RAJIV GANDHI INSTITUTE OF TECHNOLOGY

GOVERNMENT ENGINEERING COLLEGE

KOTTAYAM-686 501

DEPARTMENT OF
COMPUTER SCIENCE AND ENGINEERING

CSD 334 MINI PROJECT REPORT

FastTrack

Report Submitted by

Jessin Sunny (Reg. No: KTE22CS036)
Akhil S Nair (Reg. No: KTE22CS009)
Alwin Philip (Reg. No: KTE22CS012)
Edwin Varkey (Reg. No: KTE22CS028)

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY
THIRUVANANTHAPURAM

APRIL 2025

RAJIV GANDHI INSTITUTE OF TECHNOLOGY

GOVERNMENT ENGINEERING COLLEGE

KOTTAYAM-686 501

DEPARTMENT OF
COMPUTER SCIENCE AND ENGINEERING

Vision of the Department

To be a centre of excellence for inclusively nurturing young minds to become innovative computing professionals committed to sustainable development and societal empowerment.

Mission of the Department

- To offer a solid foundation in computing and technology for crafting competent professionals.
- To promote innovative and entrepreneurial skills of students by exposing them to the forefront of developments in the field of computing.
- To inculcate strong ethical values in the young minds to work with commitment for the progress of the nation.

RAJIV GANDHI INSTITUTE OF TECHNOLOGY

GOVERNMENT ENGINEERING COLLEGE

KOTTAYAM-686 501

DEPARTMENT OF
COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that this report entitled ***FastTrack*** *is an authentic report of the project done by* ***Jessin Sunny (Reg. No: KTE22CS036), Akhil S Nair (Reg. No: KTE22CS009), Alwin Philip (Reg. No: KTE22CS012) and Edwin Varkey (Reg. No: KTE22CS028)*** *during the academic year* ***2024-25****, in partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology in Computer Science and Engineering of APJ Abdul Kalam Technological University, Thiruvananthapuram.*

GUIDE COORDINATOR

HEAD OF THE DEPARTMENT

Acknowledgement

Every successful project is a result of dedication, hard work, and the invaluable support of many individuals. We sincerely appreciate the guidance, encouragement, and contributions of everyone who played a role in making this project a success.

We express our sincere gratitude to **Dr. Prince A., Principal, RIT Kottayam** for making the resources available at the right time without which this project would not have been a success.

We take this opportunity to express a deep sense of gratitude to **Prof. Kavitha N., Associate Professor & Head, Department of Computer Science and Engineering**.

We also take this opportunity to express our profound gratitude and deep regards to our guide **Prof. Anu Bonia Francis, Assistant Professor** for her exemplary guidance, monitoring and constant encouragement throughout the course of this project. The blessing, help and guidance given by her from time to time shall carry us a long way in the journey of life on which we are about to embark.

We also express our gratitude to Project Coordinators **Prof. Anu Bonia Francis, Assistant Professor** and **Dr. Raji R. Pillai** for the cordial support, valuable information and guidance, which helped us in completing this task through various stages.

We would also like to express our gratitude to **teaching** and **non-teaching staffs** for providing us with the necessary facilities and support during our Mini Project hours, which greatly contributed to the completion of our project

Last, but not least, we thank **almighty**, **our parents** and **friends** for their constant encouragement without which this project would not have been possible.

Jessin Sunny
Akhil S Nair
Alwin Philip
Edwin Varkey

Declaration

We, the undersigned hereby declare that the project report entitled **FastTrack**, submitted for partial fulfilment of the requirements for the award of the degree of Bachelor of Technology of the **APJ Abdul Kalam Technological University, Kerala** is a bonafide work done by us under the supervision of **Prof. Anu Bonia Francis, Assistant Professor**. This submission represents our idea in our own words where ideas or words of others have not been included; we have adequately and accurately cited and referenced the original sources. We have adhered to the ethics of academic honesty and integrity and have not misrepresented or fabricated any data, idea or on our submission. We understand that any violation of the above can result in disciplinary actions from the institute and/ or University and can evoke penal action from the sources which have not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed as the basis for awarding any degree, diploma or similar title of any other university.

Name of Students: Signatures:

Jessin Sunny (Reg. No.: KTE22CS036)

Akhil S Nair (Reg. No.: KTE22CS009)

Alwin Philip (Reg. No.: KTE22CS012)

Edwin Varkey (Reg. No.: KTE22CS028)

Batch: 2022 - 2026 March 28, 2025

Abstract

Package assignment and efficient route planning are critical aspect of logistics and delivery services. Traditional methods often result in increased delivery times, fuel consumption, and operational costs. This project aims to develop a system that assigns packages efficiently and calculates the shortest and efficient delivery route for a delivery person by leveraging graph theory principles. The solution addresses the need for optimization in delivery services, reducing travel time and resource utilization while improving customer satisfaction.

The package assignment process follows an optimized allocation strategy to ensure minimal travel distance and efficient workload distribution. Packages are first assigned to delivery personnel based on their designated quadrant, reducing unnecessary cross-region travel. If a delivery agent has additional capacity, the system dynamically reallocates packages from nearby quadrants to maximize efficiency. Furthermore, the system prioritizes urgent deliveries and adapts to real-time constraints, such as vehicle capacity and traffic conditions, ensuring timely and cost-effective deliveries. If any packages remain unassigned, they are scheduled for the next delivery cycle with high priority.

The proposed system will utilize graph-based algorithms, specifically Greedy Algorithm, to compute the shortest path between multiple delivery locations. Delivery points will be represented as nodes in a graph, and the paths connecting them as weighted edges, where the weights denote distance. The system will process input data such as the list of delivery locations and dynamically generate the efficient route for completing all deliveries.

The system consists of a web platform for company operations and a mobile app for delivery personnel. The website, built with HTML, CSS, and JavaScript, enables company registration, employee management, package uploads, and delivery assignments. The backend, developed using Python with Firebase Firestore, handles data storage and synchronization. The mobile app, built with Flutter, provides real-time navigation, package details, and notifications, ensuring seamless tracking and efficient delivery operations.

The expected outcome of this project is a reliable and user-friendly routing system that minimizes delivery time and resource usage. The solution demonstrates the practical application of graph theory in real-world scenarios and offers significant benefits to logistics companies and delivery personnel by enhancing operational efficiency and productivity.

Contents

Acknowledgement i

Declaration ii

Abstract iii

List of Figures ix

List of Algorithms x

1 Introduction 1

1.1 Objectives . . . 1
1.2 Motivation . . . 1
1.3 Scope of the Project . . . 2
1.3.1 Need . . . 2
1.3.2 Deliverables . . . 2
1.3.3 Exclusions . . . 2
1.3.4 Assumptions . . . 3
1.4 Prerequisites for the Reader . . . 3
1.5 Organization of Report . . . 3

2 System Study and Requirements Engineering 5

2.1 Existing System . . . 5
2.2 Gap Analysis . . . 6
2.3 Proposed System . . . 6
2.3.1 Problem Statement . . . 7
2.3.2 System Model . . . 7
2.4 Requirements Engineering . . . 8
2.4.1 Feasibility Study . . . 8
2.4.1.1 Economic Feasibility . . . 8
2.4.1.2 Technical Feasibility . . . 9
2.4.1.3 Behavioral Feasibility . . . 9
2.4.2 Requirements Elicitation . . . 9
2.4.3 Requirement Analysis . . . 10
2.4.3.1 Functional Requirements . . . 10
2.4.3.2 Non-Functional Requirements . . . 10
2.4.4 Requirements Specification . . . 10
2.4.4.1 Functional Requirements . . . 11

2.4.4.2 Non-Functional Requirements . . . 11
2.4.5 Requirements Validation . . . 11
2.5 Summary . . . 12

3 System Design and Methodology **13**
3.1 System Design . . . 13
3.1.1 Data Flow Diagrams . . . 14
3.1.1.1 Level 0 Data Flow . . . 14
3.1.1.2 Level 1 Data Flow . . . 15
3.1.2 Use Case Diagram . . . 16
3.1.3 System Architecture Diagram . . . 17
3.1.4 Entity-Relationship (ER) Diagram . . . 18
3.1.5 Algorithm Modules . . . 18
3.2 Detailed Design . . . 19
3.2.1 Authentication Algorithm . . . 19
3.2.2 Company Registration . . . 20
3.2.3 Update Information Algorithm . . . 21
3.2.4 Validate Registration Algorithm . . . 21
3.2.5 Package Assignment Algorithm . . . 22
3.2.6 Scan Package . . . 26
3.2.7 Retrieve efficient Route After Scanning Packages . . . 26
3.2.8 Route Optimization with Google Maps API Algorithm . . . 27
3.3 Summary . . . 27

4 Implementation **28**
4.1 Software Requirements . . . 28
4.1.1 Operating System Requirements . . . 28
4.1.2 Tools Used . . . 28
4.1.2.1 HTML . . . 28
4.1.2.2 CSS . . . 28
4.1.2.3 JavaScript . . . 29
4.1.2.4 Python . . . 29
4.1.2.5 Flutter . . . 29
4.1.2.6 Firebase . . . 29
4.1.2.7 Google Maps API . . . 30
4.1.2.8 Supabase . . . 30
4.2 Hardware Requirements . . . 30
4.2.1 Website . . . 30
4.2.2 Mobile App . . . 30

4.3 Code Implementation Details . . . 31
4.3.1 Authentication & Authorization . . . 31
4.3.2 Package Assignment . . . 31
4.3.3 Employee Assignment . . . 31
4.3.4 Route Optimization & Navigation . . . 31
4.3.5 Real-time Updates & Notifications . . . 31
4.3.6 Screenshots . . . 32
4.4 Features and Functionalities . . . 34
4.4.1 User Authentication & Management . . . 34
4.4.2 Package Handling & Assignment . . . 34
4.4.3 QR Code System . . . 34
4.4.4 Route Optimization . . . 34
4.5 Summary . . . 34

5 Testing **36**
5.1 Testing Strategies Used . . . 36
5.1.1 Unit Testing . . . 36
5.1.1.1 Black Box Testing . . . 36
5.1.1.2 White Box Testing . . . 37
5.1.2 Integration testing . . . 37
5.2 Testing Results . . . 37
5.2.1 Results of Black Box Testing . . . 37
5.2.2 Results of White Box Testing . . . 38
5.2.3 Results of Integration Testing . . . 38
5.3 Test Cases . . . 38
5.4 Summary . . . 40

6 Results and Discussion **41**
6.1 Results . . . 41
6.1.1 Login & Registration Page . . . 41
6.1.2 Package & Employee Details Page . . . 42
6.1.3 QR Code Scanning Page . . . 42
6.1.4 Route Optimization & Navigation Page . . . 43
6.2 Performance Analysis . . . 44
6.3 Challenges Faced and Solution . . . 45
6.3.1 Package Assignment . . . 45
6.3.2 Route Optimization . . . 45
6.4 Summary . . . 45

7 Conclusion and Future Scope **47**
7.1 Summary . . . 47
7.2 Recommendations for Future Work . . . 47
A.1 Introduction . . . 49
A.1.1 Purpose . . . 49
A.1.2 Document Conventions . . . 49
A.1.3 Intended Audience and Reading Suggestions . . . 49
A.1.4 Project Scope . . . 50
A.1.5 References . . . 50
A.2 Overall Description . . . 51
A.2.1 Product Perspective . . . 51
A.2.2 Product Features . . . 51
A.2.3 User Classes and Characteristics . . . 51
A.2.4 Operating Environment . . . 52
A.2.5 Design and Implementation Constraints . . . 52
A.2.6 User Documentation . . . 52
A.2.7 Assumptions and Dependencies . . . 53
A.3 System Features . . . 53
A.3.1 Route Optimization . . . 53
A.3.1.1 Description and Priority . . . 53
A.3.1.2 Stimulus/Response Sequences . . . 54
A.3.1.3 Functional Requirements . . . 54
A.3.2 Dynamic Route Adjustment . . . 54
A.3.2.1 Description and Priority . . . 54
A.3.2.2 Stimulus/Response Sequences . . . 54
A.3.2.3 Functional Requirements . . . 55
A.3.3 Proof of Delivery and Documentation . . . 55
A.3.3.1 Description and Priority . . . 55
A.3.3.2 Stimulus/Response Sequences . . . 55
A.3.3.3 Functional Requirements . . . 55
A.3.4 Dynamic Data-Driven Delivery Insights . . . 55
A.3.4.1 Description and Priority . . . 55
A.3.4.2 Stimulus/Response Sequences . . . 56
A.3.4.3 Functional Requirement . . . 56
A.3.5 Route Export and Data Integration . . . 56
A.3.5.1 Description and Priority . . . 56
A.3.5.2 Stimulus/Response Sequences . . . 56
A.3.5.3 Functional Requirements . . . 57
A.4 External Interface Requirements . . . 57

A.4.1 User Interfaces . . . 57
A.4.2 Hardware Interfaces . . . 57
A.4.3 Software Interfaces . . . 58
A.4.4 Communications Interfaces . . . 58
A.5 Other Nonfunctional Requirements . . . 58
A.5.1 Performance Requirements . . . 58
A.5.2 Safety Requirements . . . 59
A.5.3 Security Requirements . . . 59
A.5.4 Software Quality Attributes . . . 59
A.6 Other Requirements . . . 60
A.6.1 Database Requirements . . . 60
A.6.2 Internationalization Requirements . . . 60
A.6.3 Legal Requirements . . . 60
A.6.4 Backup and Recovery . . . 60
A.7 Appendix A: Glossary . . . 60
A.8 Appendix B: Analysis Models . . . 61
A.9 Appendix C: Issues List . . . 61

Software Requirements Specification **49**

Index **64**

List of Figures

3.1 Level 0 Data Flow 14
3.2 Level 1 Data Flow 15
3.3 Use Case Diagram 16
3.4 System Architecture Diagram 17
3.5 ER Diagram 18

4.1 Authentication & Authorization 32
4.2 Package Assignment 32
4.3 Employee Assignment 33
4.4 Route Optimization and Navigation 33
4.5 Notifications 33

6.1 Company Login Page 41
6.2 Company Registration Page 41
6.3 Delivery Personnel Page 42
6.4 Package Details Page 42
6.5 Employee Details Page 42
6.6 QR Code Scanning 43
6.7 Route Optimization and Navigation Page 43
6.8 Package Assignment Efficiency Analysis 44
6.9 Route Optimization Performance Analysis 44

List of Algorithms

1 Delivery Personnel or Company or Admin Authentication Algorithm 19
2 Company Registration Algorithm 20
3 Update Information 21
4 Validate Registration 21
5 Package Assignment 22
6 Categorize Packages 23
7 Assigning Algorithm for Bike and Truck Employees 24
8 Assigning Packages to Bike Employees 24
9 Truck Employee Allocation and Package Assignment 25
10 Scan Package 26
11 Retrieve efficient Route After Scanning Packages Algorithm 26
12 Route Optimization with Google Maps API 27

Chapter 1

Introduction

Efficient logistics management is crucial for seamless delivery operations, requiring effective coordination between companies and delivery personnel. This project presents an integrated platform consisting of a website and a mobile app, designed to streamline the package assignment and delivery process.

The website enables companies to register, manage employees, upload daily package details, and assign deliveries. The mobile app, developed with Flutter, provides delivery personnel with real-time navigation, package details, and notifications, ensuring smooth execution of assigned tasks.

By leveraging Firebase Firestore for seamless data synchronization, the system enhances communication between all stakeholders. The combination of automated package assignment, route optimization, and real-time tracking makes this platform a powerful tool for improving delivery efficiency and operational transparency.

1.1 Objectives

This project aims to achieve the following objectives:

1. Package Assignment: Automate the assignment of packages to available delivery personnel based on location, workload and efficiency.

2. Route Generation & Optimization: Generate optimized delivery routes to minimize travel time and enhance efficiency.

3. Employee & Delivery Management: Enable companies to register employees, manage their availability, and oversee delivery operations effectively.

1.2 Motivation

1. Optimizing Package Assignment: Manual package allocation can be time-consuming and error-prone. Automating this process ensures fair distribution of workload and improves overall efficiency.

2. Enhancing Route Planning: Delivery personnel often face delays due to inefficient routing. By integrating real-time route generation and optimization, this system helps reduce delivery time and fuel costs.

3. Improving Communication & Management: Effective coordination between companies and delivery personnel is crucial for smooth operations. The system ensures real-time updates, notifications, and seamless data synchronization, enabling better management and processing of deliveries.

1.3 Scope of the Project

1.3.1 Need

- Efficient Delivery Management: The growing demand for streamlined logistics operations necessitates a system that automates package assignment and optimizes delivery routes. This project aims to improve efficiency in managing deliveries
- Optimized Route Planning: Inefficient routing leads to increased delivery times and operational costs. The project addresses this by integrating automated route generation and optimization for faster and cost-effective deliveries.
- Seamless Communication: Effective coordination between companies and delivery personnel is crucial for smooth operations. This project ensures real-time data synchronization and notifications to enhance communication and tracking.

1.3.2 Deliverables

- Delivery Management System: A website for companies to register, manage employees, upload package details and assign deliveries.
- Mobile App for Delivery Personnel: A Flutter-based app providing real-time navigation, package details and notifications.
- Automated Package Assignment: A mechanism to allocate packages efficiently based on employee availability and workload.
- Efficient Route Generation: Implementation of a system that dynamically calculates the best delivery routes to reduce time and fuel costs.

1.3.3 Exclusions

- Third-Party Delivery Integration: The system will focus on internal company operations and will not integrate with external courier or logistics platforms.
- Offline Functionality: The system requires an active internet connection for real-time data synchronization and will not support offline operations.

- Advanced AI-Based Predictions: While route optimization will be implemented, advanced AI-driven demand forecasting and predictive analytics are not within the scope of this project.

1.3.4 Assumptions

- Active Internet Connectivity: It is assumed that both the website and mobile app users will have stable internet access for real-time data updates.
- Employee Availability: The system assumes that employees marked as available for deliveries are ready to take assignments.
- Route Feasibility: The system assumes that generated routes are feasible based on map APIs.

1.4 Prerequisites for the Reader

- Basic Understanding of Delivery Management: Readers should have a fundamental understanding of how package deliveries are assigned, routed and tracked in logistics operations.
- Knowledge of Graph Theory: Since delivery routes are represented as graphs, readers should understand concepts like nodes (locations), edges (routes), weighted graphs (distance-based costs) and shortest path algorithms.
- Familiarity with Navigation Systems: Readers should be familiar with basic mapping and navigation concepts, including route optimization, GPS-based tracking and location-based services.

1.5 Organization of Report

Chapter 1 provides an introduction to FastTrack , a software project designed to optimize delivery routes and enhance logistics efficiency. The project aims to improve workforce utilization and minimize delivery delays by integrating real-time tracking, automated personnel allocation, and route optimization. The system consists of a Flutter-based mobile application connected to a Flask backend, utilizing Firebase Firestore for real-time data synchronization and Google Maps API for navigation.

Chapter 2 covers system study and requirements engineering, detailing the functional and non-functional requirements of the application. Functional aspects include authentication, real-time route tracking, package assignment, and push notifications. The system also ensures

automatic conflict resolution by reallocating tasks when a delivery personnel is unavailable. Non-functional requirements focus on performance benchmarks, system reliability, security, and scalability. The project emphasizes user-friendly interface design with interactive maps and navigation features. External interfaces include Firebase for cloud storage, Supabase for image handling, and Google Maps API for route calculations.

Chapter 3 elaborates on system design and methodology, describing core modules such as authentication, route optimization, package assignment, and personnel management. The methodology includes algorithm selection for optimized route calculations, real-time traffic analysis, and efficient personnel allocation strategies. The chapter also covers database structuring, data flow models, and API integration to ensure seamless operation.

Chapter 4 discusses the implementation phase, detailing the tools and technologies used in the project. The mobile application is built using Flutter, while the backend is developed in Flask, with Firebase handling data synchronization. Key features such as login authentication, real-time navigation updates, package tracking, and Firebase notifications are implemented. The system dynamically updates routes and personnel assignments to ensure efficient delivery operations.

Chapter 5 focuses on testing strategies, including unit testing, integration testing, and black-box testing. Various test cases verify location tracking accuracy, route optimization efficiency, and package assignment reliability. The system undergoes rigorous validation to ensure seamless performance across different devices and network conditions.

Chapter 6 presents the results and discussion, highlighting the successful implementation of real-time navigation and optimized personnel allocation. The mobile application enables delivery personnel to access dynamically assigned routes while companies monitor deliveries through a web portal. Firebase notifications ensure instant updates. The project achieves its goals of minimizing delivery time, optimizing resource allocation, and improving last-mile logistics.

Chapter 7 concludes the document with a summary of achievements and outlines future scope. Possible enhancements include AI-driven route optimization, improved GPS accuracy, and machine learning-based predictive analytics for demand forecasting. Future work also considers expanding system capabilities for larger-scale logistics operations.

Chapter 2

System Study and Requirements Engineering

System Study and Requirements Engineering are crucial phases in the software development lifecycle, playing a pivotal role in ensuring the successful delivery of a software project. System Study involves a comprehensive analysis of the existing system, understanding its functionalities, identifying shortcomings, and determining the requirements for a new or improved system.This phase lays the foundation for Requirements Engineering, which focuses on eliciting, documenting, validating, and managing the functional and non-functional requirements of the software system.

2.1 Existing System

For the FastTrack project, there are existing systems and software solutions with similar functionalities. In the context of package assignment, various logistics platforms optimizes deliveries by considering location clustering, personnel availability, deadlines, and real-time traffic to ensure efficient distribution. Algorithms dynamically allocate packages to minimize delays and maximize resource utilization. In the context of route optimization, various logistics and delivery platforms help businesses optimize multiple routes by minimizing travel time and fuel costs. These applications leverage algorithmic methods to determine efficient routes dynamically based on real-time data.
A literature review is a critical and comprehensive analysis of existing research studies and methodologies related to package assignment and route optimization. It helps in understanding different algorithmic approaches and their efficiency in solving real-world routing problems. Following are some of the research papers and existing system related to package assignment and route optimization:

- Greedy Algorithm for Multi-Route Optimization: This study explores how the Greedy Algorithm is applied in route optimization, making decisions based on local optimization at each step to reduce total travel distance and time. Although it does not guarantee a globally optimal solution, it provides an efficient and scalable approach for large-scale delivery operations. [1]

- Bin Packing Algorithms in Logistics: This research explores the application of Bin Packing Algorithms to package assignment problems. By modeling delivery vehicles

as bins with limited capacity, packages are assigned in a way that maximizes space utilization and minimizes the number of vehicles required. Simple heuristics like the First-Fit and Best-Fit algorithms provide efficient and scalable solutions, though they may not always achieve optimal results. [2]

- Amazon Logistics: It optimizes package assignment using AI-driven algorithms that consider location, driver availability, and real-time traffic to ensure fast and efficient deliveries. Its Amazon Routing Engine (ARE) dynamically calculates optimal routes, reducing delays and operational costs. [3]

- Google Maps API: It provides route planning and optimization capabilities. It allows users to generate the shortest and fastest routes using real-time traffic data and alternative route suggestions. Businesses integrate it into their logistics systems to improve route planning accuracy. [4]

2.2 Gap Analysis

1. Package Assignment Optimization: The current system assigns packages based on predefined criteria, but it could benefit from advanced optimization techniques, such as AI-driven predictive analytics or dynamic reassignment based on real-time factors like traffic and personnel availability.

2. Employee Routing Efficiency: The navigation system calculates optimal routes, but incorporating adaptive route optimization based on live traffic data and delivery priorities could further reduce travel time and fuel costs.

2.3 Proposed System

The Fasttrack is an advanced web and mobile-based platform designed to streamline package assignment and delivery personnel management. The system offers several advantages, including:

- Smart Package Assignment: Automates package distribution using location data and employee availability.

- Minimized Travel Distance & Efficient Delivery Sequencing: The system optimizes delivery routes by assigning packages based on proximity and logical sequencing, ensuring shorter travel distances and smoother delivery flows.

Despite its advantages, the system also has some limitations:

- Performance: Managing large-scale package assignments efficiently may require cloud-based computing and optimization techniques.
- Scalability: The system should support integration with third-party logistics APIs for future expansion.
- Security & Data Privacy: Ensuring secure access and compliance with data protection regulations.

2.3.1 Problem Statement

Efficient package delivery is a critical challenge for logistics companies, requiring optimal route planning, real-time tracking, and seamless coordination between businesses and delivery personnel. Traditional package assignment methods can be inefficient, leading to delays, mismanagement, and increased operational costs. In this project, we aim to streamline this process by developing a website and mobile app that enables companies to efficiently assign deliveries, track employees in real time and optimize routes using efficient algorithms. The system will provide businesses with a user-friendly website for package management and employee assignment while equipping delivery personnel with a mobile app for navigation, status updates and push notifications.

2.3.2 System Model

The FastTrack follows a client-server architecture with a web and mobile interface. Here's an overview:

- Client-Side: It consists of website and mobile app. The website is built with HTML, CSS, and JavaScript, allows companies to log in, manage employees, upload package details and assign deliveries. An admin panel is available for verification and management tasks. The mobile app, developed in Flutter, enables delivery personnel to log in, view assigned packages, navigate routes and receive real-time push notifications.
- Server-Side: It is powered by Flask and Firebase Firestore, manages authentication, stores package and employee data and processes delivery assignments. Supabase is used for storing employee profile images. A built-in assignment algorithm optimizes delivery routes and assigns employees efficiently. The system integrates Google Maps API for live tracking and navigation.

2.4 Requirements Engineering

The process of gathering software requirements from clients and then analysing and documenting them is known as requirements engineering. The goal of requirement engineering is to develop and maintain a sophisticated and descriptive 'System Requirements Specification' document. It essentially involves the following five activities:

1. Feasibility Study
2. Requirements Elicitation
3. Requirements Analysis
4. Requirements Specification
5. Requirements Validation

2.4.1 Feasibility Study

Feasibility study is a procedure that identifies, describes and evaluates the proposed systems and selects the best system for the task under consideration. An estimate is made of whether the identified user needs may be satisfied using current software and hardware technologies. The study will decide if the proposed system will be cost-effective from a business point of view and if it can develop given existing budgetary constraints. The key considerations involved in the feasibility analysis are:

- Economic Feasibility
- Technical Feasibility
- Behavioral Feasibility

2.4.1.1 Economic Feasibility

It is more commonly known as *Cost / Benefit analysis*. Economic feasibility refers to the evaluation of whether a proposed project or investment is financially viable and justifiable. It involves assessing the potential costs and benefits associated with the project to determine whether the expected returns outweigh the investment required. Economic feasibility analysis helps stakeholders make informed decisions about whether to proceed with a project based on its financial merits.

The development includes server hosting, Firebase integration and Google Maps API usage. The benefits include improved efficiency, optimized delivery routes, reduced delays and lower operational costs. By automating package assignment and tracking, FastTrack helps businesses save time and resources, making it a financially viable solution.

2.4.1.2 Technical Feasibility

Technical feasibility refers to the assessment of whether a proposed project can be successfully implemented from a technical standpoint. It involves evaluating whether the necessary technology, resources, and expertise are available or can be acquired to develop and deploy the project effectively. Technical feasibility analysis helps determine whether the project's goals and requirements can be achieved using existing or readily available technology and resources.
The system is built using HTML, CSS and JavaScript for the website and Flutter for the mobile app, ensuring cross-platform compatibility. The backend is powered by Flask (Python) and Firebase Firestore, which provide scalable data storage and real-time synchronization. Key technologies such as Google Maps API enable route generation and live tracking, while Firebase Authentication ensures secure access for companies and employees. Supabase is used for storing profile and company data. Since all components are widely supported and well-integrated, the system is technically feasible and can be successfully implemented.

2.4.1.3 Behavioral Feasibility

This is also known as *Operational Feasibility*. Behavioral feasibility refers to the assessment of whether a proposed project is acceptable and practical from the perspective of the people who will be affected by it.
Companies benefit from package and employee assignment thereby reducing manual work. Delivery personnel receive clear navigation routes and instant updates via push notifications, improving workflow. Admins can monitor system activities seamlessly. Since the system enhances efficiency and minimizes operational challenges, user acceptance is expected to be high, ensuring smooth adoption and long-term usability.

2.4.2 Requirements Elicitation

Requirement elicitation is the process of gathering, identifying and documenting requirements for a software system or project. It involves interacting with stakeholders such as clients, end-users and subject matter experts, to understand their needs, preferences and expectations regarding the system's functionality, features and performance.
The identified key requirements are:

- User Authentication: Secure login for companies, employees and admins.
- Package Management: Companies upload package details and assign deliveries.
- Route Optimization: Delivery personnel take the efficient paths, reducing travel time, fuel costs and delays.

- Admin Panel: Monitoring and verification features for managing company registrations and system activities.

2.4.3 Requirement Analysis

The requirement analysis involves identifying and documenting the needs and expectations of stakeholders. The primary stakeholders of the system are companies, delivery personnel and administrators. This involves identifying functional and non-functional requirements for the system.

2.4.3.1 Functional Requirements

- Authentication: Companies, employees and admins must log in securely
- Registration: Registration for both companies and employee
- Package Assignment: System assigns packages to employees based on location, availability and efficient routes. Employees can update the package status.
- Employee Management: Companies can add, update or remove employees. Employees receive assigned package details in their mobile app.
- Route Optimization: System calculates the efficient delivery route.
- Admin Panel: Admins can verify company registrations and manage system operations.

2.4.3.2 Non-Functional Requirements

- Performance: System should respond to user interactions within 4 seconds. The package assignment must be done within few seconds after receiving package details, even for a high volume of packages. System should generate efficient route within few seconds after packages are assigned to employees.
- Data Integrity & Consistency: All package assignments, tracking data and employee records must be accurate and synchronized across all users.
- Compliance & Legal Considerations: The system must comply with data protection laws.

2.4.4 Requirements Specification

When a requirements specification is written, it is important to remember that the main goal is to deliver the best product possible, and not to produce a perfect requirements specification.

There are many good definitions that describe how a requirements specification should be written, but all have at least one part in common, which is the essence of requirements specification, namely the requirements. Requirements are divided into *functional requirement* and *non-functional requirements*. Functional requirements describe the functionality of the desired system that usually consists of some kind of calculation and results, given specific inputs. Non-functional requirements describe how quickly these calculations should be and how quickly the system will respond when its functionality is used. The SRS document is given below:

2.4.4.1 Functional Requirements

- User Authentication & Access Control: The system shall allow companies, employees and admins to register and log in securely. The system shall provide password reset and account recovery options.
- Package Assignment: The system shall assign packages to employees based on availability and location. It should update package status dynamically.
- Route Optimization: The system shall calculate the efficient delivery route using Google Maps API and Greedy Algorithm.
- Admin Panel: The system shall allow admins to verify new company registrations before granting access.
- Data Storage & Retrieval: The system shall store package details and employee records in Firebase Firestore.

2.4.4.2 Non-Functional Requirements

- Performance: The system shall respond to user interactions within 4 seconds of input. It should render smoothly without lagging or freezing. The system shall assign packages to employees and generates efficient route within few seconds.
- Maintainability: The system shall be developed using modular and well-documented code that is easy to understand, modify, and maintain by future developers.
- Data Integrity: The system shall ensure that all package and employee records are synchronized across the website and mobile app in real time.

2.4.5 Requirements Validation

Requirement validation is the process of ensuring that the documented requirements accurately and completely represent the needs and expectations of the stakeholders and are

feasible for implementation. It involves reviewing, verifying, and validating the requirements to confirm that they meet the intended purpose and will result in a satisfactory solution.

- Reviewing Requirements Documentation: Review the documented functional and non-functional requirements, use cases and user stories for the website and mobile app.
- Verification and Validation: Verify the requirements against predefined acceptance criteria to confirm their correctness and completeness. Validate the requirements with stakeholders to ensure that they accurately represent their needs and expectations.

2.5 Summary

Chapter 2 explores the existing system, gaps and proposed improvements for FastTrack. It reviews Amazon Logistics' AI-driven package assignment and Google Maps API's real-time route optimization, alongside literature on Greedy and Bin Packing algorithms. A gap analysis identifies the need for dynamic package reassignment and adaptive routing based on live traffic and personnel availability. The proposed system introduces package assignments and optimized delivery sequencing to enhance efficiency. FastTrack follows a client-server architecture, with a web interface for businesses, a mobile app for employees and a Flask-Firebase backend for data management and route optimization. The requirements engineering process evaluates economic, technical and behavioral feasibility to ensure smooth integration. Requirements are elicited, analyzed, specified and validated to define system functionalities such as authentication, package management, route optimization and an admin panel. A System Requirements Specification (SRS) document formalizes both functional and non-functional needs. Finally, all requirements undergo verification and validation to ensure accuracy, feasibility, and stakeholder alignment.

Chapter 3

System Design and Methodology

The purpose of the design phase is to plan a solution to the problem specified by the requirement specification. This phase is the first step in moving from the problem domain to the solution domain. In other words, starting with what is needed, design takes us toward how to satisfy the needs. The design of a system is the most critical factor affecting the quality of the software and it has a major impact on the latter phases, particularly testing and maintenance . Once the design phase is completed, we get a document with a plan for the solution which is used in implementation, testing and maintenance. The design activity is divided into two separate phases namely:

- System Design
- Detailed Design

3.1 System Design

System Design aims to identify the modules that should be present in the system, the specification of these modules and how they interact with each other to produce the desired results. At the end of the system design, all the major data structures, file formats, output formats and major modules in the system and their specifications are decided. The requirements obtained in earlier stages of the software development life cycle (SDLC) are translated into a thorough blueprint for creating the program during the phase known as system design. The architecture, parts, interfaces, and data structures must all be specified, together with the algorithms and functionalities they support. Software's scalability, maintainability, and compliance with the requirements are all ensured through system design.

3.1.1 Data Flow Diagrams

3.1.1.1 Level 0 Data Flow

The Level 0 Data Flow Diagram represents the FastTrack System's high-level interactions with its key users: Admin, Company, Delivery Personnel and Guests. Each entity exchanges information with the system, such as registration, authentication, package assignments and route details. This diagram provides an overview of data flow without detailing internal processes.

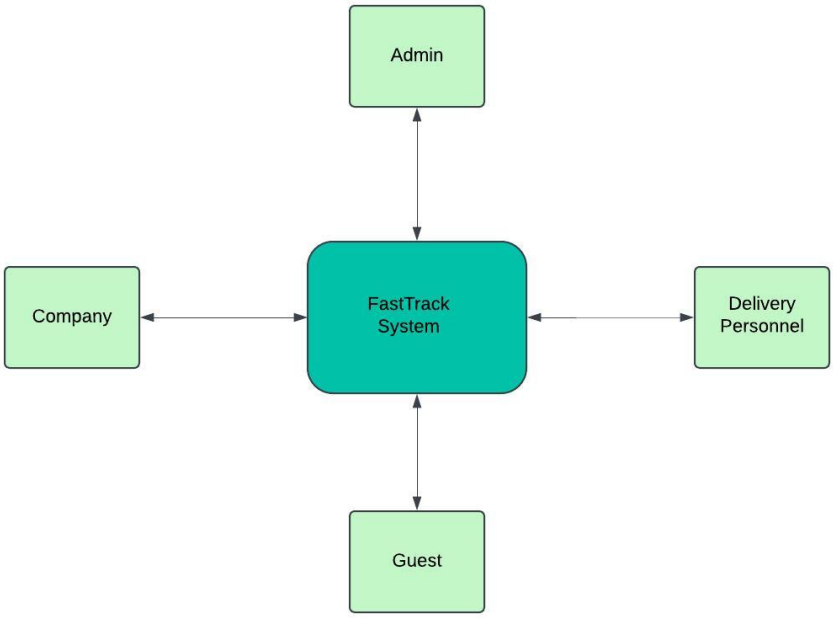

Figure 3.1: Level 0 Data Flow

3.1.1.2 Level 1 Data Flow

The Level 1 Data Flow Diagram provides a detailed breakdown of the FastTrack System's interactions. It illustrates how different entities (Admin, Company, Delivery Personnel) provide data inputs like registration, package details, and updates, which are processed and stored in the system. The system interacts with data stores for Company Information, Delivery Personnel Data, and Package Details while ensuring smooth validation, updates, and retrieval of necessary information.

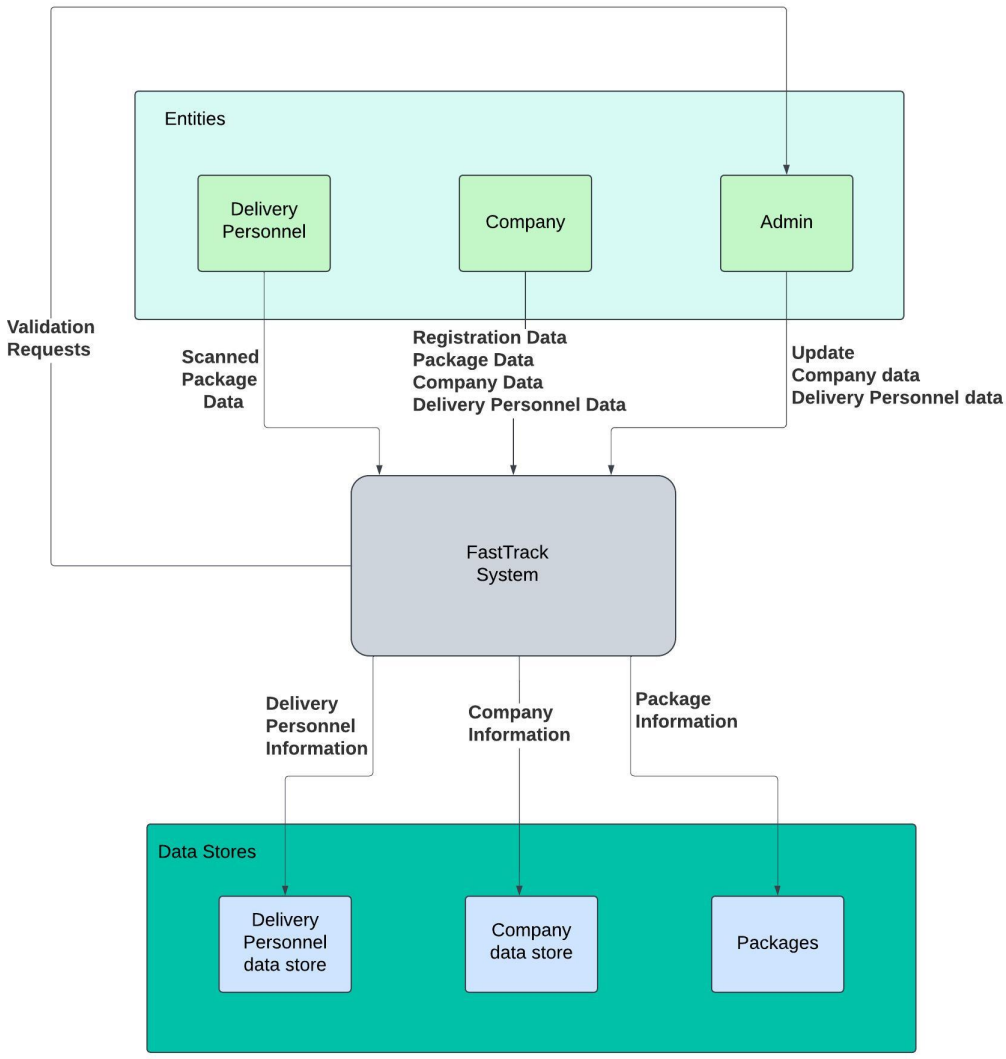

Figure 3.2: Level 1 Data Flow

3.1.2 Use Case Diagram

The Use Case Diagram for the FastTrack illustrates how different users interact with the system's functionalities. The main actors are Company, Delivery Personnel, Admin and Guest. Company can register, arrange packages and generate optimized routes. Delivery Personnel can log in, scan packages, view package details and check optimized routes. Admin is responsible for validating registrations, granting login credentials and updating system information. Guests have limited access, such as browsing the map without logging in.

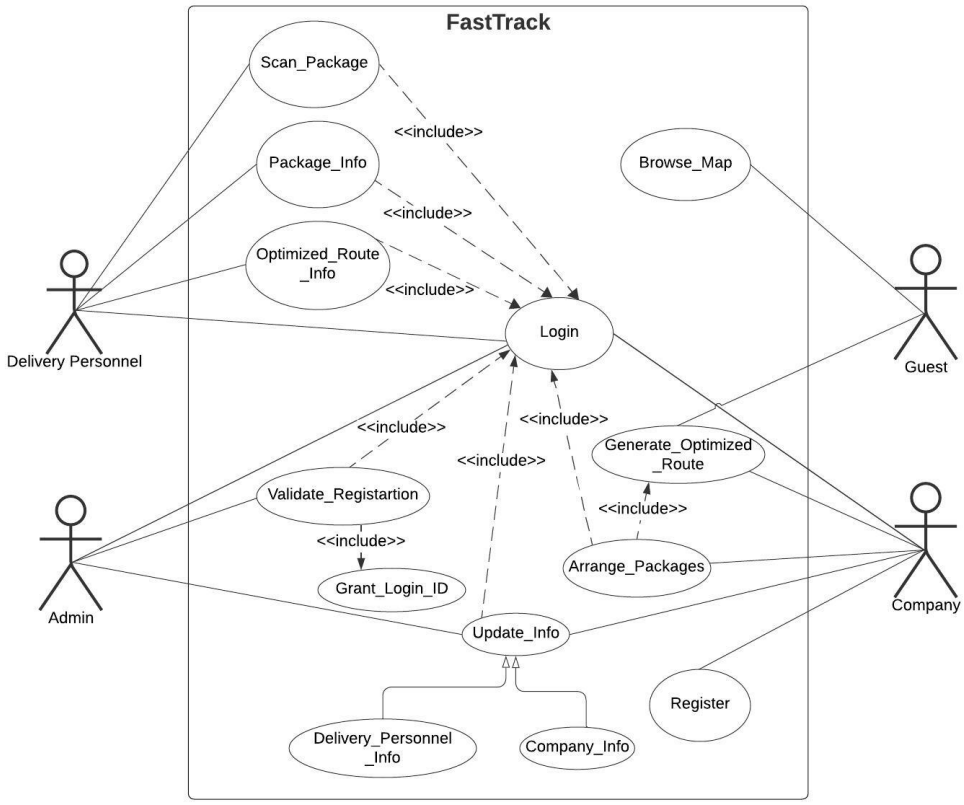

Figure 3.3: Use Case Diagram

3.1.3 System Architecture Diagram

The System Architecture of Fasttrack consists of a website, a Flutter mobile app, a Firebase database and integration with Google Maps API. The website allows companies to upload package details and assign delivery personnel, while the Flutter app enables delivery personnel to scan packages, receive optimized routes and update their delivery status in real time. Firebase acts as the central database, syncing data between the website and the app. The system also leverages Google Maps API for route optimization and navigation, ensuring efficient deliveries.

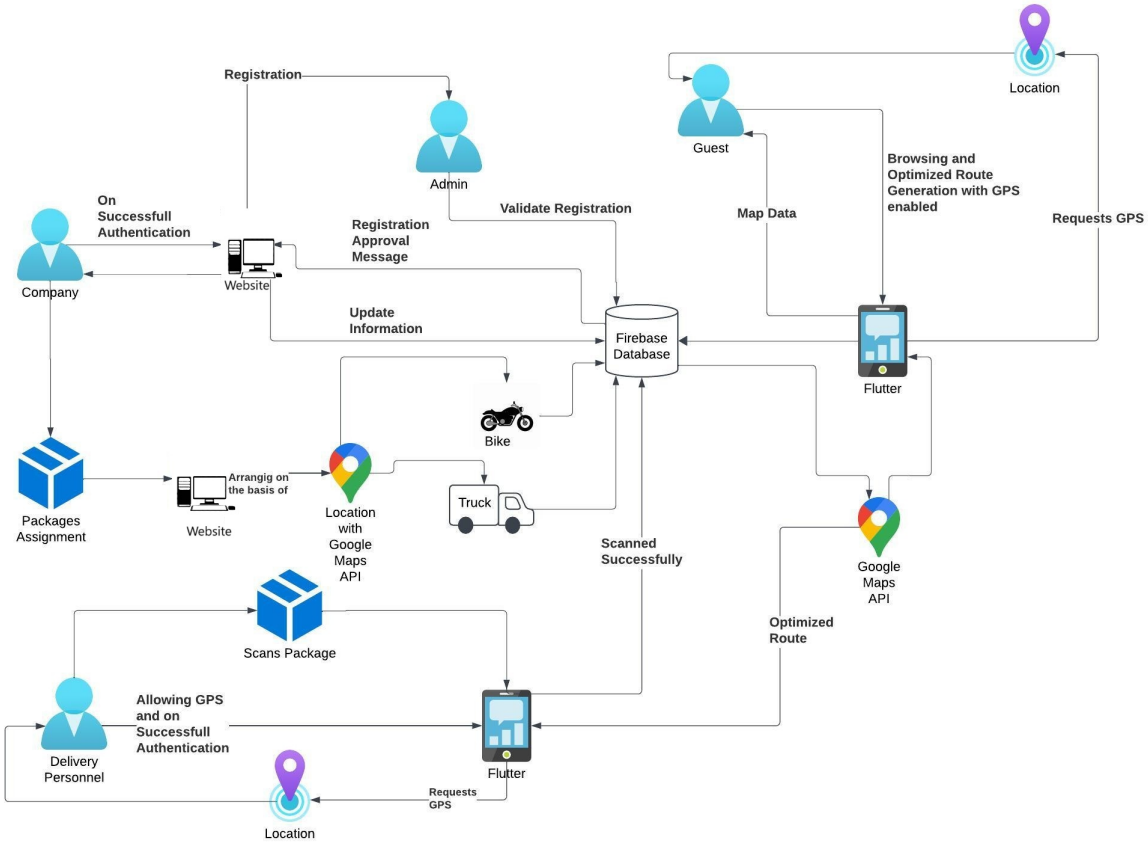

Figure 3.4: System Architecture Diagram

3.1.4 Entity-Relationship (ER) Diagram

The ER Diagram for the FastTrack represents the database structure, showing entities, their attributes and relationships. The main entities are Company, Delivery Personnel, Admin and Package. Company registers on the platform and manages delivery personnel. Delivery Personnel are assigned packages and follow optimized routes for delivery. Admin verifies company registrations and grants login credentials. Package is linked to a company and assigned to delivery personnel for delivery.

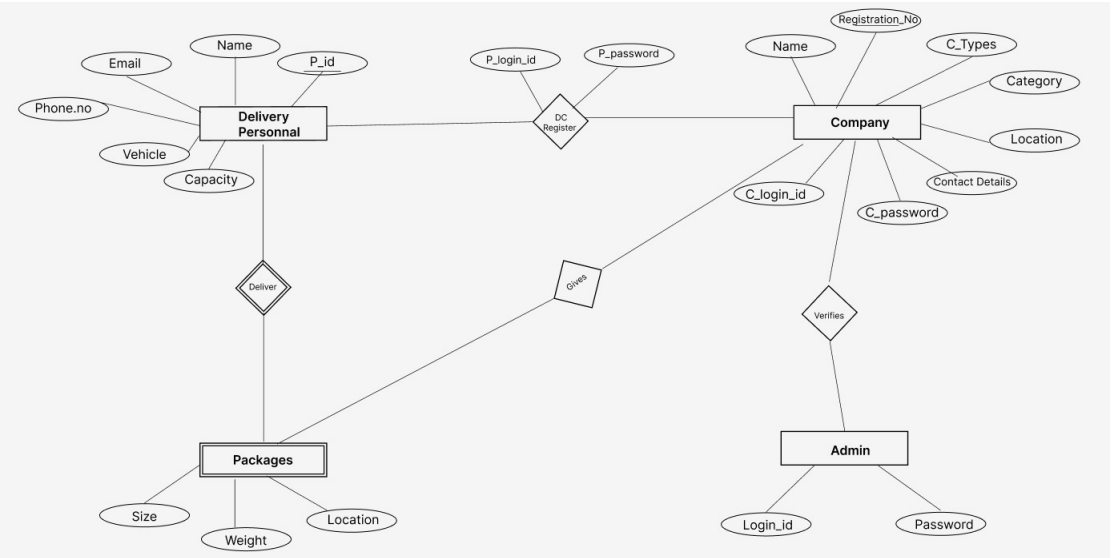

Figure 3.5: ER Diagram

3.1.5 Algorithm Modules

FastTrack incorporates the following algorithm modules:

- Authentication Algorithm
- Company Registration Algorithm
- Update Information Algorithm
- Validate Registration Algorithm
- Package Assignment Algorithm
- Scan Package Algorithm
- Retrieve Optimized Route After Scanning Packages Algorithm
- Route Optimization with Google Maps API Algorithm

3.2 Detailed Design

3.2.1 Authentication Algorithm

Algorithm 1: Delivery Personnel or Company or Admin Authentication Algorithm

Input : LoginID and Password
Output: Authentication status (Success/Failure message with token if successful)

```
1  Input the LoginID and Password from the user.
2  Search for the LoginID in the user database.
3  if the user with the LoginID is found then
4  |   Validate the Password.
5  |   if the Password matches then
6  |   |   Generate an Authentication Token.
7  |   |   Grant access.
8  |   |   Display "Authentication Successful".
9  |   end
10 |   else
11 |   |   Display "Invalid Password,Try Again!".
12 |   end
13 end
14 else
15 |   Display "User Not Found".
16 end
```

3.2.2 Company Registration

Algorithm 2: Company Registration Algorithm

Input : Company Name, Registration Number, Type of Company, Business Category, Address, Contact Details, and Verification Documents

Output: Application Number and Validation Status

```
Input Company Name, Registration Number, Type of Company, Business Category,
  Address and Contact Details.
Check if the registration number matches the official format using a regular
  expression.
if valid then
    Check if contact details are valid.
    if valid then
        Input Certificate of Incorporation, PAN Card and Proof of Address
          Documents.
        if Documents uploaded is NULL then
            Display "Please upload the required documents".
        end
        else
            Generate an application tracking number.
            Display "Validation Status: Pending Approval".
        end
    end
    else
        Display "Invalid Contact Details".
    end
end
else
    Display "Invalid Registration Number".
end
```

3.2.3 Update Information Algorithm

Algorithm 3: Update Information

Input : User Role: Admin, Company or Delivery Personnel
Update options: Address, Contact Information, Business Category
Output: Update Status (Success or Failure)

```
Input User Role: Admin, Company or Delivery Personnel.
Verify User Login using Algorithm 1.
Capture the fields the user wants to update.
Access the corresponding database table.
Update the relevant fields with the new details.
if the database update fails then
    Display "Error updating information. Please try again later".
    Set Update Status as "Failure".
    else
        Display "Information updated successfully".
        Set Update Status as "Success".
    end
end
```

3.2.4 Validate Registration Algorithm

Algorithm 4: Validate Registration

Input : Certificate of Incooperation, PAN Number and Proof of Address
Output: Validation status (Valid/Invalid)

```
Input Certificate of Incooperation, PAN Number and Proof of Address.
Ensure the Certificate of Incorporation is uploaded.
Verify that the PAN Card matches the company name format.
Check if the Proof of Address is valid.
if All validation is done then
    set Validation Status as "Valid".
    sent LoginID and temporary password to the company's email address.
    else
        set Validation Status as "Invalid".
        sent a message "Validation Failed due to incorrect document submission" to
         the company's email address.
    end
end
```

3.2.5 Package Assignment Algorithm

Algorithm 5: Package Assignment

Input : An Excel sheet, which includes package ID, location, weight, size, and date.
Average weight and size capacity of the vehicles used for delivery.
Warehouse location

Output: Table with assigned Employee ID, Package ID, Location, and Date

Dividing packages into four quadrants based on their delivery location: Quadrants = "NE": [], "NW": [], "SW": [], "SE": []

Categorize packages by Algorithm 6

Categorize packages by vehicles based on, bikes can carry packages labeled lvs, ls, lm, ms, and mm, while trucks can carry packages labeled ll, ml, hs, hm, hl, hym, and hyl.

Calculate the total weight and size for both bikes and trucks, add any remaining weight and size to the current weighted size during the calculation.

Use Algorithm 7 to find number of employees for bike and truck.

Input Number of Employees for bikes and trucks

Assign packages to bike employees by Algorithm 8

if *Total weight of Bpackage_remaining* $\geq$*125kg* **then**

  Average weight of truck = 125kg

  Average size of truck = 500cm^3

$$m = \frac{\text{Total weight of Bpackage remaining}}{\text{Average weight of truck}}$$

$$n = \frac{\text{Total size of Bpackage remaining}}{\text{Average size of truck}}$$

$$\text{Truck_employee2} = \max(m, n)$$

$$\text{Total_Truck_employee} = \text{Truck_employee} + \text{Truck_employee2}$$

end

Assign packages to truck employees by Algorithm 9

Algorithm 6: Categorize Packages

Input : package ID, location, weight and size
Output: Categorized package list

```
foreach weight in packages do
    Extract weight w and size s from p;
    if w < 2 and s < verysmall then
        Assign p to category lvs;
    end
    else if w < 2 and s < small then
        Assign p to category ls;
    end
    else if w < 2 and s < medium then
        Assign p to category lm;
    end
    else if 2 < w < 5 and s < small then
        Assign p to category ms;
    end
    else if 2 < w < 5 and s < medium then
        Assign p to category mm;
    end
    else if w < 2 and s < large then
        Assign p to category ll;
    end
    else if 2 < w < 5 and s < large then
        Assign p to category ml;
    end
    else if 5 < w < 15 and s < small then
        Assign p to category hs;
    end
    else if 5 < w < 15 and s < medium then
        Assign p to category hm;
    end
    else if 5 < w < 15 and s < large then
        Assign p to category hl;
    end
    else if w > 15 and s < medium then
        Assign p to category hym;
    end
    else if w > 15 and s < large then
        Assign p to category hyl;
    end
end
```

Algorithm 7: Assigning Algorithm for Bike and Truck Employees

```
Input  : Packages with weight, size, and location.
         Vehicle capacity (average weight and size).
Output: Assigned employees and optimized routes.
while Total_ Weight_ Vehicle ≠ 0 do
    while Quadrants changing do
        while Each employee is processing do
            while Average_ Weight ≠ 0 and Average_ Size ≠ 0 do
                Assign package;
                Update Average_Weight, Average_Size;
                if Remaining space but no suitable package then
                    if Average_ Weight > 4 then
                        Move to next quadrant (i → i+1);
                        Assign available package;
                    end
                    if Average_ Weight < 4 then
                        Move to next quadrant (i → i+1);
                        if Average_ Weight == Package_ Weight and Average_ Size
                          == Package_ Size then
                            Perform route optimization;
                            Compute shortest path (Dijkstra's algorithm);
                            if Distance < MAX_ DISTANCE then
                                Assign package and update weight/size;
                            end
                        end
                    end
                end
            end
        end
    end
end
```

Algorithm 8: Assigning Packages to Bike Employees

```
Input  : Bike employees count, available employees, and assigned packages.
Output: Updated package assignment and remaining packages.
if Bike_ employees == Available_ employeesB then
    Assign packages using assigned_packages[] from Algorithm 7;
end
if Bike_ employees > Available_ employeesB then
    Assign packages to available employees;
    Store remaining packages in Bpackages_remaining;
    Prioritize remaining packages for the next working day;
end
Calculate total weight and size of Bpackage_remaining;
```

Algorithm 9: Truck Employee Allocation and Package Assignment

Input : Total Truck Employees required (`Total_Truck_employees`)
Available Truck Employees (`Available_EmployeesT`)
Output: Assigned employees and remaining packages

```
1  if Truck_employees == Available_EmployeesT then
2  |  Assign packages to employees;
3  |  Bpackage_remaining remains unchanged and is prioritized for the next working
   |   day;
4  end
5  if Truck_employees > Available_EmployeesT then
6  |  Assign packages to Available_EmployeesT;
7  |  Store remaining packages in Tpackage_remaining and prioritize for the next
   |   working day;
8  |  Bpackage_remaining remains unchanged and prioritized for the next working
   |   day;
9  end
10 if Total_Truck_employees == Available_EmployeesT then
11 |  if Truck_employees == Available_EmployeesT - Truck_employee2 then
12 |  |  Assign packages to employees;
13 |  |  Assign Bpackage_remaining to employees;
14 |  end
15 end
16 if Total_Truck_employees > Available_EmployeesT and Truck_employees <
     Available_EmployeesT then
17 |  Emp ← Available_EmployeesT - Truck_employees;
18 |  if Truck_employees == Available_EmployeesT - Emp then
19 |  |  Assign packages to employees;
20 |  end
21 |  if Truck_employee2 > Emp then
22 |  |  Assign Bpackage_remaining to Emp;
23 |  |  Store remaining packages in Bpackages_remaining and prioritize for the next
   |  |   working day;
24 |  end
25 end
```

3.2.6 Scan Package

Algorithm 10: Scan Package

Input : Package ID (from the scanned QR code)
Output: Scanning Status (Success or Failure)

Verify Login using Algorithm 1.
Extract the Package ID from the scanned data.
Check if the Package ID exists in the system's database.
if *not found* **then**
    Display "Invalid Package ID. Please scan a valid package".
    Set Scanning Status as "Failure".
end
else
    Update the package record in the database
    Set Scanning Status as "Success."
    Display "Package successfully scanned and assigned for delivery."
end

3.2.7 Retrieve efficient Route After Scanning Packages

Algorithm 11: Retrieve efficient Route After Scanning Packages Algorithm

Input : Delivery Personnel ID
Scanned Package IDs
Output: Route Map (Efficient delivery route) or Error Message

Verify Login using Algorithm 1.
Ensure all assigned Package IDs are scanned.
if *not all assigned packages are scanned* **then**
    Display "Not all assigned packages are scanned."
end
else
    Retrieve the efficient Route generated by the company for this specific set of packages.
    if *no route exists* **then**
        Display "Optimized route not available."
    end
    else
        Display the efficient Route Map on the delivery personnel's interface.
    end
end

3.2.8 Route Optimization with Google Maps API Algorithm

Algorithm 12: Route Optimization with Google Maps API

Input : Delivery locations (latitude, longitude)
Output: Optimized delivery route and total distance

Input Delivery Locations as latitude and longitude pairs.
Send the locations to Google Maps Distance Matrix API.
Receive pairwise distances and travel times between all locations.
Set the first location as the starting point.
Initialize an empty route list and add the starting location.
Optimize Route Using Greedy Algorithm.
while *there are unvisited locations* **do**
    Find the nearest unvisited location using the distance matrix
    Add the nearest location to *route*
    Mark the location as visited
end
Add the starting location to the end of the route to complete the cycle.
For calculating the total distance, sum the distances between consecutive locations in the route using the distance matrix.
Display the optimized route sequence and integrate with Google Maps for visualization.
Print the total distance.

3.3 Summary

Chapter 3 of the software development life cycle (SDLC) delves into the design phase, transitioning from problem identification to solution planning. It encompasses two key stages: System Design and Detailed Design. The System Design phase defines key system components, interactions and data flow using DFDs, use case diagrams, system architecture and ER diagrams. The Detailed Design phase elaborates on essential algorithms, including authentication, company registration, update information, validate registration, package assignment, scan package, retrieving efficient route and route optimization.

Chapter 4

Implementation

4.1 Software Requirements

4.1.1 Operating System Requirements

Fasttrack requires a modern operating system that supports web browsers and mobile applications. The web platform runs on Windows 7 and above while the mobile app is compatible with Android 7.0 and above. A 64-bit processor with at least 4GB RAM is recommended for smooth performance. The system should support Google Chrome, Firefox, Edge, or Safari for optimal web functionality.

4.1.2 Tools Used

Various tools that are utilized for the implementation of Fasttrack are:

4.1.2.1 HTML

The FastTrack website uses HTML to structure its content, including login, package management and employee assignment sections. It employs semantic elements like <form>, <table>, , and <button> to enhance accessibility and organization. The login and registration forms use <input> fields for credentials, while <table> elements display package and employee details. Navigation is handled with <nav>, and <header> and <footer> provide structured branding. HTML ensures compatibility across devices and browsers, improving SEO and accessibility. Combined with CSS and JavaScript, it enables dynamic interactions like employee assignment. HTML also integrates seamlessly with Firebase and Flask for backend operations.

4.1.2.2 CSS

The FastTrack web application uses CSS to enhance the visual appeal and responsiveness of its interface. It employs Flexbox and Grid for layout management, ensuring a structured and adaptable design. Custom styling is applied to buttons, tables and forms for a professional look and improved usability. Media queries make the website responsive across different screen sizes, from desktops to mobile devices. Hover effects and animations improve user interaction, making navigation more intuitive. CSS variables and reusable classes ensure consistency across pages. The design follows a clean and modern aesthetic, improving the user experience for companies and employees.

4.1.2.3 JavaScript

The FastTrack website uses JavaScript to add interactivity and dynamic functionality. It handles form validation, ensuring correct user input before submission. AJAX requests enable seamless data fetching from the backend without reloading the page. Event listeners manage user interactions, such as button clicks and form submissions. JavaScript updates the UI dynamically, like displaying assigned packages without refreshing. It integrates with Firebase for real-time updates on package assignments and employee status. Third-party libraries, such as Google Maps API, enhance location-based features. The script ensures a smooth and efficient user experience across the platform.

4.1.2.4 Python

The FastTrack website uses Python for backend development, primarily with Flask. Flask handles HTTP requests and serves data to the frontend via REST APIs. It integrates with Firebase Firestore, managing company, employee and package data efficiently. Python processes uploaded Excel files, extracting package details for assignment. The employee assignment algorithm is implemented in Python, optimizing route efficiency. It also manages authentication and authorization, ensuring secure access to company and admin functionalities. Python's flexibility and Flask's lightweight nature enable a scalable and efficient backend for FastTrack.

4.1.2.5 Flutter

The FastTrack mobile app is built using Flutter, a cross-platform framework for developing high-performance applications. Dart is the primary language, enabling smooth UI and efficient state management. The app integrates with Firebase Authentication for secure employee login and Firestore for real-time package updates. Google Maps API is used for location input, navigation, and route optimization, ensuring efficient deliveries. The app supports push notifications via Firebase to keep employees updated. With Flutter's widget-based approach, the app offers a responsive and interactive user experience across Android devices.

4.1.2.6 Firebase

Firebase is essential for the FastTrack system, providing a secure and scalable backend. Firebase Authentication ensures safe logins for companies and employees. Cloud Firestore stores real-time data, including employee details, package assignments and delivery status updates. Firebase Hosting can be used to deploy the web portal efficiently. With Google's robust infrastructure, Firebase enables seamless data synchronization and real-time updates across devices. It simplifies backend management, eliminating the need for a separate server while ensuring high availability and security.

4.1.2.7 Google Maps API

The Google Maps API is used in the FastTrack to enhance location-based features in the Flutter app. It enables map rendering, real-time GPS tracking, and dynamic routing for delivery personnel. Directions API helps in generating optimized routes for multiple deliveries, ensuring efficient navigation. Geocoding API converts addresses into coordinates for precise location storage. Maps customization allows branding adjustments to align with FastTrack's theme. With real-time updates and interactive map controls, the API ensures a seamless and intuitive delivery experience.

4.1.2.8 Supabase

Supabase is used for storing images in FastTrack as a scalable and secure alternative to Firebase Cloud Storage. It provides PostgreSQL-backed storage, ensuring efficient retrieval and management of profile and package images. It provides a direct URL link for each uploaded image, making it easy to retrieve and display images in the FastTrack app. These links can be stored in Firestore alongside other user or package data for quick access.

4.2 Hardware Requirements

4.2.1 Website

1. Processor: At least an Intel i3 (or AMD equivalent), but i5/i7 is recommended for smooth multitasking.
2. RAM: Minimum 4GB, recommended 8GB or higher for handling multiple browser tabs and development tools.
3. Internet Connection: A stable broadband connection (at least 10 Mbps) for real-time interactions with cloud-based services.

4.2.2 Mobile App

1. Processor: A 1.4 GHz or higher multi-core processor to ensure smooth app execution and real-time processing.
2. RAM: At least 2GB for basic performance, with 4GB or more recommended for handling maps and real-time updates efficiently.
3. Connectivity: Stable 4G/5G, GPS and Wi-Fi support for seamless real-time data transmission and location tracking.

4.3 Code Implementation Details

4.3.1 Authentication & Authorization

FastTrack uses Firebase Authentication for secure login and access control. Employees and companies authenticate using email and password, while the system verifies credentials and maintains session states. The authentication logic ensures that only authorized users can access package details and employee assignments.

4.3.2 Package Assignment

Package details are uploaded via an Excel file, processed using Python (Flask), and stored in Firestore. Each package entry includes recipient details, location coordinates, and assigned delivery personnel. The system updates package status dynamically based on delivery progress.

4.3.3 Employee Assignment

The system calculates the required number of employees based on the uploaded package data. Companies select available employees and packages are automatically assigned using a assignment algorithm that minimizes travel distance. The assignments are updated in Firestore for real-time access by employees.

4.3.4 Route Optimization & Navigation

The Google Maps API is integrated into the mobile app to assist delivery personnel. Employees can view optimized routes and track package drop-off points. Optimized route generation in FastTrack is based on a greedy algorithm, ensuring efficient delivery route planning. The app dynamically updates routes when a delivery is completed.

4.3.5 Real-time Updates & Notifications

Firestore enables real-time synchronization of package assignments and status updates across the system. Employees receive push notifications (Firebase) for new assignments and delivery updates.

4.3.6 Screenshots

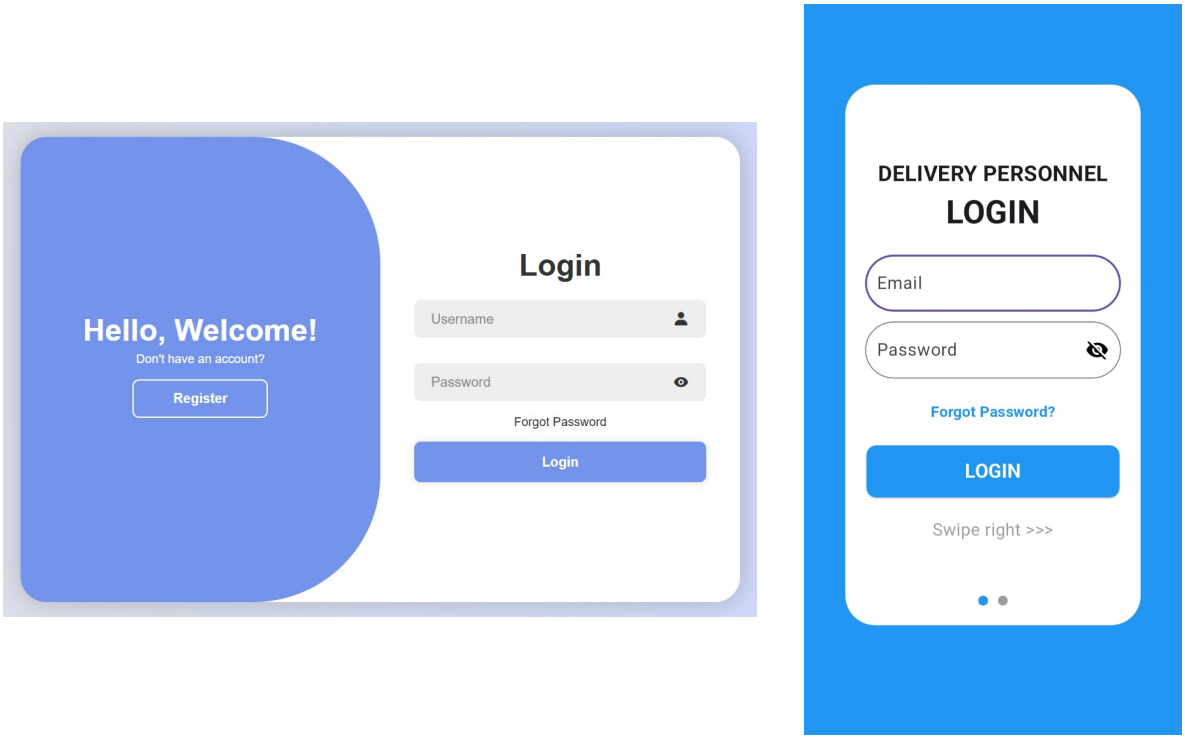

Figure 4.1: Authentication & Authorization

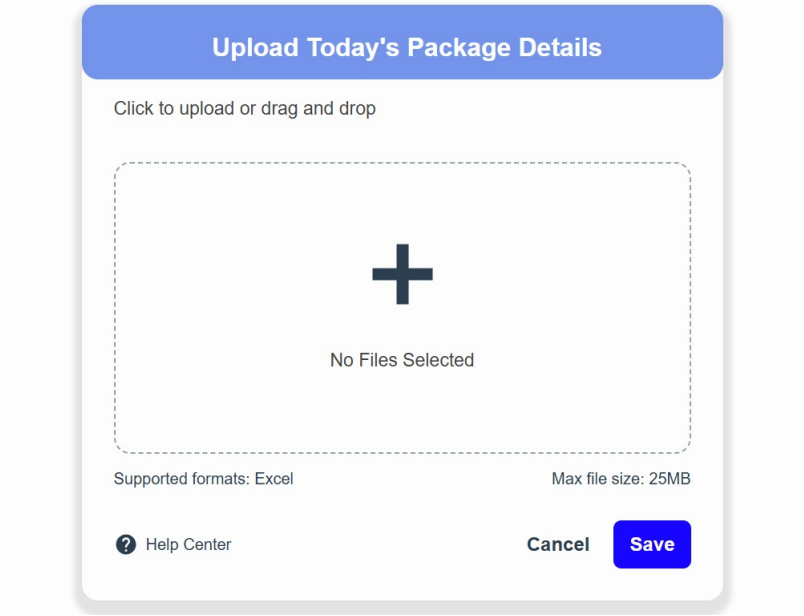

Figure 4.2: Package Assignment

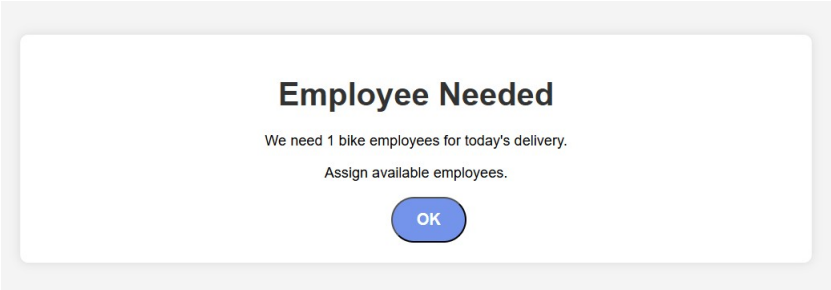

Figure 4.3: Employee Assignment

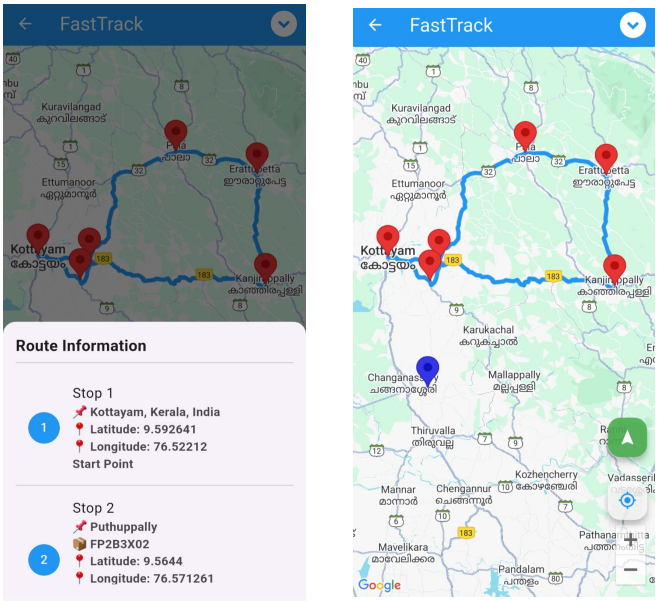

Figure 4.4: Route Optimization and Navigation

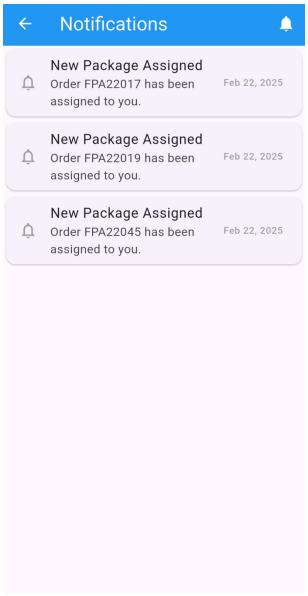

Figure 4.5: Notifications

4.4 Features and Functionalities

4.4.1 User Authentication & Management

- Companies can register with required details.
- They can login securely using Firebase Authentication.
- Companies can add, edit and manage delivery personnel.
- Each company gets a unique 5-character alphanumeric ID (FCXXX).
- Each employee gets a unique 8-character alphanumeric ID (FDXXXYYY), where XXX wil be Company ID.
- Each packet will gets a unique 8-character alphanumeric ID (FPXXXYYY), where XXX wil be Company ID.

4.4.2 Package Handling & Assignment

- Companies upload daily package details via an Excel file.
- The system calculates the required number of employees based on packages.
- Companies assign packages to available delivery personnel.

4.4.3 QR Code System

- Companies can generate QR codes for each package.
- Delivery Personnel can scan QR codes for picking their assigned package.

4.4.4 Route Optimization

- Displays the efficient path from the employee's current location to each delivery stop.
- Updates dynamically as deliveries are completed.

4.5 Summary

Chapter 4 details the implementation of the FastTrack system, focusing on the technologies and methodologies used. The section begins with the software and hardware requirements, outlining the necessary operating systems, devices, and tools such as HTML, CSS, JavaScript, Python (Flask), Flutter, Firebase and Google Maps API. It then explains the core functionalities, including

user authentication, package handling, QR code integration and real-time updates. The route optimization module ensures efficient deliveries using Google Maps API, dynamically updating paths. The backend, built with Firebase and Flask, manages data storage, user authentication and real-time synchronization. Finally, the chapter includes the screenshots of implemented code providing a visual representation . Overall, it provides a comprehensive view of the FastTrack system's implementation, highlighting key technologies and workflow.

Chapter 5

Testing

Software Testing is a method to check whether the actual software product matches expected requirements and to ensure that the software product is Defect free. It involves the execution of software/system components using manual or automated tools to evaluate one or more properties of interest. The purpose of software testing is to identify errors, gaps or missing requirements in contrast to actual requirements.

5.1 Testing Strategies Used

In this section, the strategies that were employed to test our project will be discussed. The following list outlines the testing strategies that were utilized in our project to ensure its quality and effectiveness

5.1.1 Unit Testing

Unit testing is a software testing technique where individual units or components of a software application are tested in isolation to ensure they function correctly. It is a critical practice in software development aimed at verifying the behavior of small, self-contained units of code, such as functions, methods or classes. Through systematic testing of individual units of code, developers can identify and address bugs early in the development cycle, maintain code quality and reliability, facilitate code refactoring, enhance code base understanding and prevent unintended changes in behavior due to modifications. For this project, unit testing ensures that individual components, such as login, package assignment and navigation, work correctly. The following are the main two techniques to perform unit testing:

5.1.1.1 Black Box Testing

Black Box testing is a software testing technique where the internal workings or implementation details of the system being tested are not known to the tester. This approach focuses solely on the external behavior of the software without considering its internal logic, code structure or design. Black Box Testing for FastTrack verifies functional behavior by testing login, package assignment, employee management, route optimization and real-time updates without analyzing internal code. It ensures correct system responses, secure authentication, accurate data processing and seamless navigation while handling errors and edge cases effectively.

5.1.1.2 White Box Testing

White Box testing, also known as structural or glass-box testing, involves examining the internal structure and logic of the software system. This type of testing requires access to the source code, allowing testers to design test cases based on the understanding of how the code is structured and how it operates. White Box Testing for FastTrack focuses on internal code structure, ensuring efficient algorithms, secure authentication and optimized database queries. It verifies logic flow, code coverage and performance, ensuring reliability and scalability in package assignment, employee management and route optimization.

5.1.2 Integration testing

Integration testing is a software testing technique that focuses on verifying the interactions between different modules or components of a software system after they have been integrated. It aims to identify defects or inconsistencies that may arise when multiple modules are combined to form the complete system. Integration testing is typically performed following unit testing and precedes system testing in the software development life-cycle. It ensures that individual modules or components, which may have been tested independently during unit testing, interact correctly when integrated into the larger system. Integration Testing for FastTrack ensures seamless interaction between system modules, including the web platform, mobile app, Firebase backend and Google Maps API. It verifies data flow, authentication, package assignment and real-time updates, ensuring smooth communication and functionality across all components.

5.2 Testing Results

Tests were conducted using the strategies mentioned earlier. The results obtained from these tests are as follows:

5.2.1 Results of Black Box Testing

In this project, Black Box testing, verified core functionalities without inspecting internal code. Login and authentication were tested to ensure secure access for companies and employees. Package assignment and employee management were validated for correct data processing and UI interactions. Route optimization was tested to confirm accurate navigation and real-time updates. Error handling was assessed by inputting invalid data and checking system responses. System performance was evaluated under varying loads to ensure smooth operation. The results confirmed that FastTrack functions correctly, with all key features performing as expected.

5.2.2 Results of White Box Testing

In this project, White Box testing, focused on evaluating internal logic, code structure, and flow control. Authentication modules were tested for secure session handling and proper token validation. The package assignment algorithm was analyzed to ensure correct employee selection and data integrity. Route optimization logic was examined for efficiency, ensuring minimal travel distance calculations. Code coverage tests were performed to verify all critical functions were executed under different conditions. Error handling mechanisms were assessed to ensure smooth exception management without system crashes. The results confirmed optimized code execution, proper data flow and robust security measures.

5.2.3 Results of Integration Testing

In this project, Integration testing, validated seamless communication between different system modules. The authentication module was tested with Firebase to ensure secure login and session management. The package assignment system was integrated with Firestore, verifying real-time updates and correct employee assignments. Google Maps API integration was checked for accurate route optimization and location tracking. The mobile app and web portal were tested together to ensure synchronized data flow between companies and delivery personnel. API endpoints were validated for smooth data exchange between the frontend (Flutter, JavaScript) and backend (Flask, Firebase). The results confirmed that all modules work cohesively, ensuring a reliable and efficient delivery management system.

5.3 Test Cases

Test Case ID	**Test Scenario**	**Test Steps**	**Expected Output**	**Actual Output**	**Status**
TC01	Upload Package Details	Log in as Company → Navigate to Package Upload → Upload Excel file	Packages uploaded successfully	Packages uploaded successfully	Pass
TC02	Assign Employees	Log in as Company → Navigate to Employee Assignment → Select available employees → Assign packages	Employees assigned based on workload	Assignments updated incorrectly	Fail

Test Case ID	**Test Scenario**	**Test Steps**	**Expected Output**	**Actual Output**	**Status**
TC03	Assign Employees	Log in as Company → Navigate to Employee Assignment → Select available employees → Assign packages	Employees assigned based on workload	Assignments updated correctly	Pass
TC04	Real-Time Package Updates	Log in as Company → Navigate to Package Assignment → Monitor package status	Package statuses updated in real-time	Statuses updated correctly	Pass
TC05	Route Calculation	Log in as Employee → Navigate to Assigned Deliveries → View optimized route	Route calculated based on shortest time and distance	Route displayed correctly	Pass
TC06	Real-Time ETA Update	Log in as Employee → Start navigation → Monitor ETA changes based on real-time traffic	ETA updates dynamically as conditions change	ETA updated incorrectly	Fail
TC07	Real-Time ETA Update	Log in as Employee → Start navigation → Monitor ETA changes based on real-time traffic	ETA updates dynamically as conditions change	ETA updated correctly	Pass
TC08	Multiple Deliveries Optimization	Log in as Employee → View assigned deliveries → System optimizes stops for efficiency	Deliveries arranged in the most time-efficient order	Stops optimized correctly	Pass

5.4 Summary

Software testing for FastTrack involved Unit Testing, Black Box Testing, White Box Testing and Integration Testing to ensure system reliability. Unit Testing focused on individual components, verifying functions like user authentication, package assignment and route optimization in isolation. Black Box Testing evaluated the system's functionality without examining internal code, ensuring proper user interactions, correct package assignments and accurate navigation features. White Box Testing examined the internal logic and code structure, validating algorithm efficiency, database queries and API response accuracy. Integration Testing ensured seamless data flow between modules, verifying that Firebase authentication, Firestore data storage and Google Maps API integration worked together without issues. The mobile app and web platform were tested to confirm synchronized updates for delivery personnel and businesses. All tests confirmed the system's stability, usability and accuracy in handling deliveries. The results demonstrated that FastTrack effectively meets its functional and technical requirements.

Chapter 6

Results and Discussion

6.1 Results

The following objectives were achieved through the course of this project:

- Package Assignment: The project aims to implement a package assignment system that efficiently allocates deliveries to available employees. The assignment algorithm considers factors such as package locations, employee availability and workload distribution to optimize efficiency. By ensuring balanced package distribution, the system enhances delivery management and reduces delays.
- Route Optimization: The project integrates an intelligent route optimization system that employs the Greedy Algorithm to determine the efficient delivery paths for employees. The algorithm minimizes travel time and fuel consumption. The system continuously updates routes based on delivery progress, ensuring adaptive and efficient navigation. By optimizing travel sequences, the system improves delivery speed and overall operational efficiency.

6.1.1 Login & Registration Page

The Login & Registration Page allows companies and employees to securely create accounts and access the system using Firebase Authentication. It ensures user verification, maintaining session states for seamless navigation.

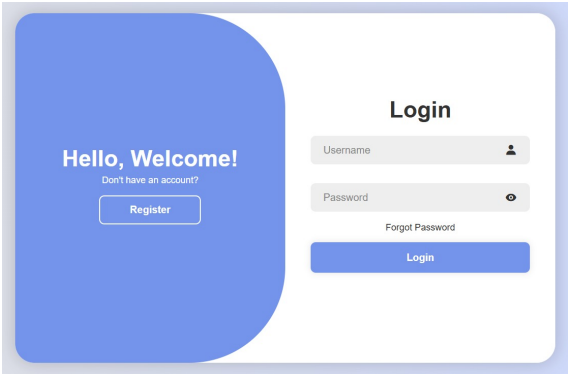

Figure 6.1: Company Login Page

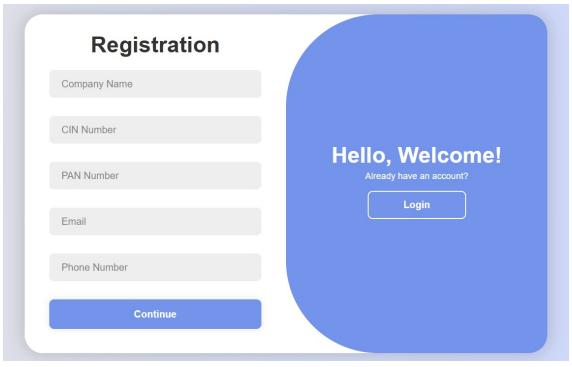

Figure 6.2: Company Registration Page

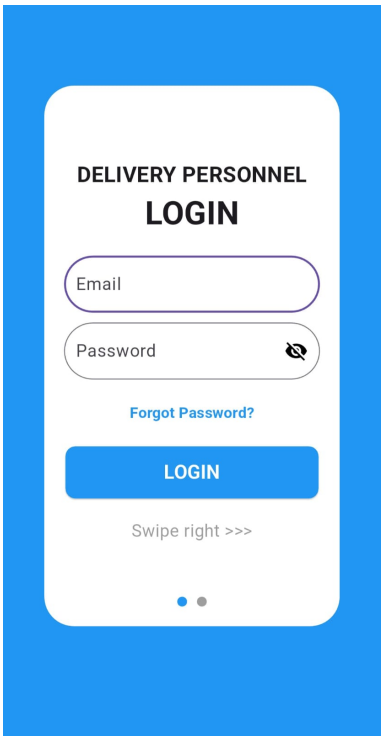

Figure 6.3: Delivery Personnel Page

6.1.2 Package & Employee Details Page

Package Details: Displays information about uploaded packages, including destination, weight and qr code ensuring efficient assignment.
Employee Details: Lists registered employees with availability status, allowing companies to allocate deliveries effectively.

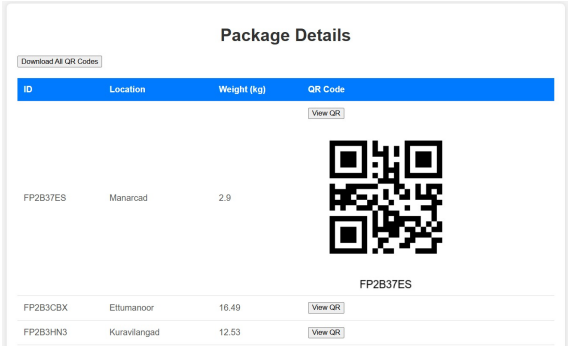

Figure 6.4: Package Details Page

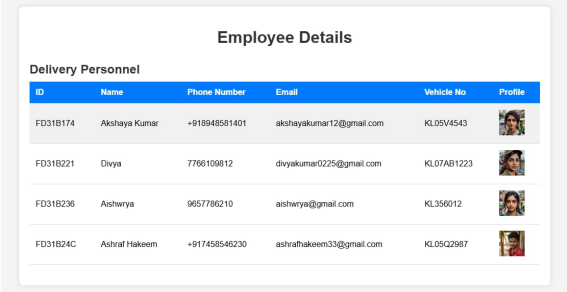

Figure 6.5: Employee Details Page

6.1.3 QR Code Scanning Page

The system enables companies to generate QR codes for packages, allowing employees to scan them for quick retrieval of delivery details. This feature ensures efficient tracking, reduces manual errors and enhances the overall package management process.

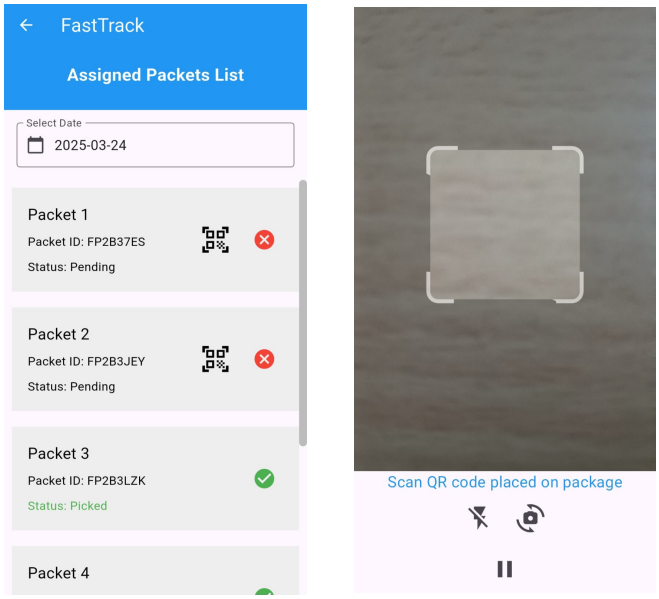

Figure 6.6: QR Code Scanning

6.1.4 Route Optimization & Navigation Page

Route Optimization: Utilizes the Greedy Algorithm and Google Maps API to calculate the efficient delivery routes, minimizing travel distance and time.

Navigation Page: Provides real-time tracking for delivery personnel, ensuring smooth and timely deliveries.

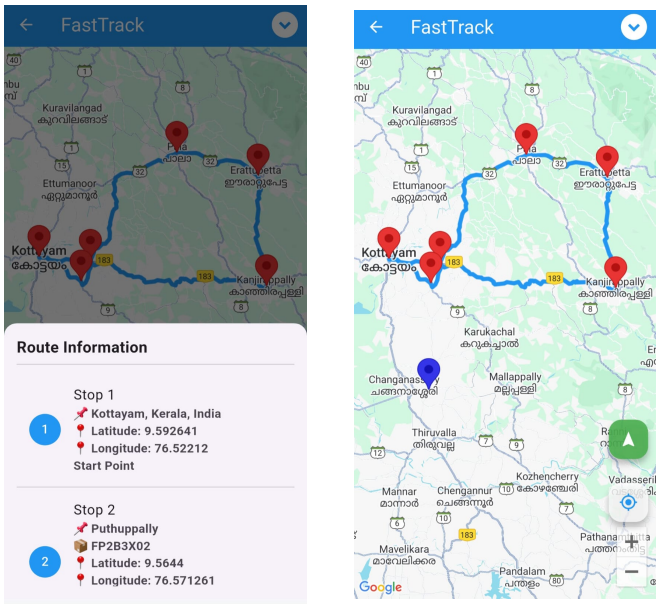

Figure 6.7: Route Optimization and Navigation Page

6.2 Performance Analysis

The main two algorithms we make use in our project are package assignment and route optimization (Greedy Algorithm).

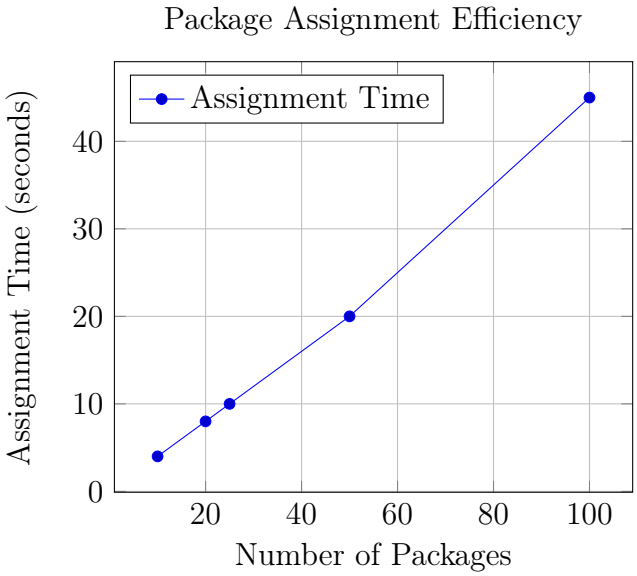

Figure 6.8: Package Assignment Efficiency Analysis

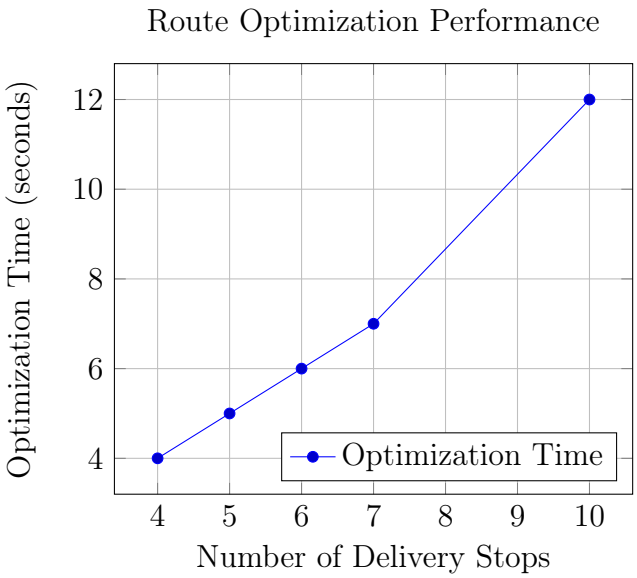

Figure 6.9: Route Optimization Performance Analysis

6.3 Challenges Faced and Solution

6.3.1 Package Assignment

- Problem: The initial employee assignment algorithm only considered the total weight of packages and the vehicle's average weight capacity. This approach often led to miscalculations, as it did not account for package size constraints or variations in real-world loading conditions. As a result, even when sufficient employees were assigned, some packages were left undelivered due to improper space utilization. Additionally, the system sometimes suggested more employees than necessary, leading to inefficiencies in workforce allocation.

- Solution: The algorithm was improved to factor in both weight and size constraints when calculating the required number of employees. Instead of relying solely on weight-based allocation, the system now calculates employee requirements based on both total weight and total volume.Additionally, package assignments were refined to distribute loads evenly across employees, preventing scenarios where small gaps in available space led to undelivered packages. This enhancement ensures that all packages are assigned efficiently while optimizing the use of available employees and transportation capacity.

6.3.2 Route Optimization

- Problem: The algorithm did not account for the final delivery location, which resulted in incomplete routes and inefficient navigation. This caused potential delays and suboptimal travel sequences for delivery personnel.

- Solution: The routing algorithm was refined to ensure that the final delivery location was always included as a required stop. Additionally, the logic was adjusted to dynamically optimize the route while considering real-time location changes, enhancing delivery efficiency and accuracy.

6.4 Summary

This chapter presents the results of the FastTrack system, highlighting key features and performance analysis. The Login & Registration Page ensures secure authentication for companies and employees using Firebase (Figure 6.1 & 6.2 & 6.3). The Package & Employee Details Page displays package information and employee availability for efficient assignment (Figure 6.4 & 6.5). A QR Code Scanning Page allows employees to scan package QR codes for quick tracking and retrieval (Figure 6.6). The Route Optimization & Navigation Page utilizes the Greedy Algorithm and Google Maps API to generate efficient delivery routes (Figure 6.7). Performance analysis graphs show package assignment efficiency and route optimization time, proving the system's effectiveness (Figure 6.8 &

6.9). Challenges, such as missing final delivery locations in routing, were resolved by refining the algorithm for accurate navigation and optimal travel sequences.

Chapter 7

Conclusion and Future Scope

7.1 Summary

This project, titled "FastTrack," is designed to enhance delivery efficiency through package assignment and route optimization system. The primary objectives were to develop a dynamic personnel allocation system, ensure timely deliveries and optimize routes in real-time. The system consists of a Flutter-based mobile application and a Flask backend, integrating Firebase Firestore for real-time data synchronization.

The study of system requirements involved defining both functional and non-functional needs, emphasizing user interaction features such as real-time route tracking and push notifications. Performance benchmarks, including response time and GPS accuracy, were considered to ensure smooth operations. The development process followed a structured Software Development Life Cycle (SDLC), incorporating core modules like authentication, route optimization and package assignment. Various algorithms were implemented to optimize the delivery workflow.

The system was built using Flutter for the mobile app, Flask for the backend, and Firebase for cloud storage. Key features include a login system, splash screen, map browsing with Google Maps, real-time navigation updates, and Firebase notifications. The integration of these technologies ensures seamless data flow between delivery personnel and companies.

Comprehensive testing strategies, including Unit Testing, Integration Testing and Black Box Testing, were conducted to validate system performance. Key functionalities, such as location tracking, route optimization, and package assignment, were rigorously tested to ensure reliability and accuracy.

The project successfully meets its objectives by minimizing delivery delays, optimizing workforce utilization and improving last-mile delivery operations. The mobile application enables delivery personnel to dynamically access assigned routes, while companies efficiently manage operations through an interactive web portal. Real-time navigation, Firebase notifications and route optimization contribute to a more efficient and streamlined delivery process.

This report comprehensively documents the development, testing, and results of the FastTrack Delivery project, demonstrating its effectiveness in achieving its goals.

7.2 Recommendations for Future Work

To enhance FastTrack's functionality, usability and efficiency, consider the following improvements:

1. Algorithm Enhancements: Explore advanced algorithms for package assignment and route optimization to improve efficiency and accuracy, leveraging AI and real-time data.

2. User Interface Refinement: Conduct user testing to refine the web and mobile UI, improving usability, accessibility and seamless interaction for businesses and employees.

3. Performance Optimization: Optimize system performance for faster package processing, real-time route updates and efficient data handling, ensuring smooth operations even with high package volumes.

Appendix A

Software Requirement Specification

A.1 Introduction

A.1.1 Purpose

This Software Requirements Specification document outlines the requirements for the FastTrack application, a platform designed to calculate and optimize delivery routes for delivery personnel efficiently. It specifies the functionalities, features, and requirements of the product. It covers aspects like location input, default location integration, route optimization algorithms, time estimation, and route visualization through an interactive map interface. The primary scope is to enhance the efficiency and reliability of delivery operations while minimizing travel time and fuel consumption. The application aims to provide an intuitive interface, seamless navigation, and support for real-time updates to improve delivery service quality and customer satisfaction.

A.1.2 Document Conventions

- This document follows a standardized format, including numbered sections, bullet points, diagrams, and tables for clarity and readability.
- Acronyms, abbreviations, and technical terms are defined upon first usage and compiled in the glossary (Appendix A) for quick reference.
- All requirements are assigned unique identifiers and listed systematically to ensure easy tracking and cross-referencing throughout the document.

A.1.3 Intended Audience and Reading Suggestions

This document is intended for the following readers:

- Developers: To understand the technical requirements and system functionality.
- Project Managers: To track progress and ensure the requirements are implemented.
- Testers: To create test cases based on requirements.
- End Users (Delivery Personnel and Administrators): To comprehend the application's functionality and benefits.

Reading Suggestions include

- Overview Sections: Recommended for all readers to gain a general understanding of the application.
- Functional and Non-functional Requirements: Crucial for developers, testers, and project managers to ensure proper implementation and verification of the system.
- User Interface and Workflow Sections: Important for end users to familiarize themselves with the application's features and usage.

A.1.4 Project Scope

The FastTrack application is designed to streamline delivery operations by optimizing routes for delivery personnel. Its key objectives include:

- Efficient Route Optimization: Calculating the most efficient delivery route for multiple destinations using advanced algorithms, reducing travel time and fuel consumption.
- Default Location Integration: Incorporating a fixed starting point for consistent and reliable route planning.
- Interactive Map Visualization: Providing clear and user-friendly route displays, along with real-time navigation support.
- Time and Distance Estimation: Offering accurate estimates for delivery time and travel distance to enhance scheduling and customer satisfaction.
- Dynamic Updates: Allowing for real-time adjustments to routes in response to changes, such as additional stops or delays.

The application aims to improve delivery efficiency, reduce operational costs, and ensure timely deliveries while enhancing the overall user experience for delivery personnel and service providers.

A.1.5 References

- Route Optimization Apps to Minimize Delivery Delays with Dynamic Routes, nandbox, 2024. Retrieved from nandbox.com
- Route4Me API and SDK Developer Documentation, Route4Me, 2024. Retrieved from support.route4me.com
- How to Use Python for Route Optimization in Logistics, Data Head Hunters Academy, 2024. Retrieved from dataheadhunters.com
- Delivery Route Optimization in Python, AskPython, 2024. Retrieved from askpython.com

A.2 Overall Description

A.2.1 Product Perspective

The FastTrack application is a new, self-contained product designed to address the challenges of delivery route optimization. Unlike existing manual or semi-automated systems, FastTrack leverages advanced algorithms and geolocation technology to create an intuitive, user-friendly platform for delivery personnel. This application operates independently, without requiring integration with any legacy systems, but can seamlessly integrate with popular GPS and mapping tools. It is designed to serve small and medium-sized businesses seeking efficient delivery solutions, enhancing operational efficiency and reducing delivery time. The concept stems from the need to simplify delivery operations, reduce costs, and improve the overall reliability of delivery services, particularly in dynamic environments with multiple destination points.

A.2.2 Product Features

The primary feature of FastTrack include:

1. Route Optimization: Users can efficiently plan routes with up to 300 stops, utilizing real-time traffic data and delivery-specific parameters for optimized navigation.

2. Dynamic Route Adjustment: Users can adapt to changing conditions through real-time route adjustments, minimizing delays caused by traffic, road closures, or weather disruptions.

3. Proof of Delivery and Documentation: Users can securely document proof of delivery through photos, notes, and signatures, ensuring seamless record-keeping and dispute resolution.

4. Dynamic Data-Driven Delivery Insights: Users can access detailed delivery performance insights, including metrics like distance, time, and fuel savings, with exportable reports for analysis and improvement.

5. Route Export and Data Integration: The system enables seamless export of route data in CSV or Excel formats, ensuring integration with logistics and operations management systems.

A.2.3 User Classes and Characteristics

- Admin: Responsible for overseeing operations, managing user accounts, monitoring system performance, and resolving issues. Requires a moderate to high level of technical expertise and familiarity with system functionalities.

- Delivery Personnel: Primarily uses the product for route optimization, proof of delivery, and real-time updates. Requires basic technical expertise and familiarity with smartphone applications.

- Technical Expertise: Includes developers and system administrators responsible for maintaining the application, integrating APIs, and ensuring system reliability. Requires advanced technical knowledge and troubleshooting skills.
- Company: Refers to stakeholders or management personnel who utilize delivery insights, performance reports, and other analytics for decision-making. Requires a high-level understanding of operational metrics but minimal technical expertise.

A.2.4 Operating Environment

- Hardware Platform:
  - Mobile devices with at least 2 GB of RAM and 16 GB of internal storage.
  - GPS enabled smartphone.
- Operating System:
  - Android (Version 8.0 and above).
  - iOS (Version 13.0 and above).

A.2.5 Design and Implementation Constraints

- Hardware Limitations: Requires smartphones with GPS functionality, a minimum of 2 GB RAM and stable internet connectivity.
- Security Considerations: Data exchanged between the app and server will be encrypted using HTTPS for secure communication.
- Programming Standards: The application will be developed using modern frameworks like Flutter for cross-platform compatibility.
- Database Technology: A cloud-based database, such as Firebase will store route details, delivery data, and user information.
- Third-party APIs: Integration with GPS services for route optimization and location tracking.
- Design Standards: Adherence to mobile app design conventions for usability and accessibility.

A.2.6 User Documentation

The following user documentation components will be delivered with the FastTrack application to ensure ease of use and proper understanding of its features:

- User Manual:A comprehensive guide in PDF format that covers installation, setup, usage, troubleshooting, and frequently asked questions.

- Online Help: Context-sensitive help integrated within the application, offering quick access to tooltips and feature explanations.
- Tutorials: Video tutorials hosted on the app, demonstrating key functionalities like route planning and navigation.
- Quick Start Guide: A concise, easy-to-follow document that provides a fast overview of the application's primary features and setup instructions.
- Release Notes: A detailed document that outlines new features, updates, and bug fixes for each version, available in PDF format.

A.2.7 Assumptions and Dependencies

Assumptions include:

- Users have access to GPS-enabled smartphones with stable internet connectivity.
- The application will primarily operate within areas where GPS accuracy and internet speed are reliable.
- Delivery routes provided by third-party services are accurate and up-to-date.

Dependencies include:

- Stable and continuous GPS and internet services for route optimization and real-time updates.
- Functional smartphones meeting the specified hardware requirements.
- Continued availability and performance of third-party APIs for GPS and location services.
- Reliable cloud-based database for storing user data, route details, and delivery updates.

A.3 System Features

A.3.1 Route Optimization

A.3.1.1 Description and Priority

- Description: Curates the best route from an extensive list of destinations (up to 200 per route) and tailored vehicle types.
- Priority: High
- Benefit: 9
- Penalty: 1

- Cost: 8
- Risk: 3

A.3.1.2 Stimulus/Response Sequences

- Stimulus: User inputs a list of destinations.
- Response: The system generates the most optimized route based on distance and time, considering current traffic conditions.

A.3.1.3 Functional Requirements

- The system should be capable of handling up to 300 stops per route without additional cost.
- The route should be optimized using real-time traffic data, considering factors such as time, distance, and mode of delivery.
- The system should offer quick optimization for routes with up to 100 stops in 5 seconds.
- The user should be able to save and duplicate routes, adding new destinations from previous routes easily.

A.3.2 Dynamic Route Adjustment

A.3.2.1 Description and Priority

- Description: Allows dynamic adjustments to routes in real-time based on changing conditions (e.g., traffic, weather, etc.).
- Priority: High
- Benefit: 9
- Penalty: 1
- Cost: 7
- Risk: 3

A.3.2.2 Stimulus/Response Sequences

- Stimulus: Real-time data indicates a traffic jam or road closure.
- Response: The system automatically adjusts the route to avoid delays, presenting the updated route to the driver.

A.3.2.3 Functional Requirements

- The system must be capable of adjusting routes in real-time based on live traffic, road closures, and weather conditions.
- The system should ensure minimum disruption when adding or removing stops in dynamic routes.

A.3.3 Proof of Delivery and Documentation

A.3.3.1 Description and Priority

- Description: Captures proof of delivery via photos, notes, and signatures.
- Priority: Medium
- Benefit: 7
- Penalty: 2
- Cost: 5
- Risk: 3

A.3.3.2 Stimulus/Response Sequences

- Stimulus: User completes a delivery.
- Response: The system captures proof (photos, notes, or signatures) for documentation purposes.

A.3.3.3 Functional Requirements

- The system should allow users to capture and save photos, notes, and signatures for each stop as proof of delivery.
- It must provide the ability to export this proof for further documentation or dispute resolution.
- The system should automatically associate proof of delivery with the corresponding stop for easy access.

A.3.4 Dynamic Data-Driven Delivery Insights

A.3.4.1 Description and Priority

- Description: Provides detailed statistics on delivery performance, including distance, time, and fuel costs saved.

- Priority: Medium
- Benefit: 7
- Penalty: 4
- Cost: 7
- Risk: 2

A.3.4.2 Stimulus/Response Sequences

- Stimulus: User requests a report or insight into delivery performance.
- Response: The system generates detailed reports on delivery efficiency, including statistics on time, distance, and fuel costs.

A.3.4.3 Functional Requirement

- The system should provide detailed performance reports on each delivery, including metrics like distance, time, fuel savings, and ETA accuracy.
- It must allow users to access and analyze historical route data for performance improvement

A.3.5 Route Export and Data Integration

A.3.5.1 Description and Priority

- Description: Facilitates exporting routes in CSV or Excel formats, ensuring compatibility with other systems.
- Priority: Medium
- Benefit: 7
- Penalty: 3
- Cost: 6
- Risk: 2

A.3.5.2 Stimulus/Response Sequences

- Stimulus: User requests an export of the current route data.
- Response: The system generates the export in CSV or Excel format for integration with other systems.

A.3.5.3 Functional Requirements

- The system should support exporting route data in CSV and Excel formats.
- It must ensure compatibility with other logistics and operations management systems.
- The export should include all relevant stop data such as addresses, times, phone numbers, and other custom fields.

A.4 External Interface Requirements

A.4.1 User Interfaces

The FastTrack application provides multiple user interfaces to facilitate interactions with the system. The application will feature for mobile interface. Below are the logical characteristics for the interfaces between the software and the users.

- Login Screen: The login screen will require users to enter credentials (username/email and password) for authentication. A Forgot Password link will also be provided.
- Home Screen: The home screen will display a dashboard with an overview of deliveries, available routes, and important notifications. A navigation bar will allow access to different sections such as Route Optimization, Settings, and User Profile.
- Route Optimization Screen: This screen will allow users to input delivery destinations and view the optimized route. It will feature a map interface with a Start Navigation button.
- Delivery Details Screen: Once a delivery is selected, users will see detailed information about the delivery, such as destination, estimated time of arrival (ETA), and route breakdown.

A.4.2 Hardware Interfaces

Device Compatibility: The application will support GPS-enabled smartphones and tablets meeting the following minimum requirements:

- RAM: 2 GB or more to ensure smooth operation of the application.
- Storage: 16 GB free space to accommodate the app and required data, including maps and route optimization data.
- Operating Systems: The application will support Android 8.0 (Oreo) or higher, and iOS 12.0 or higher to ensure compatibility with the latest device features and updates.

The mobile devices will connect to the internet via Wi-Fi or cellular networks (3G, 4G, or 5G) to communicate with the backend server for route optimization, delivery updates, and real-time notifications.

A.4.3 Software Interfaces

- Operating Systems: Android (version 8.0 or higher) and iOS (version 12.0 or higher).
- Cloud-based database: MySQL and Firebase for storing user data and product details.
- Third-party APIs: Google Maps API (Android) and Core Location API (iOS) for real-time location tracking and route optimization.
- Data exchanged between the app and server will use JSON format over HTTPS for secure transmission.

A.4.4 Communications Interfaces

- Communication Protocols: HTTP/HTTPS for secure client-server communication.
- Standards and Security: End-to-end encryption for sensitive data.
- Error Handling: Notifications for communication failures.
- Synchronization and Transfer Rates: Real-time synchronization for auction updates, product availability, and location tracking and Optimized for low-latency data transfer.

A.5 Other Nonfunctional Requirements

A.5.1 Performance Requirements

- Response Time: The system must calculate the optimal route within 2 seconds for graphs with up to 500 nodes and within 5 seconds for larger graphs (up to 2,000 nodes).
- Scalability: The system should handle up to 1,000 simultaneous delivery route calculations without significant performance degradation.
- Scalability: The system should handle up to 1,000 simultaneous delivery route calculations without significant performance degradation.
- Data Update: Real-time traffic updates, if integrated, must be processed within 10 seconds to adjust route calculations dynamically.
- Algorithm Efficiency: The implemented algorithm must operate with a time complexity in the range of $O(n^2)$ to $O(n^3)$ for graphs where n is the number of locations (destinations).

A.5.2 Safety Requirements

- Data Integrity: Ensure no loss or corruption of delivery location data during input or output processing.
- Error Handling: Provide clear error messages for invalid inputs.
- Backup and Recovery: Include backup mechanisms to save progress during route calculations, ensuring the system can recover in case of crashes.
- Regulatory Compliance: Adhere to any data protection regulations if user or customer data is involved in the system.

A.5.3 Security Requirements

- Authentication: Require secure login credentials for users accessing the system, especially for systems deployed in enterprise settings.
- Privacy: Ensure sensitive data, such as customer addresses or delivery points, is encrypted during transmission and storage.
- Access Control: Restrict administrative functionalities to authorized personnel only.
- Fraud Prevention: Detect and prevent malicious route manipulations, such as tampering with delivery point data.

A.5.4 Software Quality Attributes

- Usability: Provide a simple interface for inputting delivery points and configuring preferences.
- Modularity: Implement modular code to simplify debugging and future updates, especially for algorithm enhancements.
- Localization: Allow the use of local measurement units and integrate localized traffic data sources if real-time updates are used.
- Availability: Implement failover mechanisms to maintain functionality in case of minor hardware or software failures.
- Adaptability: Provide support for future updates, including the integration of new mapping services, delivery algorithms, or hardware components, with minimal disruption to existing functionality.

A.6 Other Requirements

A.6.1 Database Requirements

- The system must utilize a relational database management system (RDBMS) to store user data, delivery routes, location information, and performance metrics.
- The database schema must be scalable to handle an increasing number of deliveries, users, and data points as the system grows.

A.6.2 Internationalization Requirements

- Currency and distance units should be configurable according to the user's region.
- Date and time formats should be adjustable based on the user's locale to accommodate international users.

A.6.3 Legal Requirements

- The application should include clear terms and conditions and privacy policies to inform users about data usage and security measures.
- The application must comply with data protection regulations such as the General Data Protection Regulation (GDPR) for users within the EU and Data Protection Laws in India (IT Act, 2000) for users within India.

A.6.4 Backup and Recovery

- The system must include automated daily backups of all critical data, ensuring recovery in the event of data loss or corruption.
- A disaster recovery plan should be in place to restore full functionality within 24 hours of a major system failure.

A.7 Appendix A: Glossary

- API (Application Programming Interface): A set of protocols and tools that allow different software applications to communicate with each other.
- GPS (Global Positioning System): A satellite-based navigation system that provides location and time information anywhere on Earth.

- RDBMS (Relational Database Management System): A database management system that stores data in a structured format using rows and columns.
- Route Optimization: The process of determining the most efficient path or route for delivery based on various factors like distance, time, and traffic.
- UI (User Interface): The means by which a user interacts with the system or application, typically through visual elements like buttons, menus, and forms.
- API Key: A unique identifier used to authenticate requests to an API service.
- ETA (Estimated Time of Arrival): The predicted time a delivery will arrive at a given destination.
- GDPR (General Data Protection Regulation): A regulation in the EU law on data protection and privacy in the European Union and the European Economic Area.
- Firebase: A platform developed by Google that provides cloud-based backend services like real-time databases, authentication, and hosting for mobile and web applications.
- Cloud-based Database: A type of database that is hosted on the cloud, allowing for remote storage and access over the internet.

A.8 Appendix B: Analysis Models

- Data Flow Diagram (DFD): A diagram showing how data flows through the system. The DFD will include external entities (users, APIs), processes (route optimization, delivery tracking), data stores (delivery data, user profiles), and data flows.
- Use Case Diagram: A visual representation of the system's functional requirements, detailing how users (delivery drivers, administrators) will interact with the system (e.g., input destinations, request optimized routes).
- Entity-Relationship Diagram (ERD): A diagram that shows the relationships between the different data entities (e.g., Users, Routes, Deliveries, Locations) within the system.
- Class Diagram: A diagram that models the object-oriented structure of the system, showing the classes (e.g., User, Route, Delivery) and their relationships.

A.9 Appendix C: Issues List

- TBD: Integration of third-party APIs: The exact details of integrating the Google Maps API and Core Location API still need to be finalized, including the authentication process and API limits.

- Data Synchronization: There is an ongoing discussion about how the real-time delivery updates will synchronize across multiple devices and servers.
- Traffic Data Accuracy: Ensuring the accuracy and timeliness of real-time traffic data is a challenge, especially in regions with poor coverage or inconsistent data feeds.
- User Interface Design: The exact layout and user flow for the route optimization and delivery tracking screens need further refinement to ensure a seamless user experience.
- Security for User Data: The security measures for sensitive user data, especially delivery addresses and contact details, are being reviewed to ensure full compliance with GDPR and other data protection laws.
- Device Compatibility Issues: Further testing is required to confirm that the app will function properly across all supported devices, especially for different screen sizes and operating systems.
- Route Optimization Algorithm Performance: More tests are needed to ensure the optimization algorithm can efficiently handle large datasets and produce accurate routes in real-time.

References

[1] Jordan Makansi. A greedy quantum route-generation algorithm. *arXiv preprint*, arXiv:2405.03054, 2024.

[2] Application of bin packing algorithms to package assignment problems. *Logistics and Optimization Research*, 2024. Explores Bin Packing Algorithms for package assignment, modeling delivery vehicles as bins to optimize space utilization and minimize vehicle usage.

[3] Amazon Logistics. Ai-driven package assignment and route optimization. *Amazon Research*, 2024. Utilizes AI-driven algorithms for package assignment and the Amazon Routing Engine (ARE) for route optimization.

[4] Google Developers. *Google Maps API Documentation*, 2024. Accessed: 2025-03-25.

[5] Google Developers. *Flutter: Build Apps for Any Screen*, 2024. Accessed: 2025-03-25.

[6] Google Firebase. *Cloud Firestore Documentation*, 2024. Accessed: 2025-03-25.

[7] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT Press, 2022.

[8] A multi-algorithm approach for operational human resources workload balancing in last-mile urban package delivery systems. *arXiv preprint*, 2024.

[9] Flask Community. *Flask: Web Development, One Drop at a Time*, 2024. Accessed: 2025-03-25.

Index

Behavioral Feasibility, 9
Black Box Testing, 36
Detailed Design, 19
Economic Feasibility, 8
Feasibility Study, 8
Functional Requirements, 11
Implementation, 28
Integration Testing, 37
Existing System, 5
Non-Functional Requirements, 11
Proposed System, 6
Requirements Analysis, 10
Requirements Elicitation, 9
Requirements Specification, 10
Requirements Validation, 11
System Design, 13
Technical Feasibility, 9
Testing, 36
Unit Testing, 36
White Box Testing, 37