Complete Document

- The Trucker Tracker

Stakeholder: Marcy Pullard & Ken Pullard (Pullin' Freight, LLC)

Team-member:

- **Taixiang(Max) Zeng (taixianz@usc.edu)**
- **Aarav Malpani (malpani@usc.edu)**

Counseling Faculty: Prof. Jeffrey Miller

Project Description:

This project builds up the online operational system of Pullin' Freight, LLC, a small, but growing, Los Angeles based, DBE/MBE certified trucking company. The system consists of a doorway website, an AWS EC2 server, a mobile app and an operation dashboard. The system supports all the services Pullin' Freight LLC is providing to users, as well as daily operational management.

Glossary

Database	Place to store all the transactions of all the users
Server	A central management that controls the flow of data between database and end users
UI	Short for "User Interface", meaning the appearance of the app
React Native	Outside library by Facebook that generates code for both iOS and Android platform at the same time
AWS	"Amazon Web Services", a web computation service provided by Amazon

High-level Requirement

Website (https://www.pullinfreightllc.com)

Pullin' Freight company website serves as a doorway to the company's business. The main page demonstrates Pullin' Freight's primary business information, including company introduction, trucking services provided, past projects, etc. The website also supports driver application portal, where visitors can submit an application form and become a driver if qualified.

The website is hosted by WiX server, which supports all the functionality. In the Dashboard page (after logged in), the administrator is able to perform managerial operations of the website, including checking analytics, responding to communication messages, marketing the site (SEO), and editing content. WiX also provides information encryption functionality to secure sensitive information (driver applications for example).

Mobile app (iOS & Android):

The Pullin' Freight app is the primary venue for users to pick up tasks, submit Bill of Ladings, view bill history, and edit account information. After a user has logged in, he or she will be directed to Jobs Page. The user is then able to see all the "Today" as well as "Future" jobs assigned and take the desired job.

After a job is finished, the user fills out a physical Bill of Lading form in the truck and then enter the same information in the app. The user can then view the submitted bills in the History page. The user can log out the app by pressing the "Sign Out" button.

Operation Dashboard (Desktop Program):

The Operation Dashboard supports all managerial activities required by Fleet Manager, including shipper management, jobs management, viewing submitted Bill of Ladings and driver information, as well as generating invoice logs. The program is in sync with driver apps and can be live updated.

Server (AWS EC2 Web Server)

The server manages all of the requests and communications between the mobile apps and backend database.

Database (AWS RDS Database)

The RDS database hosts a MySQL relational database that stores all the user and job data.

Functionality Specification (Implemented)

Website

a. Main Page

The main page displays primary Pullin' Freight's business information for the people interested with a simplistic and intuitive layout design. The website includes:

- ***About Us*** section
- ***Services*** section
- ***Projects*** section
- ***Contact*** Information & ***Inquiry*** submission
- ***Outside link*** (Instagram)

b. Application Page

- The "Apply Now" button directs the user to a separate Application Page.
- Visitors are able to fill out the required areas and upload files on the form before submitting the application.

c. Administrator

- The administrator of the website can log into administrator account on the WiX server, where he or she can view website analytics, respond to communication messages, market the website(SEO functionalities), set up automations, and edit the website.
- Administrator will get email notifications if an application has been submitted, and able to review the uploaded file in "Form Submission" section inside dashboard.

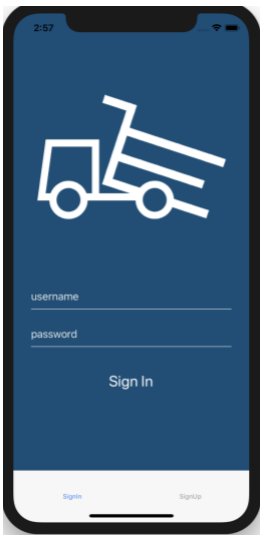
d. Information Encryption

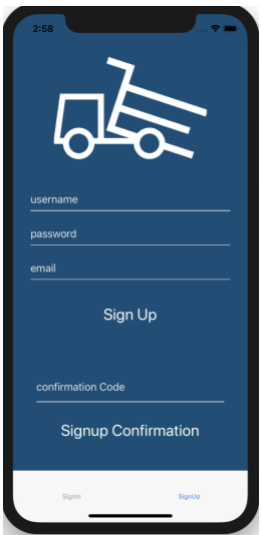
- WiX provides encryption to secure all the submitted information.

Mobile app (iOS & Android)

a. App Download

- After a user gets approved from the driver application, he/she is qualified to become a driver, or "sub-haulers", for Pullin' Freight. The user is then invited to download the Pullin' Freight app and set up a driver account.

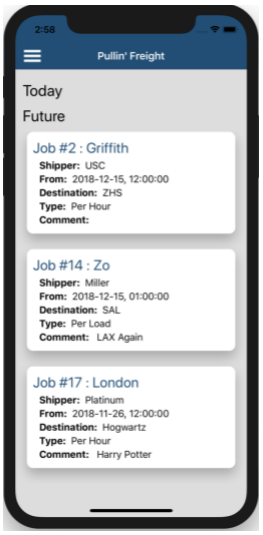




b. Account Sign Up & Sign In

- The first page provides the user with the option to sign up a new account or sign in with an existing one. The user cannot go to any other page if not signed in.
- In the Sign Up page, the user needs to enter a username (not taken by other people), a password (requirement customizable in AWS Cognito settings) and an email address to receive the confirmation code.
    - After the user hit "Sign Up", he or she will get an email with confirmation code. After entering the code in the app, the new user account is stored in the AWS server.
    - The user is then able to sign in with the new account credential.

c. Job Feed Page (Arriving page)



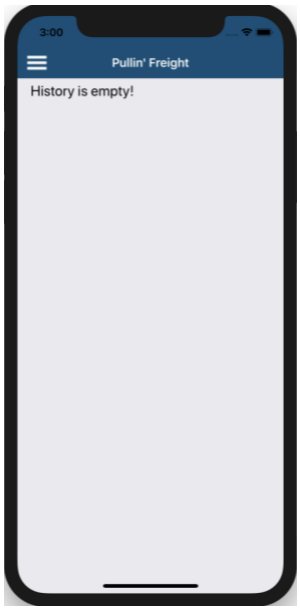
- After logging in, the user arrives at the Job Feed Page, which displays all the jobs dispatched by Fleet Manager in "Today" and "Future" sections respectively.
- Sliding down the job list refreshes the job feeds.
- When there is no jobs assigned to the user, the page displays "No Jobs Available!".

d. Bill of Lading Submission

- After the user confirms taking the job with the Fleet Manager (text/email), he or she is responsible for completing the job.
- After the job is finished, the user is required to submit the job's invoice information (the same information as on the physical copy of Bill of Lading) by selecting the job in the job list and filling out the required fields.

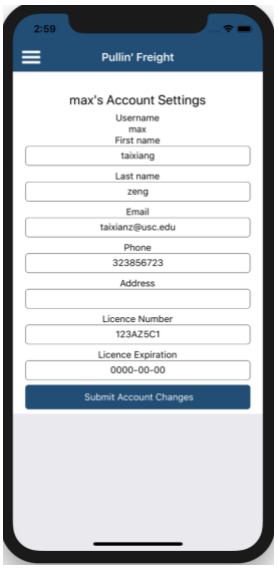
e. Job History

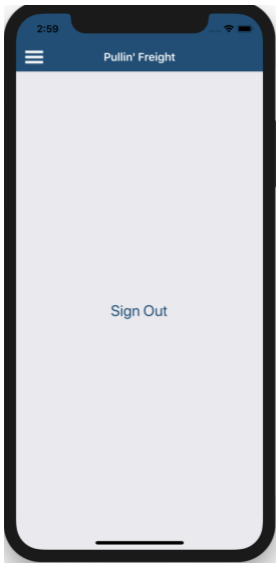
- In the Side menu, the user can navigate to Job History Page, which displays all the past jobs of the user based on the Bill of Lading submissions.
- Clicking on specific job on the bill page, the detailed information of that specific job is displayed. The user can cross check this with the company's billing information.
- Sliding down the job list refreshes past jobs list.
- When there is no jobs assigned to the user, the page shows "History is empty!".



f. Account Settings

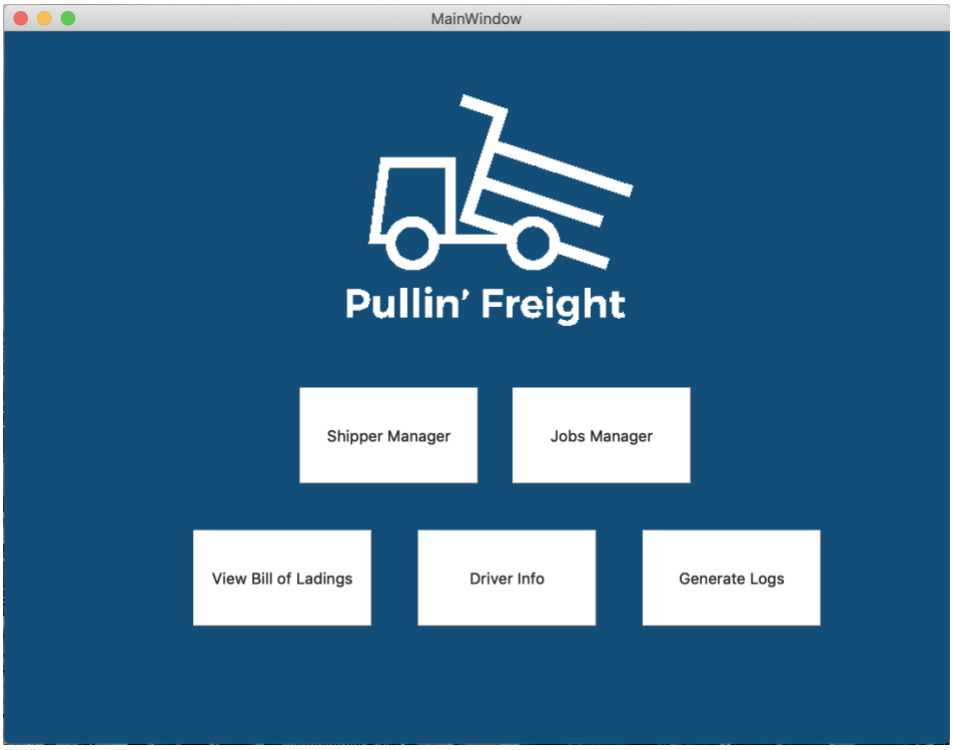
- In the account page, information (driver license, billing info, etc.) about the user is displayed.
- The user can edit all the fields of the account information except the username, which is set during account sign up.





g. Sign Out

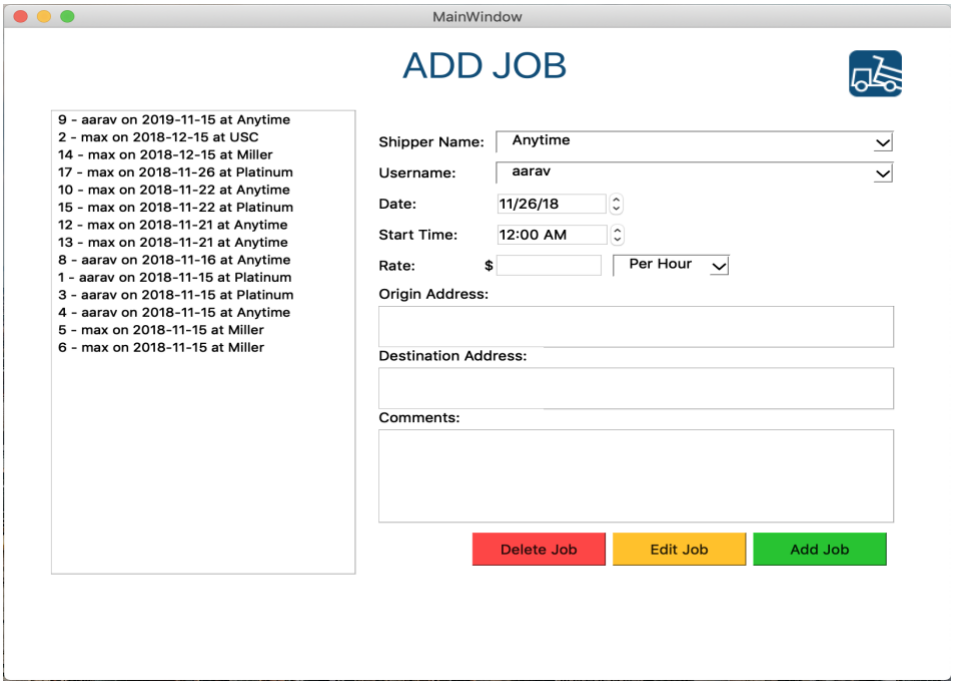
- The Sign Out page lets the user sign out of the app.



Operation Dashboard

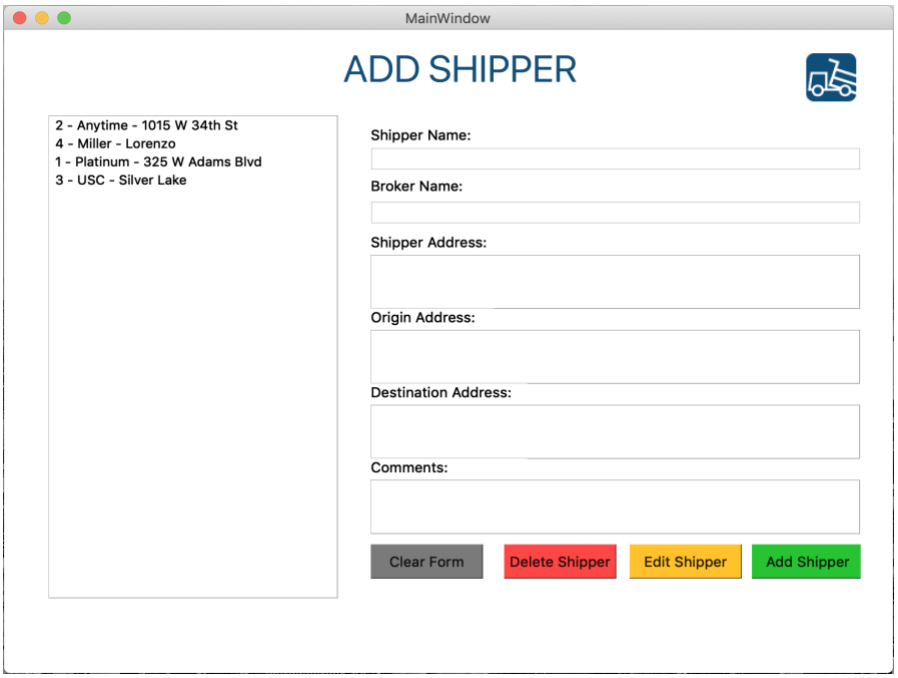
a. **Jobs tab**

a.1. Add a new jobs for a user
a.2. Edit a jobs
a.3. Delete jobs
a.4. Display a list of all jobs



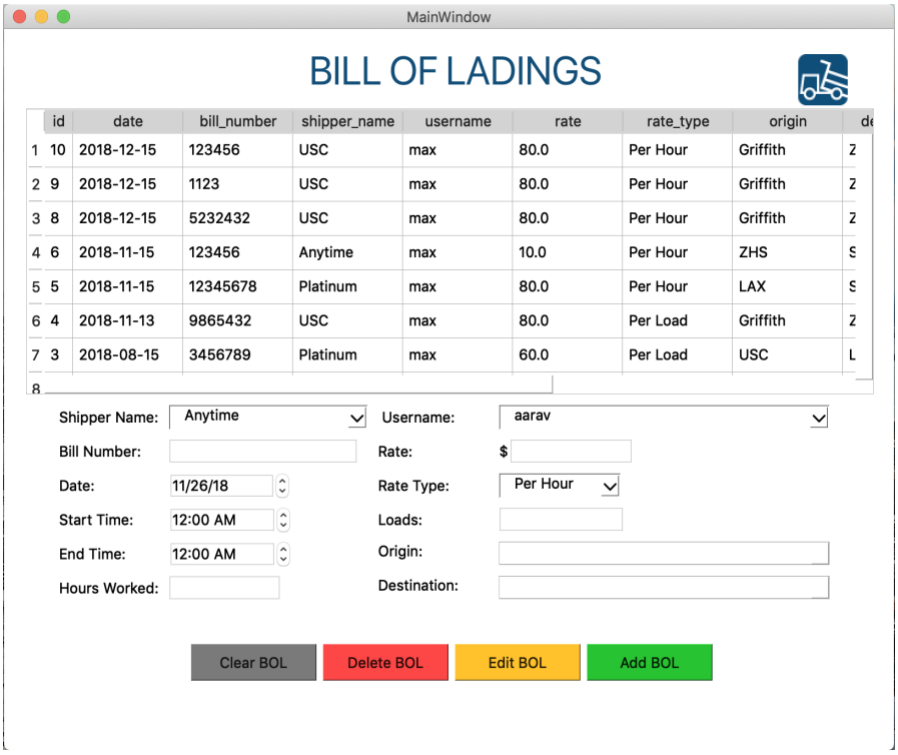
b. **Shipper tab**

b.1. Add a new shipper
b.2. Edit shipper
b.3. Delete shipper



c. **Bill of Ladings tab**

c.1. See past jobs
c.2. Add a new bill

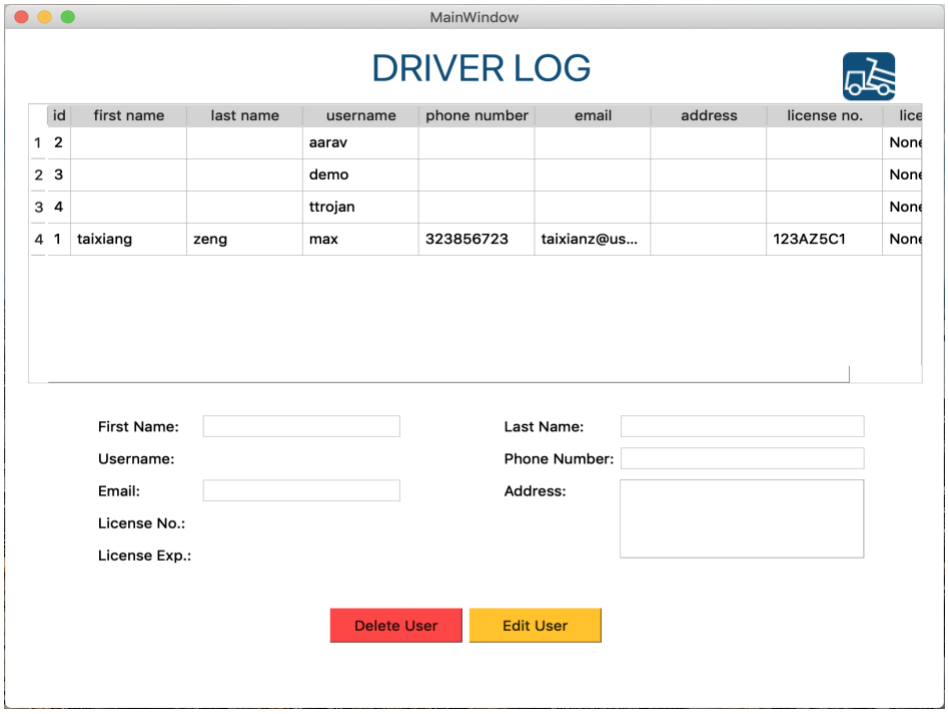


c.3. Edit bill

d. **Driver Logs**

d.1. View driver information

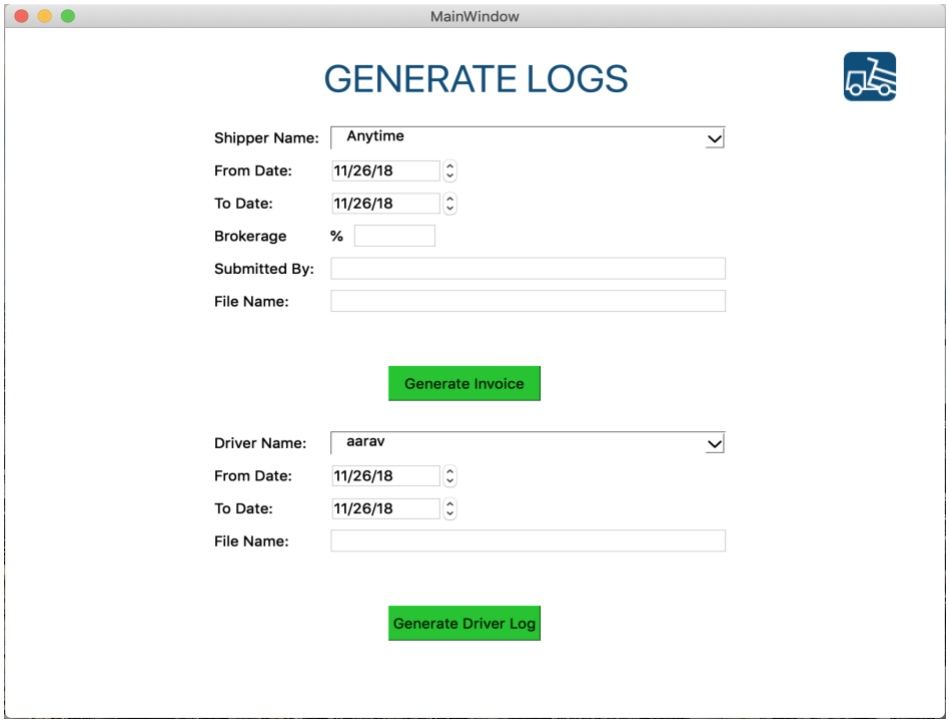
d.2. Edit driver information



e. **Generate Logs**

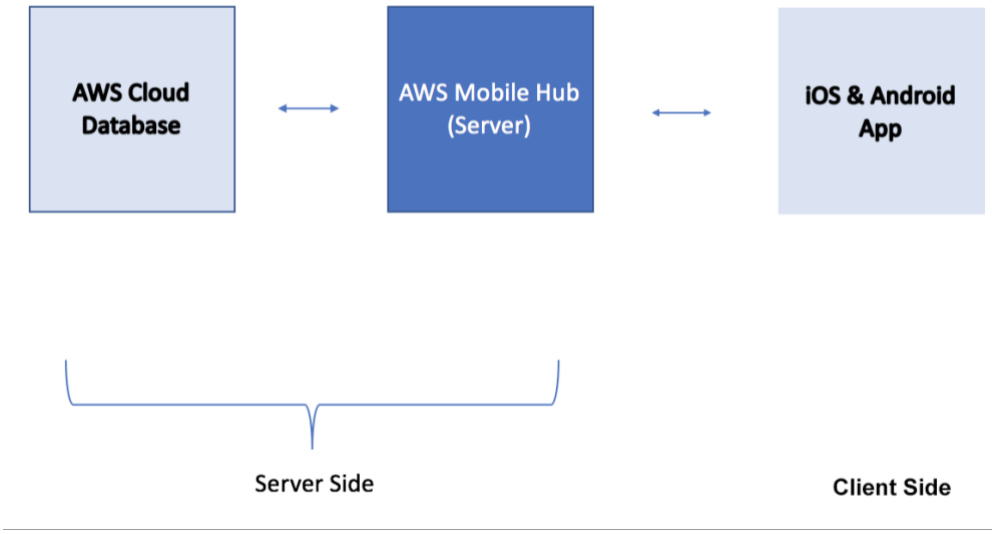
e.1. Generate invoice for shippers

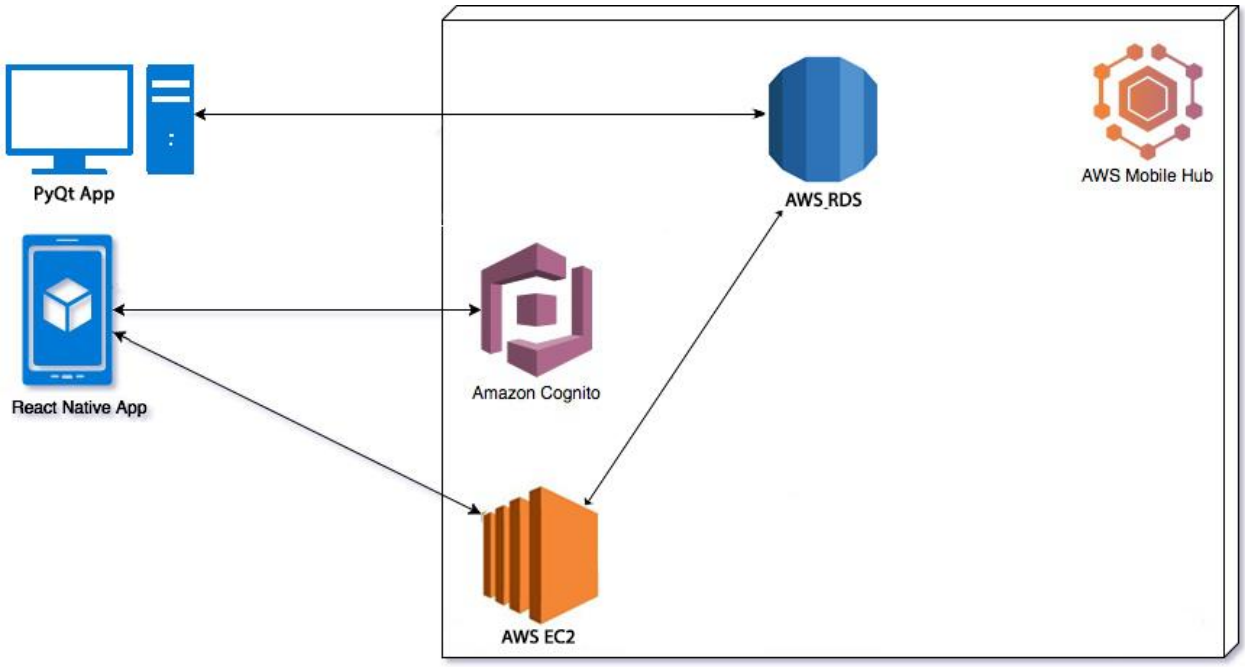
e.2. Generate invoice for drivers



System Architecture

"Three-Tier Pattern" is the principal design pattern for both mobile and dashboard app. The Three-Tier Pattern adopts "separation of concerns" design principle that splits the application into sections based on their functionalities. Each is separated into three components: Display tier, Business Logic tier, and Database tier. Display Tier encompasses all the screens as well as the logic to update them; Business Logic Tier is the mid-way server that handles data retrieval and data update to the back-end database; the Database Tier consists of MySQL database hosted by AWS RDS Database.





System Detailed Design

Website

- The domain "www.pullinfreightllc.com" is hosted by Google Domains
- The website is programmed and edited using WiX Editor. The premium account of WiX enables several functionalities needed and "123 Form" plug-in for free. Most of further modifications can be done using the WiX editor.

Mobile App (iOS & Android)

Programming Language: React Native

Outside Library: AWS Mobile Hub, react-navigation, react-native-cardview

Software Requirements:

To run on emulators:

- Android Studio
- XCode (need MacOS)

To run on Expo simulator:

- Expo Cli

Recommended Hardware Requirements:

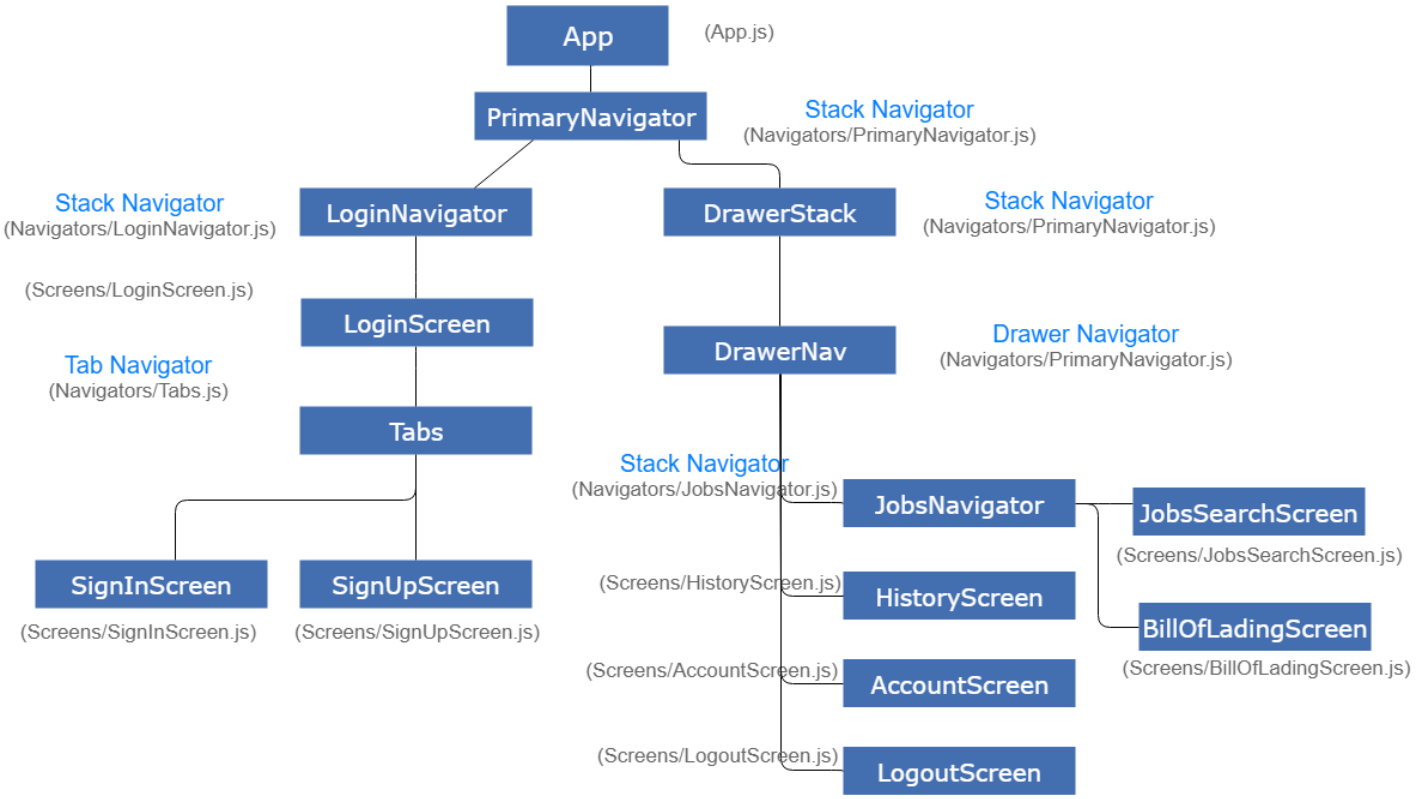
Microsoft® Windows® 7/8/10 (32- or 64-bit)

- 4 GB RAM minimum, 8 GB RAM recommended
- 2 GB of available disk space minimum
- 4 GB Recommended (500 MB for IDE + 1.5 GB for Android SDK and emulator system image)

MacOS (32- or 64-bit)

- Mac® OS X® 10.10 (Yosemite) or higher, up to 10.14 (macOS Mojave)
- 1.4 GHz of Intel Based CPU
- 4 GB RAM minimum, 8 GB RAM recommended
- 2 GB of available disk space minimum
- 4 GB Recommended (500 MB for IDE + 1.5 GB for Android SDK and emulator system image)

Navigation Structure Break down:



Server

Host: AWS EC2 Web server

Scripting Language: PHP

Communication Mechanism: JSON (Called using "Fetch" API in React Native)

Script List:

a. **DBConfig.php:**
   contains configurations needed to connect AWS MySQL database; already included in other script files.

b. **newUser.php:**
   Called when the new user account is confirmed by AWS Cognito. Creates a new user account and stores the username into database.

c. **getAccount.php:**
   Gets account information from database given the username.

d. **getTodayJobs.php:**
   Retrieves assigned jobs to the specific user on today by username. Job list returned is displayed in the "Today" section in Jobs Page.

e. **getFutureJobs.php:**
   Retrieves assigned jobs to the specific user in the future by username. Job list returned is displayed in the "Future" section in Jobs Page.

f. **getHistory.php:**
   Retrieves a user's past job history. Called in History Screen.

g. **submitAccount.php:**
   Submits the edits to account information into the database.

h. **submitBill.php:**
   Inserts the newly created bill of lading information into the backend database.

Database (AWS RDS - MySQL database)

Storage Allowed: 100.00 TB

Connection:

Hostname: pullinfreightllc.cpunxilbhe7v.us-west-1.rds.amazonaws.com

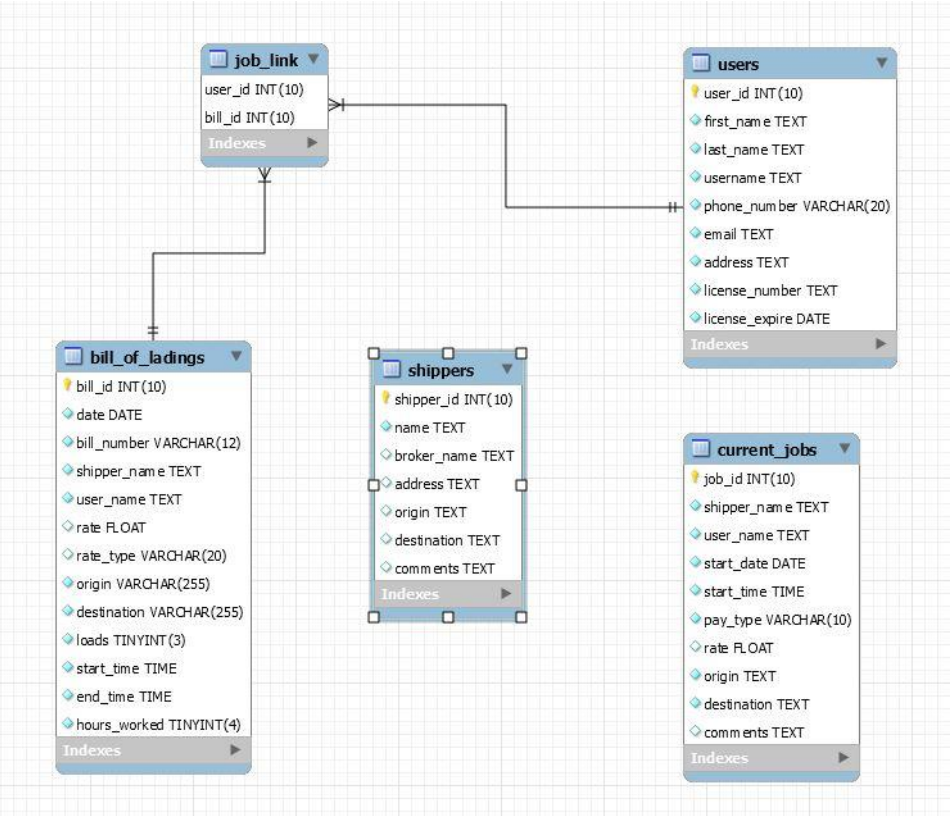
Connection method: Standard (TCP/IP)

Username: pullinfreight

Port: 3306

Tables

- users
- shippers
- current_jobs
- bill_of_ladings
- job_link


job_link
user_id INT(10)
bill_id INT(10)
Indexes
users
user_id INT(10)
first_name TEXT
last_name TEXT
username TEXT
phone_number VARCHAR(20)
email TEXT
address TEXT
license_number TEXT
license_expire DATE
Indexes
bill_of_ladings
bill_id INT(10)
date DATE
bill_number VARCHAR(12)
shipper_name TEXT
user_name TEXT
rate FLOAT
rate_type VARCHAR(20)
origin VARCHAR(255)
destination VARCHAR(255)
loads TINYINT(3)
start_time TIME
end_time TIME
hours_worked TINYINT(4)
Indexes
shippers
shipper_id INT(10)
name TEXT
broker_name TEXT
address TEXT
origin TEXT
destination TEXT
comments TEXT
Indexes
current_jobs
job_id INT(10)
shipper_name TEXT
user_name TEXT
start_date DATE
start_time TIME
pay_type VARCHAR(10)
rate FLOAT
origin TEXT
destination TEXT
comments TEXT
Indexes

Deployment

Mobile App

Pullin' Freight is developed using React Native technology. As a result, the application package contains both iOS and Android components of the app. To run the ap simulation in the desired environment, respective mobile emulator must be installed on the machine. Node.JS is also required for the simulation requirement. All other libraries are included in the package file already.

To install Node and NPM if not installed already, type "brew install node" in the command line (or "sudo brew install node" if run into any errors).

Set up Instructions for MacOS

1. In the command line, type "**npm install -g react-native-cli**" (or "**sudo npm install -g react-native-cli**" if run into any errors).

2. Download, decompress pullinfreight_reactnative.zip file to the intended file location.

3. Navigate to the file location inside command line.

4. Type "**react-native run-ios**" or "**react native run-android**" respectively for the desired emulation environment in the command line.

Other means of running the application in a simulated enviornment can be found on:
https://facebook.github.io/react-native/docs/getting-started.html

For simulation in Windows and Linux environment and further details please refer to:
https://facebook.github.io/react-native/docs/getting-started.html

For further troubleshooting informations, please refer to:
https://facebook.github.io/react-native/docs/troubleshooting#content

Operation Dashboard

The Operation Dashboard is a desktop app developed using Python & PyQt5. The executable can only been run in Unix environments (on Mac OS but not on Windows).

To start the program, a step by step instruction is given below:

1. Copy the decompressed “Fleet Console” executable to intended location.

2. Double click the “Fleet Console” executable to start the program.