Advanced Web project

Project README - Table of Contents

Description

This is our major project called SHIPLY. It is a postal office app with real-time data, responsive and well-built for ideal user experience. Gracefully tested and deployed on platforms, we followed project specsifications smoothly.

Project Overview

Shiply is an app which includes three major milestones - Client app, Driver app and a simulation of a parcel locker. They are all connected not only in UI, but also tighten up with database functionality.

Our team

Our team is our pride! Get to know the talented individuals who brought this project to life.

Rachna Kumar

Rachna has proven to be a motivated and responsible team player. She takes charge of frontend development and app design of main client app, ensuring a seamless user experience. Rachna implemented all frontend architecture and layout.

Hakan Asmaolgu

As the mastermind behind Figma design, styling, and layout, Hakan is ensuring smooth design for all our apps: client, driver and parcels app. His responsibilities also include frontend development of main client app. Also, Hakan has been dividing tasks and organising our work into more connected to each other.

Shovan Das

Shovan is developing the driver app and locker app. His expertise extends to full-stack implementation for both applications. Shovan takes charge of MySQL driver and lockers responsibilities, ensuring smooth connectivity for drivers, lockers, and clients.

Shovan has also enhanced client app to make it more responsive and introduced new tech stack into our frontend.

Mariia Glushenkova

Mariia is developing full-stack main client app. She takes charge of developing the client app and managing the backend for client, including MySQL and Firebase. Mariia is also responsible for seamless deployments of all apps and hosting MySQL.

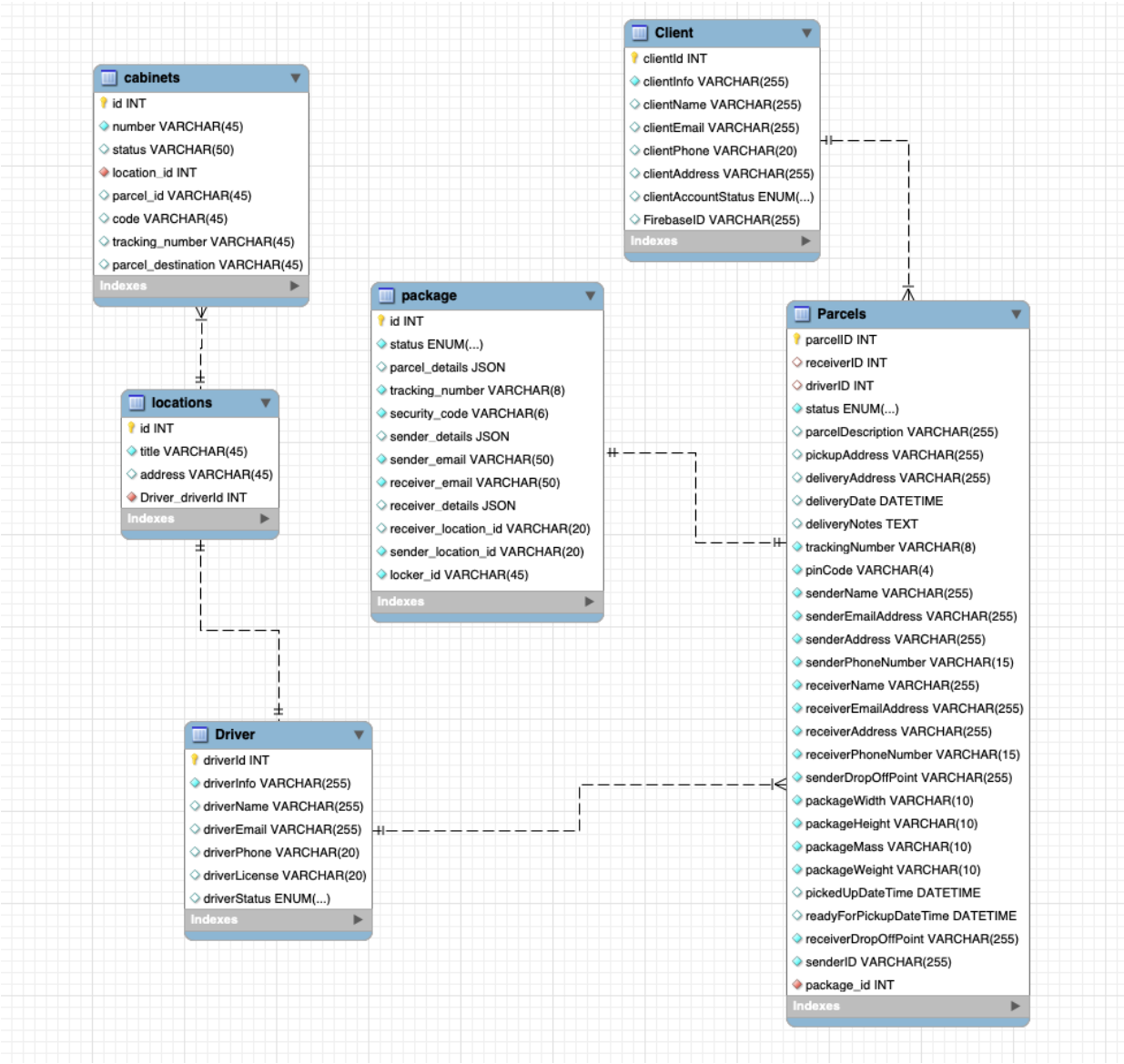
Project Components

Our apps are following same architecture and structure. Our main app "Shiply-app" is the brain of all our apps. It consists of client frontend and backend for all apps.

Other apps like driver ap and touchscreen app are frontend-based, sendind requests to our backend.

1. **Application Architecture**
    - Application architecture is following default React application principles on frontend and backend is structured as per database tables: driver endpoints, parcels endpoints, client endpoints, location endpoints etc.
    -
2. **Database Structure**
    - For all our apps, we use hosted Amazon RDS database MySQL.

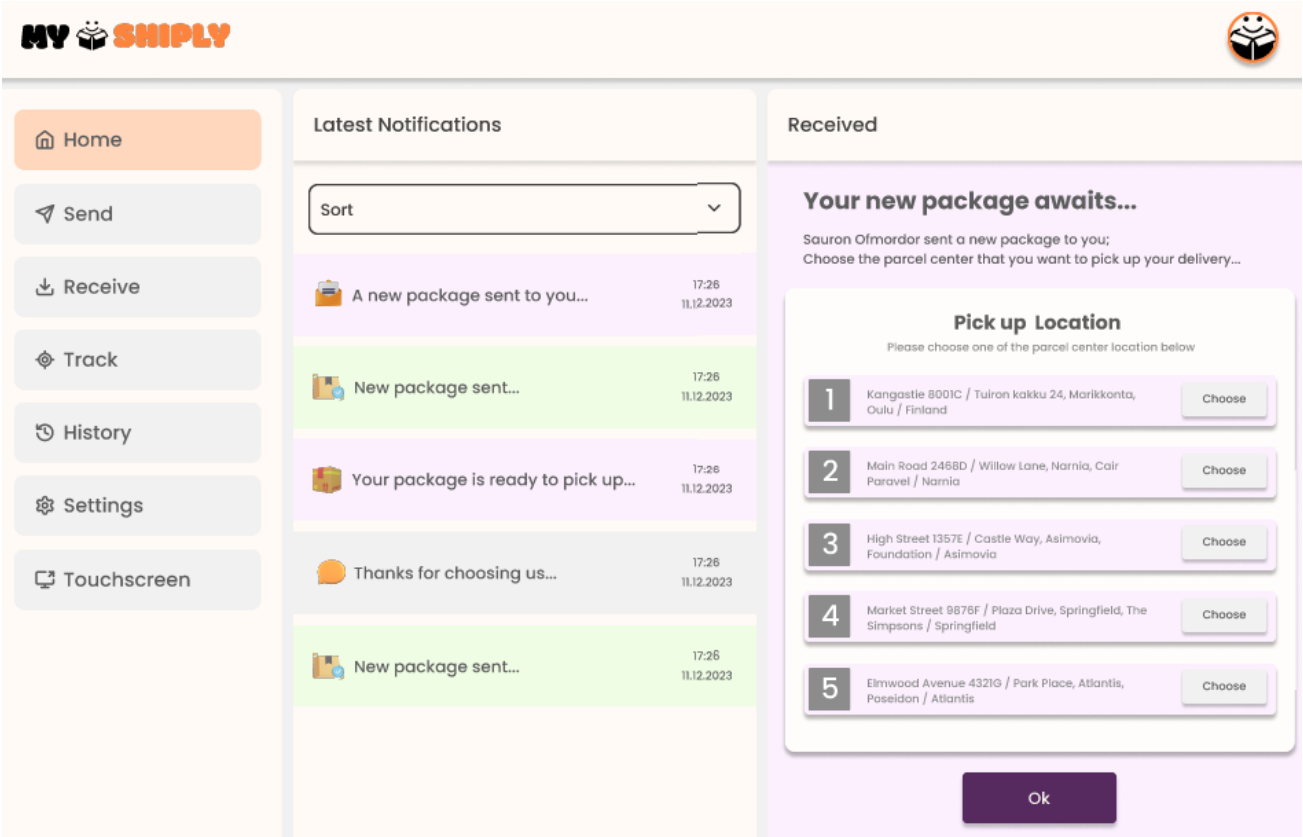
      Here is our diagram with all important tables and key properties.



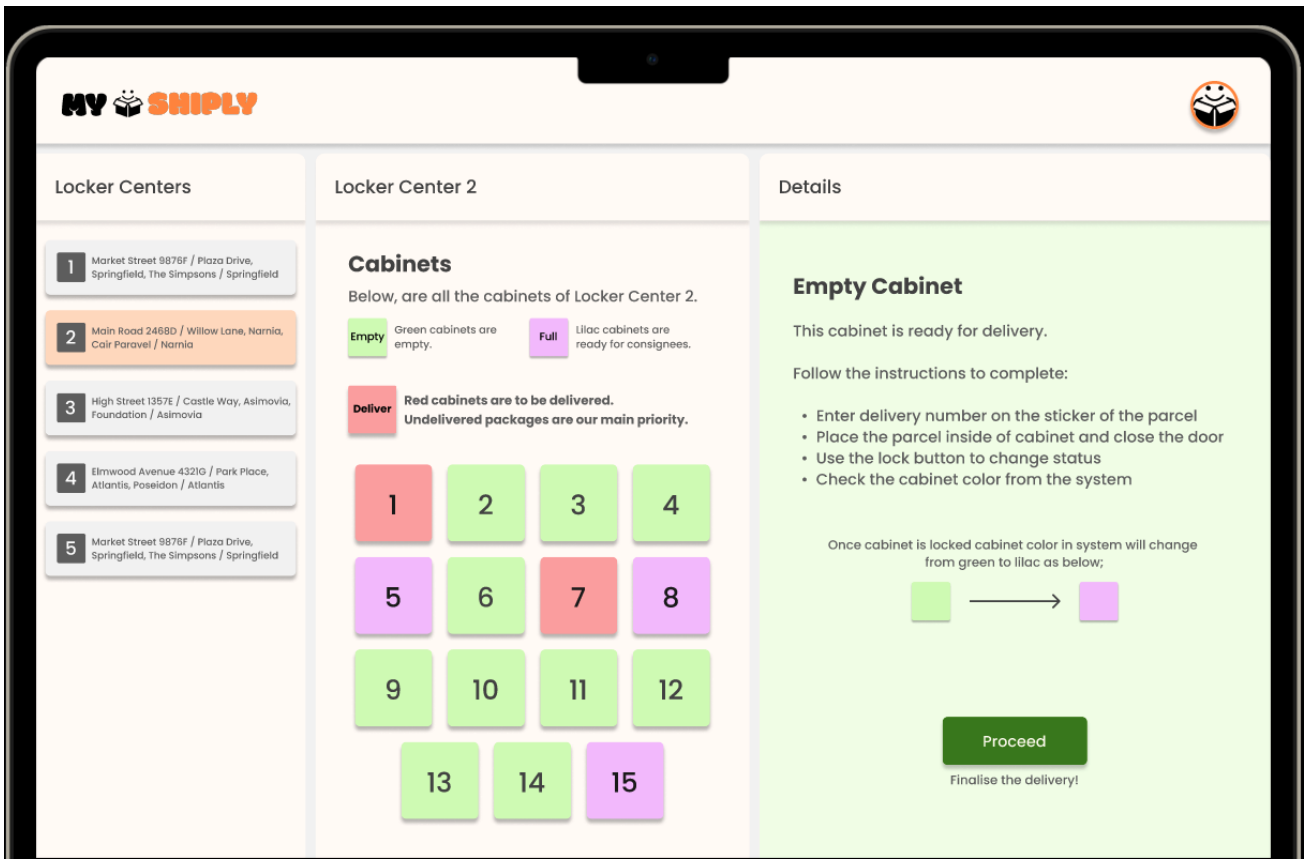
      As per model, Parcels table is the one which connects other tables. It has key-to-key relationship with Client and Driver tables. Locatioins and cabinets table are created for handling different location of parcel cetners.

3. **Interface Description**
    - Interface is lightweight and powerful. There are different sections of websites to send, track, receive and look at history.
    - For driver app there are all possible packages and possibility to tak them to parcel locker
    - Parcel locker is simulating aka Posti lokers for sending, receiving packages.
4. **UI Plan**
    - Our UI is made with Figma design, here is the link for main client app design https://www.figma.com/proto/jijAvzmo9h8uR9TxKqVhCe/Shiply-Main-Design?type=design&node-id=396-3280&scaling=scale-down&page-id=0%3A1&starting-point-node-id=15%3A4 (Click to go to touchscreen to see parcel simulator)
    - Click "Log in as driver" to see driver app

Client app



Driver app



Parcel simulator (Touchscreen) app

Robot app

· Application to automatically generate new parcel deliveries to consumer users. This is a backend application only. This is the parcel generator “robot”.

Deployed in a file robot.ts in the backend folder.

🖥️ Frontend

Tech Stack

- **React** - A JavaScript library for building user interfaces.
- **Vite** - A fast, opinionated frontend build tool.
- **TypeScript** - A typed superset of JavaScript that compiles to plain JavaScript.
- **Tailwind CSS** - A utility-first CSS framework.
- **Tailwind Prettier Plugin** - A Prettier plugin for formatting Tailwind CSS classes.
- **ESLint** - A pluggable linting utility for JavaScript and TypeScript.
- **PostCSS** - A tool for transforming CSS with JavaScript.
- **Autoprefixer** - A PostCSS plugin to parse CSS and add vendor prefixes.
- **shadcn/ui** - Beautifully designed components that you can copy and paste into your apps.

⚙️ Prerequisites

Make sure you have the following installed on your development machine:

- Node.js (version 16 or above)
- npm (package manager)

🚀 Setup

Follow these steps to get started with the app development:

1. Clone the repository which you want to develop:

```
git clone https://github.com/Oamk-DIN22SP/Shiply-App.git
```

2. Navigate to the project's directory:

```
cd shiply-app/frontend
```

3. Install the dependencies:

```
npm install
```

4. Start the development server:

```
npm run dev
```

📜 Available Scripts

- npm dev - Starts the development server.
- npm build - Builds the production-ready code.
- npm lint - Runs ESLint to analyze and lint the code.
- npm preview - Starts the Vite development server in preview mode.

📂 Project Structure

The project structure follows a standard React application layout:

```
shiply-app/frontend/
  ├── node_modules/      # Project dependencies
  ├── public/            # Public assets
  ├── src/               # Application source code
  |   ├── components/    # React components
  |   |   └── ui/        # shadc/ui components
  |   ├── styles/        # CSS stylesheets
  |   ├── lib/           # Utility functions
  |   ├── App.tsx        # Application entry point
  |   └── index.tsx      # Main rendering file
  ├── .eslintrc.json     # ESLint configuration
  ├── index.html         # HTML entry point
  ├── postcss.config.js  # PostCSS configuration
  ├── tailwind.config.js # Tailwind CSS configuration
  ├── tsconfig.json      # TypeScript configuration
  └── vite.config.ts     # Vite configuration
```

🖥️ Backend

Tech Stack

- **Node.js** - A JavaScript runtime built on Chrome's V8 JavaScript engine.
- **Express** - A fast, unopinionated, minimalist web framework for Node.js.
- **TypeScript** - A typed superset of JavaScript that compiles to plain JavaScript.

  Installation

  Follow same procedure as frontend installation. Just connect to MySQL with env file and Firebase with your app credentials.

- **Configuration for backend**

  Ensure that you follow env.example file for backend configuration.

Additional Resources

Working Application URLs

Login with username test@gmail.com and password testtest

Main Client URL: https://shiply-frontend.onrender.com/

Backend URL: https://shiply-server.onrender.com/

Touchscreen URL: https://shiply-touchscreen.onrender.com

Driver URL: https://shiply-driver.onrender.com

API Documentation and Testing

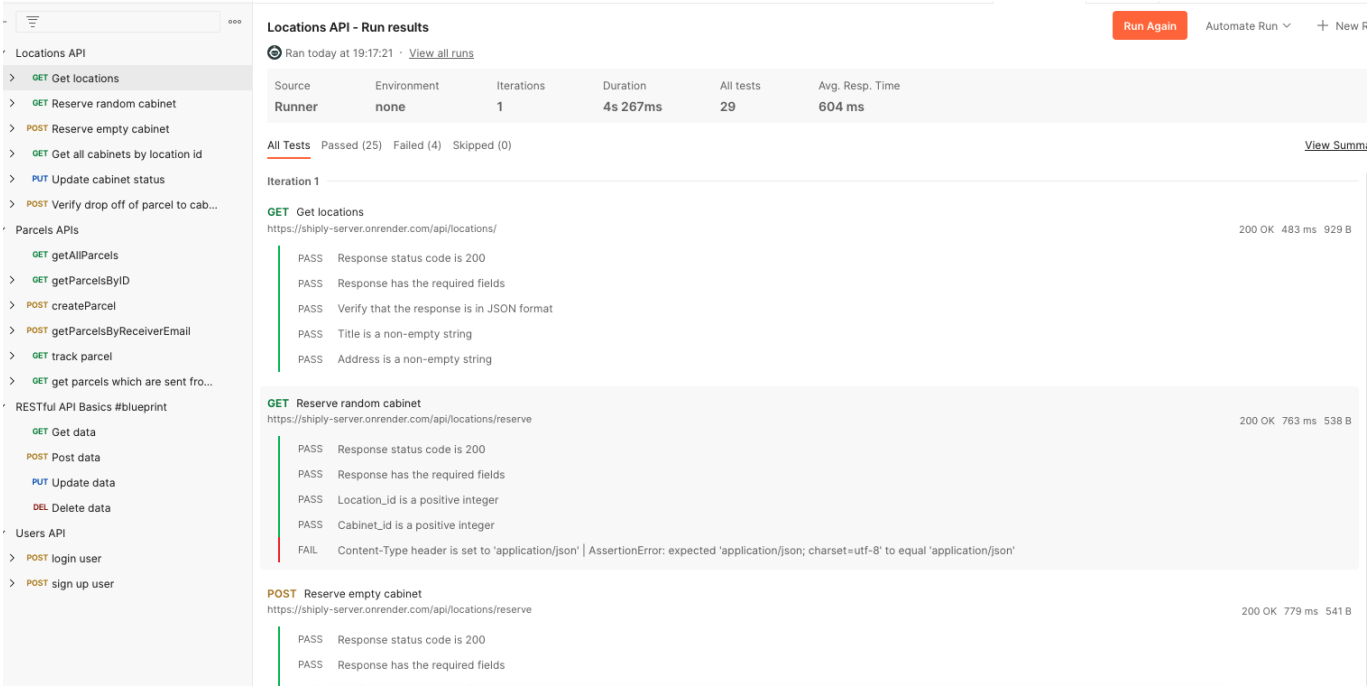
Tested consumer app and driver app accordingly to test plan API endpoints.

Here is Location API endpoints and tests for them can be seen by clicking on each endpoint.

Location APIs

Parcel APIs

Users APIs



Include a link to the test plan document for comprehensive testing information.