are named index.test.js and db.test.js and utilize a mock data object named mock_data.json.

- Database unit tests
    - New user test
        - Expects that the user is added to the database and can be retrieved
    - Get user test
        - Expects that the user can be queried by the username and retrieved even in the event of a server crash.
    - Post-ride test
        - Expects that a ride can be added to several internal data structures and can be updated dynamically.
    - Find-ride test
        - Expects that the k nearest rides to a given location and time can be found in a query.
    - Update-ride test
        - Expects that a ride can be updated and removed if the departure time has passed.

- Index (API) unit tests
    - Signup test
        - Expects that a new user can be created and inserted into the database.
    - Login test
        - Expects a user can be queried from the database and returned to the client.
    - Post-ride test
        - Expects that ride parameters can be retrieved and used to create a ride in the database.
    - Find-ride test
        - Expects that a ride request can be received and queried in the database.