Report For Parcel Delivery System

March 26, 2023

1 Abstract

This report presents the design, implementation, and testing of a parcel delivery system developed using Spring Boot, React, MongoDB and Junit. The system was developed to simplify the process of sending and receiving parcels.

The report begins with an introduction to the project, explaining the motivations and goals behind its development. This is followed by a discussion of the design and architecture of the system, which includes the database schema, API endpoints, and front-end interface developed using React.

The development process section details the methodologies and tools used during the software development lifecycle, including the use of agile development methodologies and continuous integration and delivery practices.

The implementation section provides a detailed description of the system's features, including user registration and login,placing order. The testing and validation section covers the various types of tests performed, including unit tests to ensure the system's functionality and quality.

The user interface section presents the system's visual design and layout, highlighting user interaction and navigation. The conclusion section summarizes the report and discusses the main findings, limitations, and future work.

Based on the findings, we recommend further development of the system by incorporating additional features such as automated scheduling of pickups and deliveries, real-time tracking of delivery vehicles, and integration with popular e-commerce platforms.

Overall, the system offers a user-friendly and efficient solution for parcel delivery, and the use of Spring Boot, React, and MongoDB provide a scalable and robust architecture for future development.

2 Introduction

A parcel delivery system is an essential service that allows customers to send and receive packages quickly and efficiently. To streamline this process, an app can be developed using Spring Boot, MongoDB, and React, providing users with an intuitive and easy-to-use interface.

The app's main use case for recipients is to place orders for package deliveries. The recipient will log in to the app, enter their delivery address, package details. Once the order is placed, the app will notify the driver assigned to the delivery.

Another important use case for recipients is to provide feedback on the delivery experience. The recipient can rate the delivery and leave a review, which will be visible to other recipients using the app.

Additionally, recipients can view all feedback provided by other users. This feature provides valuable insight into the quality of service offered by the delivery service, enabling recipients to make more informed decisions when selecting a courier.

For drivers, the app provides use cases such as taking orders, delivering orders, providing feedback, and viewing all feedback.. Once they accept an order, they can use the app to navigate to the delivery location and complete the delivery process.

After completing the delivery, drivers can provide feedback on the experience, including any issues or challenges they faced during the delivery process. This feedback is essential for the delivery service to continuously improve and provide a better experience for all users.

Finally, drivers can also view all feedback provided by other users. This feature enables drivers to gain insights into the quality of service provided by the delivery service, allowing them to identify areas for improvement and enhance their performance.

Overall, the Spring Boot, MongoDB, and React-based parcel delivery system app provides an intuitive and efficient way for recipients and drivers to manage the delivery process. With features such as order placement, feedback, and real-time tracking, the app provides a comprehensive delivery service experience for all users.

3 Development Process

The development process for creating a parcel delivery system with a REST API using Spring Boot and testing it with JUnit, followed by the development of the frontend using React, can be summarized in the following steps:

I began by designing the system, identifying the key features and requirements, and planning how to implement them. I then set up the development environment, installing and configuring the necessary tools, such as the Java Development Kit, Spring Boot, and JUnit.

Next, I developed the REST API using Spring Boot, defining the endpoints and implementing the logic required to perform the necessary operations, such as creating new orders, updating order status, and retrieving order information. I ensured that the API followed RESTful principles and was easy to use and understand.

After developing the REST API, I tested it thoroughly using JUnit. I wrote unit tests for each endpoint and integration tests to test the entire API, ensuring that it worked correctly and met the requirements.

I then developed the frontend of the parcel delivery system using React. This involved designing the user interface and implementing the logic required to interact with the REST API, such as creating new orders, updating order status, and retrieving order information.

Once the frontend was developed, I integrated it with the REST API using Axios, a popular JavaScript library, to make HTTP requests to the backend and handle the responses.

Finally, I tested the system end-to-end, performing tests such as user acceptance testing to ensure that the system was user-friendly and met the requirements.

In conclusion, the development process for creating a parcel delivery system with a REST API using Spring Boot and testing it with JUnit, followed by the development of the frontend using React, required careful planning, designing, and testing at each stage of the development process.

4 Tools Used for this project and Justification of My Selections

For the creation of restful api I have used spring boot.

4.1 Why Spring Boot

Spring Boot is a popular framework for building Java-based web applications because it simplifies development and reduces boilerplate code. Its automatic configuration and embedded servers make it easy to build standalone, production-ready applications quickly. Additionally, Spring Boot offers testing and monitoring tools, making it a convenient and efficient choice for developers.

For the DataBase I have used MongoDB

4.2 Why MongoDB

MongoDB is a popular NoSQL database because of its flexibility, scalability, and performance. It allows developers to store and manage large amounts of data easily and efficiently, and its dynamic schema and powerful query language make it easy to adapt to changing data requirements. Additionally, MongoDB's distributed architecture and automatic sharding make it highly scalable, able to handle large amounts of data and traffic. Overall, MongoDB is a powerful and flexible database solution that is widely used by developers and enterprises around the world.

For the Testing of Restful api I have used postman and Junit

4.3 Why Postman

Postman is a popular API development tool because it simplifies the process of testing, documenting, and sharing APIs. It provides an easy-to-use interface for

sending requests, inspecting responses, and testing API endpoints. Additionally, Postman offers a range of features such as automated testing, team collaboration, and API documentation, making it a convenient and efficient choice for developers. Overall, Postman is a powerful tool for building and testing APIs, and its user-friendly interface and range of features make it a go-to choice for many developers and enterprises.

4.4 Why Junit

JUnit is a popular Java testing framework because it simplifies the process of writing and running tests for Java applications. It provides a range of annotations, assertions, and test runners that make it easy to write and run tests, and its integration with popular Java IDEs like Eclipse and IntelliJ makes it convenient for developers. Additionally, JUnit offers support for parameterized tests, test suites, and other advanced testing features, making it a versatile choice for developers. Overall, JUnit is a powerful and widely-used testing framework for Java applications, and its simplicity and range of features make it a go-to choice for many developers and enterprises.

For the Frontend of the application I have used React.

4.5 Why React

React is a popular JavaScript library for building user interfaces because it simplifies the process of building complex UIs with reusable components. It provides a declarative programming model that allows developers to write UIs as a function of their state, making it easier to reason about and test code. Additionally, React's virtual DOM and efficient rendering algorithm make it fast and performant, even for complex UIs. Overall, React is a powerful and widely-used library for building user interfaces, and its simplicity and efficiency make it a popular choice for many developers and enterprises.

5 Design

5.1 Actors

- Driver
- Recipient

5.2 Business Requirements

- Place orders
- Real-time tracking
- Driver management

- Route optimization
- Payment processing
- Feedback and ratings
- Security and privacy

5.3 Functional Requirements

- **FR 1**: Register User
- **FR 2**: Login User
- **FR 3**: Take Order
- **FR 4**: Place Order
- **FR 5**: Deliver Order
- **FR 6**: Provide Feedback
- **FR 7**: View All Feedbacks

5.4 Non-Functional Requirements

- **FR 1**: Stability: System version should be stable and bugs free.
- **FR 2**: Availability: System should be Cross Platform and available on all platforms.
- **FR 3**: Security: System should have secure payment gateway. Security should be the main priority to avoid any system breaches and to avoid any data loss.
- **FR 4**: Scalability: System should have the ability to handle growth, especially in handling more users and evolving concurrently with the business needs.
- **FR 5**: Relaibility: Probability of Failure should be less.
- **FR 6**: Usability: Easy to use and user-friendly interface • NFR7: Maintainability: System should be maintainable. If
- **FR 7**: Maintainability: System should be maintainable. If business want some new features in the system, those changes must not affect the reliability and quality of the system.

5.5 Use Case Diagram

UseCase Diagram for parcel delivery system is shown in Figure 1.

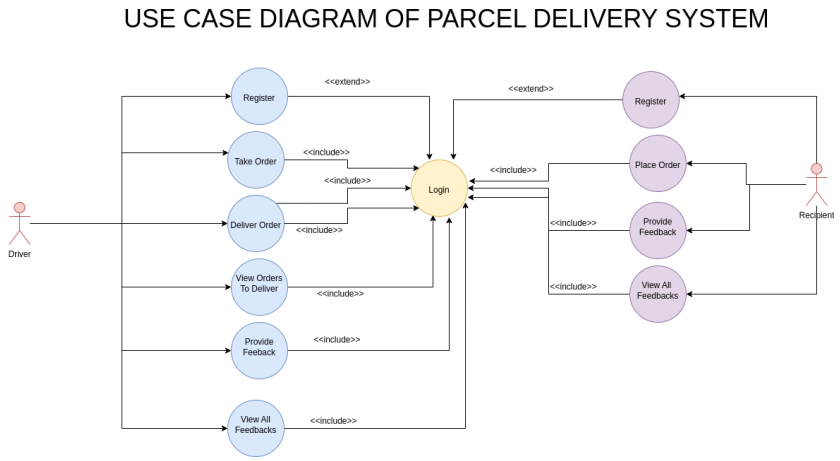

Figure 1: UseCase Diagram

5.6 Swimlane Diagram

Swimlane Diagrams for parcel delivery system is shown in fugure 2 and 3.

5.7 Class Diagram

Class Diagram for parcel delivery system is shown in figure 4.

6 Result of Testing

I have tested all the api using Junit and postman and all the tests are passed.

7 Recommendation for Furthur Development

Here are some recommendations for further development of a parcel delivery system:

- Payment methods: Integrate multiple payment methods to make it easier for customers to pay for their orders. Offer options such as credit card, debit card, PayPal, Google Pay, Apple Pay, and more.
- Date and time of delivery: Allow customers to select a preferred delivery date and time at the time of placing their order. This will give them greater control over when their package will arrive and help prevent missed deliveries.
- Order tracking: Implement a real-time order tracking system that allows customers to track their package from the moment it is dispatched until it

SWIMLANE DIAGRAM 2: Driver

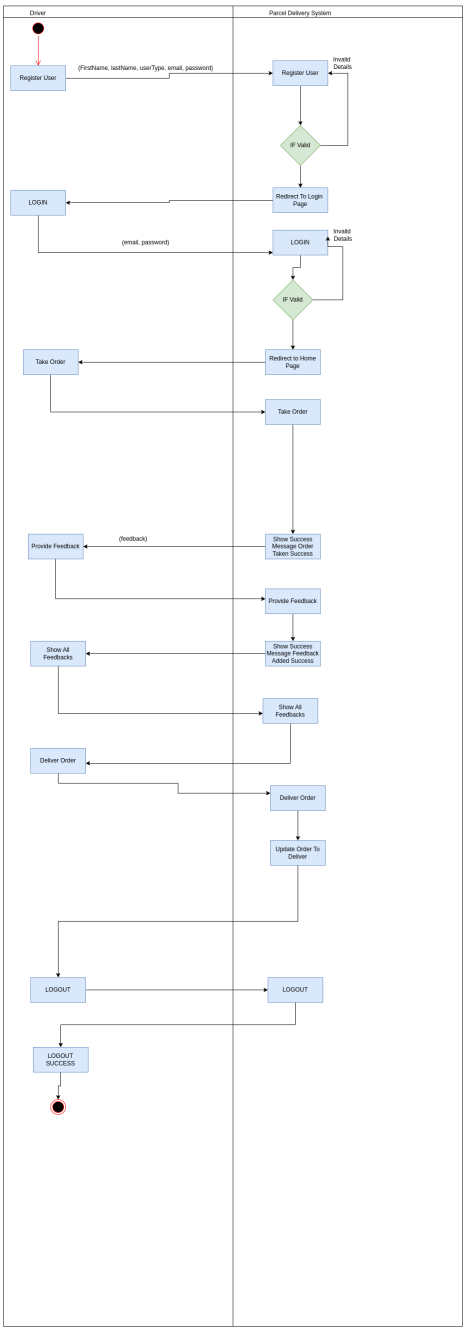

Figure 2: Swimlane Diagram 1

SWIMLANE DIAGRAM 1: Recipient

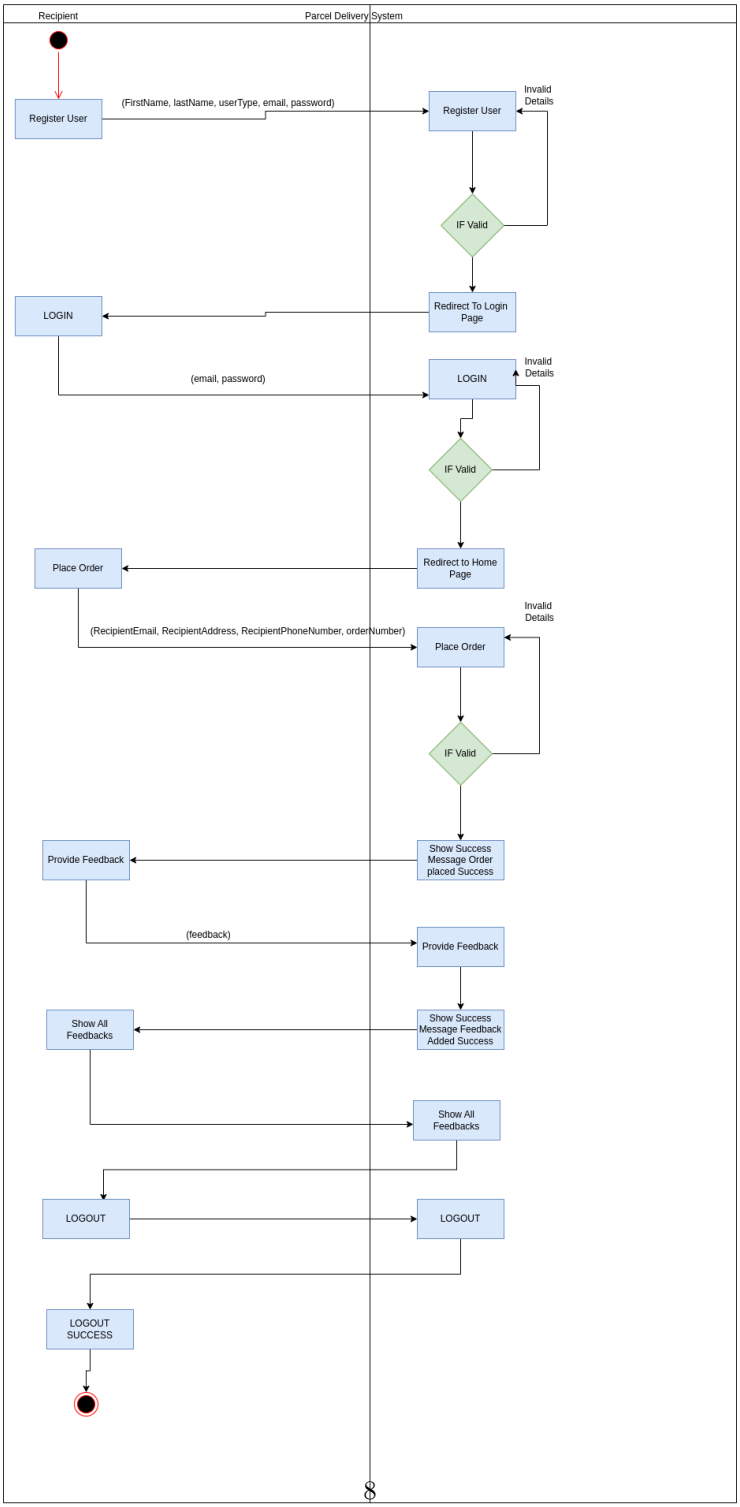

Figure 3: Swimlane Diagram 2

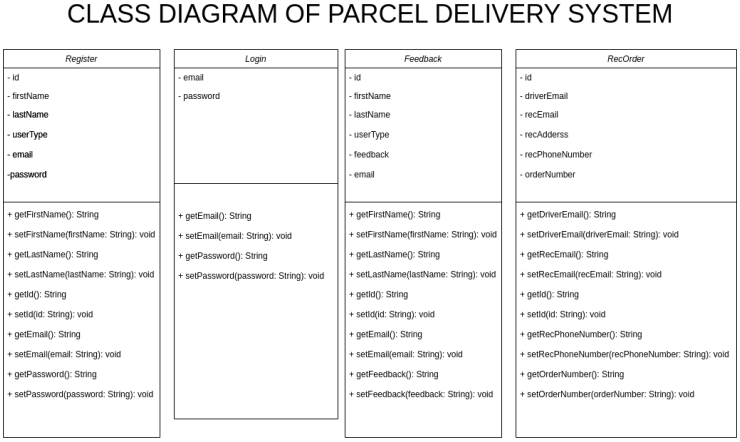

Figure 4: Class Diagram

reaches its final destination. This will help customers stay informed about the status of their delivery and avoid any surprises.

- Location tracking: Use GPS technology to track the location of delivery drivers in real-time. This will allow customers to see the estimated time of arrival of their package and make any necessary adjustments to their schedule.

- Change details of the order: Allow customers to make changes to their order, such as the delivery address or preferred delivery date, up until the moment the package is dispatched. This will give them greater flexibility and reduce the risk of missed deliveries.

By implementing these features, you can create a more user-friendly and efficient parcel delivery system that meets the needs of today's customers.