**Post Office Management System**

Parjal Rai | Prithvi Kulkarni | Zephania Molla | Zainab Bikar

A picture containing text, outdoor, building, bin

Description automatically generated

Table of Contents

[Overview 3](#_Toc120784765)

[Description 3](#_Toc120784766)

[Lessons Learned 3](#_Toc120784767)

[Wireframes 3](#_Toc120784768)

[Product Backlog 5](#_Toc120784769)

[Functional Requirements 7](#_Toc120784770)

[Technical Architecture 8](#_Toc120784771)

[Running Files 12](#_Toc120784772)

[NPM Install and Start 12](#_Toc120784773)

[Apollo Express 12](#_Toc120784774)

[React Redux App 12](#_Toc120784775)

[.ENV File for Apollo Express 12](#_Toc120784776)

[Documentation 12](#_Toc120784777)

[Logging 12](#_Toc120784778)

# Overview

## Description

Post Office Management System organisation provides various delivery services. The organisation can deliver letters and parcels to users. The website showcases depot locations and the services available at those locations, driver, parcel, and letter details.

## Lessons Learned

Prithvi:

* Learnt how to work efficiently in a team, planning with a bigger group and ways the team should operate.
* Learnt how to use dates with graphql.
* Testing getting 75%+ code coverage.

Zainab

* Learnt how to do testing on server and react.
* Learnt how to test react form and button components.
* Learnt how to use mock data to test CRUD operations for react.
* Improved debugging programme skills.

Parjal

* Learning how to do the graphQL test.
* Learning how to do the redux test using jest and snapshots.

Zephania

* Learnt how to work in a focused, organised, and communicative group setting.
* Learnt how to utilise git projects for collaborative projects involving simultaneous changed being made to the project by different group members.
* Learnt how to use jest to test graphQL apollo server and React Redux applications.
* Learnt how to test app component.
* Learnt how to do snapshot tests on components and functions within.
* Learnt how to test buttons within a react page.

## Wireframes

Graphical user interface, application, Teams

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Text

Description automatically generated

Graphical user interface, text, application

Description automatically generated

# Product Backlog

The ID letters stand for Design, MongoDB, Graph QL, React Redux and Testing.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Name: V | Description | Priority | Status |
| D.01 | Create wireframes | Wireframes to showcase front-end | Low | Complete |
| M.01 | Create Letter Collection | Letter collection fields | Low | Complete |
| M.02 | Create Letter type def | Letter model for graphql | Low | Complete |
| M.03 | Create resolvers | Allows CRUD for letter collection | Low | Complete |
| M.04 | Create Script for Letter Collection | Lets us see Letter documents | Low | Complete |
| R.01 | Create React redux file | Allows us to manipulate mongo database objects | High | Complete |
| R.02 | Create Header and Footer file | Allows navigation | Low | Complete |
| R.03 | Redux reducers | Lets us use actions to change state | High | Complete |
| R.04 | React website | Lets user manipulate data | Medium | Complete |
| T.01 | Create letterList test | Test graphql | High | Complete |
| T.02 | Create CRUD test | Graphql CRUD operations mocked  75+% coverage | High | Complete |
| T.03 | Create react Letter test | Test react Letter component | High | Complete |
| T.04 | Create react CRUD test | React CRUD operations mocked 75+% coverage | High | Complete |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Name: Parjal | Description | Priority | Status |
| D.01 | Create wireframes | Wireframes to showcase front-end | Low | Complete |
| M.01 | Create a script for Parcel Collection | Script document | Medium | Complete |
| M.02 | Create a model for Parcel Collection | Gives field info of Parcel Collection | Medium | Complete |
| M.03 | Create resolvers | Includes queries (get) and mutations (create, update and delete) for Parcel | Medium | Complete |
| M.03 | Create typedefs | Parcel model for graphQL | Medium | Complete |
| R.01 | Redux Parcel Slice | Reducer logic and actions for parcel | High | Complete |
| R.02 | React Parcel Components | Create Parcel components which tests for CRUD operations | High | Complete |
| R.03 | React Parcel  Frontend | Display the relevant info of the Parcel component | High | Complete |
| T.01 | GraphQL test | Create tests for CRUD operations with 75% coverage | High | Complete |
| T.02 | Redux test | Create tests for CRUD operations on react with 75% coverage | High | Complete |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Name: Zainab | Description | Priority | Status |
| D.01 | Create wireframes | Wireframes to showcase front-end | Low | Complete |
| M.01 | Create Drivers Script | Contains MongoDB CRUD | Medium | Complete |
| M.02 | Create Drivers Collection | Drivers' collection: name, email, and password. | Medium | Complete |
| M.03 | Create Drivers Typedef | Drivers model for Graphql | Medium | Complete |
| M.04 | Create Drivers resolver | CRUD for Drivers collection | Medium | Complete |
| R.01 | Redux Drivers Slice File | Collection of Redux reducer logic and actions for drivers feature. | High | Complete |
| R.02 | React Drivers Reducers | Use of actions to change state | High | Complete |
| R.03 | React Front End | Front-end page for drivers with CRUD capabilities | High | Complete |
| T.01 | GraphQL Test Driver Server | Contains resolver and a function which creates a new server Apollo Server instance. | High | Complete |
| T.02 | GraphQL Test Get Driver Query Test | Contains query to test, a link to the external mock server, and the capability to start and stop the server. | High | Complete |
| T.03 | Create React CRUD test | Tests react components in each of the pages: 75%+ coverage | High | Complete |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Name: Zephania | Description | Priority | Status |
| D.01 | Create wireframes | Wireframes to showcase front-end | Low | Complete |
| M.01 | Create Depots Script | Script to populate database and perform CRUD operations on data | Medium | Complete |
| M.02 | Create project script | Script combining all collection scripts with operation commands | Low | Complete |
| M.03 | Create Depots collection | Run script to create collection | High | Complete |
| M.04 | Create Depots model | Create the model containing information about fields of depot collections | High | Complete |
| M.05 | Create Depots typedef | Methods to query and mutate the database are declared | High | Complete |
| M.06 | Create Depots resolver | CRUD methods to be performed on the database are created | High | Complete |
| R.01 | Redux Depots slice | Creates methods for the Apollo server to perform on the database | High | Complete |
| R.02 | React Depots components | Create components that use data from the database | High | Complete |
| R.03 | React Depots front end | Display the relevant component features in the react redux web app | Medium | Complete |
| T.01 | Create DepotList test | Test graphql | High | Complete |
| T.02 | Create CRUD test | Graphql CRUD operations mocked  75+% coverage | High | Complete |
| T.03 | Create react Depot test | Test react Letter component | High | Complete |
| T.04 | Create react CRUD test | React CRUD operations mocked 75+% coverage | High | Complete |

# Functional Requirements

|  |  |  |
| --- | --- | --- |
| 1 | Queries | Description |
| PK | GetALetter(id) | Retrieves the letter with corresponding id |
| PK | GetAllLetters | Retrieves all letters |
| ZB | getAllDrivers | Returns all drivers |
| ZB | GetADriver(id) | Returns a driver using Id |
| PR | GetAParcel(id) | Returns a parcel with the matching id |
| PR | GetAllParcels | Returns all parcels from the database |
| ZM | GetADepot(id) | Returns the depot document from the array with the matching id to the mutation request |
| ZM | GetAllDepots | Returns all depot documents from database |

|  |  |  |
| --- | --- | --- |
| 2 | Mutations | Description |
| PK | CreateALetter(Letter) | Create a letter with the letter parameters filled |
| PK | UpdateALetter(id, Letter) | Update a letter with the existing id and letter parameters filled |
| PK | DeleteALetter(id) | Delete a letter with the corresponding id |
| ZB | CreateADriver | Create a Driver |
| ZB | UpdateADriver | Update a driver using an existing id |
| ZB | DeleteADriver | Delete a driver with the corresponding id |
| PR | CreateAParcel | Creates a parcel object. |
| PR | UpdateAParcel | Updates field of a parcel with a given id. |
| PR | DeleteAParcel | Delete a parcel with a corresponding id. |
| ZM | CreateADepot | Creates a depot document in the database with a user given city and workers |
| ZM | UpdateADepot | Updates the selected depot document from the database and updates city and worker with user given inputs |
| ZM | DeleteADepot | Removes the selected depot document from the database |

Please see the scripts for CRUD and see express for data types.

|  |  |  |
| --- | --- | --- |
| 3 | Operations Used | Description |
| PK | Letters Collections | CRUD + operations |
| ZM | Depots Collections | Insert many, find many, find one, update one, delete many. Not equal, increment, set, exists, less than |
| ZB | Drivers Collections: CRUD, sort, limit, count. | Insert many, insert one, update one, delete one, find by name and limit to 2. |
| PR | Parcels Collections | CRUD + Operations |
|  | Data Types Used for Project | String, Number, Array, Date, Boolean |

# Technical Architecture

|  |  |  |
| --- | --- | --- |
| 4: Dependency | Description | Use |
| @apollo/client: 3.7.1 | State management library for JS which helps you manage local and remote data with graphQL. | Slicer |
| @reduxjs/toolkit: 1.9.0 | Set of tools to simplify redux development. It includes utilities to create and manage redux stores, write redux actions and reducers. | Slicer |
| @testing-library/jest-dom: 5.16.5 | Companion library for testing library providing custom dom element matchers for jest. | Redux Testing Matching |
| @testing-library/react: 13.4.0 | Testing library to test DOM tree rendered by react on the browser. Helps you write tests on how users would use the website. | Component testing |
| @testing-library/user-event: 14.4.3 | Provides advanced simulation of browser interactions. | Component testing |
| @types/graphql: 14.5.0 | Stub TypeScript definitions entry for graphql, which provides its own types definitions. | Redux Interfaces |
| @types/node: 17.0.45 | Contains type definitions for Node.js |  |
| @types/react: 18.0.25 | Contains type definitions for Node.js |  |
| @types/react-dom: 18.0.9 | Contains type definitions for react dom |  |
| @types/react-router-dom: 5.3.3 | Contains type definitions for react router dom |  |
| bootstrap: 5.2.3 | Front end dev framework. | React Components |
| graphql: 16.6.0 | The JavaScript reference implementation for GraphQL, a query language for APIs. | Slicer |
| mocha: 10.1.0 | JavaScript test framework for Node.js & The Browser | React testing |
| react: 18.2.0 | A JS library for building UI | React project |
| react-bootstrap: 2.6.0 | Bootstrap components built for react | React Components |
| react-dom: 18.2.0 | This package serves as the entry point to the DOM and server renderers for React | React Components |
| react-redux: 8.0.5 | React uses Redux for building the user interface. | React Components |
| react-router-dom: 6.4.3 | Npm package that enables you to implement dynamic routing in a web app. | App tsx |
| redux: 4.2.0 | Redux is a state container designed used to write apps that behave consistently across client, server, and native environments. | React project |
| redux-mock-store: 1.5.4 | A mock store for testing Redux async action creators and middleware. | React testing |
| reselect: 4.1.7 | A library for creating memoized "selector" functions. Commonly used with Redux | React Components |
| typedoc: 0.23.21 | TypeDoc converts comments in TypeScript source code into rendered HTML documentation or a JSON model. | App tsx |
| typescript: 4.9.3 | TypeScript extends JavaScript and improves the developer experience. | React Components |
| @graphql-tools/mock: 8.7.12 | Mock your GraphQL data based on a schema. | Graphql testing |
| @graphql-tools/schema: 9.0.10 | GraphQL server uses a schema to describe the shape of your available data. | Graphql |
| @types/jest: 29.2.3 | Contains type definitions for Jest | Graphql testing |
| husky: 8.0.1 | Modern native Git hooks made easy. | Gitlab |
| jest: 29.3.1 | Jest is a simple, lightweight testing framework that provides a variety of testing capabilities to JavaScript and TypeScript projects. |  |
| prettier: 2.7.1 | Code formatter | All |
| pretty-quick: 3.1.3 | By default, pretty-quick will check your prettier configuration file for any overrides you define to support formatting of additional file extensions. |  |
| supertest-graphql: 1.1.4 | Test API | Graphql testing |
| ts-jest: 29.0.3 | A Jest transformer with source map support that lets you use Jest to test projects written in TypeScript. |  |
| ts-node: 10.9.1 | ts-node is a TypeScript execution engine and REPL for Node.js |  |
| @apollo/server: 4.2.1 | It's a way to build a production-ready, self-documenting GraphQL API that can use data from any source. | Graphql |
| @types/bcryptjs: 2.4.2 | Contains type definitions for bcryptjs | Graphql |
| @types/express: 4.17.14 | Contains type definitions for express | Graphql |
| apollo-server-express: 3.10.4 | The Express integration of Apollo Server, a GraphQL server that works with many Node.js HTTP server frameworks. | Graphql |
| bcryptjs: 2.4.3 | A JS implementation of the bcrypt password hashing function that allows you to easily create a hash out of a password string . | Used to hash the password |
| cors: 2.8.5 | CORS is a node.js package for providing a Connect/Express middleware that can be used to enable CORS with various options. | Graphql |
| dotenv: 16.0.3 | Loads env variables from the .env file. | Graphql |
| express: 4.18.2 | Express is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications. | Graphql |
| graphql: 16.6.0 | GraphQL.js is a general-purpose library and can be used both in a Node server and in the browser. | Graphql |
| mongoose: 6.7.0 | Manage collections on MongoDB. | Graphql |
| winston: 3.8.2 | Winston is the most popular logging library for Node. Js | Graphql |

|  |  |  |
| --- | --- | --- |
| 5 | Database Collections | Description |
| PK | Letters Collections | This collection contains address, delivered, dispatched and delivery class. |
| ZM | Depots Collections | This collection contains city, workers and an array of services. |
| ZB | Drivers Collections | This collection contains driver's login details: name, email, and password. |
| PR | Parcels Collections |  |

|  |  |  |
| --- | --- | --- |
| 6 | React Components | Description |
| PK | Header | Navbar for navigation |
| PK | Footer | footer |
| PK | Letter | Lets user see all letters |
| PK | Create Letter | Lets user create a letter |
| PK | Delete Letter | Lets user delete a letter |
| PK | Edit Letter | Lets user update a letter |
| ZB | Driver | Lets user see all driver |
| ZB | Create Driver | Lets user create a driver |
| ZB | Delete Driver | Lets user delete a driver |
| ZB | Update Driver | Lets user update a driver |
| PR | Parcel | Lets user see all parcel |
| PR | Create Parcel | Lets user create a parcel |
| PR | Delete Parcel | Lets user delete a parcel |
| PR | Update Parcel | Lets user update a parcel |
| ZM | Depot | Lets user see all depot |
| ZM | Create Depot | Lets user create a depot |
| ZM | Delete Depot | Lets user delete a depot |
| ZM | Update Depot | Lets user update a depot |
| ZM | Depot Details | Lets user view table of services for each depot |
| ZM | Depot Page | Lets user view details, update and delete components (in one page) |

# **Running Files**

* Clone post-office-management-system Linked [here](https://git.fdmgroup.com/Prithvi.Kulkarni/post-office-management-system.git).
* Before running the express folder, the databases are created as using the dump folder.
* Unzip the folder called dump to recreate the database.

## NPM Install and Start

### Apollo Express

* Navigate to **post-office-graphql-server**
* Use this command to reinstall the modules folder: **npm install.**
* Run this command on the terminal: **npm run start.**

### React Redux App

* Navigate to **post-office-react-redux**
* Use this command to reinstall the modules folder: **npm install.**
* Run this command on the terminal: **npm run start.**

### .ENV File for Apollo Express

NODE\_ENV= 'development';

SERVER\_HOST = '127.0.0.1';

SERVER\_PORT= 4556

### Documentation

Documentation was done for the react redux application App.tsx.

npx typedoc src/App.tsx

### Logging

Logging was done for the apollo express application.