Department of Computer Science

Graduation Project

Final Report

Project 27

Travello

Indoor Positioning System based on BLE enhanced with KNN approach

Peter Joseph	Jan Kadry
Samer Hany	Bishoy Botros
Ephraim Yousef	Jack Hany
Demiana Emad	Rawan Ahmed

Academic supervisor: Dr. Tarek Salah Abd El-Azeem
Assisting supervisor: Eng. Salma Tarek

July 2024

Abstract

This documentation introduces Travello, an advanced indoor positioning system designed for optimized navigation and luggage tracking within airports. Utilizing Bluetooth Low Energy (BLE) technology, Travello provides real-time location data to passengers, facilitating seamless traversal through airport terminals, departure gates, and amenities, while also enabling luggage monitoring. The paper details Travello's design, implementation, and evaluation, highlighting its operational efficiency and user-centric functionalities, with potential contributions to enhancing airport logistics and passenger satisfaction. Indoor positioning systems (IPS) have become essential for various applications, addressing the limitations of GPS in indoor environments. This paper synthesizes advancements in IPS technologies, encompassing Wi-Fi, ZigBee, and Bluetooth, to enhance accuracy and reliability. It discusses the challenges of indoor environments and the need for tailored positioning solutions, exploring applications such as Real-Time Location Systems (RTLS), indoor navigation, and first-responder location systems. Bluetooth positioning is crucial within IPS but faces challenges due to unstable signal strengths. To address this, the paper introduces a sophisticated positioning methodology that integrates weighted K-nearest neighbors and adaptive bandwidth mean shift techniques. This method utilizes signal strength information for multiple candidate locations, refined for maximum density and accuracy. This research contributes to augmenting IPS capabilities, promoting enhanced indoor location-based services across diverse domains, thus enriching passenger experiences and optimizing airport operations. By leveraging Travello's features, airports can streamline passenger flow, reduce wait times, and improve overall efficiency, ultimately enhancing the travel experience for all stakeholders involved.

Acknowledgements

We would like to express our deepest gratitude to all those who have supported and contributed to the successful completion of our project. First and foremost, we are grateful to our project supervisor, Dr. Tarek Abd El-Azeem, for his invaluable guidance, insightful feedback, and unwavering support throughout the entire process. His expertise and encouragement were pivotal in navigating the challenges and complexities of this project.

Furthermore, we would like to express our sincere admiration and profound thanks to our remarkable teaching assistant, Eng. Salma Tarek, whose unwavering dedication and invaluable contributions have left an indelible impact on our project. Finally, we would like to thank our families for their continuous encouragement, patience, and understanding, which sustained us through the long hours of research and development.

This project would not have been possible without the collective effort and dedication of everyone involved. Thank you.

Contents

Chapter 1

Planning

1.1. Problem definition

Airports are often large, complex, and crowded environments that can be challenging for passengers to navigate. The process of finding gates, amenities, and services can be stressful, particularly for those unfamiliar with the layout. Additionally, the management and tracking of luggage present significant challenges, leading to frequent issues with lost or misplaced bags. These problems can result in increased stress for travelers, reduced satisfaction, and inefficiencies in airport operations.

The primary issues faced by passengers include:

1. **Difficulty in Navigation**

   Airports typically have intricate layouts with multiple terminals, levels, and numerous points of interest. This complexity can cause confusion and delay, particularly during peak travel times

   when passengers are in a hurry. 2% to 8% of travelers miss their flights due to navigational challenges.

2. **Luggage Tracking Issues**

   The process of checking in, handling, and retrieving luggage involves multiple stages, each presenting opportunities for errors and mishandling. Lost or delayed luggage can cause significant inconvenience, distress, and even financial loss to travelers. Current tracking systems often fail to provide passengers with clear visibility into the status and location of their bags.

1.2. Project overview

Indoor Positioning Systems (IPS) are technologies designed to locate objects or people inside buildings or other covered areas, where traditional GPS technology fails to provide accurate positioning due to signal attenuation and multipath effects.

Our project is an indoor positioning system (IPS) designed specifically for airports to enhance user navigation and experience. Leveraging Bluetooth Low Energy (BLE) technology, it provides real-time location tracking, enabling passengers to effortlessly find their way through complex airport environments. The IPS technology employed is pivotal in offering accurate indoor navigation, which traditional GPS systems struggle to provide within enclosed spaces.

1.3. Project Objectives

1 **Navigate Passengers**

The system is designed to assist users in navigating through the often complex and crowded terminal environments. Airports can be daunting places, especially for infrequent travelers or those unfamiliar with the layout. The system enables passengers to effortlessly find their way. The system delivers step-by-step directions, interactive maps, and timely notifications, ensuring travelers can move efficiently and confidently from check-in to boarding. The project’s aim is to reduce the stress associated with navigating large airport spaces.

2 Luggage tracking

Enables users to track passengers' luggage, addressing a common concern among travelers: the safety and location of their bags. QR code technology, the system provides tracking of checked and carry-on luggage throughout the airport. Travelers can receive updates on the status and location of their bags, from check-in and security screening to loading onto the aircraft and arrival at the destination. This not only reduces the anxiety associated with lost or misplaced luggage but also enhances operational efficiency for airport staff by streamlining the baggage handling process. With Travello, passengers can enjoy peace of mind knowing their luggage is accounted for and easily retrievable, ensuring a seamless and stress-free journey.

1.4. Expected Time Plan

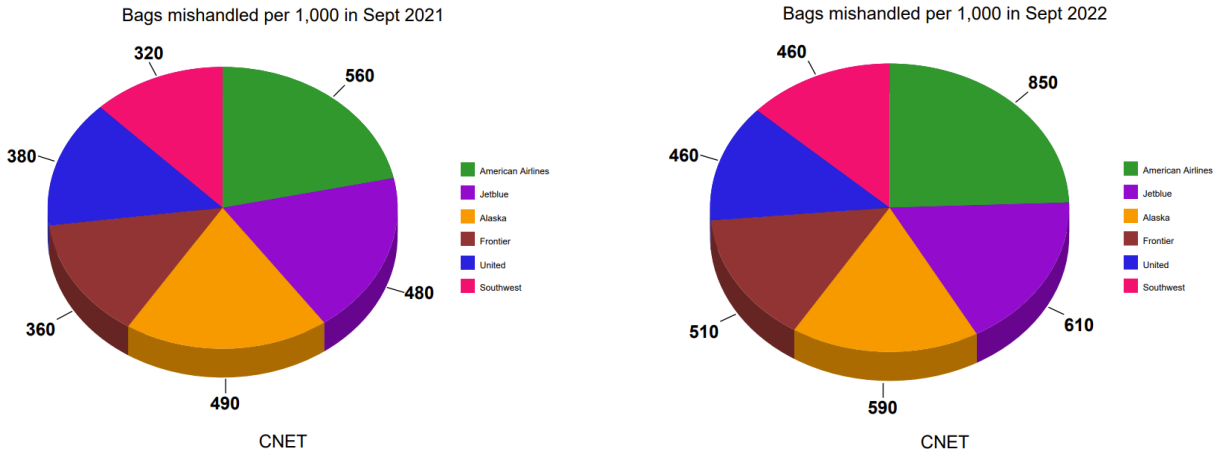
1.5. Motivation

The project is driven by the growing need to enhance the passenger experience within airport environments. As global air travel continues to increase, airports face the challenge of accommodating more passengers while maintaining high standards of service and efficiency. Navigating large and complex airport terminals can be daunting, leading to stress and confusion for travelers. Additionally, the frequent issues of lost or misplaced luggage further exacerbate the travel experience, causing frustration and inconvenience.

Our motivation stems from the desire to address these common problems and to leverage modern technology to create a seamless, stress-free journey for all airport users. By utilizing Bluetooth Low Energy (BLE) and other advanced tracking technologies, we aim to provide precise location services that simplify navigation and enhance luggage tracking. The ultimate goal is to improve overall passenger satisfaction, facilitate airport operations, and contribute to a more efficient and enjoyable travel experience.

According to CNET

Every year the number of mishandled bags exceeds more and more



1.6. Scoped Motivation

The specific focus of the Travello project is to implement a robust indoor positioning system (IPS) that targets two primary areas of improvement within airport operations: user navigation and luggage tracking.

Chapter 2

Survey

2.1. Introduction to Indoor Positioning System

> Global Positioning System (GPS) has been developed in outdoor environments in recent years. GPS offers a wide range of applications in outdoor areas, including military, weather forecasting, vehicle tracking, mapping, farming, and many more. In an outdoor environment, an exact location, velocity, and time can be determined by using GPS. Rather than emitting satellite signals, GPS receivers passively receive them. However, due to No Line-of-Sight (NLoS), low signal strength, and low accuracy, GPS is not suitable to be used indoors. As consequence, the indoor environment necessitates a different Indoor Positioning System (IPS) approach that is capable to locate the position within a structure. IPS systems provide a variety of location-based indoor tracking solutions.
>
> N. Syazwani C. J, Nur Haliza Abdul Wahab, Noorhazirah Sunar, Sharifah H. S. Ariffin, Keng Yinn Wong and Yichiet Aun, "Indoor Positioning System: A Review" International Journal of Advanced Computer Science and Applications (IJACSA), 13(6), 2022

2.2. Similar projects

In our research, we examined several airports worldwide that have already implemented indoor positioning systems (IPS) similar to our project. These airports utilize advanced technologies like Bluetooth Low Energy (BLE), Wi-Fi, and RFID to enhance passenger navigation and improve luggage tracking. By exploring the practical applications and benefits of these existing systems, we gained valuable insights into how IPS can significantly improve the airport experience.

These airports are:

- Miami International Airport (MIA)
- San Francisco International Airport (SFO)
- London Heathrow Airport (LHR)
- Hong Kong International Airport (HKG)
- Dubai International Airport (DXB)
- Amsterdam Airport Schiphol (AMS)
- Helsinki Airport (HEL)
- Frankfurt Airport (FRA)
- Singapore Changi Airport (SIN)
- Dallas/Fort Worth International Airport (DFW)

We explored the features that these systems apply and what are the main points they focus on such as: Accuracy, system integration, cost, coverage, security, accessibility, user experience, maintenance, real-time updates, feedback mechanism.

We tried to apply as much as possible, it helped us in making our system and we tried to apply as much as possible to our project.

Criteria	MIA	SFO	AMS	HEL	SIN	LHR	HKG	DXB	FRA	DFW
Technology Used	BLE, WiFi	BLE, WiFi	BLE, WiFi, UWB	BLE, WiFi	BLE, WiFi	BLE, WiFi, RFID	BLE, WiFi, RFID	BLE, WiFi, GPS	BLE, WiFi	BLE, WiFi
Accuracy	1-3 meters	1-3 meters	<1 meter	1-3 meters	1-3 meters	<1 meter	<1 meter	1-3 meters	1-3 meters	1-3 meters
Coverage	Full terminal coverage	Full terminal coverage	Full terminal coverage	Full terminal coverage	Full terminal coverage	Full terminal coverage	Full terminal coverage	Full terminal coverage	Full terminal coverage	Full terminal coverage
User Experience	Excellent	Excellent	Excellent	Excellent	Excellent	Excellent	Excellent	Excellent	Excellent	Excellent
System Integration	Seamless with airport systems	Seamless with airport systems	Seamless with airport systems	Seamless with airport systems	Seamless with airport systems	Seamless with airport systems	Seamless with airport systems	Seamless with airport systems	Seamless with airport systems	Seamless with airport systems
Cost	High	High	High	High	High	High	High	High	High	High

2.3. Technologies used to Apply IPS

Wi-Fi Positioning Systems (WPS)

- Utilizes existing Wi-Fi access points to triangulate the position of devices.
- Measures signal strength (RSSI) from multiple access points.
- Typically offers room-level accuracy.
- Commonly used in environments with extensive Wi-Fi infrastructure.

Fen Liu, Jing Liu, Yuqing Yin, Wenhan Wang, “A Survey on WiFi-Based Indoor Positioning Technologies”, *IET Communications*, 2020.

Bluetooth Low Energy (BLE)

- Employs small, battery-powered beacons emitting Bluetooth signals.
- Devices calculate their position based on signal proximity and strength.
- Offers high accuracy within a limited range.
- Ideal for retail environments and indoor navigation.

Uday Agarwal, “Indoor Positioning with BLE Beacons”, 2017.

Ultra-Wideband (UWB)

- Uses short-range radio waves with very high bandwidth.
- Provides centimeter-level accuracy by measuring the time of flight of signals.
- Suitable for precise tracking in dense environments.
- Common in industrial settings and asset tracking.

Hao Zhang, Qing Wang, Chao Yan, Jiujing Xu, Bo Zhang, "Research on UWB Indoor Positioning Algorithm under the Influence of Human Occlusion and Spatial NLOS", *Remote Sens*, 2022.

Radio Frequency Identification (RFID)

- Leverages RFID tags and readers to determine location.
- Tags can be passive or active, with active tags offering better range.
- Often used for asset tracking and inventory management.
- Limited accuracy but low cost and easy implementation.

Ching-Sheng Wang, Chien-Liang Chen, You-Ming Guo, "A Real-Time Indoor Positioning System Based on RFID and Kinect" In J. J. Park, L. T. Yang, & C. Lee (Eds.), *Information Technology Convergence* (pp. 575-585), 2013.

2.4. Positioning Methods

Received Signal Strength Indicator (RSSI)

- **Signal Strength Measurement**
  - Measure the signal strength (RSSI) of the received signal from multiple reference points.
  - RSSI decreases as the distance between the transmitter and receiver increases.
- **Path Loss Model**
  - Use a path loss model to relate RSSI values to distances.
  - Common models include the log-distance path loss model, which accounts for environmental factors.
- **Trilateration**
  - Combine distance estimates from at least three reference points.
  - Use trilateration to compute the device's position by finding the intersection of the circles with radii equal to the calculated distances.
- **Error Mitigation**
  - Mitigate errors due to multipath propagation and signal interference.
  - Implement filtering techniques and calibrate the system for improved accuracy.

Angle of Arrival (AoA)

- **Signal Angle Measurement**
    - Use antenna arrays to measure the angle at which signals arrive.
    - Calculate the AoA by comparing the phase difference or time difference of arrival at different antennas.
- **Multiple Reference Points**
    - Collect AoA measurements from multiple reference points or anchor nodes.
    - Each AoA measurement provides a line along which the device is located.
- **Triangulation**
    - Use triangulation to determine the device's position.
    - Calculate the intersection point of the lines defined by the AoA measurements from different reference points.
- **Calibration and Error Correction**
    - Perform system calibration to correct for biases and errors in AoA measurements.
    - Use algorithms like Kalman filtering to integrate and smooth the position estimates.

Wenzhao Shu, Shuai Wang, “An Indoor Positioning System of Bluetooth AOA Using Uniform Linear Array Based on Two-point Position Principle”, *In IEEE SENSORS JOURNAL, VOL.30, NO.8,* 2022.

Chapter 3

Tools & Technologies

3.1. Hardware

BLE beacons using ESP32

The ESP32 is a powerful and versatile microcontroller developed by Espressif Systems. It features a dual-core processor, integrated Wi-Fi and Bluetooth capabilities, and a rich set of peripherals, making it ideal for a wide range of applications in IoT, wearables, and smart home devices. The ESP32's robust connectivity options allow for seamless integration into various wireless networks, while its high processing power and extensive memory support complex tasks and data handling. Additionally, the ESP32 supports a variety of programming environments, including the Arduino IDE and Espressif's own ESP-IDF, offering flexibility and ease of development. Its low power consumption and efficient power management make it suitable for battery-powered projects, further enhancing its appeal in diverse technological applications.

3.2. Development Utilities

Arduino IDE

The Arduino Integrated Development Environment (IDE) is a user-friendly platform designed for coding, compiling, and uploading programs to Arduino microcontrollers and compatible boards. It features a simple, intuitive interface that allows developers to write code in the Arduino programming language, which is based on C/C++. The IDE provides a range of built-in libraries and examples, facilitating rapid development and prototyping of projects in areas such as robotics, IoT, and automation. With its extensive community support, the Arduino IDE is an accessible tool for both beginners and experienced developers, enabling the creation of innovative and interactive electronic projects with ease.

Flutter

Flutter is an open-source UI software development kit created by Google, designed for building natively compiled applications for mobile, web, and desktop from a single codebase. It uses the Dart programming language and offers a rich set of pre-designed widgets that enable developers to create visually appealing, responsive, and performant user interfaces. Flutter's hot-reload feature allows for real-time code changes and fast iteration, significantly speeding up the development process.

Firebase

Firebase is a comprehensive app development platform provided by Google, offering a suite of cloud-based tools and services to support the development, deployment, and scaling of mobile and web applications. It includes features such as real-time database management, authentication, cloud storage, hosting, and machine learning capabilities. Firebase simplifies backend development with its serverless architecture, allowing developers to focus on building high-quality user experiences.

.NET WPF

Windows Forms, part of the Microsoft .NET framework, is a graphical user interface (GUI) class library used for developing rich desktop applications for Windows. It provides a comprehensive set of controls and components for building visually appealing and interactive user interfaces. With Windows Forms, developers can design forms and windows using a drag-and-drop interface in Visual Studio, allowing for rapid development and easy customization.

3.3. Bluetooth Low Energy (BLE)

Bluetooth Low Energy (BLE) is a wireless communication technology designed for short-range data transmission with minimal power consumption, making it ideal for battery-powered devices. Introduced as part of the Bluetooth 4.0 specification, BLE supports a wide range of applications, including wearable devices, smart home products, health and fitness monitors, and IoT devices. It operates on the same 2.4 GHz frequency band as classic Bluetooth but is optimized for low latency and efficient energy usage. BLE's ability to maintain long battery life while ensuring reliable connectivity has made it a popular choice for modern wireless communication needs, particularly in scenarios where conserving energy is crucial

3.4. Advantages of BLE-based Indoor Positioning

1. **Energy Efficiency**
   BLE's primary advantage lies in its low power consumption, making it ideal for battery-powered devices and long-term deployments. It enables devices to operate for extended periods without frequent recharging

2. **Small Form Factor**
   BLE devices are compact and can be easily integrated into various applications, from wearable devices to building infrastructure.

3. **Cost-Effectiveness**
   BLE technology is relatively affordable compared to other indoor positioning technologies, making it a viable option for various budgets.

4. **Wide Adoption**
   The widespread adoption of BLE in smartphones and other devices ensures a readily available infrastructure for indoor positioning applications.

3.5. Challenges and Limitations of BLE-based Indoor Positioning

1. **Signal Attenuation**
   Signal strength can be affected by obstacles and interference, leading to potential inaccuracies.

2. **Beacon Management**
   Maintaining and managing a large number of beacons can be complex, requiring regular calibration and updates.

3.6. Use Cases and Applications of BLE-based Indoor Positioning

1. **Navigation**
   Providing indoor navigation in shopping malls, airports, hospitals, and other complex environments, enhancing user experience and accessibility.
2. **Asset Tracking**
   Monitoring the location of equipment, tools, or inventory in warehouses, factories, and logistics centers, improving efficiency and security.
3. **Employee Tracking**
   Monitoring the location of employees in a work environment, ensuring safety, productivity, and efficient resource allocation.
4. **Marketing & Analytics**
   Gathering insights on customer behavior and movement patterns in retail stores, museums, and other public spaces, optimizing marketing strategies.

3.7. Quick Response Code (QR)

QR code technology involves the use of square-shaped barcodes that can be scanned using a smartphone or QR code reader. These codes store data, such as URLs, contact information, or text, in a compact and easily readable format. They are widely used for various applications, including marketing, ticketing, and payment systems, due to their ability to quickly link users to digital content. QR codes can be customized in design and color, making them both functional and visually appealing. Their ease of use and versatility have made them a popular tool for connecting the physical and digital worlds.

Chapter 4

Methodology

4.1. Our Methodology

Agile is a type of software development methodology that anticipates the need for flexibility and applies a level of pragmatism to the delivery of the finished product. Agile software development requires a cultural shift in many companies because it focuses on the clean delivery of individual pieces or parts of the software and not on the entire application.

Benefits of Agile include its ability to help teams in an evolving landscape while maintaining a focus on the efficient delivery of business value. The collaborative culture facilitated by Agile also improves efficiency throughout the organization as teams work together and understand their specific roles in the process. Finally, companies using Agile software development can feel confident that they are releasing a high-quality product because testing is performed throughout development. This provides the opportunity to make changes as needed and alert teams to any potential issues.

Agile has largely replaced waterfall as the most popular development methodology in most companies but is itself at risk of being eclipsed or consumed by the growing popularity of DevOps.

4.2. The four values of Agile

In 2001, 17 software development professionals gathered to discuss concepts around the idea of lightweight software development and ended up creating the Agile Manifesto. The Manifesto outlines the four core values of Agile, and although there has been debate about whether the Manifesto has outlived its usefulness, it remains at the core of the Agile movement.

The four core values outlined in the Agile Manifesto are as follows:

1. **Individual interactions are more important than processes and tools**

   People drive the development process and respond to business needs. They are the most important part of development and should be valued above processes and tools. If the processes or tools drive development, then the team will be less likely to respond and adapt to change and, therefore, less likely to meet customer needs.

2. **A focus on working software rather than thorough documentation**

   Before Agile, a large amount of time was spent documenting the product throughout development for delivery. The list of documented requirements was lengthy and would cause long delays in the development process. While Agile does not eliminate the use of documentation, it streamlines it in a way that provides the developer with only the information that is needed to do the work -- such as user stories. The Agile Manifesto continues to place value on the process of documentation, but it places higher value on working software

3. Collaboration instead of contract negotiations

Agile focuses on collaboration between the customer and project manager, rather than negotiations between the two, to work out the details of delivery. Collaborating with the customer means that they are included throughout the entire development process, not just at the beginning and end, thus making it easier for teams to meet the needs of their customers. For example, in Agile, the customer can be included at different intervals for demos of the product. However, the customer could also be present and interact with the teams daily, attend all meetings and ensure the product meets their desires.

4. A focus on responding to change

Traditional software development used to avoid change because it was considered an undesired expense. Agile eliminates this idea. The short iterations in the Agile cycle allow changes to easily be made, helping the team modify the process to best fit their needs rather than the other way around. Overall, Agile software development believes change is always a way to improve the project and provide additional value.

4.3. The 12 principles of Agile

The Agile Manifesto also outlined 12 core principles for the development process. They are as follows:

1. Satisfy customers through early and continuous delivery of valuable work.
2. Break big work down into smaller tasks that can be completed quickly.
3. Recognize that the best work emerges from self-organized teams.
4. Provide motivated individuals with the environment and support they need and trust them to get the job done.
5. Create processes that promote sustainable efforts.
6. Maintain a constant pace for completed work.
7. Welcome changing requirements, even late in a project.
8. Assemble the project team and business owners daily throughout the project.
9. Have the team reflect at regular intervals on how to become more effective, then tune and adjust behavior accordingly.
10. Measure progress by the amount of completed work.
11. Continually seek excellence.
12. Harness change for a competitive advantage.

4.4. The Agile software development cycle

The Agile software development cycle can be broken down into the following six steps:

- concept
- inception
- iteration/construction
- release
- production
- retirement

The first step, concept, involves the identification of business opportunities in each potential project as well as an estimation of the time and work that will be required to complete the project. This information can then be used to prioritize projects and discern which ones are worth pursuing based on technical and economic feasibility.

During the second step, inception, team members are identified, funding is established, and the initial requirements are discussed with the customer. A timeline should also be created that outlines the various responsibilities of teams and clearly defines when work is expected to be completed for each sprint. A sprint is a set period during which specific work must be completed and made ready for review.

The third step, iteration/construction, is when teams start creating working software based on requirements and continuous feedback. The Agile software development cycle relies on iterations -- or single development cycles -- that build upon each other and lead into the next step of the overall development process until the project is completed. Each iteration typically lasts between two to four weeks, with a set completion date. The goal is to have a working product to launch at the end of each iteration.

Multiple iterations occur throughout the development cycle and they each possess their own workflow. A typical iteration flow consists of the following:

- defining requirements based on the product backlog, sprint backlog and customer and stakeholder feedback.
- developing software based on the set requirements.
- conducting QA testing, internal and external training and documentation.
- delivering and integrating the working product into production; and
- gathering customer and stakeholder feedback on the iteration to define new requirements for the next sprint.

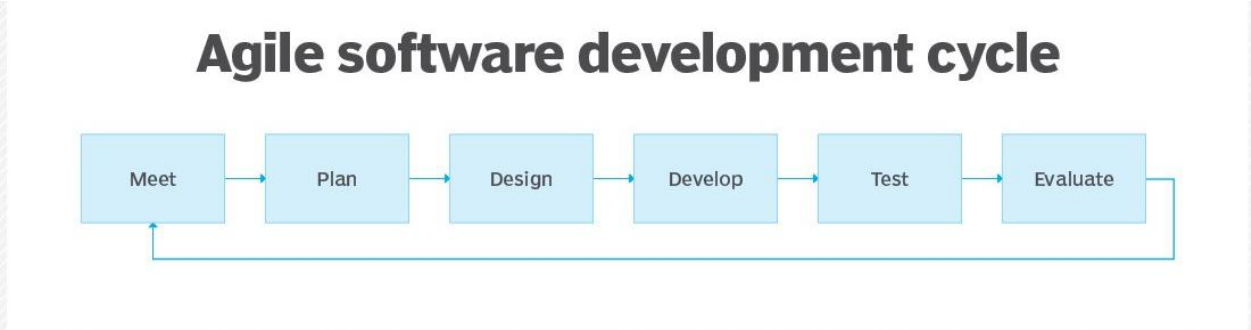
The fourth step, release, involves final QA testing, resolution of any remaining defects, finalization of the system and user documentation and, at the end, release of the final iteration into production.

After the release, the fifth step, production, focuses on the ongoing support necessary to maintain the software. The development teams must keep the software running smoothly while also teaching users exactly how to use it. The production phase continues until the support has ended or the product is planned for retirement.

The final step, retirement, incorporates all end-of-life activities, such as notifying customers and final migration. The system release must be removed from production. This is usually done when a system

needs to be replaced by a new release or if the system becomes outdated, unnecessary or starts to go against the business model.

Throughout the Agile cycle, different features can be added to the product backlog, but the entire process should consist of repeating each step over and over until every item in the backlog has been satisfied. This makes the Agile cycle more of a loop than a linear process. At any time, an enterprise can have multiple projects occurring simultaneously with iterations that are logged on different product lines and a variety of internal and external customers providing different business needs.



4.5. Agile vs Waterfall methodology

Agile and Waterfall are both Software Development Lifecycle (SDLC) methodologies that have been widely adopted in the IT industry.

The Waterfall framework was designed to enable a structured and deliberate process for developing high quality information systems within project scope. The spirit of becoming more adaptive through the real-world implementation of a software project plan gave way to the Agile methodology.

Both Waterfall and Agile require organizations to follow certain operating principles—but the practice often departs from the principles. It is therefore important to understand what Agile and Waterfall mean and how they differ as you make your choice for an SDLC framework that best suits your development goals.

Waterfall:

is a classical SDLC methodology that follows logical progression of linear and sequential phases within the project lifecycle process.

Some of the key principles in Waterfall include:

- Sequential structure. Typically includes phases such as Requirements, Design, Implementation, Verification, and Maintenance.
- Strong focus on documentation. Extensive details that define project requirements and implementation process.
- Low customer involvement. Requirements must be agreed early during the project lifecycle. Once requirements are defined, the development process is strongly focused on meeting the agreed requirements

Agile principles:

The Agile SDLC model is designed to facilitate change and eliminate waste processes (similar to Lean). It replaces a command-and-control style of Waterfall development with an approach that prepares for and welcomes changes.

4.6. Issues with the Waterfall approach

The comparison between basic principles of Waterfall and Agile methodologies point to some key issues with the Waterfall model, especially when considering that it remained the de facto SDLC standard for decades:

- **The lack of communication** introduces highly varied interpretation of requirements and the documentation between team members.
- **The lack of adaptability** renders the model unsuitable for most consumer-focused software projects where the requirements must follow unpredictable, dynamic, and rapidly evolving external change agents.
- **The strict focus on fulfilling original requirements** discourages mistakes and change. As a result, creativity, innovation, and novelty are suppressed. Lack of flexibility prevents experiences that help identify areas of improvement or change for the better.
- **The lack of communication** between engineering teams, customers, and real-world **users** creates the lack of technical empathy. As a result, many expensive innovations fail to gain market traction, whereas agile startup firms focusing on specific customer problems gain popularity rapidly in comparison.

4.7. Issues with Agile

At the same time, the Agile SDLC methodology hasn't proven to be a silver bullet. Although the principles of the Agile model aim to solve problems that may arise from the Waterfall approach, many organizations fail to realize the promised advantages.

This issue emerges due to the following reasons:

- **Organizations follow Agile for planning**, but swiftly transition to "fast waterfall" sprints in implementation. The agility isn't truly designed and embedded during the execution.
- **There's no coherent system design** with practical provisions for improving or changing software functionality. As a result, iterating on customer feedback only adds to complexity. The organization is forced to follow the traditional Waterfall model as it fixes each issue based on feedback.

- **The Agile mindset isn't truly followed across the organization.** Once the team meetings end, individuals may struggle to adapt—unless a system of change is introduced. For instance, siloed organizational departments and lack of collaboration tools restrict the ability to be truly collaborative and Agile in practice.
- **Technical debt accumulates fast** during the Agile process, especially if the sprints are solely focused on bringing new functional improvements instead of fixing quality issues early during the SDLC process.
- **Evaluating the wrong metrics** for project progress gives a false picture of success. Critical performance lapses aren't discovered until too late into the SDLC pipeline.
-
- **Following Agile isn't entirely sufficient**. The value-driven perspective requires organizations to identify the hurdles that prevent them from achieving the goals of the Agile SDLC methodology: fast-paced delivery of high-quality software, lower waste process, and satisfied customers.

Agile	Waterfall
Iterative development in short sprints	Sequential development process in pre-defined phases
Flexible and adaptive methodology	The process is documented and follows the fixed structure and requirements agreed in the beginning of the process
Feedback-based approach: Sprints lead to short build updates that are evaluated on and guide the future direction of the development process.	Limited and delayed feedback: The software quality and requirements fulfilment isn't evaluated until the final phase of the development processes when testers and customer feedback is requested.
A provision for adaptability: Project development requirements and scope is expected to change over the course of the iterative development process.	The requirements and scope are definitive once agreed upon.
The SDLC phases overlap and begin early in the SDLC: planning, requirements, designing, developing, testing, and maintenance.	The SDLC phases are followed in order, with no overlap. Members of one functional group are not involved in another phase that doesn't belong to their job responsibilities.
Follows a mindset of collaboration and communication. The requirements, challenges, progress, and changes are discussed between all stakeholders on a continuous basis.	Follows a project-focused mindset with the aim of fully completing the SDLC process.
Responsibilities and hierarchical structure can be interchangeable between team members.	Fixed individual responsibilities, particularly in management positions.
All team members focused on end-to-end completion (achieved sequentially) for the projects.	Team members focused on their responsibilities only during their respective SDLC phases.

Chapter 5

Our Project

5.1. Logo and Name

5.2. Plan:

The development and deployment of Travello follow a structured plan, encompassing all phases from initial design to final implementation and ongoing support. Our plan is divided into several key stages:

1. **Initiation**:

- Defining project goals and objectives.
- Conducting feasibility studies and risk assessments.
- Securing stakeholder buy-in and project funding.

2. **Planning:**

- Detailed project scoping and requirements gathering.
- Developing the project timeline and milestones.
- Resource allocation and team formation.
- Establishing the project management framework and tools.

3. **Design:**

- Creating system architecture and technical specifications.
- Designing user interface (UI) and user experience (UX) elements.
- Developing wireframes and prototypes for feedback.

4. Development:

- Setting up BLE beacons and infrastructure within the airport.
- Coding and integrating the mobile application and backend systems.
- Implementing navigation and luggage tracking features.
- Conducting iterative development cycles with regular reviews.

5. Testing:

- Conducting unit, integration, and system testing.
- Performing usability tests.
- Identifying and resolving any issues or bugs.

6. Maintenance and Support:

- Regular inspections and updates of BLE beacons and software to maintain optimal performance.
- Airport staff are designated to generate QR codes for luggage, assist passengers with questions or emergencies, and maintain communication through the Travello app. Additionally, an admin interface is planned to oversee and manage the flow of passengers and staff movements, ensuring smooth operations and addressing any issues promptly

5.3. Project Backlogs

The project backlog is a comprehensive list of all tasks, features, enhancements, and fixes needed to complete Travello. This backlog is managed and prioritized to guide the development team throughout the project lifecycle.

1. User Stories

Each task is formulated as a user story, capturing the needs and expectations from the end-user's perspective. For example:

- "As a passenger, I want to receive updates on my luggage location so that I can ensure it arrives with me."
- "As an admin, I want to monitor passenger flow to optimize resource allocation."

2. Prioritization

We use a prioritization system to focus on delivering the most critical features first. Items are ranked based on their importance to the project's goals and user needs.

3. Estimation

Each backlog item is estimated in terms of effort required, helping in planning and resource allocation.

Detailed Backlog Items

1. Implement QR Code Generation

Title	Description	Priority	Estimation	Status	Duties	Functionality	Acceptance Criteria
Implement QR Code Generation	Develop a system for staff to generate and attach QR codes to passenger luggage for tracking at specific checkpoints.	High	8 story points	Delivered	Design QR code generation algorithm Create UI for staff to generate QR codes Integrate QR codes with tracking system	Allow staff to generate unique QR codes for each piece of luggage. QR codes must be scannable and link to the luggage's tracking information.	Staff can generate QR codes quickly and attach them to luggageQR codes link correctly to tracking system.

2. Setup Checkpoint Scanning System

Title	Description	Priority	Estimation	Status	Duties	Functionality	Acceptance Criteria
Setup Checkpoint Scanning System	Establish QR code scanners at key checkpoints to monitor luggage movement.	High	10 story points	Delivered	Install and configure QR code scanners Ensure scanners are integrated with the tracking system	Scanners must accurately read QR codes and update luggage location in the system. Reliable operation in various airport conditions.	QR code scanners are operational at all checkpoints. Luggage locations update correctly when scanned.

3. Design User Interface for Navigation

Title	Description	Priority	Estimation	Status	Duties	Functionality	Acceptance Criteria
Design User Interface for Navigation	Develop UI for guiding passengers through the airport, ensuring ease of use and clarity.	Medium	5 days	Delivered	Create wireframes and design mockups Develop UI components. Integrate with backend navigation system	The UI must provide clear, real-time directions and updates. Users should find it easy to navigate to gates and amenities.	UI is user-friendly and intuitive Passengers can navigate the airport effectively using the app.

4. Create Admin Dashboard

Title	Description	Priority	Estimation	Status	Duties	Functionality	Acceptance Criteria
Create Admin Dashboard	Develop a dashboard for admins to track passenger and staff movements, and oversee operational flow.	High	7 days	Delivered	Design and develop dashboard UI Integrate tracking and monitoring data Implement reporting features	The dashboard should display real-time data on passenger and staff movements. Provide tools for operational oversight and reporting.	Admins can monitor and manage the flow effectively Real-time updates and reporting are accurate and timely.

5. Develop Staff Support System

Title	Description	Priority	Estimation	Status	Duties	Functionality	Acceptance Criteria
Develop Staff Support System	Implement a support feature where staff can respond to user inquiries and emergencies through the Travello app.	Medium	6 story points	Delivered	Develop communication platform for support ntegrate emergency response tools Ensure user accessibility	Staff can communicate with users through the app. Provide tools for emergency support and handling inquiries.	Staff can assist users effectively through the app Emergency and inquiry response tools are functional and reliable.

6. Integration with Airport Systems

Title	Description	Priority	Estimation	Status	Duties	Functionality	Acceptance Criteria
Integration with Airport Systems	Integrate Travello with existing airport systems for data on flights, gates, and security.	High	8 days	Not Started	Identify and connect to airport data sources Develop data integration APIs Ensure real-time data flow	Real-time integration with flight, gate, and security information. Reliable data exchange between systems.	Travello displays accurate and up-to-date airport informationData integration is seamless and reliable.

7. Conduct Usability Testing

Title	Description	Priority	Estimation	Status	Duties	Functionality	Acceptance Criteria
Conduct Usability Testing	Plan and execute tests to gather feedback on user experience and functionality.	Medium	3 days	Delivered	Develop testing scenarios Recruit test users Collect and analyze feedback	Feedback must provide insights into user experience Identify areas for improvement in the app's functionality.	Usability tests are conducted with representative users. Feedback is used to make iterative improvements.

8. Regular Maintenance Plan

Title	Description	Priority	Estimation	Status	Duties	Functionality	Acceptance Criteria
Regular Maintenance Plan	Outline procedures for the ongoing maintenance of BLE beacons, QR scanners, and the Travello app.	Medium	4 story points	Not Started	Develop maintenance schedules Define update protocols Plan for regular inspections	Ensure consistent performance of beacons, scanners, and app Address issues proactively through scheduled maintenance.	Maintenance protocols are documented, and actionable Regular checks and updates are performed as scheduled.

9. Security and Privacy Implementation

Title	Description	Priority	Estimation	Status	Duties	Functionality	Acceptance Criteria
Security and Privacy Implementation	Implement robust security measures to protect user data and ensure privacy.	High	5 story points	Delivered	Implement data encryption Develop secure authentication methods Ensure compliance with regulations	Protect user data from unauthorized access and breachesEnsure compliance with data protection laws.	Data is securely encrypted. User privacy and data protection measures meet regulatory standards.

Backlog Management

Regular backlog reviews are conducted to refine items, update priorities, and ensure alignment with project goals. These sessions involve stakeholders and development team members to incorporate feedback and make necessary adjustments.

5.4. Project Parts

Travello is composed of several integral parts, each contributing to its overall functionality and user experience. These parts include both hardware and software components that work together to provide accurate navigation and real-time luggage tracking within the airport.

1. Bluetooth Low Energy (BLE) Beacons

- **Role**: BLE beacons are the cornerstone of Travello's indoor positioning system. They emit signals at regular intervals, which are picked up by the user's device to determine their location within the airport.
- **Deployment**: Beacons are strategically placed throughout the airport to ensure comprehensive coverage. Key areas include entrance gates, check-in counters, security checkpoints, departure gates, and baggage claim areas.
- **Maintenance**: Beacons require minimal maintenance but are monitored regularly to ensure optimal performance and battery life.

2. Mobile Application

- **Functionality**: The Travello mobile app is the primary user interface for passengers. It provides real-time navigation, luggage tracking, and personalized notifications. The app leverages BLE signals to pinpoint the user's location and guide them through the airport.
- **Design**: The app is designed with user-friendliness in mind, featuring an intuitive layout and clear visual directions. It supports both iOS and Android platforms to ensure accessibility for a broad range of users.
- **Features**: Key features include interactive maps, turn-by-turn directions, luggage updates, and customizable alerts about flight status and gate changes.

3. Backend System

- Infrastructure: The backend system processes data from BLE beacons and the mobile app to deliver location information and notifications. It also manages user profiles, preferences, and system settings.
- Data Management: The system handles the storage and retrieval of location data, ensuring secure and efficient data processing. Robust encryption methods are employed to protect user privacy.
- Integration: The backend integrates with existing airport systems, such as flight information databases and security protocols, to provide a seamless user experience.

4. Navigation and Tracking Algorithms

- **Positioning Algorithms**: Travello employs advanced algorithms to enhance positioning accuracy, such as weighted K-nearest neighbors and adaptive bandwidth mean shift techniques. These algorithms enhance signal strength data to determine the user's precise location.
- **Route Optimization**: The system calculates the most efficient routes through the airport, considering factors like current location, destination, and potential obstacles or delays.
- **Real-Time Updates**: The algorithms continuously update location and route information based on real-time data, ensuring that users receive the most accurate and timely guidance.

5. User Interface (UI) and User Experience (UX)

- **Design Principles**: The UI and UX are designed to be simple and intuitive, reducing the cognitive load on users and making navigation straightforward. The design accommodates diverse user needs, including those with accessibility requirements.
- **Feedback and Iteration**: User feedback is integral to the design process. Usability testing sessions are conducted to gather insights and refine the interface, ensuring that it meets user expectations and enhances their experience.

6. Analytics and Reporting

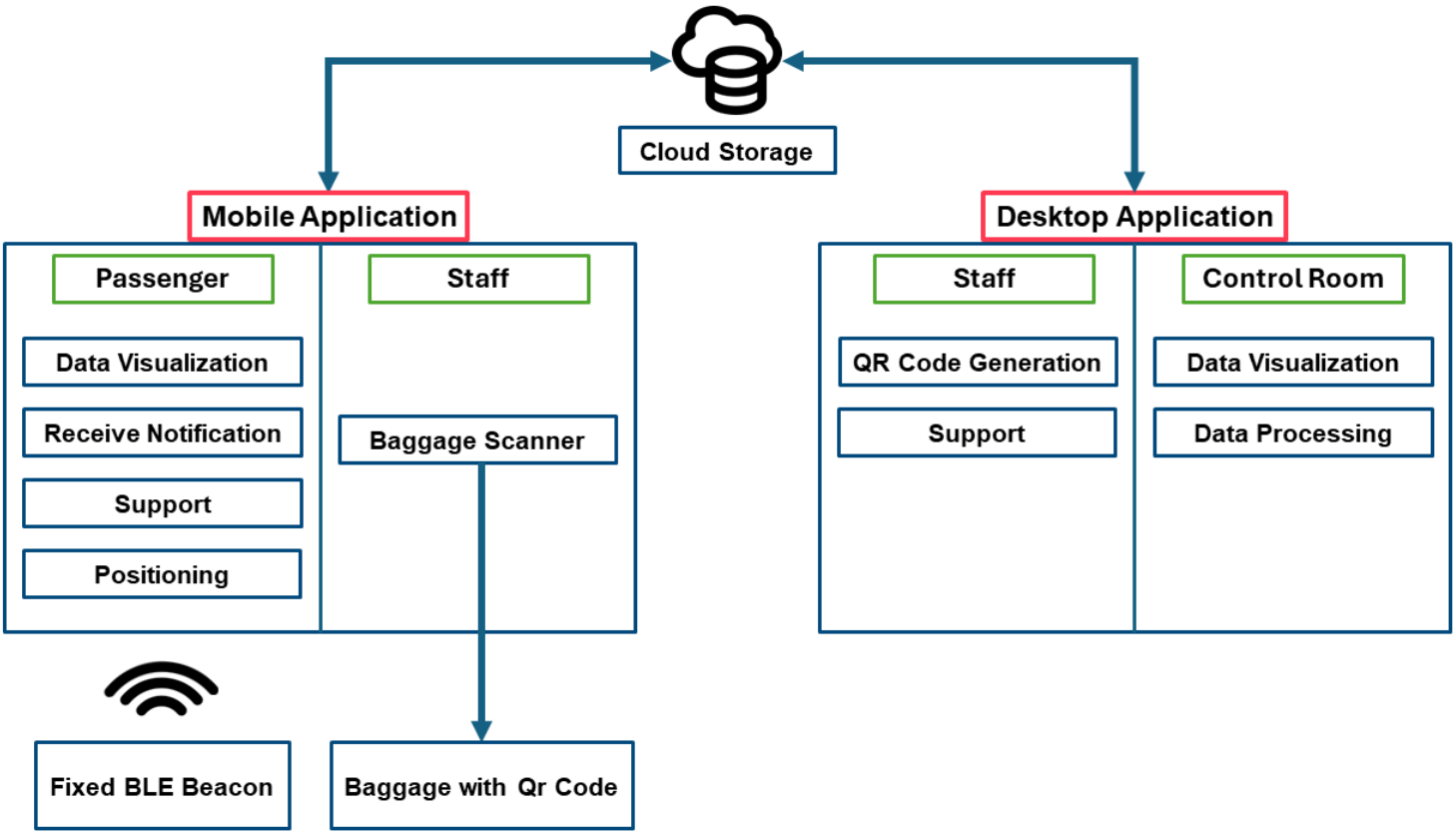
- **Passenger Flow Analysis**: Travello provides detailed analytics on passenger movement and behavior within the airport. This data helps airport authorities understand congestion patterns and optimize resource allocation.
- **Luggage Tracking Reports**: The system generates reports on luggage movement and location, aiding in the quick resolution of any issues and improving operational efficiency.
- **Usage Metrics**: Travello tracks user interactions and system performance, providing insights into app usage, feature popularity, and areas for improvement.

By combining these components, Travello offers a comprehensive and reliable solution for indoor navigation and luggage tracking, significantly enhancing the travel experience for airport passengers and streamlining operations for airport authorities.

Chapter 6

System Design

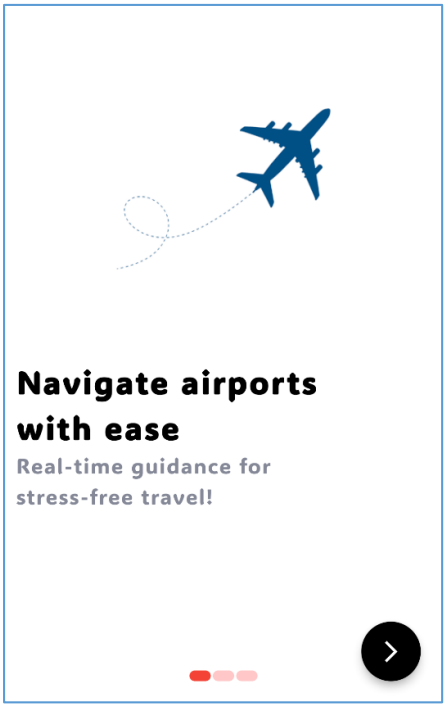
6.1. System Architecture

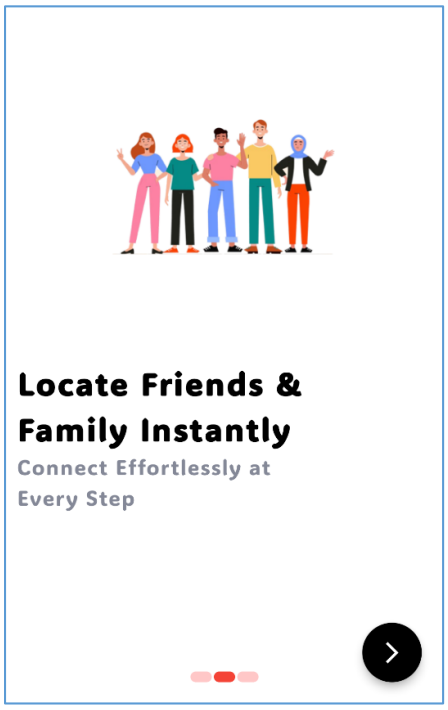


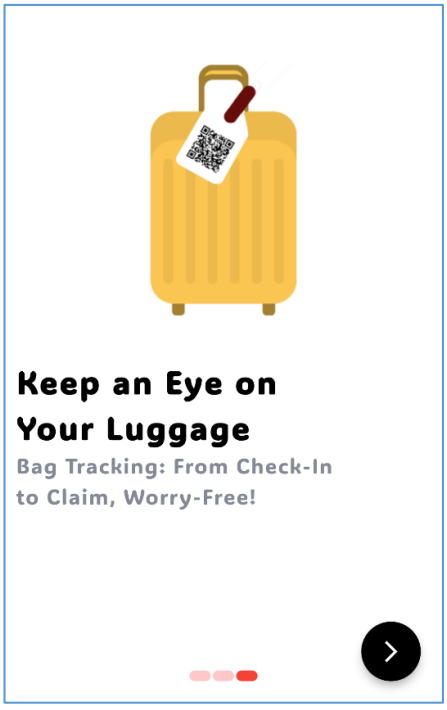
6.2. User Interface

Splash Screen

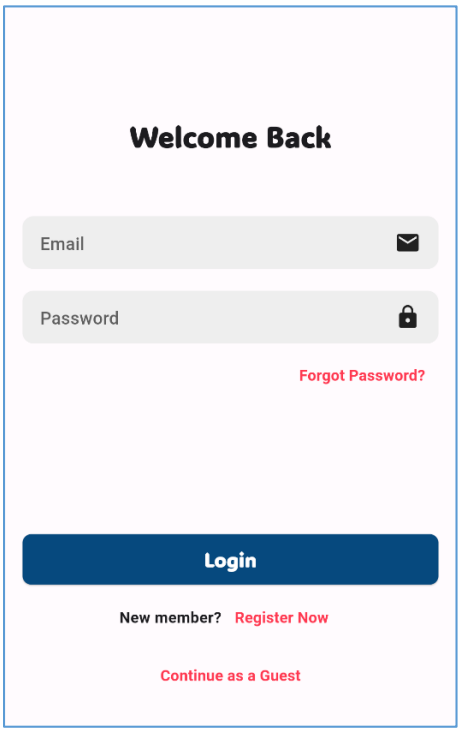
Onboarding Screens





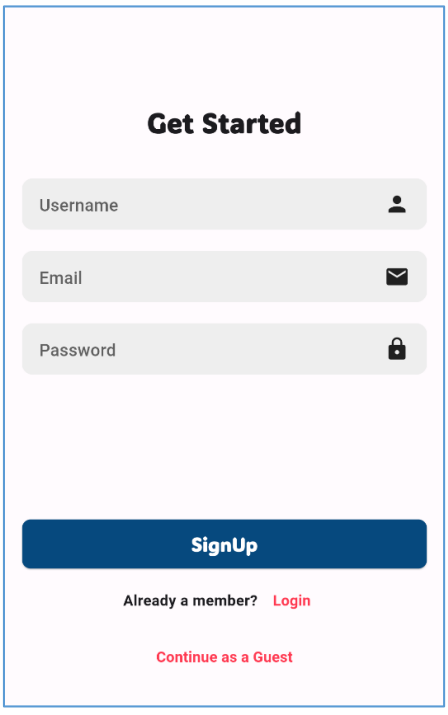


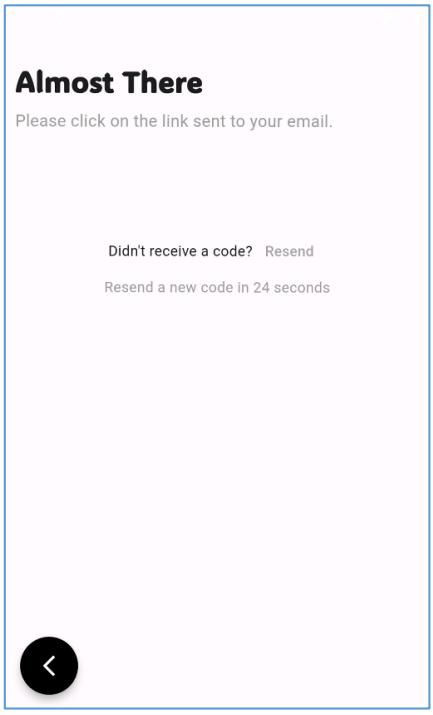
Login Screen where user enter his credentials correctly to login into the application. The user can change his forgotten password by clicking on Forget Password then the user will receive an email to change password. The user can continue as a guest but will not have a complete access to all system's features.



For new members they can create a new account provide their username, real email address and strong password. Then a verification link is sent to their email address.

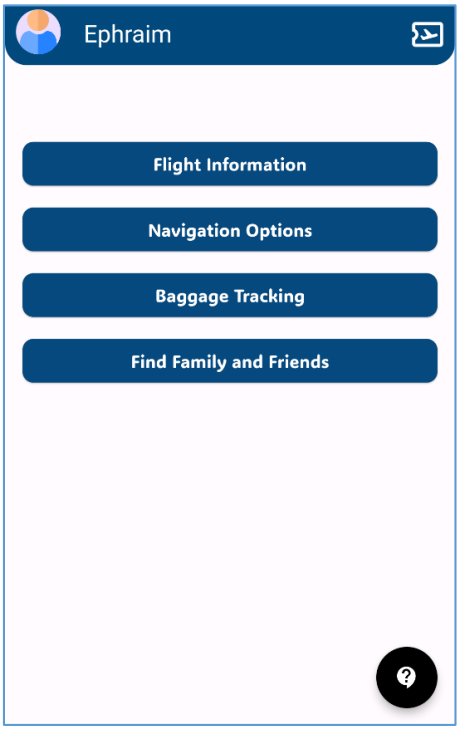
Registration Screen



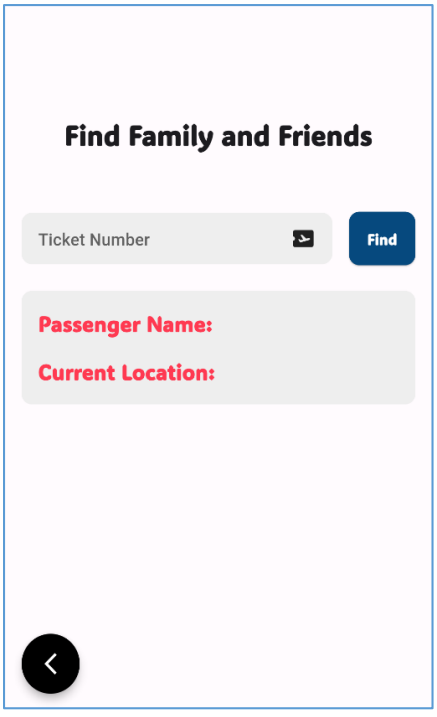


Verification Screen

Menu Screen, After Login into account the passenger has some options and services to use. The passenger can View Profile, get his flight information, navigate through airport, track his baggage, Find Family & Friends or Contact Support.

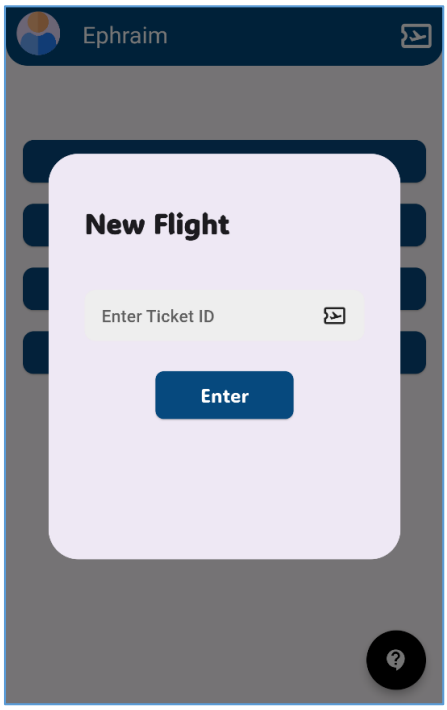


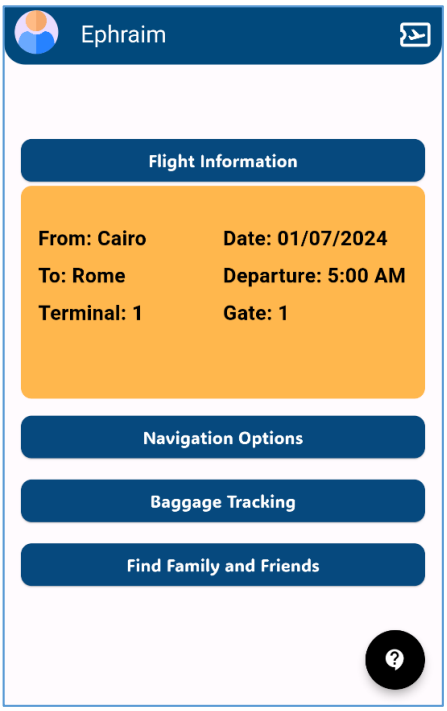
Find Family & Friends Screen, for both membered and guest user they can use Find Family & Friends feature by entering the ticket number of the person they want to find.



The passenger must enter his ticket number in the specified place to get information about the ticket and will be used later for tracking baggage.

New Flight Screen

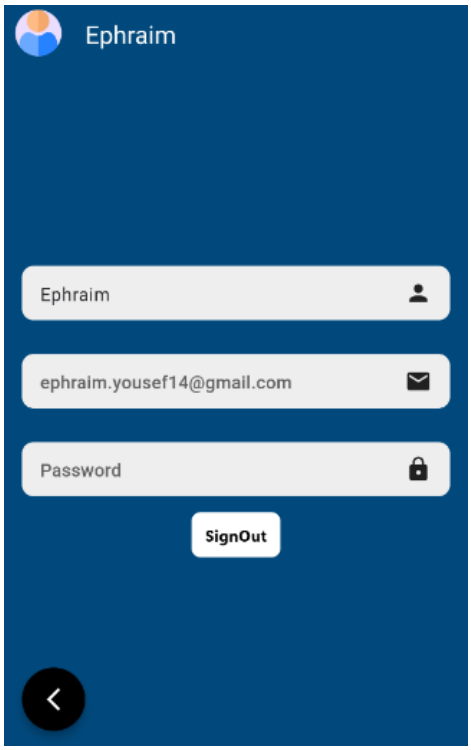




Flight Info Screen

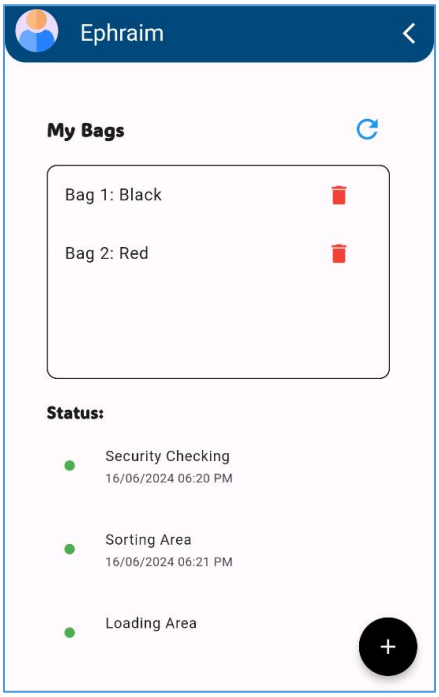
Profile Screen, Where passenger can sign-out from the account

.

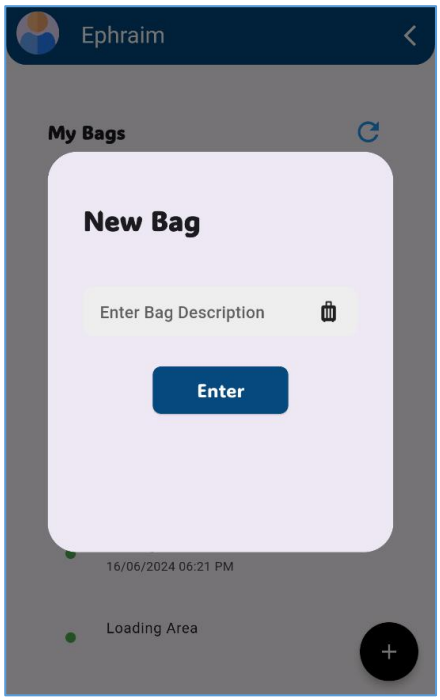


Passengers can add new baggage by pressing on the plus button then type a notable description for the added bag to be easily identified during tracking. The baggage can be deleted easily if it was added by mistake, there is message appear when the passenger tries to delete a bag to ensure that the deletion process is intended

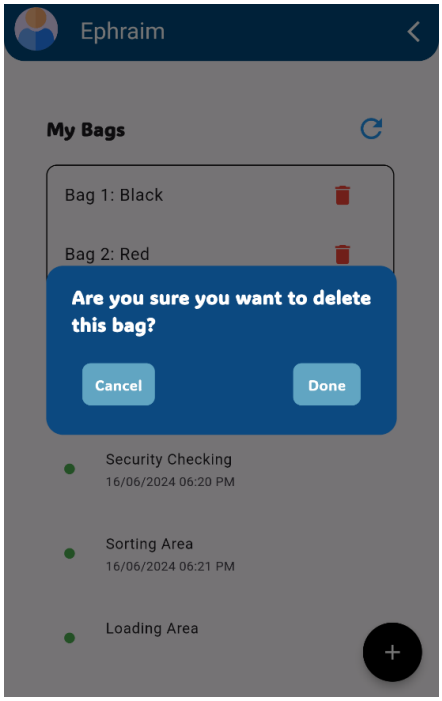
Baggage Tracking Screen



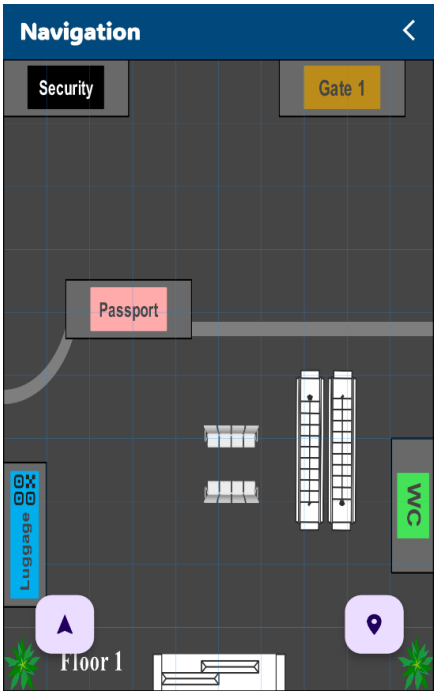
New Bag Screen

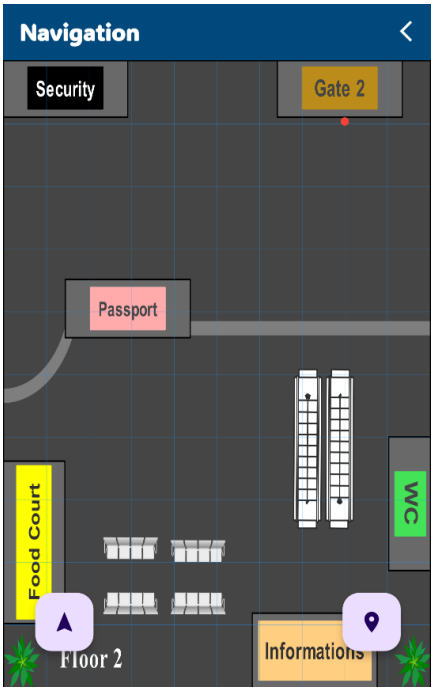


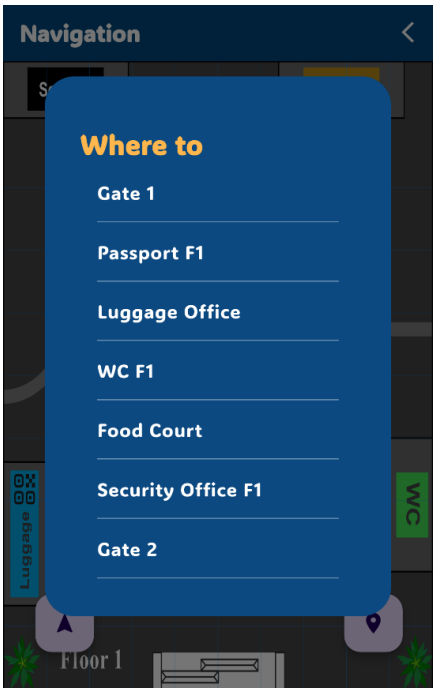
Deleting Message

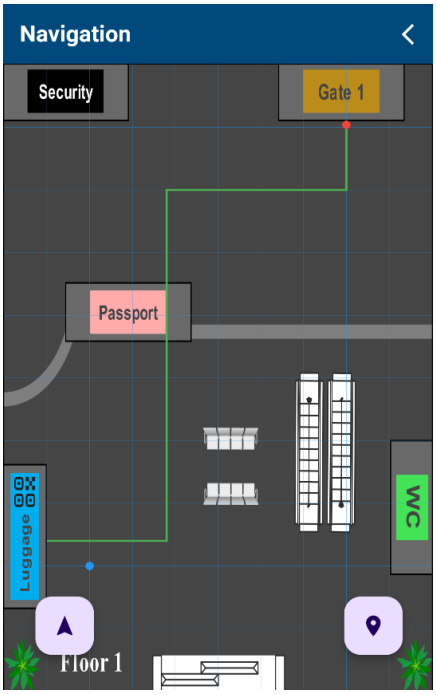


In **Navigation Screen** the passenger starts to select the location to be reached. Once the location selected mobile start calculate the passenger location giving a direct route according to the location area where the passenger existed and the targeted location.



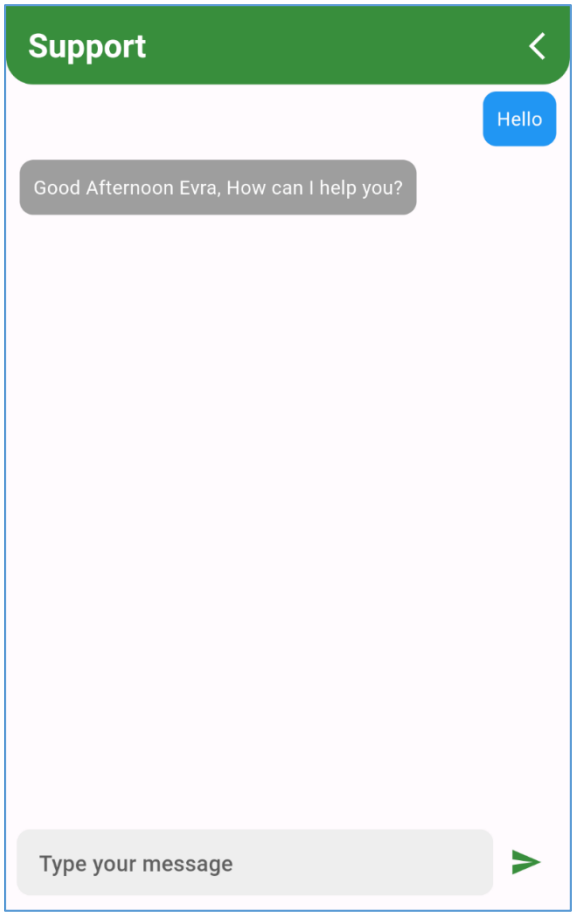




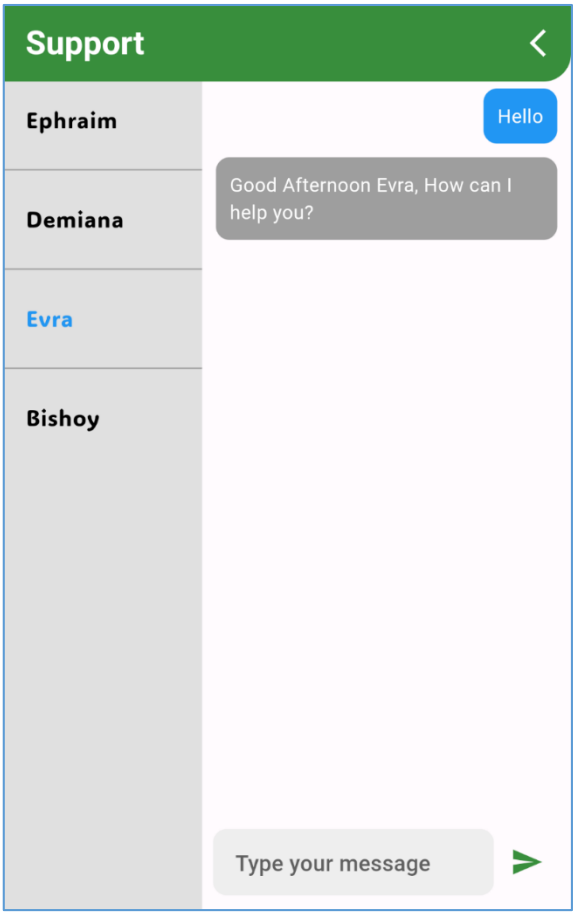


Customer support feature provides a real time response between the passenger and the support team.

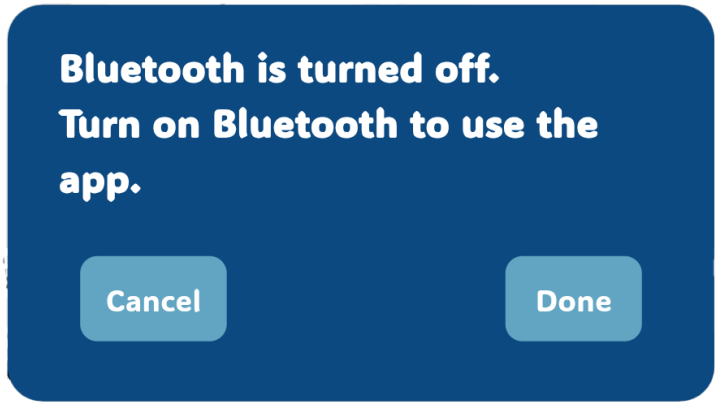
Client Side



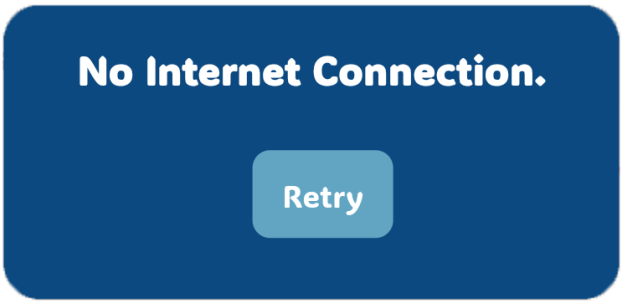
Agent Side



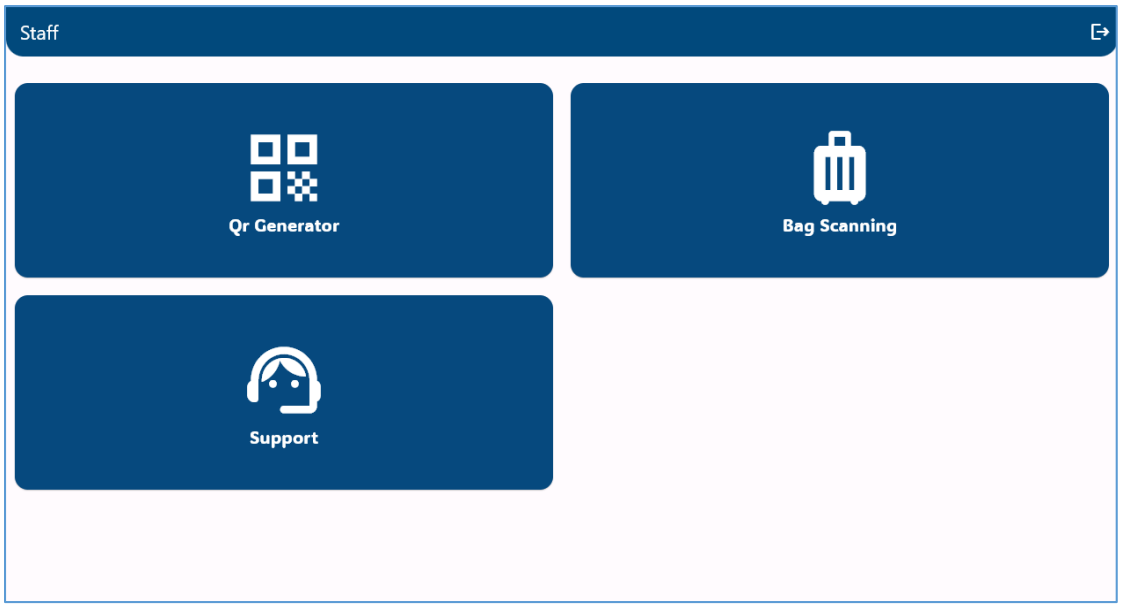
Bluetooth Check Message As our system is based on Bluetooth technology (BLE) the passenger must enable Bluetooth services to be able to use the application and take advantage of the application's features. There is checker method that works during the entire runtime to keep checking for Bluetooth whether is enabled or not.



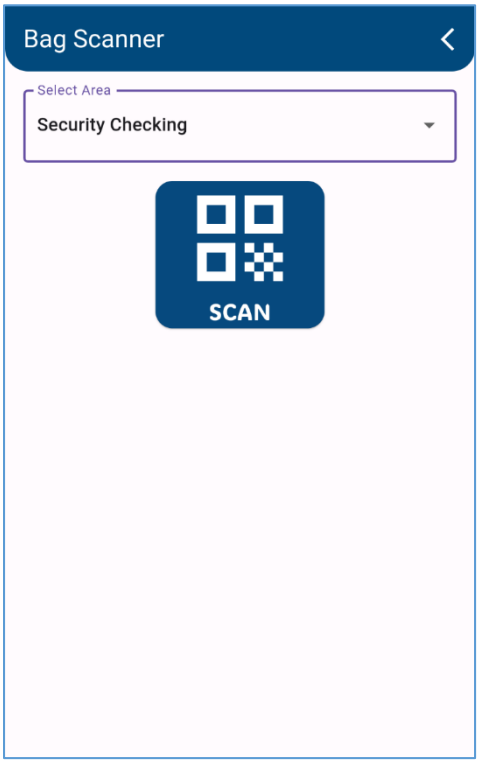
Internet Check Message, mainly our system depends on the internet connection for transferring data and many other processes so to ensure that the devices is having an internet connection. We made a method to continuously checks for connection and if there is connection this message is shown until the connection is restored.



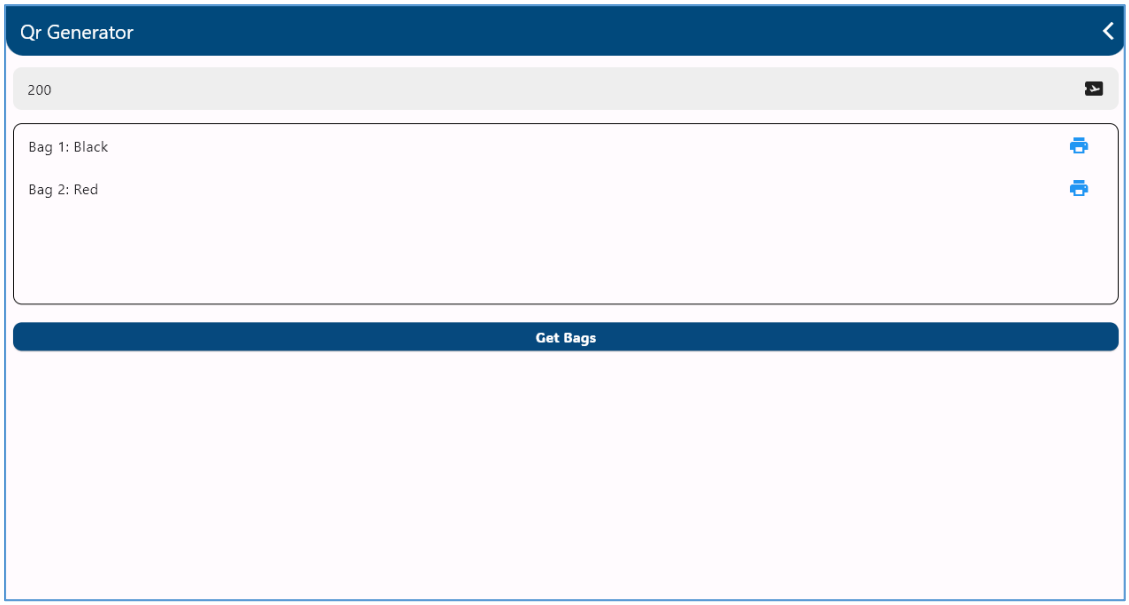
Staff menu where the staff based on his job select between these options.



Baggage Scanning Screen, For the staff responsible for scanning baggage. He must select the location of the checkpoint where he exists then press scan button to start scan the coming bags. Once the bag is scanned a notification is sent to the passenger owns the bag telling him the location and time where his bag was scanned.

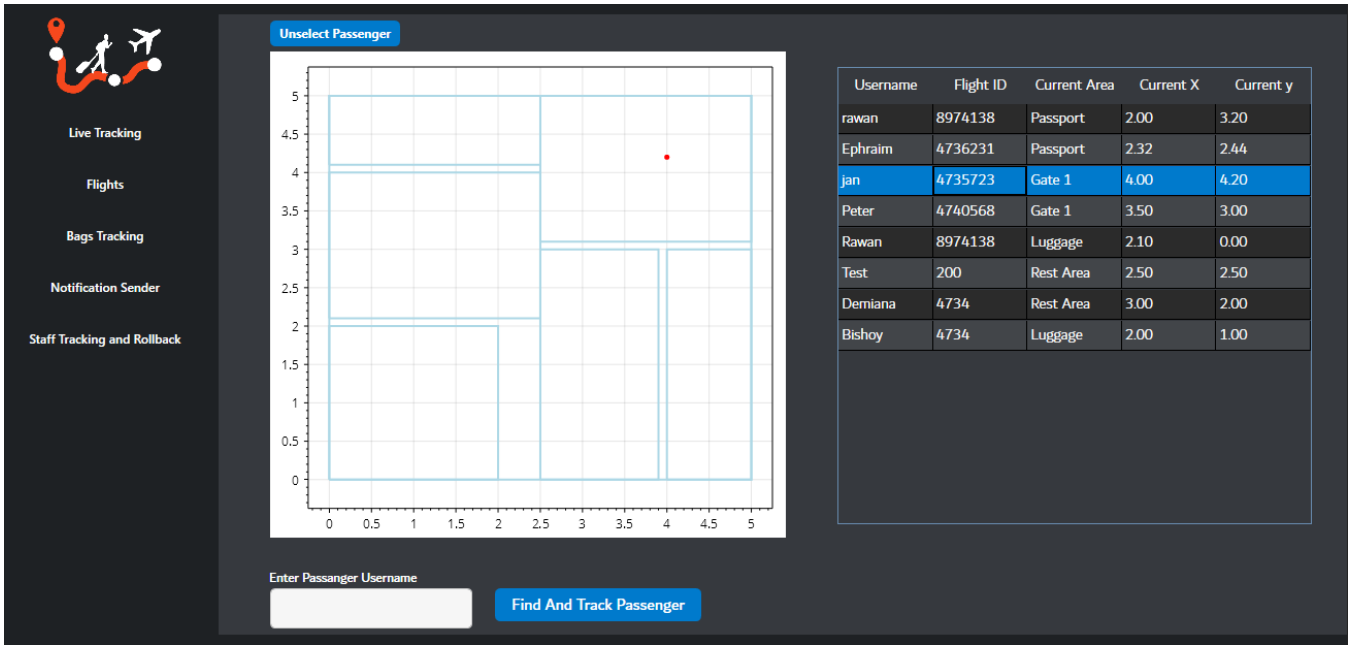


QR Code Generation Screen, For the staff responsible for generating QR codes for baggage. Simply enter the ticket number and then he gets all bags added by ticket owner. Then he can print QR code for each bag.

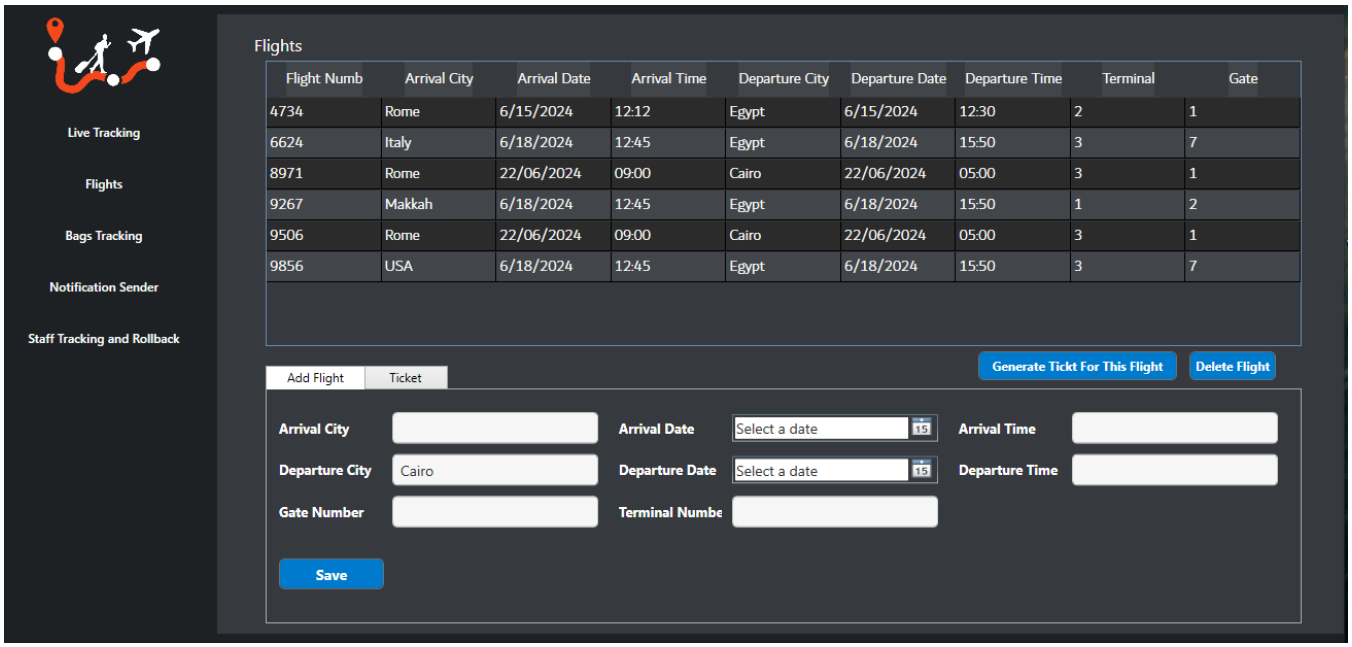


Control Room User interface

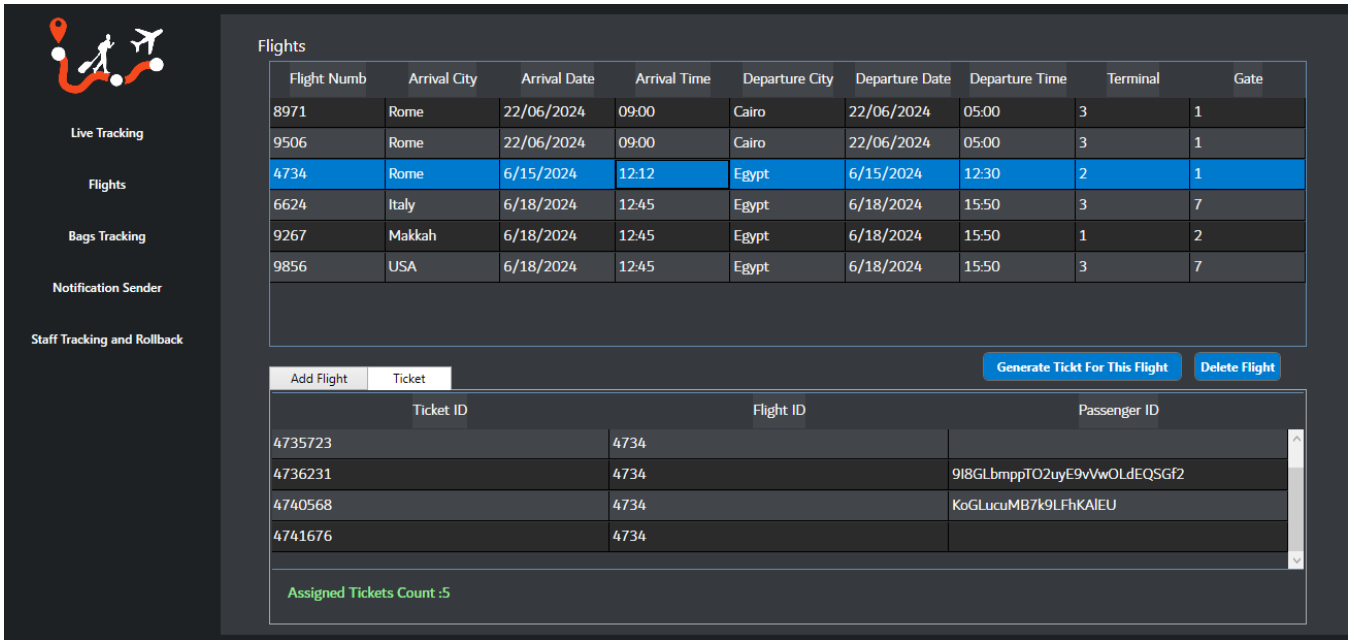
Live Tracking Screen Real-time monitoring of airport passengers with the ability to select specific individuals to display their data in a DataGrid and show their current location on a map.



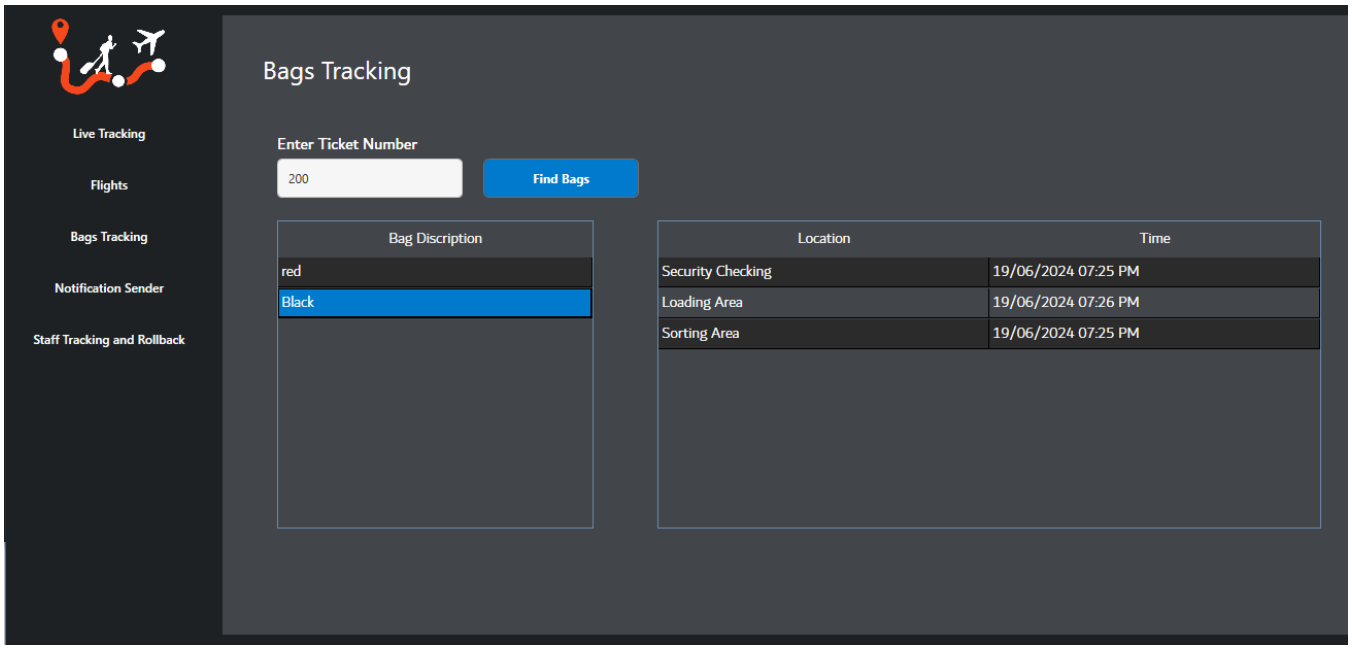
Flights Screen Displays comprehensive flight information, allows addition and deletion of flights, facilitates ticket generation for specific flights, and shows all tickets associated with each flight, including passenger assignment.



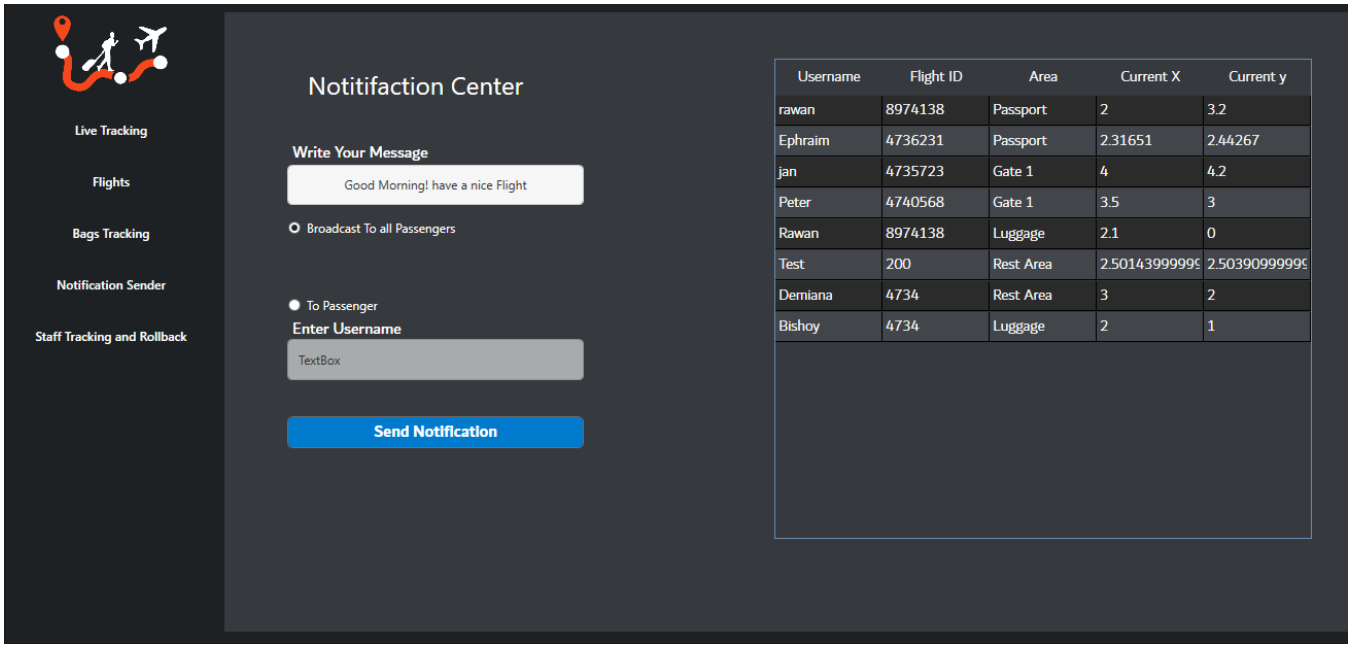
“Creating Flight”



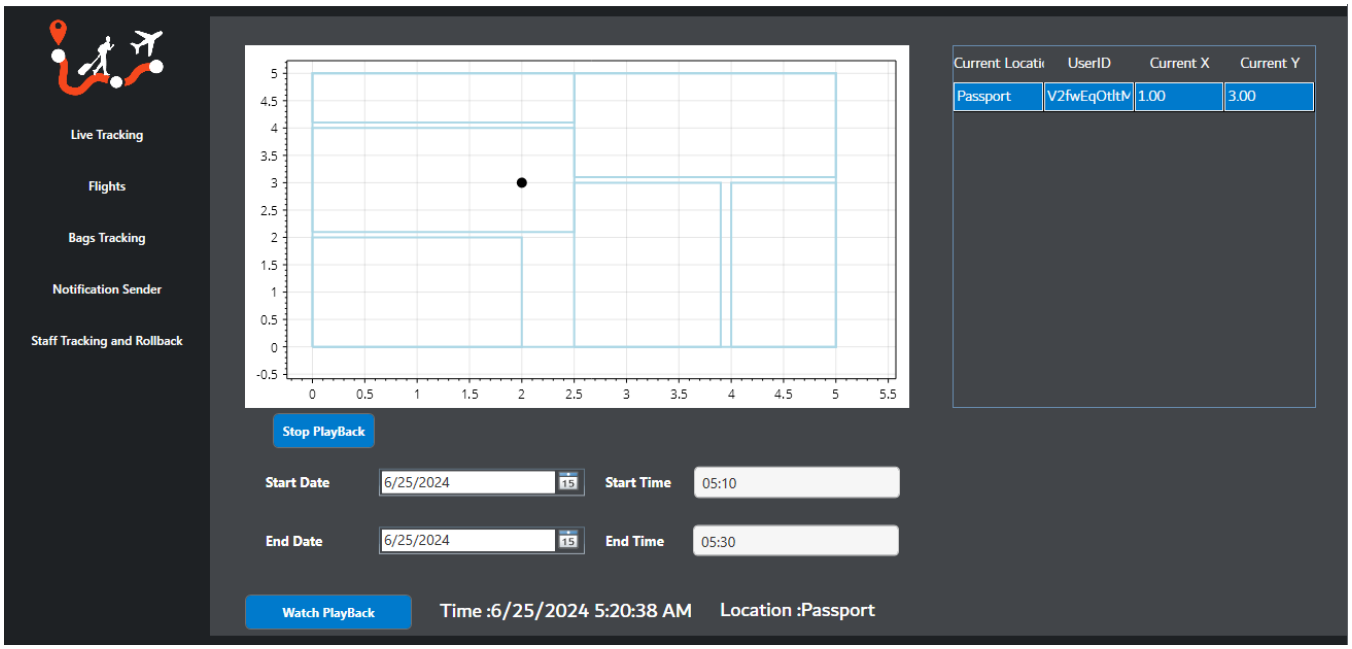
Bags Tracking Screen Tracks each passenger's bags, logs their movement through checkpoints, records last scanned locations by staff, facilitates lost bag retrieval with individual bag descriptions per passenger (multiple bags per passenger), and allows bag retrieval via passenger ticket identification.



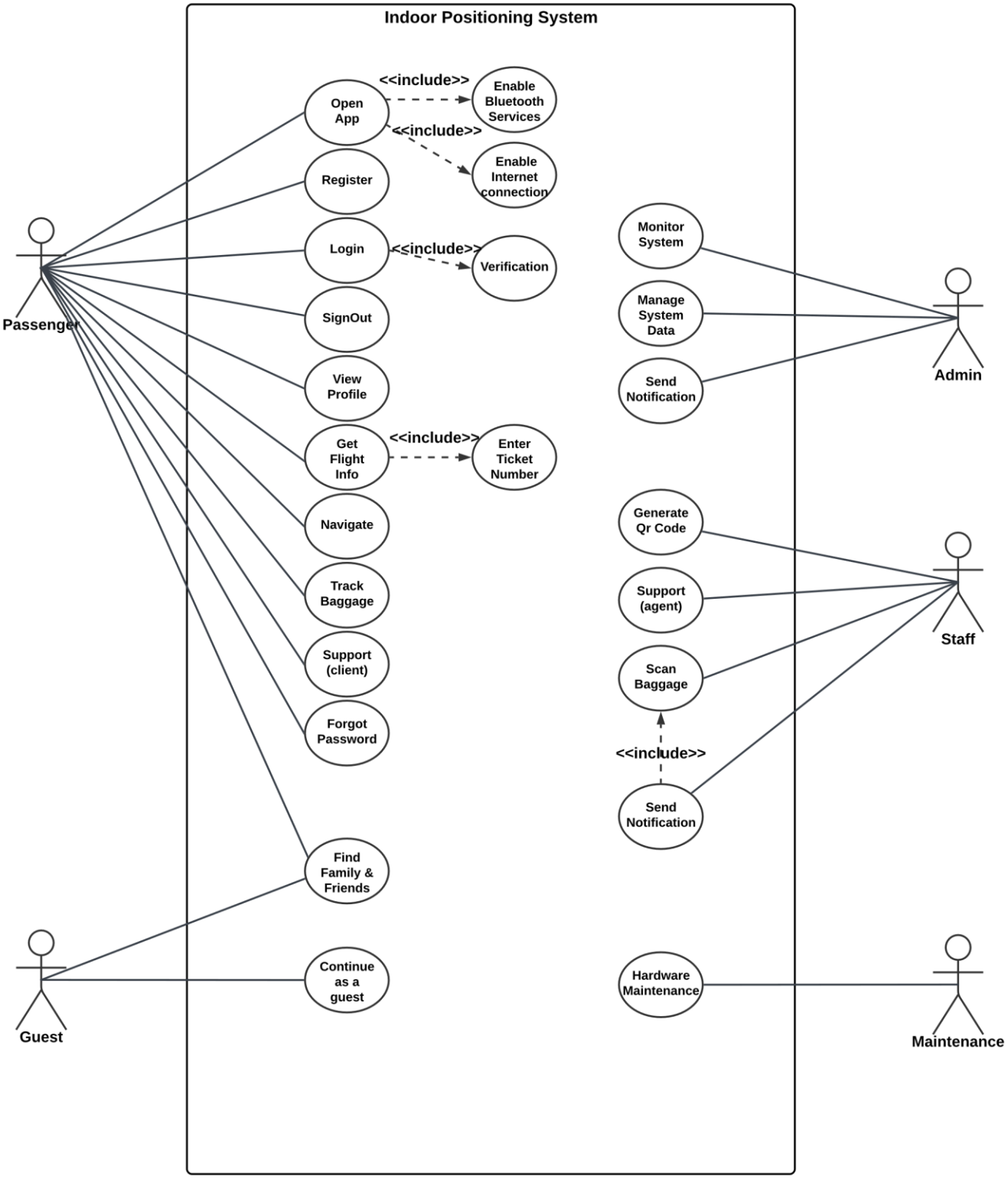
Notification Center Screen Enables sending notifications to broadcast to all passengers, specific individuals selected from a DataGrid, or passengers in specific areas, facilitating emergency alerts and general passenger communications.



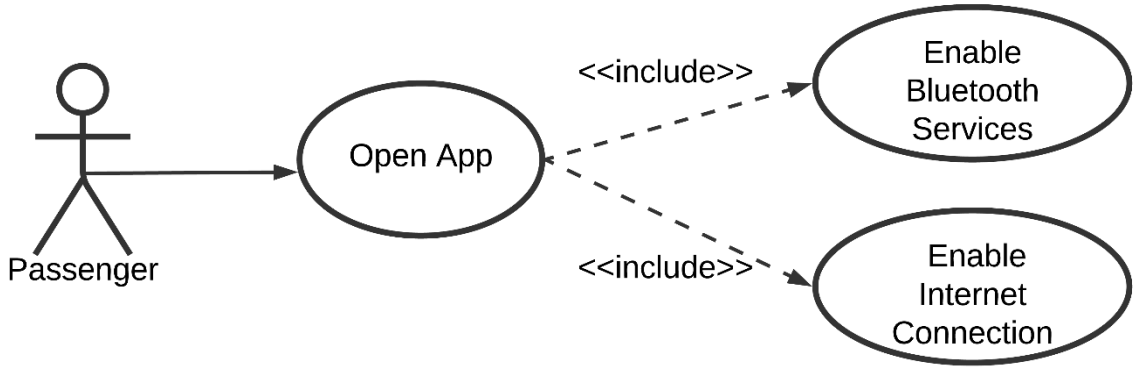
Staff Tracking Screen Real-time tracking of staff locations with a day history playback, displaying each staff member's movements throughout the day with timestamps and corresponding locations.



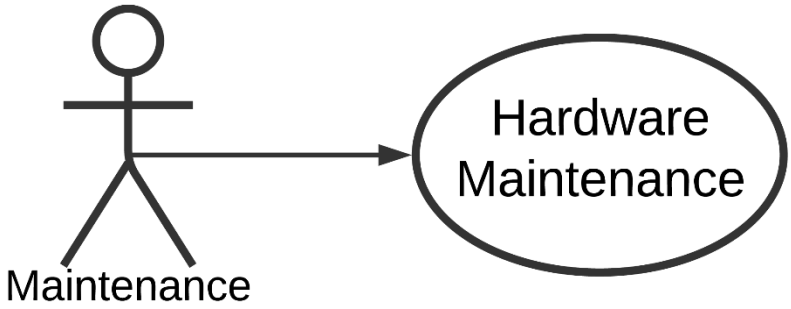
6.3. Use-Case


Indoor Positioning System
Open App
<<include>>
Enable Bluetooth Services
<<include>>
Enable Internet connection
Register
Login
<<include>>
Verification
SignOut
View Profile
Get Flight Info
<<include>>
Enter Ticket Number
Navigate
Track Baggage
Support (client)
Forgot Password
Find Family & Friends
Continue as a guest
Passenger
Guest
Monitor System
Manage System Data
Send Notification
Admin
Generate Qr Code
Support (agent)
Scan Baggage
<<include>>
Send Notification
Staff
Hardware Maintenance
Maintenance

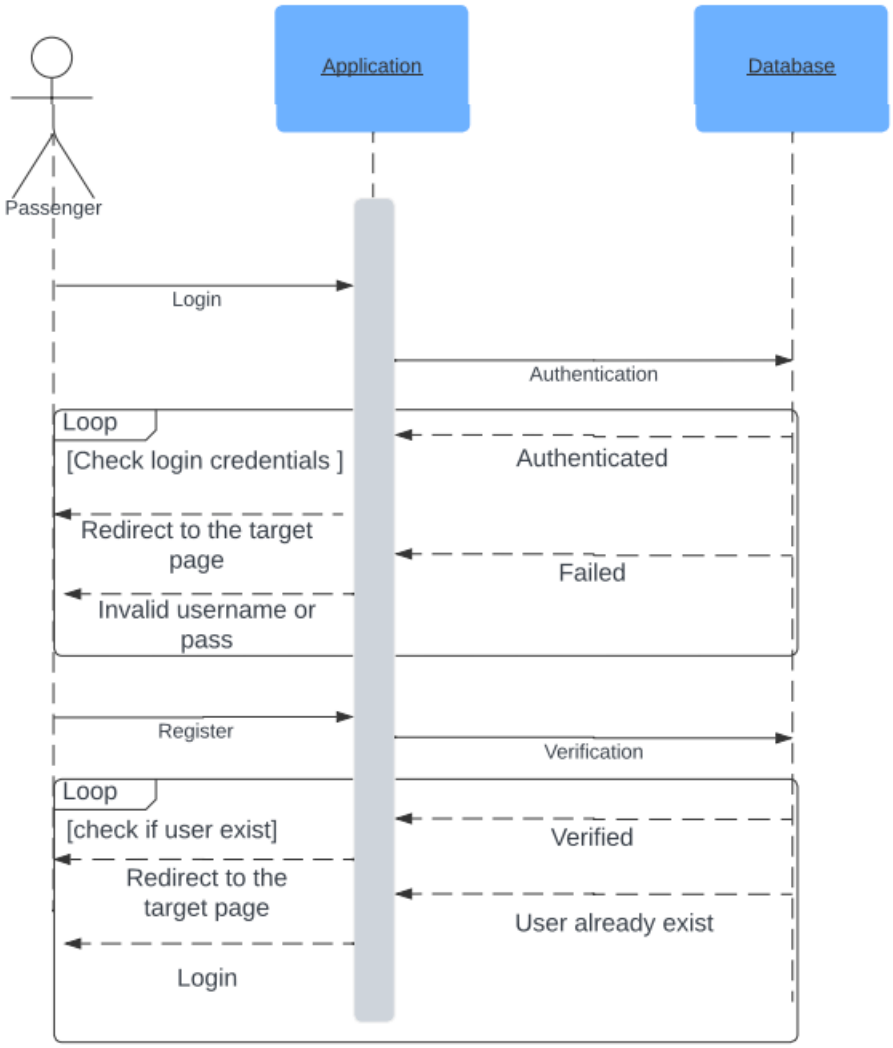
Passenger open the app

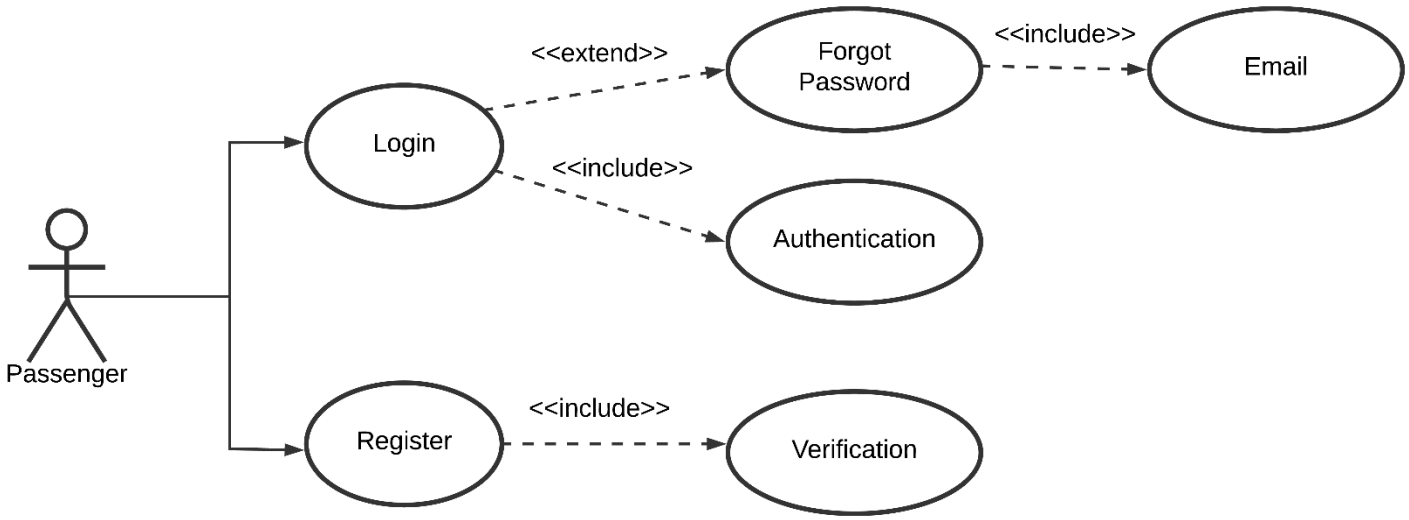


Maintenance

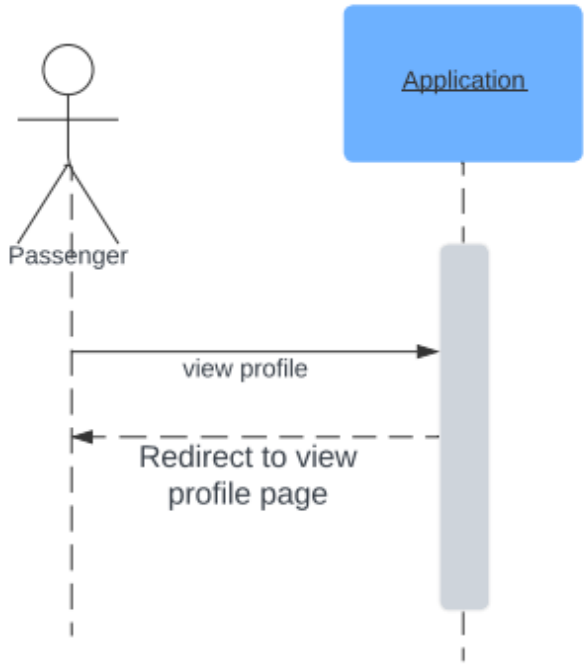


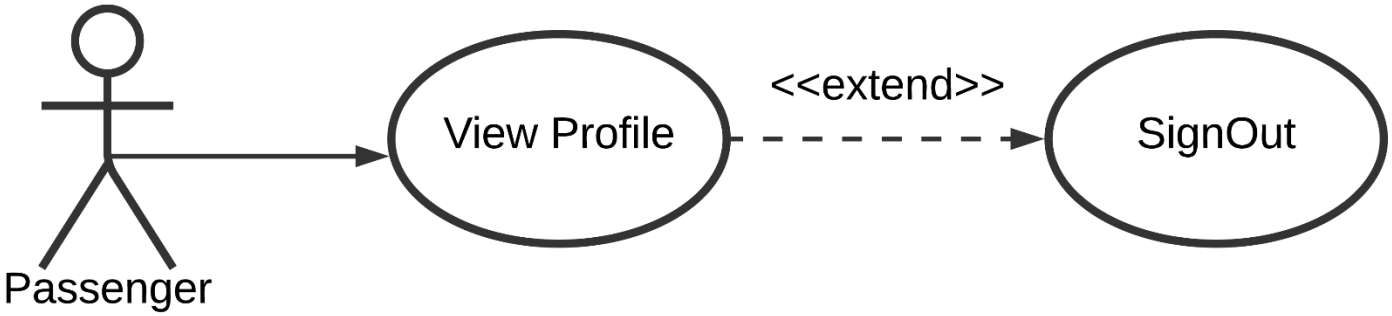
Passenger login and register



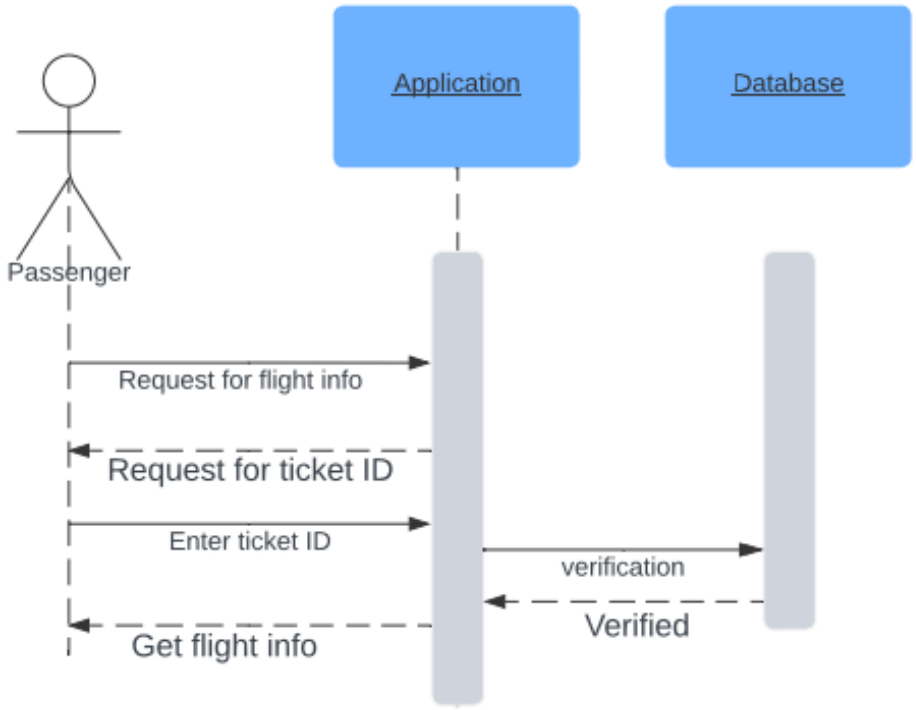


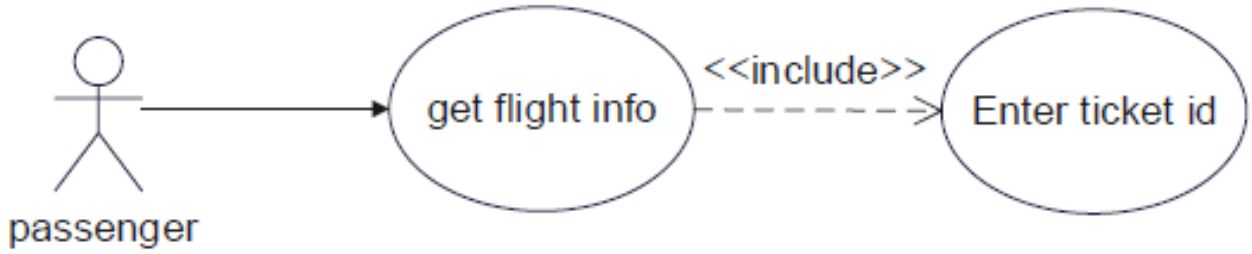
Passenger view profile



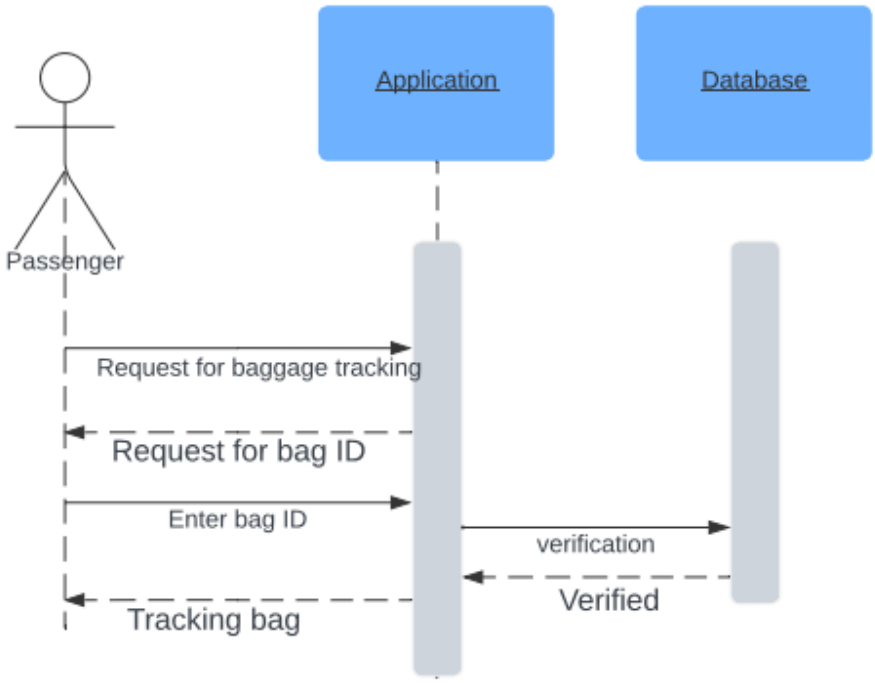


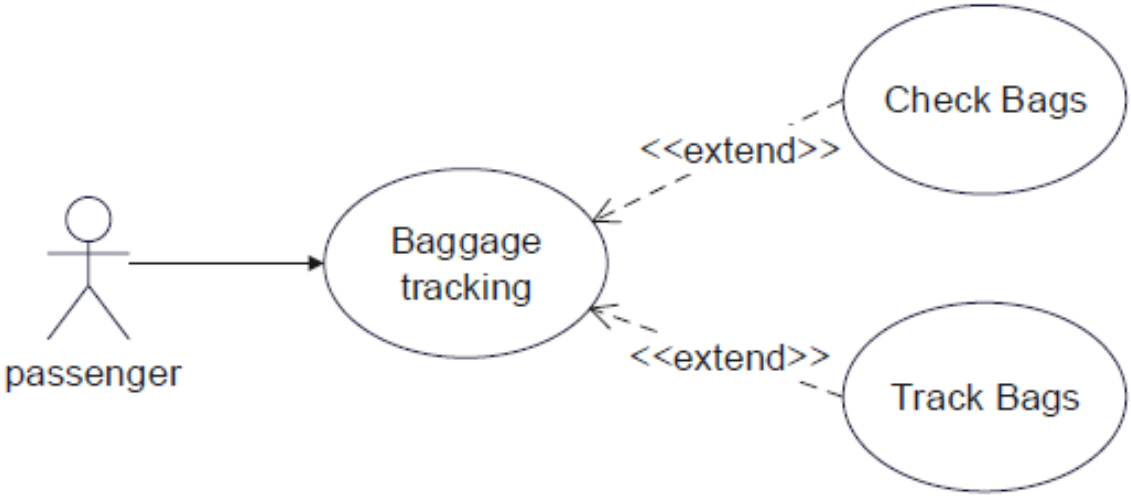
Passenger get flight info



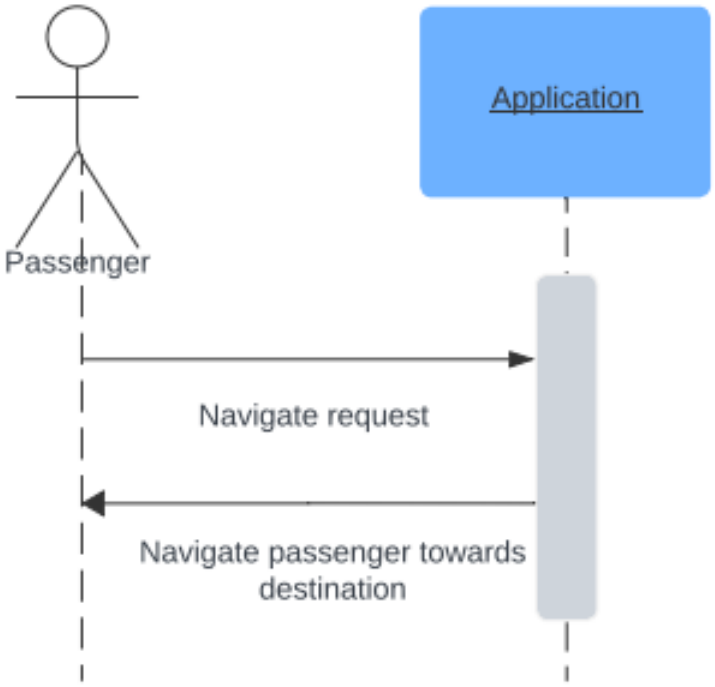


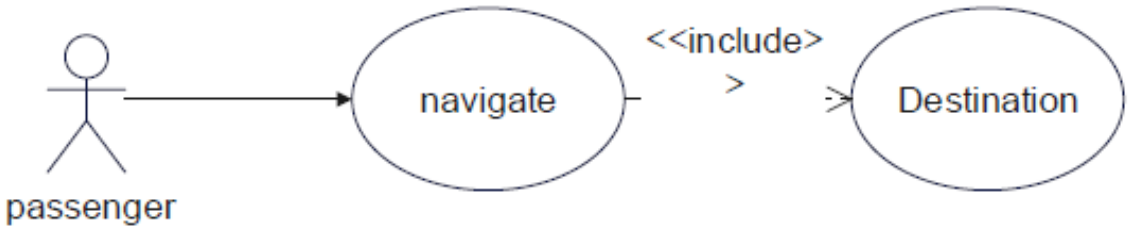
Passenger track baggage



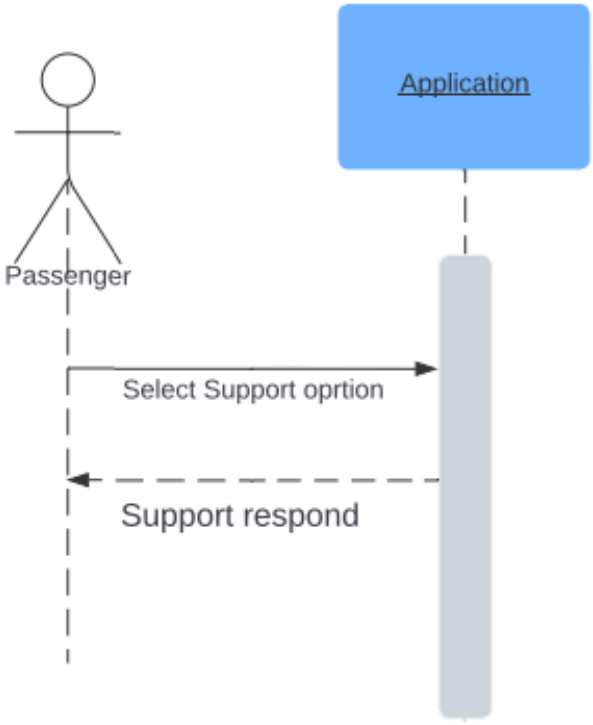


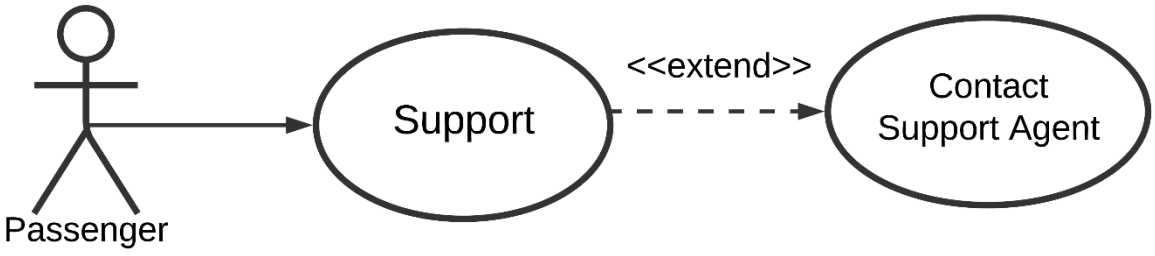
Passenger Navigation



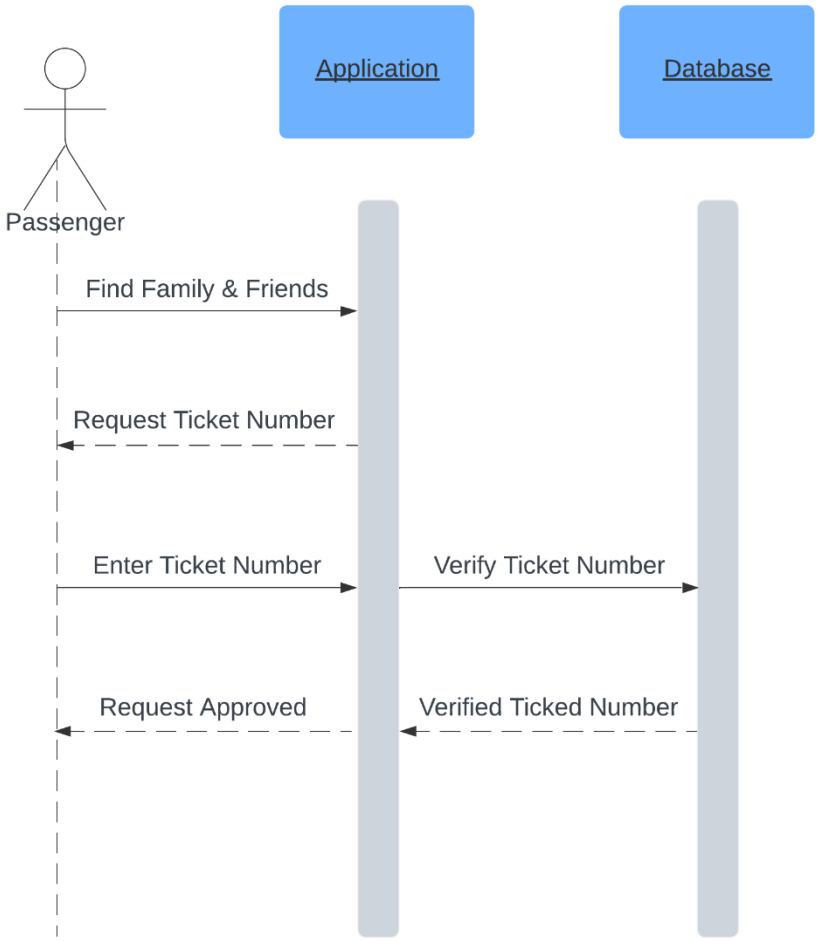


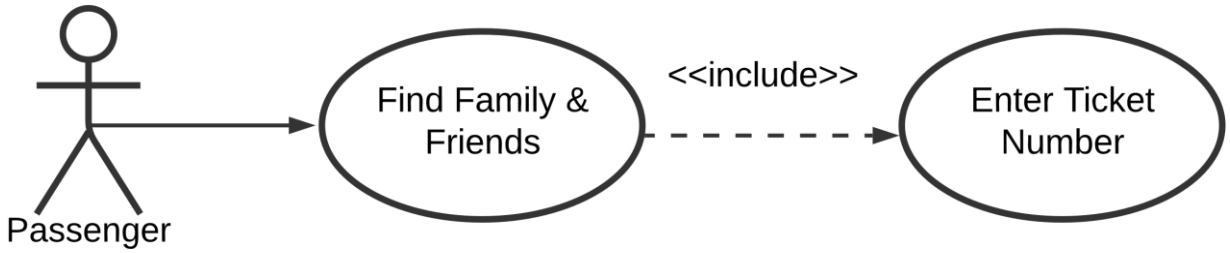
Passenger support



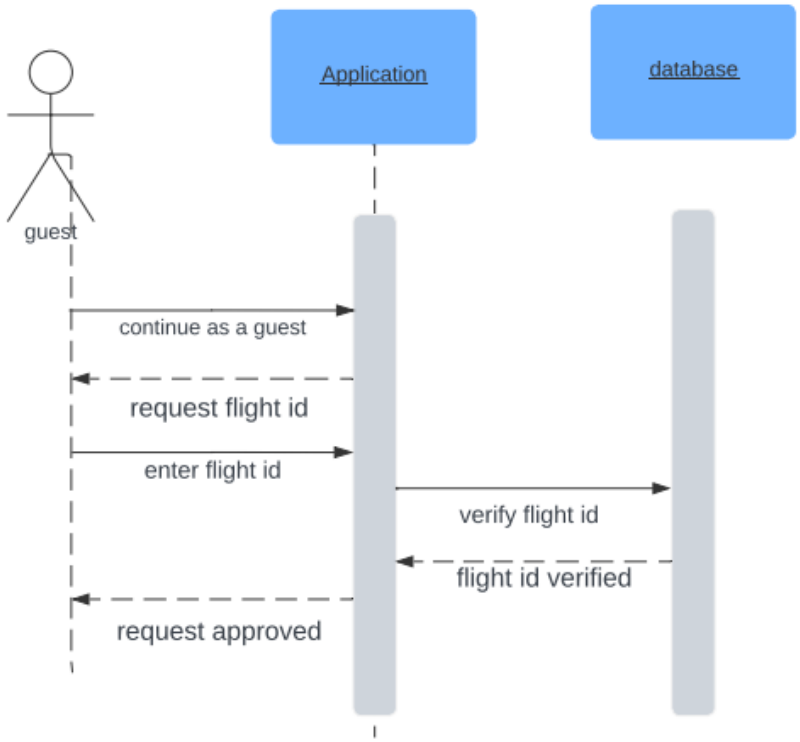


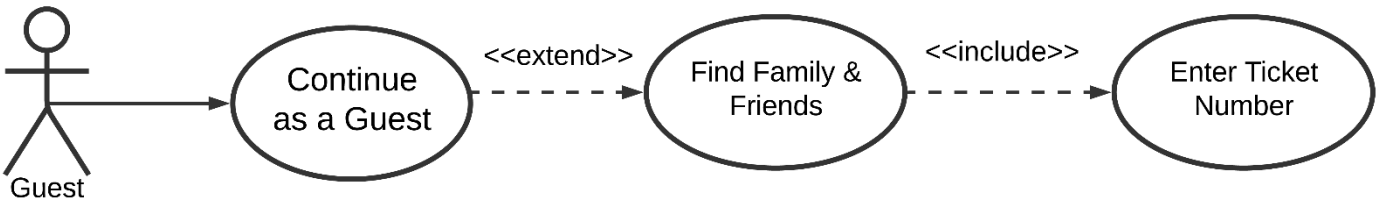
Find Family & Friends



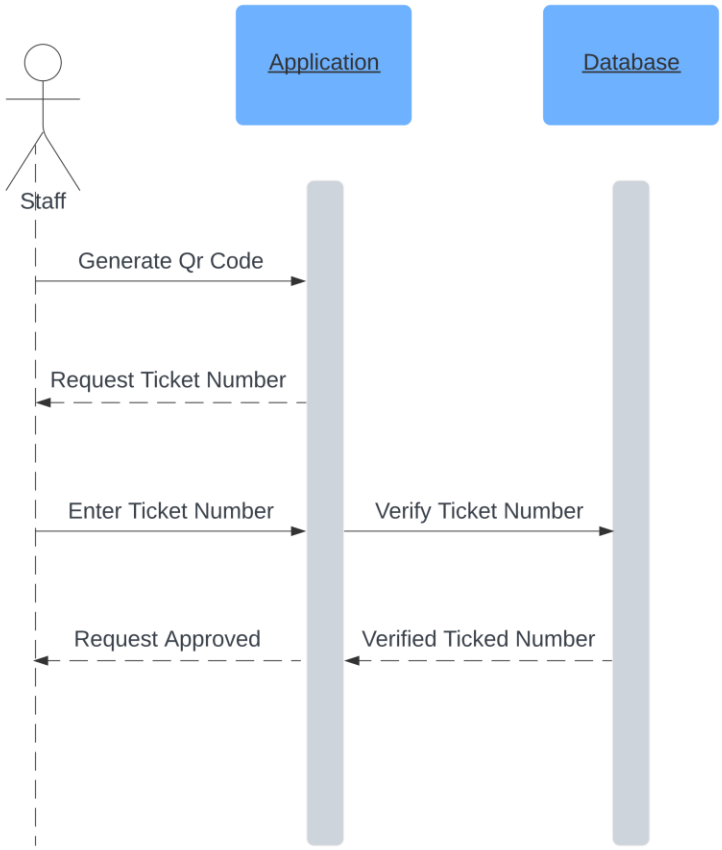


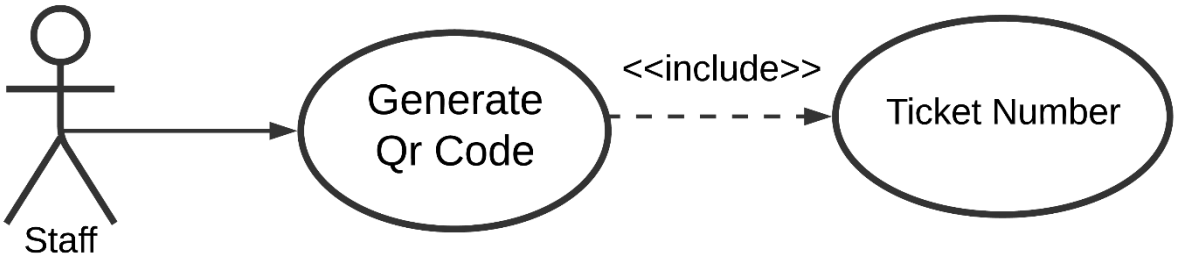
Continue as a guest



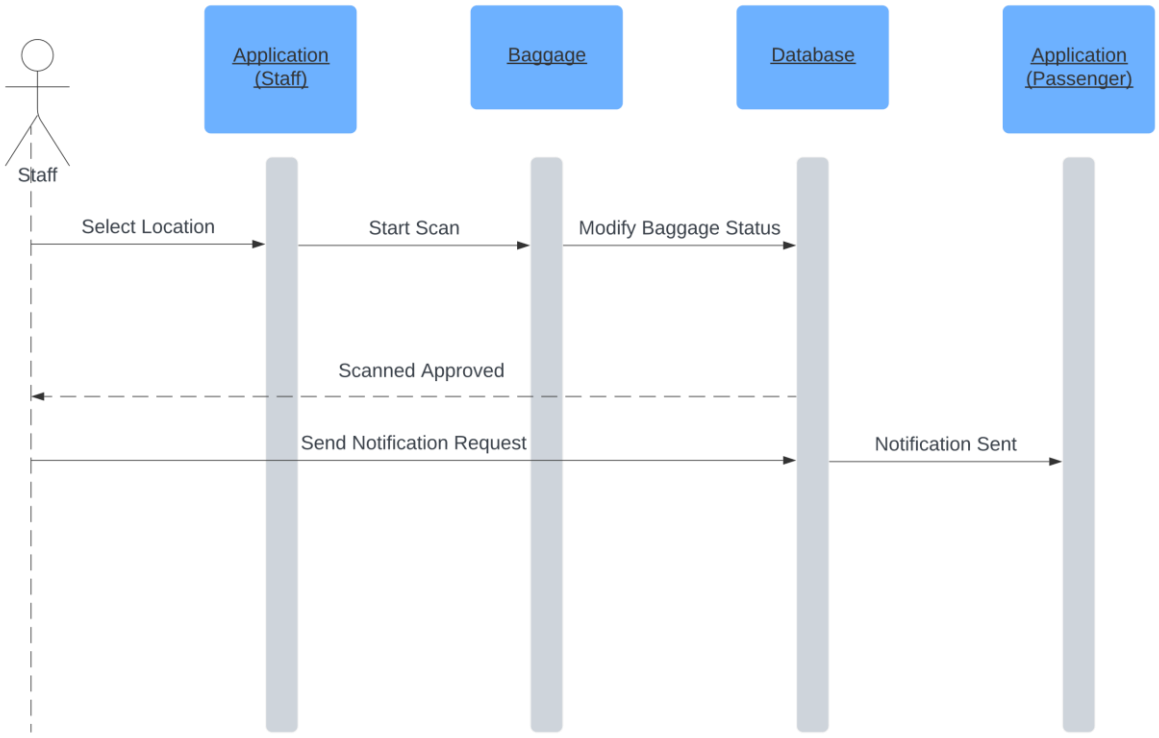


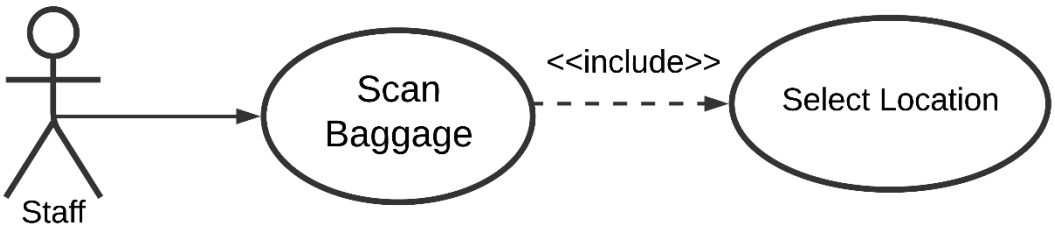
Staff Qr code Generator

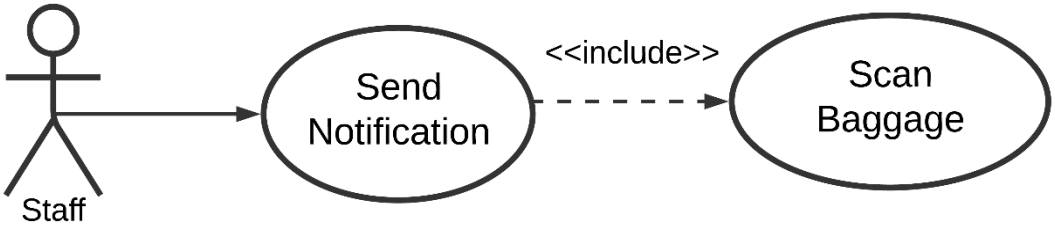




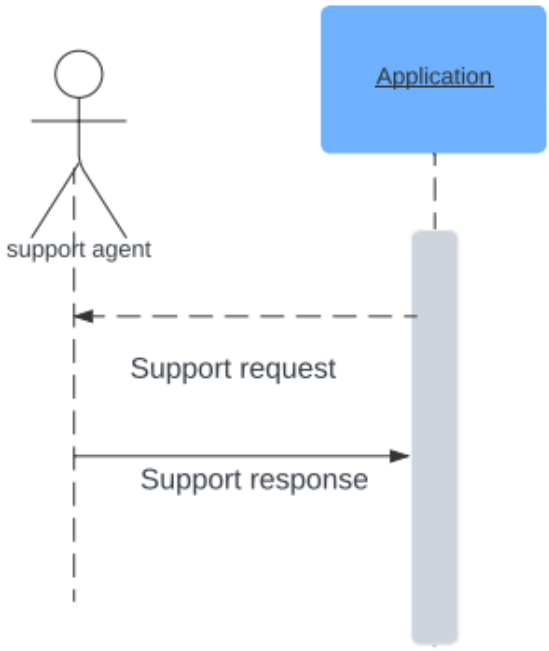
Staff Scan Baggage

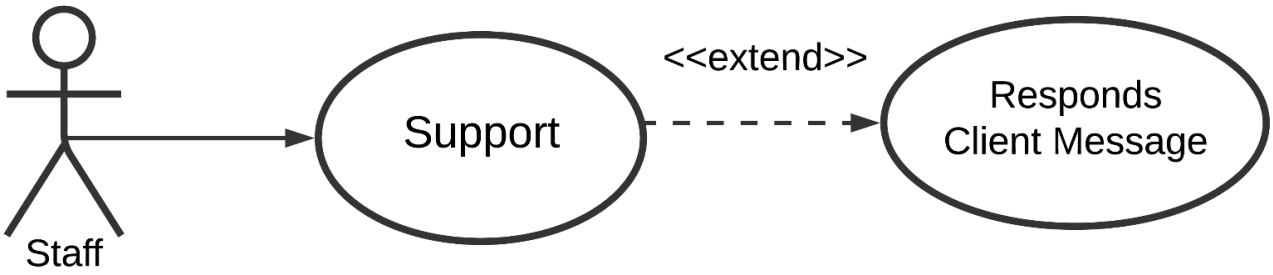




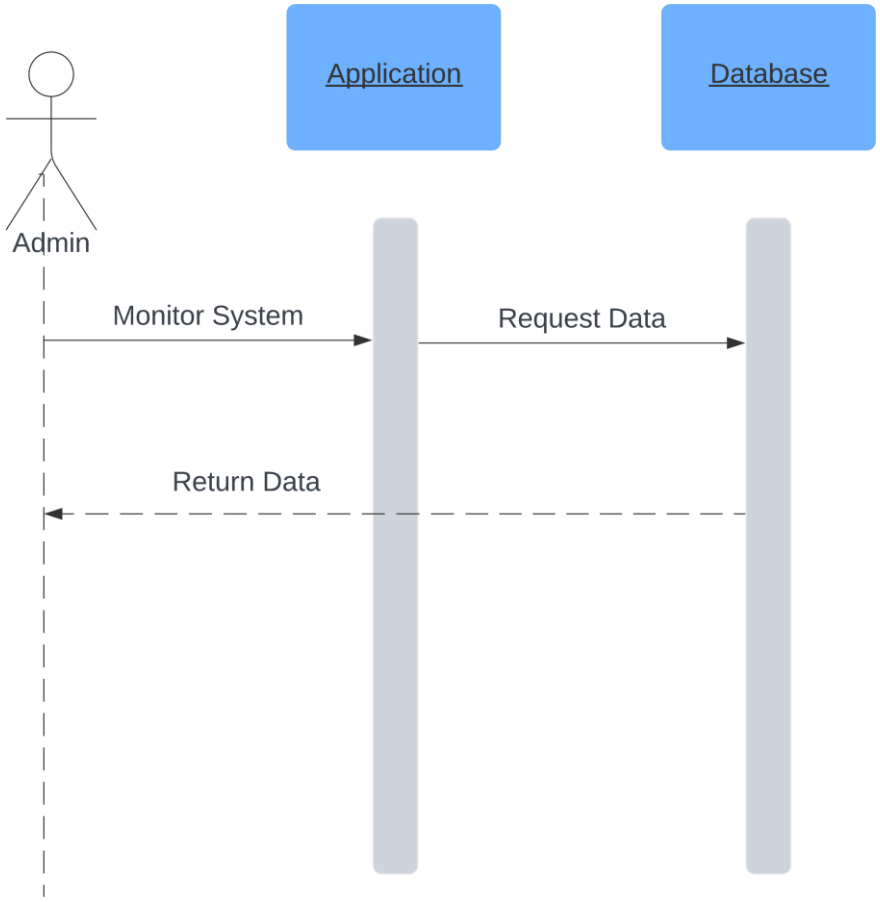


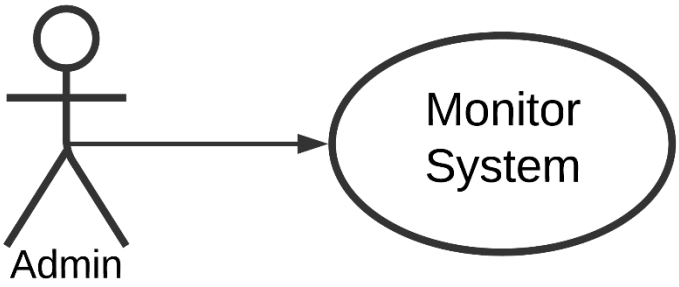
Staff Support



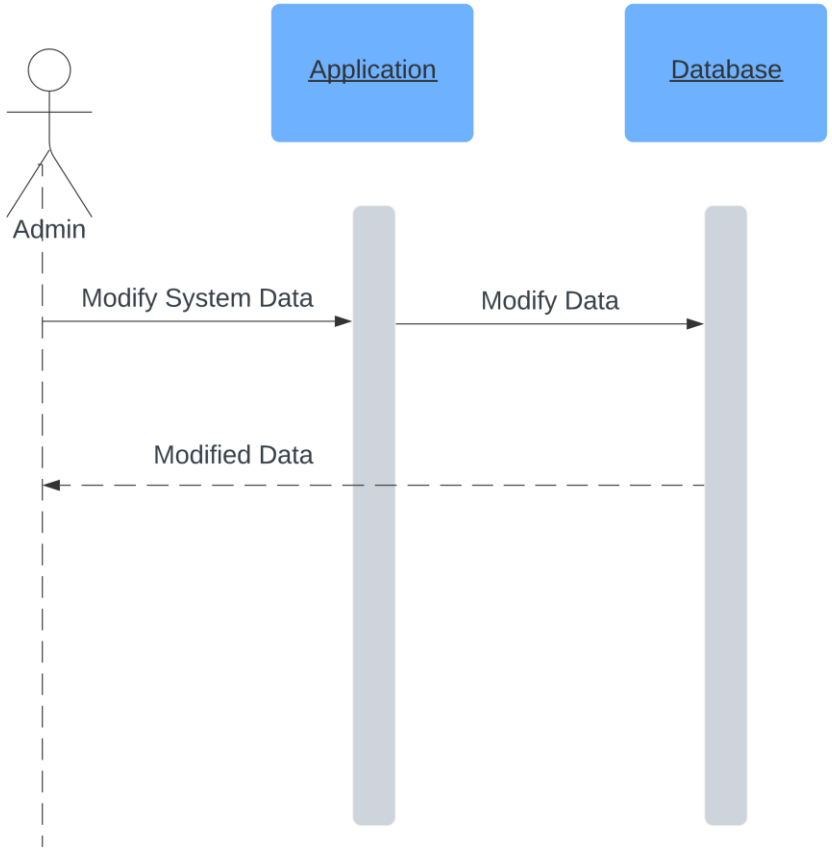


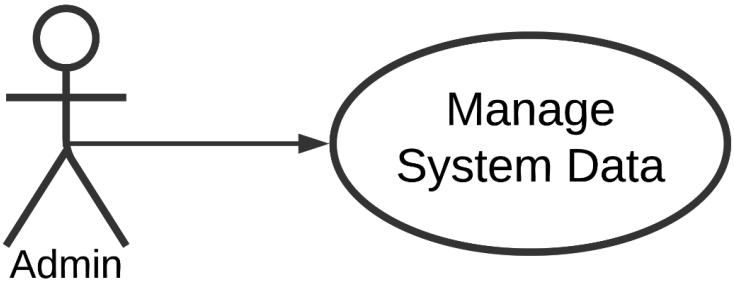
Admin Monitor System



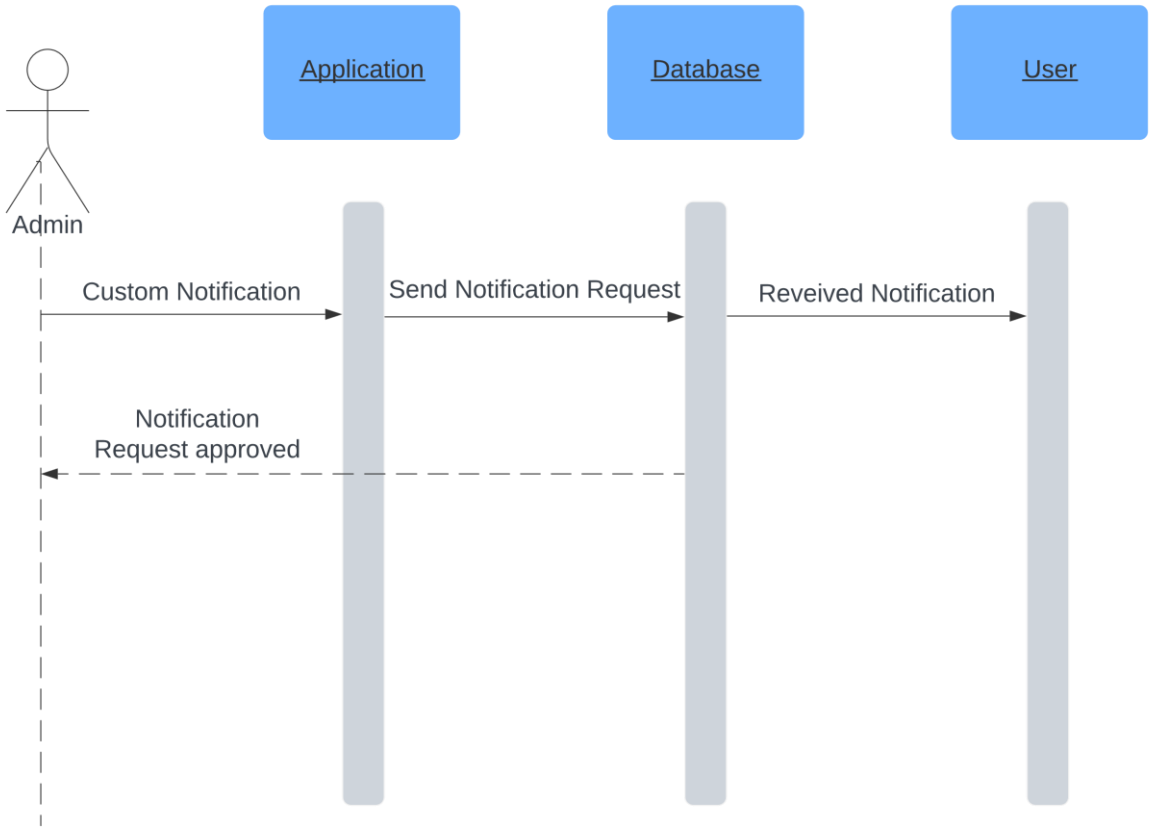


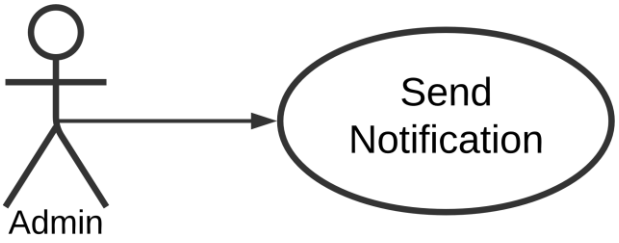
Admin Modify System Data





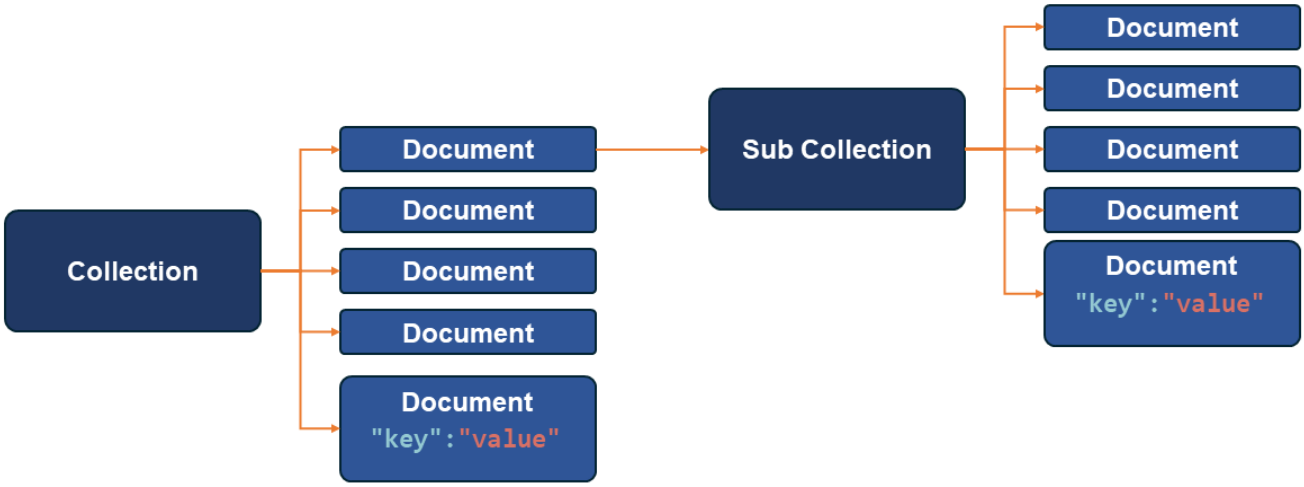
Admin Send Notifications

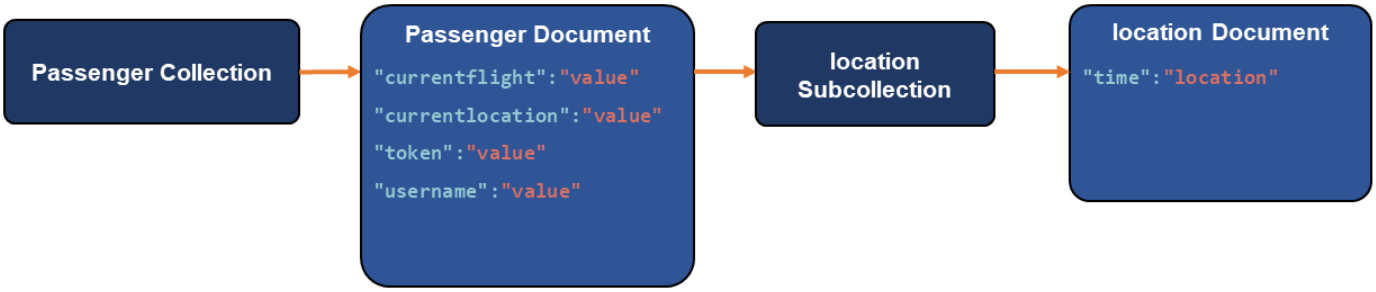


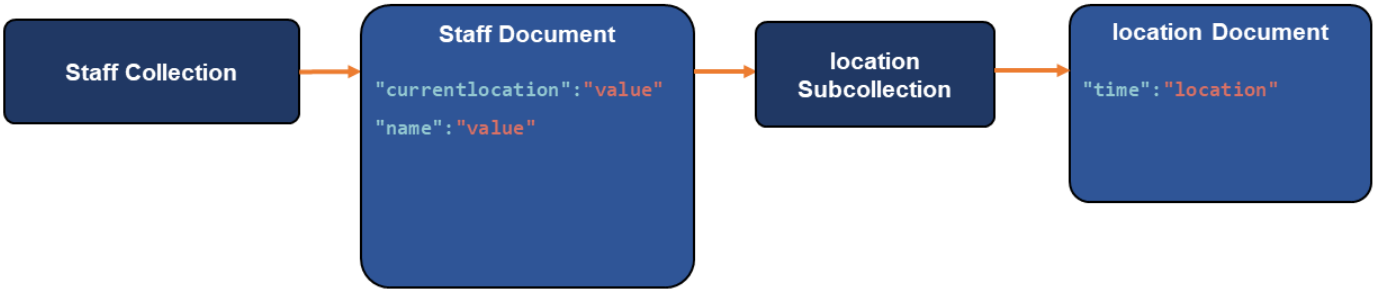


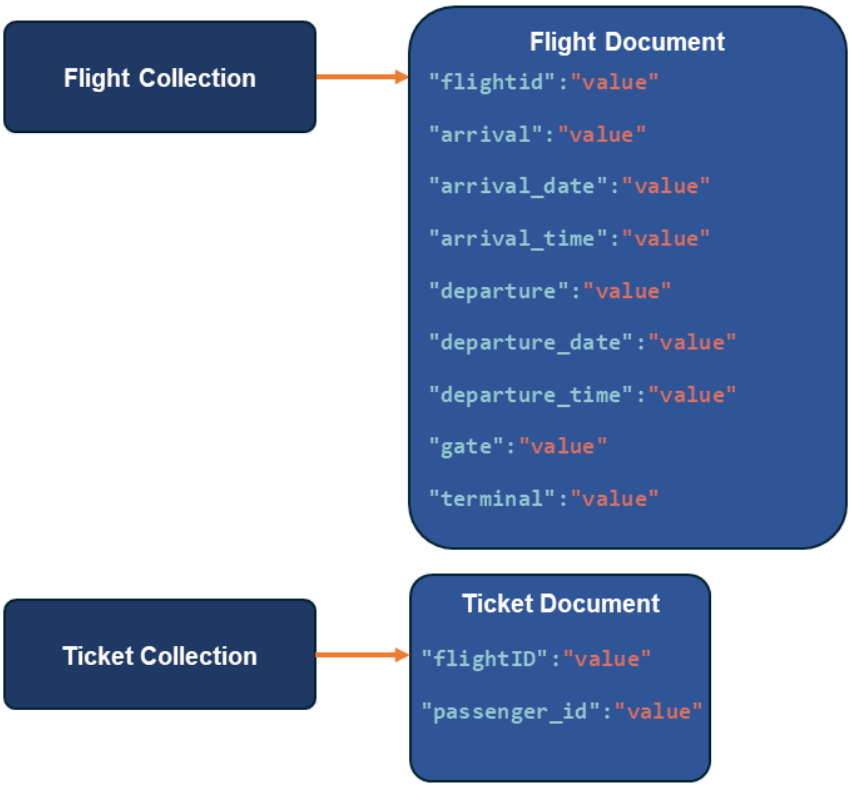
6.4. Database Schema

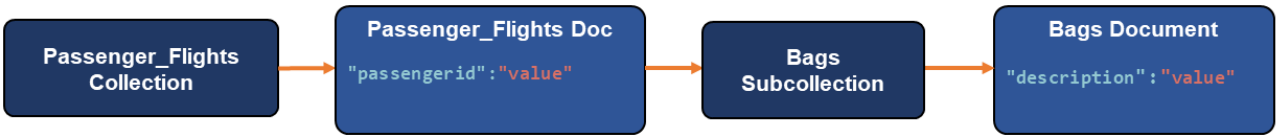
Firebase Firestore Database Strucutre



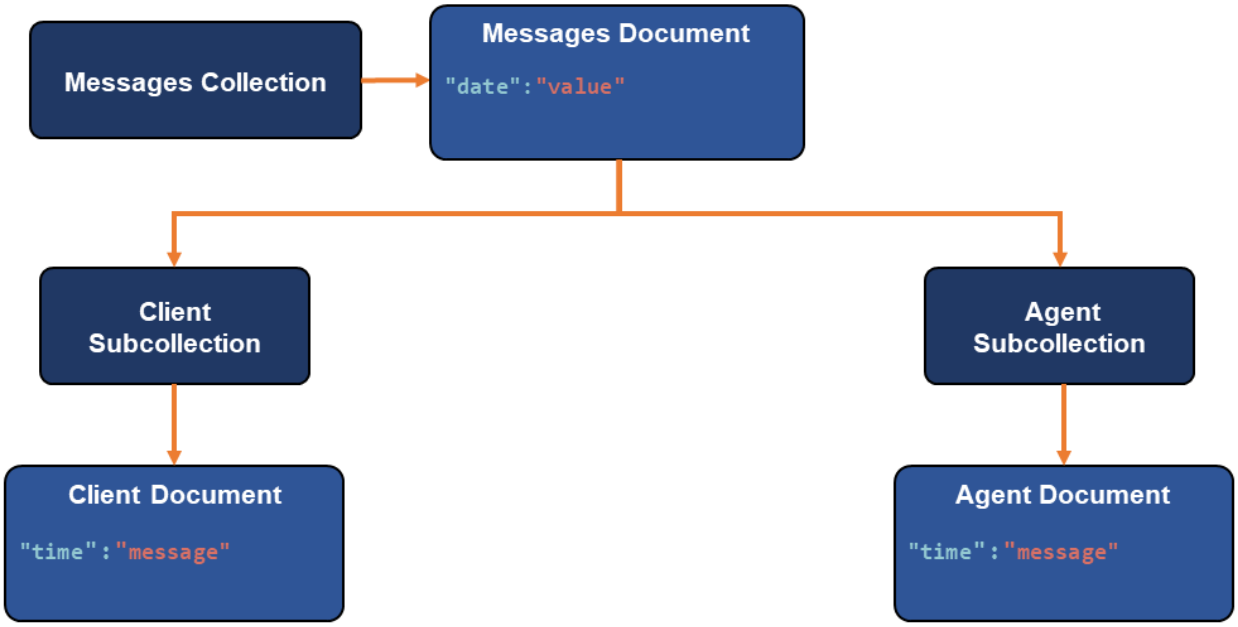





Flight Collection
Flight Document
"flightid":"value"
"arrival":"value"
"arrival_date":"value"
"arrival_time":"value"
"departure":"value"
"departure_date":"value"
"departure_time":"value"
"gate":"value"
"terminal":"value"
Ticket Collection
Ticket Document
"flightID":"value"
"passenger_id":"value"


Passenger_Flights Collection
Passenger_Flights Doc
"passengerid":"value"
Bags Subcollection
Bags Document
"description":"value"


Bags Document
"description":"value"
Status Subcollection
Status Document
"time":"location"


Messages Collection
Messages Document
"date":"value"
Client Subcollection
Agent Subcollection
Client Document
"time":"message"
Agent Document
"time":"message"

Chapter 7

Implementation

7.1. Passenger Implementation

Main (Startup)

In the main file we established configuration setup for the connection with firebase database, Defining all routes for navigation inside the application.

```
import 'dart:io';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:firebase_messaging/firebase_messaging.dart';
import 'package:flutter/material.dart';
import 'package:Travello/screens/Splash.dart';
import 'package:Travello/screens/Welcome1.dart';
import 'package:Travello/screens/Welcome2.dart';
import 'package:Travello/screens/Welcome3.dart';
import 'package:Travello/screens/BluetoothScan.dart';
import 'package:Travello/screens/Register.dart';
import 'package:Travello/screens/Login.dart';
import 'package:Travello/screens/Verification.dart';
import 'package:Travello/screens/Menu.dart';
import 'package:Travello/screens/agentchat.dart';
import 'package:Travello/screens/baggage.dart';
import 'package:Travello/screens/bagqrgen.dart';
import 'package:Travello/screens/bluetoothcheck.dart';
import 'package:firebase_core/firebase_core.dart';
import 'package:Travello/screens/securitytracking.dart';
import 'package:Travello/screens/userchat.dart';
import 'package:Travello/screens/findff.dart';
import 'package:Travello/screens/profile.dart';
import 'package:Travello/screens/staffmenu.dart';
import 'package:Travello/screens/staffscanner.dart';
import 'package:Travello/screens/internetcheck.dart';

void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  if (Platform.isAndroid) {
    await Firebase.initializeApp(
        options: const FirebaseOptions(
            apiKey: 'AIzaSyAs5pu-La60fsbgpDu2ZVHWbElLnGMfjcw',
            appId: '1:930623352397:android:9c7e34b734d40440123604',
            messagingSenderId: '930623352397',
            projectId: 'travello-2ed88'));
  } else if (Platform.isWindows) {
    await Firebase.initializeApp(
        options: const FirebaseOptions(
```

```
      apiKey: "AIzaSyAeQZJnBJk2PBBC-GCd36vgMgNyFxlvSHo",
      authDomain: "travello-2ed88.firebaseapp.com",
      databaseURL: "https://travello-2ed88-default-rtdb.firebaseio.com",
      projectId: "travello-2ed88",
      storageBucket: "travello-2ed88.appspot.com",
      messagingSenderId: "930623352397",
      appId: "1:930623352397:web:80f8196bb14f8623123604",
    ));
  }
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      home: MyHomePage(),
    );
  }
}
class MyHomePage extends StatefulWidget {
  @override
  _MyHomePageState createState() => _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {
  BluetoothCheck bluetoothCheck = BluetoothCheck();
  final _firebasemessaging = FirebaseMessaging.instance;
  InternetCheck internetCheck = InternetCheck();

  @override
  void initState() {
    FirebaseAuth.instance.authStateChanges().listen((User? user) {
      if (user == null) {
        print('user is currently signed out!');
      } else {
        print('user signed in');
        if (!Platform.isWindows) {
          getToken();
        }
      }
    });
    super.initState();
    if (Platform.isAndroid) {
```

```
      BluetoothCheck.startBluetoothCheck(context);
    }
    internetCheck.checkConnection(context);
  }
  Future<void> getToken() async {
    String? token = await _firebasemessaging.getToken();
    try {
      String currentUserUid = FirebaseAuth.instance.currentUser!.uid;
      await FirebaseFirestore.instance
          .collection('passenger').doc(currentUserUid).update({
        "token": token,
      });
      print("Document added successfully!");
      print(token);
    } catch (e) {
      print("Error: $e");
    }
  }
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      routes: {
        '/': (context) => const Splash(),
        '/welcome1': (context) => const Welcome1(),
        '/welcome2': (context) => const Welcome2(),
        '/welcome3': (context) => const Welcome3(),
        '/bluetooth': (context) => const BluetoothScanner(),
        '/Register': (context) => RegistrationScreen(),
        '/Login': (context) => LoginScreen(),
        '/Verification': (context) => Verification(),
        '/menu': (context) => const MenuScreen(),
        '/Profile': (context) => const Profile(),
        '/baggage': (context) => const Baggage(),
        '/qrgen': (context) => const QRGen(),
        '/staffmenu': (context) => const StaffMenu(),
        '/staffscanner': (context) => const StaffScanner(),
        '/guestscreen': (context) => GuestSrceen(),
        '/client': (context) => const ClientScreen(),
        '/agent': (context) => const AgentScreen(),
        '/securitytracking':(context) => const SecurityTracking(),
      },
    );
  }
}
```

Splash Screen

We took advantage of lottie package to enable us using animation in splash screen

```
import 'dart:async';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:lottie/lottie.dart';

class Splash extends StatefulWidget {
  const Splash({Key? key});

  @override
  State<Splash> createState() => _SplashState();
}

class _SplashState extends State<Splash> {
  void initState() {
    super.initState();
    startTimer();
  }

  startTimer() {
    var duration = const Duration(seconds: 4, milliseconds: 0);
    return Timer(duration, route);
  }

  route() async {
    User? currentUser = FirebaseAuth.instance.currentUser;
    if (currentUser == null) {
      Navigator.pushReplacementNamed(context, '/welcome1');
    } else {
      String? email = currentUser.email;
      if (email != null && email == 'staff@travello.com') {
        Navigator.pushReplacementNamed(context, '/staffmenu');
      } else if (email!.contains('.security@travello.com')) {
        Navigator.pushReplacementNamed(context, '/securitytracking');
      } else {
        Navigator.pushReplacementNamed(context, '/menu');
      }
    }
  }
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: const Color(0xFF014F86),
```

```
      body: SingleChildScrollView(
        child: content(),
      ),
    );
  }

  Widget content() {
    return Center(
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          Transform.scale(
            scale: 0.75,
            child: Container(
              margin: EdgeInsets.only(top: 50.0, bottom: 20.0),
              child: Lottie.asset('lib/raw/splash2.json'),
            ),
          ),
          const Text(
            'Travello',
            style: TextStyle(
              color: Colors.white,
              fontSize: 40,
              fontWeight: FontWeight.bold,
              fontFamily: 'ADLaMDisplay',
              letterSpacing: 2.5,
            ),
          ),
        ],
      ),
    );
  }
}
```

Onboarding Screen

We designed three onboarding screens for the new members to represent the main idea of our system.

```
import 'package:flutter/material.dart';
import 'package:Travello/widget/nextfloatingbutton.dart';
import 'package:Travello/widget/introscreens.dart';
import 'package:Travello/widget/roundedrectangle.dart';

class Welcome1 extends StatefulWidget {
  const Welcome1({Key? key}) : super(key: key);

  @override
  State<Welcome1> createState() => _Welcome1State();
}

class _Welcome1State extends State<Welcome1> {
  Color firstRec = Colors.red;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.white,
      body: SingleChildScrollView(
        child: ConstrainedBox(
          constraints: BoxConstraints(
            minHeight: MediaQuery.of(context).size.height,
          ),
          child: content(),
        ),
      ),
    );
  }
  Widget content() {
    return Stack(
      children: [
        Center(
          child: Column(
            mainAxisAlignment: MainAxisAlignment.start,
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
              SizedBox(height: 40),
              Container(
                alignment: Alignment.topCenter,
                margin: const EdgeInsets.only(top: 10.0),
                child: Transform.scale(
```

```
            scale: 0.7,
            child: Image.asset('images/Plane.png'),
          ),
        ),
        const CustomTextWidget(
          text: 'Navigate airports\nwith ease',
          fontSize: 30,
          fontWeight: FontWeight.bold,
          letterSpacing: 2,
          margin: EdgeInsets.only(left: 10.0, top: 10),
        ),
        const CustomTextWidget(
          text: 'Real-time guidance for\nstress-free travel!',
          color: Color(0xFF848897),
          fontSize: 20,
          fontWeight: FontWeight.normal,
          letterSpacing: 1,
          margin: EdgeInsets.only(left: 10.0),
        ),
      ],
    ),
  ),
  Positioned(
    bottom: 20,
    right: 20,
    child: NextFButton(
      onPressed: () {
        Navigator.pushReplacementNamed(context, '/welcome2');
        setState(() {
          firstRec = const Color(0xFFFFC7C7);
        });
      },
    ),
  ),
  Positioned(
    bottom: 20,
    left: 0,
    right: 0,
    child: Row(
      mainAxisAlignment: MainAxisAlignment.center,
      children: [
        RoundedRectangle(
          color: firstRec,
        ),
        SizedBox(width: 2),
```

```
              RoundedRectangle(
                color: const Color(0xFFFFC7C7),
                onPressed: () =>
                    Navigator.pushReplacementNamed(context, '/welcome2'),
              ),
              SizedBox(width: 2),
              RoundedRectangle(
                color: const Color(0xFFFFC7C7),
                onPressed: () =>
                    Navigator.pushReplacementNamed(context, '/welcome3'),
              ),
            ],
          ),
        ),
      ],
    );
  }
}
```

Login Screen

As we care about privacy & security of the user. The system authenticate user's identity every time trying to login.

```
import 'dart:io';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:fluttertoast/fluttertoast.dart';

class LoginScreen extends StatefulWidget {
  @override
  _LoginScreenState createState() => _LoginScreenState();
}

class _LoginScreenState extends State<LoginScreen> {
  TextEditingController email = TextEditingController();
  TextEditingController password = TextEditingController();

  void forgetPassword() async {
    if (email.text.isNotEmpty) {
      try {
        await FirebaseAuth.instance.sendPasswordResetEmail(email: email.text);
        Fluttertoast.showToast(
          msg: "Password reset email sent successfully.",
        );
      } on FirebaseAuthException catch (e) {
```

```
      String errorMessage =
          "Error sending password reset email. Please try again later.";
      if (e.code == 'user-not-found') {
        errorMessage = "Email address not found.";
      }
      Fluttertoast.showToast(
        msg: errorMessage,
      );
    } catch (e) {
      Fluttertoast.showToast(msg: "Error: $e");
    }
  } else {
    Fluttertoast.showToast(
      msg: "Please enter your email address.",
    );
  }
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    body: SingleChildScrollView(
      child: Container(
        padding: EdgeInsets.all(16.0),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.stretch,
          children: [
            SizedBox(height: 100.0),
            Container(
              alignment: Alignment.center,
              child: Text(
                'Welcome Back',
                style: TextStyle(
                  fontSize: 26,
                  fontWeight: FontWeight.bold,
                  fontFamily: 'ADLaMDisplay',
                ),
              ),
            ),
            SizedBox(height: 50.0),
            _roundedTextField('Email', Icons.email, controller: email),
            SizedBox(height: 10.0),
            _roundedTextField('Password', Icons.lock,
                obscureText: true, controller: password),
            Container(
```

```
  alignment: Alignment.centerRight,
  child: TextButton(
    onPressed: () {
      forgetPassword();
    },
    child: Text(
      "Forgot Password?",
      style: TextStyle(
        fontSize: 14,
        fontWeight: FontWeight.bold,
        color: Color(0xFFFF3951),
      ),
    ),
  ),
),
SizedBox(height: 120.0),
ElevatedButton(
  style: ElevatedButton.styleFrom(
    padding: EdgeInsets.symmetric(vertical: 10.0),
    backgroundColor: Color(0xFF01497C),
    shape: RoundedRectangleBorder(
      borderRadius: BorderRadius.circular(8.0),
    ),
  ),
  onPressed: () async {
    if (email.text.isEmpty || password.text.isEmpty) {
      Fluttertoast.showToast(msg: 'Please fill your credentials');
    } else {
      try {
        final credential = await FirebaseAuth.instance
            .signInWithEmailAndPassword(
                email: email.text, password: password.text);
        if (Platform.isWindows ||
            Platform.isAndroid &&
                email.text == "staff@travello.com") {
          Navigator.pushReplacementNamed(context, '/staffmenu');
        } else if (email.text
            .contains('.security@travello.com')) {
          Navigator.pushReplacementNamed(
              context, '/securitytracking');
        } else {
          if (FirebaseAuth.instance.currentUser!.emailVerified ==
              false) {
            Navigator.pushReplacementNamed(
                context, '/Verification');
```

```
            } else {
              Navigator.pushReplacementNamed(context, '/menu');
            }
          }
        } on FirebaseAuthException catch (e) {
          Fluttertoast.showToast(
              msg: "email or password is incorrect");
          if (e.code == 'user-not-found') {
            Fluttertoast.showToast(
                msg: 'No user found for that email.');
          } else if (e.code == 'wrong-password') {
            Fluttertoast.showToast(
                msg: 'Wrong password provided for that user.');
          }
        }
      }
    },
    child: Text(
      'Login',
      style: TextStyle(
        fontSize: 18,
        color: Colors.white,
        fontWeight: FontWeight.bold,
        fontFamily: 'ADLaMDisplay',
      ),
    ),
  ),
),
Container(
  margin: EdgeInsets.symmetric(vertical: 5),
  child: Row(
    mainAxisAlignment: MainAxisAlignment.center,
    children: [
      Text(
        'New member?',
        style: TextStyle(
          fontWeight: FontWeight.bold,
        ),
      ),
      TextButton(
        onPressed: () {
          Navigator.pushReplacementNamed(context, '/Register');
        },
        child: Text(
          'Register Now',
          style: TextStyle(
```

```
                        fontWeight: FontWeight.bold,
                        color: Color(0xFFFF3951),
                      ),
                    ),
                  ),
                ],
              ),
            ),
            TextButton(
              onPressed: () {
                Navigator.pushReplacementNamed(context, '/guestscreen');
              },
              child: Text(
                'Continue as a Guest',
                style: TextStyle(
                  fontWeight: FontWeight.bold,
                  color: Color(0xFFFF3951),
                ),
              ),
            ),
          ],
        ),
      ),
    ),
  );
}
```

Register Screen

We designed and built simple registration form for the ease of use.

```
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:fluttertoast/fluttertoast.dart';

class RegistrationScreen extends StatefulWidget {
  @override
  _RegistrationScreenState createState() => _RegistrationScreenState();
}

class _RegistrationScreenState extends State<RegistrationScreen> {
  TextEditingController username = TextEditingController();
  TextEditingController email = TextEditingController();
  TextEditingController password = TextEditingController();
  TextEditingController rpassword = TextEditingController();
```

```
  GlobalKey<FormState> formstate = GlobalKey<FormState>();
  CollectionReference user = FirebaseFirestore.instance.collection('passenger');

Future<void> addName(String name) async {
    try {
      String currentUserUid = FirebaseAuth.instance.currentUser!.uid;
      await FirebaseFirestore.instance
          .collection('passenger')
          .doc(currentUserUid)
          .set({"username": name, "passenger_id": currentUserUid});
      print("Document added successfully!");
    } catch (e) {
      print("Error: $e");
    }
  }
  Future<bool> isUsernameAvailable(String username) async {
    try {
      QuerySnapshot querySnapshot = await FirebaseFirestore.instance
          .collection('passenger')
          .where('username', isEqualTo: username)
          .get();
      return querySnapshot.docs.isEmpty;
    } catch (e) {
      print("Error checking username availability: $e");
      return false;
    }
  }
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: SingleChildScrollView(
        child: Padding(
          padding: const EdgeInsets.all(16.0),
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.stretch,
            children: [
              Container(
                alignment: Alignment.center,
                margin: const EdgeInsets.only(top: 100),
                child: const Text(
                  'Get Started',
                  style: TextStyle(
                      fontSize: 26,
                      fontWeight: FontWeight.bold,
                      fontFamily: 'ADLaMDisplay'),
```

```
  ),
),
const SizedBox(height: 30),
_roundedTextField('Username', Icons.person, controller: username),
const SizedBox(height: 10),
_roundedTextField('Email', Icons.email, controller: email),
const SizedBox(height: 10),
_roundedTextField('Password', Icons.lock,
    obscureText: true, controller: password),
const SizedBox(height: 10),
const SizedBox(height: 120),
ElevatedButton(
  style: ElevatedButton.styleFrom(
    padding: const EdgeInsets.symmetric(vertical: 10),
    backgroundColor: const Color(0xFF01497C),
    shape: RoundedRectangleBorder(
      borderRadius: BorderRadius.circular(8.0),
    ),
  ),
  onPressed: () async {
    if (email.text.isEmpty ||
        password.text.isEmpty ||
        username.text.isEmpty) {
      Fluttertoast.showToast(msg: 'Please fill your credentials');
    } else {
      try {
        final credential = await FirebaseAuth.instance
            .createUserWithEmailAndPassword(
          email: email.text,
          password: password.text,
        );
        addName(username.text);
        FirebaseAuth.instance.currentUser!.emailVerified
            ? Navigator.pushReplacementNamed(context, '/Login')
            : Navigator.pushReplacementNamed(
                context, '/Verification');
      } on FirebaseAuthException catch (e) {
        if (e.code == 'weak-password') {
          Fluttertoast.showToast(
              msg: 'The password provided is too weak.');
        } else if (e.code == 'email-already-in-use') {
          Fluttertoast.showToast(
              msg: 'The account already exists for that email.');
        }
      } catch (e) {
```

```
          print(e);
        }
      }
    },
    child: const Text(
      'SignUp',
      style: TextStyle(
          fontSize: 18,
          color: Colors.white,
          fontWeight: FontWeight.bold,
          fontFamily: 'ADLaMDisplay'),
    ),
  ),
  Container(
    margin: const EdgeInsets.symmetric(vertical: 5),
    child: Row(
      mainAxisAlignment: MainAxisAlignment.center,
      children: [
        const Text(
          'Already a member?',
          style: TextStyle(fontWeight: FontWeight.bold),
        ),
        TextButton(
          onPressed: () {
            Navigator.pushReplacementNamed(context, '/Login');
          },
          child: const Text(
            'Login',
            style: TextStyle(
              fontWeight: FontWeight.bold,
              color: Color(0xFFFF3951),
            ),
          ),
        ),
      ],
    ),
  ),
  Container(
    child: TextButton(
      onPressed: () {
        Navigator.pushReplacementNamed(context, '/guestscreen');
      },
      child: const Text(
        'Continue as a Guest',
        style: TextStyle(
```

```
                        fontWeight: FontWeight.bold, color: Color(0xFFFF3951)),
                  ),
                ),
              )
            ],
          ),
        ),
      ),
    );
  }
```

Verification Screen

For security purposes the email address used by the user to create an account must be verified first to be able to use the application. This done but pressing on the link sent to the user's email.

```
import 'dart:async';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:fluttertoast/fluttertoast.dart';
import 'package:Travello/widget/backfloatingbutton.dart';

class Verification extends StatefulWidget {
  @override
  _VerificationState createState() => _VerificationState();
}
class _VerificationState extends State<Verification> {
  List<TextEditingController> controllers =
      List.generate(6, (index) => TextEditingController());
  int _timerCount = 30;
  bool _resendButtonEnabled = false;
  late Timer _timer;
  @override
  void initState() {
    super.initState();
    FirebaseAuth.instance.currentUser!.sendEmailVerification();
    _startTimer();
    checkVerification();
  }

  void _startTimer() {
    _timer = Timer.periodic(const Duration(seconds: 1), (timer) {
      checkVerification();
      if (_timerCount > 0) {
        setState(() {
          _timerCount--;
```

```
      });
    } else {
      setState(() {
        _resendButtonEnabled = true;
        _timer.cancel();
      });
    }
  });
}
Future<void> checkVerification() async {
  User? user = FirebaseAuth.instance.currentUser;
  await user?.reload();
  if (user?.emailVerified == true) {
    Fluttertoast.showToast(msg: "Email Verified!");
    Navigator.pushReplacementNamed(context, '/Login');
  }
}
@override
Widget build(BuildContext context) {
  return Scaffold(
    body: SafeArea(
      child: Container(
        height: MediaQuery.of(context).size.height,
        child: Stack(
          children: [
            Container(
              height: double.infinity,
              child: Center(
                child: Padding(
                  padding: const EdgeInsets.all(10.0),
                  child: Column(
                    crossAxisAlignment: CrossAxisAlignment.start,
                    children: [
                      const SizedBox(height: 20),
                      const Text(
                        'Almost There',
                        style: TextStyle(
                            fontSize: 28,
                            fontWeight: FontWeight.bold,
                            fontFamily: 'ADLaMDisplay'),
                      ),
                      const SizedBox(height: 7),
                      const Text(
                        'Please click on the link sent to your email.',
                        style: TextStyle(fontSize: 16, color: Colors.grey),
```

```
            ),
            const SizedBox(height: 80),
            const SizedBox(height: 10),
            Row(
              mainAxisAlignment: MainAxisAlignment.center,
              children: [
                const Text("Didn't receive a code?"),
                TextButton(
                  onPressed: _resendButtonEnabled
                      ? () {
                          FirebaseAuth.instance.currentUser!
                              .sendEmailVerification();
                          setState(() {
                            _resendButtonEnabled = false;
                            _timerCount = 30;
                            _timer.cancel();
                            _startTimer();
                          });
                        }
                      : null,
                  child: const Text('Resend'),
                ),
              ],
            ),
            const SizedBox(height: 1),
            Center(
              child: Text(
                'Resend a new code in $_timerCount seconds',
                style: const TextStyle(
                    color: Colors.grey, fontSize: 14),
              ),
            ),
          ],
        ),
      ),
    ),
  ),
  Positioned(
    bottom: 16.0,
    left: 16.0,
    child: BackFButton(
      onPressed: () {
        Navigator.pushReplacementNamed(context, '/Register');
      },
    ),
```

```
            ),
          ],
        ),
      ),
    ),
  );
  }

  @override
  void dispose() {
    _timer.cancel();
    super.dispose();
  }
}
```

Menu Screen

Clear and simple menu contains all features and services can be used by the passenger.

```
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:fluttertoast/fluttertoast.dart';
import 'package:Travello/screens/ticket.dart';
import 'package:Travello/widget/nextfloatingbutton.dart';
import 'package:Travello/widget/menubuttons.dart';


class MenuScreen extends StatefulWidget {
  const MenuScreen({Key? key}) : super(key: key);

  @override
  _MenuScreenState createState() => _MenuScreenState();
}

class _MenuScreenState extends State<MenuScreen> {
  bool _isExpanded = false;
  Map<String, dynamic> flightInfo = {};

  @override
  void initState() {
    getName();
    getFlightData();
    super.initState();
  }

  @override
```

```
Widget build(BuildContext context) {
  final Size screenSize = MediaQuery.of(context).size;
  return Scaffold(
    appBar: AppBar(
      backgroundColor: const Color(0xFF01497C),
      shape: const RoundedRectangleBorder(
        borderRadius: BorderRadius.only(
          bottomLeft: Radius.circular(20.0),
          bottomRight: Radius.circular(20.0),
        ),
      ),
      leading: GestureDetector(
        onTap: () {
          Navigator.pushReplacementNamed(context, '/Profile');
        },
        child: Container(
          padding: const EdgeInsets.all(5),
          margin: const EdgeInsets.only(bottom: 5, left: 5),
          child: const CircleAvatar(
            radius: 20.0,
            backgroundImage: AssetImage('images/avatar.png'),
          ),
        ),
      ),
      title: FutureBuilder<String>(
        future: getName(),
        builder: (BuildContext context, AsyncSnapshot<String> snapshot) {
          if (snapshot.connectionState == ConnectionState.waiting) {
            return const Center(
              child: CircularProgressIndicator(
                valueColor: AlwaysStoppedAnimation<Color>(Colors.white),
              ),
            );
          } else if (snapshot.hasError) {
            return Text('Error: ${snapshot.error}');
          } else {
            return Text(
              snapshot.data!,
              style: const TextStyle(color: Colors.white),
            );
          }
        },
      ),
      actions: [
        IconButton(
```

```
onPressed: () {
  showDialog(
    context: context,
    builder: (BuildContext context) {
      return AlertDialog(
        content: SingleChildScrollView(
          child: SizedBox(
            height: MediaQuery.of(context).size.height * 0.5,
            width: MediaQuery.of(context).size.height * 0.7,
            child: Padding(
              padding: const EdgeInsets.all(10),
              child: Stack(
                children: [
                  Column(
                    children: [
                      Container(
                        alignment: Alignment.topLeft,
                        margin: const EdgeInsets.only(top: 20),
                        child: const Text(
                          'New Flight',
                          style: TextStyle(
                              fontSize: 26,
                              fontWeight: FontWeight.bold,
                              fontFamily: 'ADLaMDisplay'),
                        ),
                      ),
                      const SizedBox(height: 40),
                      _roundedTextField('Enter Ticket ID',
                          Icons.airplane_ticket_outlined,
                          controller: ticketno),
                      const SizedBox(height: 22),
                      ElevatedButton(
                        style: ElevatedButton.styleFrom(
                          padding: const EdgeInsets.symmetric(
                              vertical: 0),
                          backgroundColor: const Color(0xFF01497C),
                          shape: RoundedRectangleBorder(
                            borderRadius:
                                BorderRadius.circular(8.0),
                          ),
                          minimumSize: Size(130,45),
                        ),
                        onPressed: () {
                          getFlightData();
                          addFlightToPassenger();
```

```
                                currentFlight();
                                Navigator.pop(context);
                              },
                              child: const Text(
                                'Enter',
                                style: TextStyle(
                                    fontSize: 18,
                                    color: Colors.white,
                                    fontFamily: 'ADLaMDisplay'),
                              ),
                            ),
                          ),
                        ],
                      ),
                    ],
                  ),
                ),
              ),
            );
          },
        );
      },
      icon: const Icon(
        Icons.airplane_ticket_outlined,
        color: Colors.white,
        size: 34,
      ),
    ),
  ],
),
body: LayoutBuilder(
  builder: (context, constraints) {
    return SingleChildScrollView(
      child: ConstrainedBox(
        constraints: BoxConstraints(
          minHeight: constraints.maxHeight,
        ),
        child: Padding(
          padding: const EdgeInsets.all(16.0),
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.stretch,
            children: [
              const SizedBox(height: 50.0),
              MenuButton(
                onPressed: () {
```

```
        setState(() {
          _isExpanded = !_isExpanded;
        });
      },
      text: 'Flight Information',
    ),
    ClipRRect(
      borderRadius: BorderRadius.circular(10.0),
      child: AnimatedContainer(
        duration: const Duration(milliseconds: 500),
        height: _isExpanded ? 200.0 : 0.0,
        width: MediaQuery.of(context).size.width,
        color: Colors.orange[300],
        child: SingleChildScrollView(
          child: Padding(
            padding: const EdgeInsets.all(16.0),
            child: Row(
              mainAxisAlignment: MainAxisAlignment.spaceEvenly,
              children: [
                Expanded(
                  child: Column(
                    crossAxisAlignment:
                        CrossAxisAlignment.start,
                    children: [
                      const SizedBox(height: 20),
                      Text(
                        'From: ${flightInfo['departure']}',
                        style: const TextStyle(
                            color: Colors.black,
                            fontSize: 19,
                            fontWeight: FontWeight.bold),
                      ),
                      const SizedBox(height: 8),
                      Text(
                        'To: ${flightInfo['arrival']}',
                        style: const TextStyle(
                            color: Colors.black,
                            fontSize: 19,
                            fontWeight: FontWeight.bold),
                      ),
                      const SizedBox(height: 8),
                      Text(
                        'Terminal: ${flightInfo['terminal']}',
                        style: const TextStyle(
                            color: Colors.black,
```

```
                                fontSize: 19,
                                fontWeight: FontWeight.bold),
                          ),
                        ],
                      ),
                    ),
                    Expanded(
                      child: Column(
                        crossAxisAlignment:
                            CrossAxisAlignment.start,
                        children: [
                          const SizedBox(height: 20),
                          Text(
                            'Date: ${flightInfo['departure_date']}',
                            style: const TextStyle(
                                color: Colors.black,
                                fontSize: 19,
                                fontWeight: FontWeight.bold),
                          ),
                          const SizedBox(height: 8),
                          Text(
                            'Departure: ${flightInfo['departure_time']}',
                            style: const TextStyle(
                                color: Colors.black,
                                fontSize: 19,
                                fontWeight: FontWeight.bold),
                          ),
                          const SizedBox(height: 8),
                          Text(
                            'Gate: ${flightInfo['gate']}',
                            style: const TextStyle(
                                color: Colors.black,
                                fontSize: 19,
                                fontWeight: FontWeight.bold),
                          ),
                        ],
                      ),
                    ),
                  ],
                ),
              ),
            ),
          ),
        ),
      ),
    ),
  ),
  const SizedBox(height: 12.0),
```

```
                    MenuButton(
                      onPressed: () {
                        Navigator.pushReplacementNamed(context, '/bluetooth');
                      },
                      text: 'Navigation Options',
                    ),
                    const SizedBox(height: 12.0),
                    MenuButton(
                      onPressed: () {
                        Navigator.pushReplacementNamed(context, '/baggage');
                      },
                      text: 'Baggage Tracking',
                    ),
                    const SizedBox(height: 12.0),
                    MenuButton(
                      onPressed: () {
                        Navigator.pushReplacementNamed(context, '/guestscreen');
                      },
                      text: 'Find Family and Friends',
                    ),
                  ],
                ),
              ),
            ),
          );
        },
      ),
      floatingActionButton: NextFButton(
        onPressed: () async {
          Navigator.popAndPushNamed(context, '/client');
        },
        icon: Icons.contact_support,
      ),
      floatingActionButtonLocation: FloatingActionButtonLocation.endFloat,
    );
}

Future<String> getName() async {
  try {
    String currentUserUid = FirebaseAuth.instance.currentUser!.uid;
    DocumentSnapshot documentSnapshot = await FirebaseFirestore.instance
        .collection('passenger')
        .doc(currentUserUid)
        .get();
```

```
    if (documentSnapshot.exists) {
      Map<String, dynamic> data =
          documentSnapshot.data() as Map<String, dynamic>;
      String name = data['username'];
      return name;
    } else {
      throw Exception("Document does not exist");
    }
  } catch (e) {
    print("Error: $e");
    throw e;
  }
}

Future<void> initializeData() async {
  try {
    await Future.wait([
      getName(),
    ]);
    print('Initialization complete');
  } catch (e) {
    print('Initialization error: $e');
  }
}

Future<void> getFlightData() async {
  try {
    FirebaseFirestore firestore = FirebaseFirestore.instance;
    String currentUserUid = FirebaseAuth.instance.currentUser!.uid;
    DocumentSnapshot passengerDoc =
        await firestore.collection('passenger').doc(currentUserUid).get();

    if (passengerDoc.exists) {
      Map<String, dynamic> current =
          passengerDoc.data() as Map<String, dynamic>;
      if (current.containsKey('currentflight') &&
          current['currentflight'] != null &&
          current['currentflight'].isNotEmpty) {
        String name = current['currentflight'];
        DocumentSnapshot ticketDoc =
            await firestore.collection('ticket').doc(name).get();
        if (ticketDoc.exists) {
          Map<String, dynamic> data =
              ticketDoc.data() as Map<String, dynamic>;
          int ticket = data['flightID'];
```

```
          DocumentSnapshot flightDoc = await firestore
              .collection('flight')
              .doc(ticket.toString())
              .get();
          if (flightDoc.exists) {
            Map<String, dynamic> flightData =
                flightDoc.data() as Map<String, dynamic>;
            flightInfo = {
              'arrival': flightData['arrival'],
              'arrival_time': flightData['arrival_time'],
              'departure': flightData['departure'],
              'departure_date': flightData['departure_date'],
              'departure_time': flightData['departure_time'],
              'gate': flightData['gate'],
              // 'seat': flightData['seat'],
              'terminal': flightData['terminal']
            };
          } else {
            throw Exception("Flight document does not exist");
          }
        } else {
          Fluttertoast.showToast(msg: "Please Enter Valid Ticket");
        }
      } else {
        Fluttertoast.showToast(msg: "Please Enter Your Ticket number");
      }
    } else {
      throw Exception("Passenger document does not exist");
    }
  } catch (e) {
    print("Error: $e");
    Fluttertoast.showToast(msg: "Flight not exists");
  }
}
```

New Flight Screen

The passenger must enter his ticket number in the specified place to get information about the ticket and will be used later for tracking baggage.

```
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';

class TicketNumber extends StatefulWidget {
  const TicketNumber({super.key});
  @override
  State<TicketNumber> createState() => _TicketNumberState();
}

TextEditingController ticketno = TextEditingController();

class _TicketNumberState extends State<TicketNumber> {
   @override
     void initState() {
    super.initState();
  }
  @override
  Widget build(BuildContext context) {
    final Size screenSize = MediaQuery.of(context).size;
    return AlertDialog(
      content: SingleChildScrollView(
        child: Container(
          height: screenSize.height * 0.5,
          width: screenSize.height * 0.7,
          child: Padding(
            padding: const EdgeInsets.all(10),
            child: Stack(
              children: [
                Column(
                  children: [
                    Container(
                      alignment: Alignment.topLeft,
                      margin: const EdgeInsets.only(top: 20),
                      child: const Text(
                        'New Flight',
                        style: TextStyle(
                          fontSize: 26,
                          fontWeight: FontWeight.bold,
                        ),
                      ),
```

```
                    ),
                    const SizedBox(height: 40),
                    _roundedTextField(
                        'Enter Ticket ID', Icons.airplane_ticket_outlined,
                        controller: ticketno),
                    const SizedBox(height: 22),
                    ElevatedButton(
                      style: ElevatedButton.styleFrom(
                        padding: const EdgeInsets.symmetric(vertical: 0),
                        backgroundColor: const Color(0xFF01497C),
                        shape: RoundedRectangleBorder(
                          borderRadius: BorderRadius.circular(8.0),
                        ),
                        minimumSize: Size(
                            130, 45),
                      ),
                      onPressed: () {
                        getFlightData();
                        addFlightToPassenger();
                        currentFlight();
                        Navigator.pop(context);
                      },
                      child: const Text(
                        'Enter',
                        style: TextStyle(fontSize: 18, color: Colors.white),
                      ),
                    ),
                  ],
                ),
              ],
            ),
          ),
        ),
      ),
    );
  }
}

Future<void> addFlightToPassenger() async {
  try {
    String currentUserUid = FirebaseAuth.instance.currentUser!.uid;

    await
FirebaseFirestore.instance.collection('passengerflights').doc(ticketno.text).set({
      "passengerid":currentUserUid
```

```
    });
    print("Document added successfully!");
  } catch (e) {
    print("Error: $e");
  }
}

Future<void> currentFlight() async {
  try {
    String currentUserUid = FirebaseAuth.instance.currentUser!.uid;
    await
FirebaseFirestore.instance.collection('passenger').doc(currentUserUid).update({
      "currentflight": ticketno.text,
    });
    print("Document updated successfully!");
  } catch (e) {
    print("Error: $e");
  }
}
}
```

Find Family & Friends Screen

for both membered and guest user they can use Find Family & Friends feature by entering the ticket number of the person they want to find.

```
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:fluttertoast/fluttertoast.dart';
import 'package:Travello/widget/backfloatingbutton.dart';


class GuestSrceen extends StatefulWidget {
  @override
  _GuestSrceenState createState() => _GuestSrceenState();
}

class _GuestSrceenState extends State<GuestSrceen> {
  TextEditingController ticket = TextEditingController();
  String currentLocation = "";
  String passengerName = "";

  Future<void> getPassengerLocation() async {
    try {
      FirebaseFirestore firebaseFirestore = FirebaseFirestore.instance;
      DocumentSnapshot ticketDoc = await firebaseFirestore
          .collection('passengerflights')
```

```
        .doc(ticket.text)
        .get();

    if (ticketDoc.exists) {
      String passengerId = ticketDoc['passengerid'];

      firebaseFirestore
          .collection('passenger')
          .doc(passengerId)
          .snapshots()
          .listen((passengerInfo) {
        if (passengerInfo.exists) {
          setState(() {
            if (passengerInfo.data()!.containsKey('currentlocation')) {
              currentLocation = passengerInfo['currentlocation'];
            } else {
              Fluttertoast.showToast(msg: "Passenger information not found");
            }
            if (passengerInfo.data()!.containsKey('username')) {
              passengerName = passengerInfo['username'];
            } else {
              Fluttertoast.showToast(msg: "Passenger information not found");
            }
          });
        } else {
          Fluttertoast.showToast(msg: "Passenger information not found");
        }
      });
    } else {
      Fluttertoast.showToast(msg: "Incorrect Ticket or not exists");
    }
  } catch (e) {
    print('Error: $e');
  }
}

route() async {
  User? currentUser = FirebaseAuth.instance.currentUser;

  if (currentUser == null) {
    Navigator.pushReplacementNamed(context, '/Login');
  } else {
    Navigator.pushReplacementNamed(context, '/menu');
  }
}
```

```
@override
Widget build(BuildContext context) {
  return Scaffold(
    body: Stack(
      children: [
        SingleChildScrollView(
          child: Container(
            margin: const EdgeInsets.only(top: 100),
            height: MediaQuery.of(context).size.height,
            child: Padding(
              padding: const EdgeInsets.all(16.0),
              child: Column(
                crossAxisAlignment: CrossAxisAlignment.stretch,
                children: [
                  Container(
                    alignment: Alignment.center,
                    margin: const EdgeInsets.only(top: 10),
                    child: const Text(
                      'Find Family and Friends',
                      style: TextStyle(
                          fontSize: 26,
                          fontWeight: FontWeight.bold,
                          fontFamily: 'ADLaMDisplay'),
                    ),
                  ),
                  const SizedBox(height: 50),
                  Row(
                    crossAxisAlignment: CrossAxisAlignment.center,
                    children: [
                      Expanded(
                        child: _roundedTextField(
                          'Ticket Number',
                          Icons.airplane_ticket,
                          controller: ticket,
                        ),
                      ),
                      const SizedBox(
                          width: 16),
                      ElevatedButton(
                        style: ElevatedButton.styleFrom(
                          padding: const EdgeInsets.symmetric(vertical: 15),
                          backgroundColor: const Color(0xFF01497C),
                          shape: RoundedRectangleBorder(
                            borderRadius: BorderRadius.circular(10.0),
                          ),
```

```
        ),
        onPressed: () async {
          if (ticket.text != "") {
            getPassengerLocation();
          } else {
            Fluttertoast.showToast(
                msg: "Please Enter Ticket Number");
          }
        },
        child: const Text(
          'Find',
          style: TextStyle(
              fontSize: 14,
              color: Colors.white,
              fontWeight: FontWeight.bold,
              fontFamily: 'ADLaMDisplay'),
        ),
      ),
    ],
  ),
  const SizedBox(height: 20),
  Center(
    child: Container(
      padding: const EdgeInsets.all(16),
      decoration: BoxDecoration(
        borderRadius: BorderRadius.circular(10),
        color: Colors.grey[200],
      ),
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          Padding(
            padding: const EdgeInsets.only(bottom: 8),
            child: Row(
              children: [
                const Text(
                  'Passenger Name: ',
                  style: TextStyle(
                      fontSize: 20,
                      fontWeight: FontWeight.bold,
                      color: Color(0xFFFF3951),
                      fontFamily: 'ADLaMDisplay'),
                ),
                Text(
                  passengerName,
```

```
                        style: const TextStyle(
                            fontSize: 16,
                            fontWeight: FontWeight.bold,
                            fontFamily: 'ADLaMDisplay'),
                      ),
                    ],
                  ),
                ),
                Padding(
                  padding: const EdgeInsets.only(top: 8),
                  child: Row(
                    children: [
                      const Text(
                        'Current Location: ',
                        style: TextStyle(
                            fontSize: 20,
                            fontWeight: FontWeight.bold,
                            color: Color(0xFFFF3951),
                            fontFamily: 'ADLaMDisplay'),
                      ),
                      Text(
                        currentLocation,
                        style: const TextStyle(
                            fontSize: 16,
                            fontWeight: FontWeight.bold,
                            fontFamily: 'ADLaMDisplay'),
                      ),
                    ],
                  ),
                ),
              ],
            ),
          ),
        ],
      ),
    ),
  ),
),

Positioned(
  bottom: 16.0,
  left: 16.0,
  child: BackFButton(
    onPressed: () {
```

```
                route();
              },
            ),
          ),
        ],
      ),
    );
  }
}
```

Baggage Tracking Screen

For each ticket owned by passenger there is a list of bags attached to that ticket added by the passenger. The passenger can retrieve and display all his baggage, also he can choose any of these bags to track it

```
import 'dart:async';
import 'package:flutter/material.dart';
import 'package:fluttertoast/fluttertoast.dart';
import 'package:Travello/screens/newbag.dart';
import 'package:Travello/widget/nextfloatingbutton.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:cloud_firestore/cloud_firestore.dart';

class Baggage extends StatefulWidget {
  const Baggage({Key? key}) : super(key: key);

  @override
  State<Baggage> createState() => _BaggageState();
}
class _BaggageState extends State<Baggage> {
  @override
  void initState() {
    getBags();
    super.initState();
  }
  List<QueryDocumentSnapshot> data = [];
  List<QueryDocumentSnapshot> status = [];
  StreamSubscription<QuerySnapshot>? statusSubscription;

  @override
  void dispose() {
    statusSubscription?.cancel();
    super.dispose();
  }
  Future<void> getBags() async {
    try {
      FirebaseFirestore firestore = FirebaseFirestore.instance;
```

```
    String currentUserUid = FirebaseAuth.instance.currentUser!.uid;
    DocumentSnapshot passengerDoc =
        await firestore.collection('passenger').doc(currentUserUid).get();

    if (passengerDoc.exists) {
      Map<String, dynamic> current =
          passengerDoc.data() as Map<String, dynamic>;
      String name = current['currentflight'];
      DocumentReference passengerRef =
          firestore.collection('passengerflights').doc(name);
      QuerySnapshot bagsQuery = await passengerRef.collection('bags').get();
      setState(() {
        data = bagsQuery.docs;
      });
    } else {
      print('Passenger document not found for user: $currentUserUid');
    }
  } catch (e) {
    Fluttertoast.showToast(
        msg: "Please enter Ticket number to track luggages");
  }
}

Future<void> getStatus(String docId) async {
  try {
    FirebaseFirestore firestore = FirebaseFirestore.instance;
    String currentUserUid = FirebaseAuth.instance.currentUser!.uid;
    DocumentReference passengerDocRef =
        firestore.collection('passenger').doc(currentUserUid);
    passengerDocRef.snapshots().listen((passengerDocSnapshot) async {
      if (passengerDocSnapshot.exists) {
        Map<String, dynamic> current =
            passengerDocSnapshot.data() as Map<String, dynamic>;
        String name = current['currentflight'];
        DocumentReference passengerRef =
            firestore.collection('passengerflights').doc(name);
        DocumentReference statusRef =
            passengerRef.collection('bags').doc(docId);

        statusSubscription = statusRef
            .collection('status')
            .snapshots()
            .listen((statusSnapshot) {
          setState(() {
            status = statusSnapshot.docs;
```

```
          });
        });
      } else {
        print('Passenger document not found for user: $currentUserUid');
      }
    });
  } catch (e) {
    print('Error retrieving current flight: $e');
  }
}

Future<String> getName() async {
  try {
    String currentUserUid = FirebaseAuth.instance.currentUser!.uid;
    DocumentSnapshot documentSnapshot = await FirebaseFirestore.instance
        .collection('passenger')
        .doc(currentUserUid)
        .get();

    if (documentSnapshot.exists) {
      Map<String, dynamic> userData =
          documentSnapshot.data() as Map<String, dynamic>;
      String name = userData['username'];
      return name;
    } else {
      throw Exception("Document does not exist");
    }
  } catch (e) {
    print("Error: $e");
    throw e;
  }
}

@override
Widget build(BuildContext context) {
  final screenSize = MediaQuery.of(context).size;
  return Scaffold(
    appBar: AppBar(
      backgroundColor: const Color(0xFF01497C),
      shape: const RoundedRectangleBorder(
        borderRadius: BorderRadius.only(
          bottomLeft: Radius.circular(20.0),
          bottomRight: Radius.circular(20.0),
        ),
      ),
```

```
leading: GestureDetector(
  onTap: () {
    Navigator.pushReplacementNamed(context, '/Profile');
  },
  child: Container(
    padding: const EdgeInsets.all(5),
    margin: const EdgeInsets.only(bottom: 5, left: 5),
    child: const CircleAvatar(
      radius: 20.0,
      backgroundImage: AssetImage('images/avatar.png'),
    ),
  ),
),
title: Builder(
  builder: (context) => FutureBuilder<String>(
    future: getName(),
    builder: (BuildContext context, AsyncSnapshot<String> snapshot) {
      if (snapshot.connectionState == ConnectionState.waiting) {
        return const Center(
          child: CircularProgressIndicator(
            valueColor: AlwaysStoppedAnimation<Color>(Colors.white),
          ),
        );
      } else if (snapshot.hasError) {
        return Text('Error: ${snapshot.error}');
      } else {
        return Text(
          snapshot.data!,
          style: const TextStyle(color: Colors.white),
        );
      }
    },
  ),
),
actions: [
  IconButton(
    onPressed: () {
      Navigator.pushReplacementNamed(context, '/menu');
    },
    icon: const Icon(
      Icons.arrow_back_ios_new_outlined,
      color: Colors.white,
    ),
  ),
],
```

```
),
body: SingleChildScrollView(
  child: Container(
    margin: const EdgeInsets.all(40),
    child: Column(
      crossAxisAlignment: CrossAxisAlignment.start,
      children: [
        Row(
          mainAxisAlignment: MainAxisAlignment.spaceBetween,
          children: [
            const Text(
              'My Bags',
              style: TextStyle(
                  fontSize: 18,
                  fontWeight: FontWeight.bold,
                  fontFamily: 'ADLaMDisplay'),
            ),
            IconButton(
              onPressed: getBags,
              icon: const Icon(
                Icons.refresh,
                color: Colors.blue,
                size: 30,
              ),
            ),
          ],
        ),
        const SizedBox(height: 10),
        Container(
          width: screenSize.width * 0.8,
          height: screenSize.height * 0.3,
          decoration: BoxDecoration(
            border: Border.all(color: Colors.black),
            borderRadius: BorderRadius.circular(10),
          ),
          child: ListView.builder(
            itemCount: data.length,
            itemBuilder: (context, index) {
              String docId = data[index].id;
              int itemNumber = index + 1;
              String description = data[index]['description'];
              return GestureDetector(
                onTap: () {
                  getStatus(docId);
                  print(docId);
```

```
  print('Tapped on Bag $itemNumber: $description');
  for (var item in status) {
    print(item.data());
  }
},
child: ListTile(
  title: Text('Bag $itemNumber: $description'),
  trailing: IconButton(
    icon: const Icon(
      Icons.delete,
      color: Colors.red,
    ),
    onPressed: () {
      showDialog(
        context: context,
        builder: (BuildContext context) {
          return AlertDialog(
            backgroundColor: const Color(0xFF01497C),
            shape: RoundedRectangleBorder(
              borderRadius: BorderRadius.circular(16.0),
            ),
            content: SingleChildScrollView(
              child: Padding(
                padding: const EdgeInsets.all(0),
                child: Column(
                  children: [
                    const Text(
'Are you sure you want to delete this bag?',
                      style: TextStyle(
                          color: Colors.white,
                          fontSize: 18,
                          fontWeight: FontWeight.bold,
                          fontFamily: 'ADLaMDisplay'),
                    ),
                    const SizedBox(height: 20),
                    Row(
                      mainAxisAlignment:
                          MainAxisAlignment.spaceBetween,
                      children: [
                        Container(
                          margin: const EdgeInsets.only(
                              left: 10),
                          child: TextButton(
                            style: TextButton.styleFrom(
                              backgroundColor:
```

```
                const Color(0xFF61A5C2),
            shape:
                RoundedRectangleBorder(
              borderRadius:
                  BorderRadius.circular(
                      8.0),
            ),
          ),
          child: const Text(
            'Cancel',
            style: TextStyle(
                color: Colors.white,
                fontSize: 14,
                fontFamily:
                    'ADLaMDisplay'),
          ),
          onPressed: () {
            Navigator.pop(context);
          },
        ),
      ),
      Container(
        margin: const EdgeInsets.only(
            right: 10),
        child: TextButton(
          style: TextButton.styleFrom(
            backgroundColor:
                const Color(0xFF61A5C2),
            shape:
                RoundedRectangleBorder(
              borderRadius:
                  BorderRadius.circular(
                      8.0),
            ),
          ),
          child: const Text(
            'Done',
            style: TextStyle(
                color: Colors.white,
                fontSize: 14,
                fontFamily:
                    'ADLaMDisplay'),
          ),
          onPressed: () async {
            try {
```

```
FirebaseFirestore
    firestore =
    FirebaseFirestore
        .instance;
String currentUserUid =
    FirebaseAuth.instance
        .currentUser!.uid;
DocumentSnapshot
    passengerDoc =
    await firestore
        .collection(
            'passenger')
        .doc(
            currentUserUid)
        .get();
if (passengerDoc.exists) {
  Map<String, dynamic>
      current =
      passengerDoc.data()
          as Map<String,
              dynamic>;
  String name = current[
      'currentflight'];
  DocumentReference
      bagRef = firestore
          .collection(
              'passengerflights')
          .doc(name)
          .collection(
              'bags')
          .doc(data[index]
              .id);
  await bagRef.delete();
  setState(() {
    data.removeAt(index);
  });
  Fluttertoast.showToast(
      msg:
          "Bag is Deleted!");
  Navigator.pop(context);
  }
} catch (e) {
  print(
      'Error deleting bag: $e');
  Fluttertoast.showToast(
```

```
                                                msg:
                                                    "Please try again!");
                                            }
                                          },
                                        ),
                                      )
                                    ],
                                  )
                                ],
                              ),
                            ),
                          ),
                        );
                      },
                    );
                  },
                ),
              ),
            );
          },
        ),
      ),
      const SizedBox(height: 20),
      const Text(
        'Status:',
        style: TextStyle(
            fontSize: 16,
            fontWeight: FontWeight.bold,
            fontFamily: 'ADLaMDisplay'),
      ),
      Container(
        height: screenSize.height * 0.3,
        child: ListView.builder(
          itemCount: status.length,
          itemBuilder: (context, index) {
            Map<String, dynamic> data =
                status[index].data() as Map<String, dynamic>;
            String location = data['location'];
            String time = data['time'];
            return Padding(
              padding: const EdgeInsets.symmetric(vertical: 4.0),
              child: ListTile(
                leading: const Icon(
                  Icons.circle,
                  size: 12,
```

```
                        color: Colors.green,
                      ),
                      title: Text(
                        location,
                        style: const TextStyle(fontSize: 14),
                      ),
                      subtitle: Text(
                        time,
                        style: const TextStyle(fontSize: 12),
                      ),
                    ),
                  );
                },
              ),
            ),
          ],
        ),
      ),
    ),
    floatingActionButton: Positioned(
      bottom: 20,
      right: 20,
      child: NextFButton(
        icon: Icons.add,
        onPressed: () {
          showDialog(
            context: context,
            builder: (BuildContext context) {
              return const NewBag();
            },
          );
        },
      ),
    ),
  );
 }
}
```

New Bag Screen

```
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:fluttertoast/fluttertoast.dart';

class NewBag extends StatefulWidget {
  const NewBag({super.key});

  @override
  State<NewBag> createState() => _NewBagState();
}

TextEditingController bagdesc = TextEditingController();
TextEditingController ticketno = TextEditingController();

Map<String, dynamic> flightInfo = {};

class _NewBagState extends State<NewBag> {
  GlobalKey<FormState> formstate = GlobalKey<FormState>();
  String currentUserUid = FirebaseAuth.instance.currentUser!.uid;
  CollectionReference bags = FirebaseFirestore.instance.collection('bags');

  Future<String> getUsernameName() async {
    try {
      String currentUserUid = FirebaseAuth.instance.currentUser!.uid;
      DocumentSnapshot documentSnapshot = await FirebaseFirestore.instance
          .collection('passenger')
          .doc(currentUserUid)
          .get();

      if (documentSnapshot.exists) {
        Map<String, dynamic> data =
            documentSnapshot.data() as Map<String, dynamic>;
        String name = data['username'];
        return name;
      } else {
        throw Exception("Document does not exist");
      }
    } catch (e) {
      print("Error: $e");
      throw e;
    }
  }
```

```
Future<void> addbag() async {
  try {
    FirebaseFirestore firestore = FirebaseFirestore.instance;
    String currentUserUid = FirebaseAuth.instance.currentUser!.uid;
    DocumentSnapshot passengerDoc =
        await firestore.collection('passenger').doc(currentUserUid).get();

    if (passengerDoc.exists) {
      Map<String, dynamic> current =
          passengerDoc.data() as Map<String, dynamic>;
      String name = current['currentflight'];
      DocumentReference passengerRef =
          firestore.collection('passengerflights').doc(name);
      await passengerRef
          .collection('bags')
          .add({"description": bagdesc.text});
      bagdesc.clear();
    } else {
      print('Passenger document not found for user: $currentUserUid');
    }
  } catch (e) {
    print('Error retrieving current flight: $e');
  }
}

@override
Widget build(BuildContext context) {
  final Size screenSize = MediaQuery.of(context).size;
  return AlertDialog(
    content: SingleChildScrollView(
      child: Container(
        height: screenSize.height * 0.5,
        width: screenSize.height * 0.7,
        child: Padding(
          padding: const EdgeInsets.all(10),
          child: Stack(
            children: [
              Column(
                children: [
                  Container(
                    alignment: Alignment.topLeft,
                    margin: const EdgeInsets.only(top: 20),
                    child: const Text(
                      'New Bag',
                      style: TextStyle(
```

```
                        fontSize: 26,
                        fontWeight: FontWeight.bold,
                        fontFamily: 'ADLaMDisplay'),
                  ),
                ),
                const SizedBox(height: 40),
                _roundedTextField('Enter Bag Description', Icons.luggage,
                    controller: bagdesc),
                const SizedBox(height: 22),
                ElevatedButton(
                  style: ElevatedButton.styleFrom(
                    padding: const EdgeInsets.symmetric(vertical: 0),
                    backgroundColor: const Color(0xFF01497C),
                    shape: RoundedRectangleBorder(
                      borderRadius: BorderRadius.circular(8.0),
                    ),
                    minimumSize: const Size(
                        130, 45),
                  ),
                  onPressed: () {
                    if (bagdesc.text.isNotEmpty) {
                      addbag();
                      Navigator.pop(context);
                    } else {
                      Fluttertoast.showToast(
                          msg: "Please Enter Description");
                    }
                  },
                  child: const Text(
                    'Enter',
                    style: TextStyle(
                        fontSize: 18,
                        color: Colors.white,
                        fontFamily: 'ADLaMDisplay'),
                  ),
                ),
              ],
            ),
          ],
        ),
      ),
    ),
  );
}
```

Support Screen

We provided an important feature to help need or concerns of our users.

```
import 'dart:async';
import 'package:flutter/material.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:intl/intl.dart';
import 'package:fluttertoast/fluttertoast.dart';

class ClientScreen extends StatefulWidget {
  const ClientScreen({super.key});

  @override
  _ClientScreenState createState() => _ClientScreenState();
}

class _ClientScreenState extends State<ClientScreen> {
  final TextEditingController messageController = TextEditingController();
  final List<Map<String, String>> clientMessages = [];
  final List<Map<String, String>> agentMessages = [];
  List<Map<String, String>> allMessages = [];
  final ScrollController _scrollController = ScrollController();
  late StreamSubscription clientMessagesSubscription;
  late StreamSubscription agentMessagesSubscription;

  @override
  void initState() {
    super.initState();
    fetchMessages();
  }
  @override
  void dispose() {
    clientMessagesSubscription.cancel();
    agentMessagesSubscription.cancel();
    super.dispose();
  }
  Future<void> sendMessage() async {
    try {
      FirebaseFirestore firestore = FirebaseFirestore.instance;
      String currentUserUid = FirebaseAuth.instance.currentUser!.uid;
      DateTime now = DateTime.now();
      String timestamp = DateFormat('dd/MM/yyyy hh:mm:ss.SSS a').format(now);
      DocumentReference clientMessagesRef = firestore
          .collection('messages')
```

```
        .doc(currentUserUid)
        .collection('client')
        .doc('$currentUserUid c');
    await clientMessagesRef.set({
      timestamp: messageController.text,
    }, SetOptions(merge: true));

    DocumentReference clientAttribute =
        firestore.collection('messages').doc(currentUserUid);
    await clientAttribute.set({"date": timestamp});
    messageController.clear();
  } catch (e) {
    print(e);
    Fluttertoast.showToast(msg: "An error occurred, please try again");
  }
}

void fetchMessages() {
  try {
    FirebaseFirestore firestore = FirebaseFirestore.instance;
    String currentUserUid = FirebaseAuth.instance.currentUser!.uid;

    clientMessagesSubscription = firestore
        .collection('messages')
        .doc(currentUserUid)
        .collection('client')
        .snapshots()
        .listen((snapshot) {
      if (!mounted) return;
      setState(() {
        clientMessages.clear();
        for (var doc in snapshot.docs) {
          Map<String, dynamic> data = doc.data();
          data.forEach((key, value) {
            clientMessages.add({key: value});
          });
        }
      });
    });
    agentMessagesSubscription = firestore
        .collection('messages')
        .doc(currentUserUid)
        .collection('agent')
        .snapshots()
        .listen((snapshot) {
```

```
      if (!mounted) return;
      setState(() {
        agentMessages.clear();
        for (var doc in snapshot.docs) {
          Map<String, dynamic> data = doc.data();
          data.forEach((key, value) {
            agentMessages.add({key: value});
          });
        }
      });
    });
  } catch (e) {
    print(e);
    Fluttertoast.showToast(msg: "An error occurred, please try again");
  }
}

Widget _roundedTextField(
    {TextEditingController? controller, required String hintText}) {
  return Container(
    margin: const EdgeInsets.symmetric(vertical: 5),
    decoration: BoxDecoration(
      borderRadius: BorderRadius.circular(10),
      color: Colors.grey[200],
    ),
    child: Padding(
      padding: const EdgeInsets.symmetric(horizontal: 16),
      child: Row(
        children: [
          Expanded(
            child: TextField(
              controller: controller,
              decoration: InputDecoration(
                hintText: hintText,
                border: InputBorder.none,
              ),
            ),
          ),
        ],
      ),
    ),
  );
}

@override
```

```
Widget build(BuildContext context) {
  allMessages = [...clientMessages, ...agentMessages];
  allMessages.sort((a, b) => a.keys.first.compareTo(b.keys.first));

  return Scaffold(
    appBar: AppBar(
      backgroundColor: Colors.green[700],
      shape: const RoundedRectangleBorder(
        borderRadius: BorderRadius.only(
          bottomLeft: Radius.circular(0.0),
          bottomRight: Radius.circular(20.0),
        ),
      ),
      title: const Text('Support'),
      titleTextStyle: const TextStyle(
          fontSize: 24,
          fontWeight: FontWeight.bold,
          fontFamily: 'ADLaMDisplay'),
      actions: [
        IconButton(
            onPressed: () {
              Navigator.pushReplacementNamed(context, '/menu');
            },
            icon: const Icon(
              Icons.arrow_back_ios_new,
              color: Colors.white,
            ))
      ],
    ),
    body: Column(
      children: [
        Expanded(
          child: ListView.builder(
            controller: _scrollController,
            itemCount: allMessages.length,
            itemBuilder: (context, index) {
              bool isClientMessage = clientMessages
                  .map((msg) => msg.keys.first)
                  .contains(allMessages[index].keys.first);

              return Container(
                alignment: isClientMessage
                    ? Alignment.centerRight
                    : Alignment.centerLeft,
                padding:
```

```
                    const EdgeInsets.symmetric(horizontal: 10, vertical: 5),
                child: Container(
                  decoration: BoxDecoration(
                    color: isClientMessage ? Colors.blue : Colors.grey,
                    borderRadius: BorderRadius.circular(10),
                  ),
                  padding: const EdgeInsets.all(10),
                  child: Text(
                    allMessages[index].values.first,
                    style: const TextStyle(color: Colors.white),
                  ),
                ),
              );
            },
          ),
        ),
        Padding(
          padding: const EdgeInsets.all(8.0),
          child: Row(
            children: [
              Expanded(
                child: _roundedTextField(
                  hintText: 'Type your message',
                  controller: messageController,
                ),
              ),
              IconButton(
                icon: Icon(Icons.send, color: Colors.green[700]),
                onPressed: () {
                  if (messageController.text != "") {
                    sendMessage();
                  }
                },
              ),
            ],
          ),
        ),
      ],
    ),
  );
 }
}
```

Profile Screen

```
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:Travello/widget/backfloatingbutton.dart';



class Profile extends StatefulWidget {
  const Profile({Key? key}) : super(key: key);

  @override
  _ProfileState createState() => _ProfileState();
}

class _ProfileState extends State<Profile> {
  bool _isExpanded = false;

  // Add three TextEditingController
  TextEditingController _nameController = TextEditingController();
  TextEditingController _emailController = TextEditingController();
  TextEditingController _password = TextEditingController();

  String nametext = "";
  String emailtext = "";

  @override
  void initState() {
    initializeData();
    super.initState();
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        backgroundColor: const Color(0xFF01497C),
        shape: const RoundedRectangleBorder(
          borderRadius: BorderRadius.only(
            bottomLeft: Radius.circular(0.0),
            bottomRight: Radius.circular(0.0),
          ),
        ),
        leading: GestureDetector(
          onTap: () {},
```

```
      child: Container(
        padding: const EdgeInsets.all(5),
        margin: const EdgeInsets.only(bottom: 5, left: 5),
        child: const CircleAvatar(
          radius: 20.0,
          backgroundImage: AssetImage('images/avatar.png'),
        ),
      ),
    ),
    title: FutureBuilder<String>(
      future: getName(),
      builder: (BuildContext context, AsyncSnapshot<String> snapshot) {
        if (snapshot.connectionState == ConnectionState.waiting) {
          return const Center(
            child: CircularProgressIndicator(
              valueColor: AlwaysStoppedAnimation<Color>(Colors.white),
            ),
          );
        } else if (snapshot.hasError) {
          return Text('Error: ${snapshot.error}');
        } else {
          return Text(
            snapshot.data!,
            style: const TextStyle(color: Colors.white),
          );
        }
      },
    ),
  ),
  body: Stack(
    children: [
      Container(
        color: const Color(0xFF01497C),
        padding: const EdgeInsets.all(16.0),
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            _roundedTextField(nametext, Icons.person,
                controller: _nameController),
            const SizedBox(height: 10),
            _roundedTextField(emailtext, Icons.email,
                controller: _emailController),
            const SizedBox(height: 10),
            _roundedTextField('Password', Icons.lock,
                obscureText: true, controller: _password),
```

```
                TextButton(
                  onPressed: () async {
                    await FirebaseAuth.instance.signOut();
                    Navigator.pushReplacementNamed(context, '/Login');
                  },
                  style: TextButton.styleFrom(
                    backgroundColor: Colors.white,
                    shape: RoundedRectangleBorder(
                      borderRadius: BorderRadius.circular(8.0),
                    ),
                  ),
                  child: const Text(
                    'SignOut',
                    style: TextStyle(
                        color: Colors.black, fontFamily: 'ADLaMDisplay'),
                  ),
                ),
              ],
            ),
          ),
          Positioned(
            bottom: 16.0,
            left: 16.0,
            child: BackFButton(
              onPressed: () {
                Navigator.pushReplacementNamed(context, '/menu');
              },
            ),
          ),
        ],
      ),
    );
  }
}
```

Navigation Screen

As our system based on BLE, first of all we needed to use a package to enable us start listening for surroundings ble devices and add them to a list. With the help of flutter_blue_package we could detect our fixed ble beacons fetching required data from advertised packets that help us in positioning process. Also we used fl_chart package to design a grid that emulates a real indoor map and plot points indicating position of the passenger.

```
import 'dart:async';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:flutter_blue_plus/flutter_blue_plus.dart';
import 'dart:math';
import 'package:fl_chart/fl_chart.dart';
import 'package:fluttertoast/fluttertoast.dart';
import 'package:intl/intl.dart';

class BluetoothScanner extends StatefulWidget {
  const BluetoothScanner({super.key});

  @override
  _BluetoothScannerState createState() => _BluetoothScannerState();
}

class _BluetoothScannerState extends State<BluetoothScanner> {
  List<String> locations = [
    'Gate 1',
    'Passport F1',
    'Luggage Office',
    'WC F1',
    'Food Court',
    'Security Office F1',
    'Gate 2',
    'Passport F2',
    'Food Court',
    'Security Office F2',
    'Information Office',
    'WC F2'
  ];
  List<ScanResult> scanResultList = [];
  List<FlSpot> spots = [];
  List<Point> trilaterationResults = [];
  List<Map> data = [];
  Timer? _timer;
  Timer? timerL;
```

```
var scanMode = 0;
var isCalculating = false;
var isScanning = false;
final mp = -40.0;
final List<String> macAddressesToTrack = [
  '40:22:D8:77:2E:26',
  '40:22:D8:77:26:E6',
  'A0:B7:65:61:74:2A'
];
var myController1 = TextEditingController();
var myController2 = TextEditingController();
var myController3 = TextEditingController();
var myController4 = TextEditingController();
var myController5 = TextEditingController();
List<List<double>> dataPointsList = [];
List<FlSpot> wc = [const FlSpot(4.5, 1.5)];
List<FlSpot> luggage = [const FlSpot(0.5, 1.2)];
List<FlSpot> spotsLocations = [];
String locationselected = "";
double pointX = 0;
double pointY = 0;
String imagepath = 'images/airport.png';
double d1 = 0;
double d2 = 0;
double d3 = 0;
List<FlSpot> luggageToPassport = [
  const FlSpot(0.5, 1.2),
  const FlSpot(1.5, 1.2),
  const FlSpot(1.5, 2.8),
];
List<FlSpot> luggageToGate1 = [
  const FlSpot(0.5, 1.2),
  const FlSpot(1.9, 1.2),
  const FlSpot(1.9, 4),
  const FlSpot(4, 4),
  const FlSpot(4, 4.55)
];
List<FlSpot> luggageToSecurity = [
  const FlSpot(0.5, 1.2),
  const FlSpot(1.9, 1.2),
  const FlSpot(1.9, 4),
  const FlSpot(0.7, 4),
  const FlSpot(0.7, 4.55)
];
List<FlSpot> luggageToWC = [
```

```
  const FlSpot(0.5, 1.2),
  const FlSpot(4.5, 1.2),
];
List<FlSpot> wcToPassport = [
  const FlSpot(4.5, 1.2),
  const FlSpot(1.9, 1.2),
  const FlSpot(1.9, 2.8),
];
List<FlSpot> wcToGate1 = [
  const FlSpot(4.5, 1.2),
  const FlSpot(1.9, 1.2),
  const FlSpot(1.9, 4),
  const FlSpot(4, 4),
  const FlSpot(4, 4.55)
];
List<FlSpot> wcToSecurity = [
  const FlSpot(4.5, 1.2),
  const FlSpot(1.9, 1.2),
  const FlSpot(1.9, 4),
  const FlSpot(0.7, 4),
  const FlSpot(0.7, 4.55)
];
List<FlSpot> passportToGate1 = [
  const FlSpot(1.5, 3.25),
  const FlSpot(1.5, 4),
  const FlSpot(4, 4),
  const FlSpot(4, 4.55)
];
List<FlSpot> passportToSecurity = [
  const FlSpot(1.5, 3.25),
  const FlSpot(1.5, 4),
  const FlSpot(0.7, 4),
  const FlSpot(0.7, 4.55)
];
List<FlSpot> securityToGate1 = [
  const FlSpot(0.7, 4.55),
  const FlSpot(0.7, 4),
  const FlSpot(4, 4),
  const FlSpot(4, 4.55)
];
List<FlSpot> route = [];
List<FlSpot> defaultRoute = [];

@override
void initState() {
```

```
  setLocation();
  super.initState();
  // Listen to scan results in the initState method
  FlutterBluePlus.scanResults.listen((results) {
    // Sort the list by RSSI
    results.sort((a, b) => b.rssi.compareTo(a.rssi));

    // Update state
    if (isScanning) {
      setState(() {
        scanResultList = results;
      });
    }
  });
}

@override
void dispose() {
  if (BluetoothAdapterState.unavailable == false) {
    FlutterBluePlus.stopScan();
  }
  _timer?.cancel();
  timerL?.cancel();
  super.dispose();
}

/* Start, Stop */
void toggleState() {
  isScanning = !isScanning;

  if (isScanning) {
    print("Start Scan");
    FlutterBluePlus.startScan(
        continuousUpdates: true,
        androidScanMode: const AndroidScanMode(0),
        androidUsesFineLocation: true);
  } else {
    print("Stop Scan");
    FlutterBluePlus.stopScan();
  }

  setState(() {});
}
/* Scan Mode */
void scan() async {
```

```
  if (isScanning) {
    scanResultList.clear();
  }
}
/* device TxPower */
Widget deviceTX(ScanResult r) {
  return Text('Tx: ${r.advertisementData.txPowerLevel.toString()}');
}

/* device RSSI */
Widget deviceSignal(ScanResult r) {
  return Text(r.rssi.toString());
}

/* device MAC address */
Widget deviceMacAddress(ScanResult r) {
  return Text(r.device.remoteId.str);
}

/* device name */
Widget deviceName(ScanResult r) {
  String name;

  if (r.device.platformName.isNotEmpty) {
    name = r.device.platformName;
  } else if (r.advertisementData.advName.isNotEmpty) {
    name = r.advertisementData.advName;
  } else {
    name = 'N/A';
  }
  return Text(name);
}

/* Calculate Distance */
double calculateDistance(double measuredPower, double rssi) {
  var distance = pow(10, ((measuredPower - rssi) / (10 * 2))).toDouble();
  return distance;
}

Point<double> trilateration(double d1, double d2, double d3) {
  // Known points in the room
  var p1 = const Point<double>(0, 0);
  var p2 = const Point<double>(5, 0);
  var p3 = const Point<double>(0, 5);
```

```
  var A = 2 * p2.x - 2 * p1.x;
  var B = 2 * p2.y - 2 * p1.y;
  var C = pow(d1, 2) -
      pow(d2, 2) -
      pow(p1.x, 2) +
      pow(p2.x, 2) -
      pow(p1.y, 2) +
      pow(p2.y, 2);
  var D = 2 * p3.x - 2 * p2.x;
  var E = 2 * p3.y - 2 * p2.y;
  var F = pow(d2, 2) -
      pow(d3, 2) -
      pow(p2.x, 2) +
      pow(p3.x, 2) -
      pow(p2.y, 2) +
      pow(p3.y, 2);

  var x = (C * E - F * B) / (E * A - B * D);
  var y = (C * D - A * F) / (B * D - A * E);

  return Point<double>(x, y);
}

/* BLE icon widget */
Widget leading(ScanResult r) {
  return const CircleAvatar(
    backgroundColor: Colors.cyan,
    child: Icon(
      Icons.bluetooth,
      color: Colors.white,
    ),
  );
}

void onTap(ScanResult r) {
  if (r.device.remoteId.str == '40:22:D8:77:2E:26') {
    if (scanResultList.contains(r)) {
      double d1 = calculateDistance(mp, r.rssi.toDouble());
      myController1.text = d1.toString();
    } else {}
  }
}

String? locationRanges(double x, double y) {
  if (x >= 0 && x <= 2 && y >= 0 && y <= 2) {
```

```
    return "Luggage";
  } else if (x >= 4 && x <= 5 && y >= 0 && y <= 3) {
    return "WC";
  } else if (x >= 2.5 && x <= 5 && y >= 3.1 && y <= 5) {
    return "Gate 1";
  } else if (x >= 0 && x <= 2.5 && y >= 2.1 && y <= 4) {
    return "Passport";
  } else if (x >= 0 && x <= 2.5 && y >= 4.1 && y <= 5) {
    return "Secuirty Office";
  } else if (x >= 2.5 && x <= 3.9 && y >= 0 && y <= 3) {
    return "Rest Area";
  } else {
    return null;
  }
}

List<FlSpot>? getDirection() {
  String? location = locationRanges(pointX, pointY);
  if ((location == "Luggage" && locationselected == "Passport F1") ||
      (location == "Passport" && locationselected == "Luggage Office")) {
    return luggageToPassport;
  } else if ((location == "Luggage" && locationselected == "Gate 1") ||
      (location == "Gate 1" && locationselected == "Luggage Office")) {
    return luggageToGate1;
  } else if ((location == "Luggage" && locationselected == "WC F1") ||
      (location == "WC" && locationselected == "Luggage Office")) {
    return luggageToWC;
  } else if ((location == "Luggage" &&
          locationselected == "Security Office") ||
      (location == "Security Office" &&
          locationselected == "Luggage Office")) {
    return luggageToSecurity;
  } else if ((location == "WC" && locationselected == "Passport F1") ||
      (location == "Passport" && locationselected == "WC F1")) {
    return wcToPassport;
  } else if ((location == "WC" && locationselected == "Gate 1") ||
      (location == "Gate 1" && locationselected == "WC F1")) {
    return wcToGate1;
  } else if ((location == "WC" && locationselected == "Security Office") ||
      (location == "Security Office" && locationselected == "WC F1")) {
    return wcToSecurity;
  } else if ((location == "Passport" && locationselected == "Gate 1") ||
      (location == "Gate 1" && locationselected == "Passport F1")) {
    return passportToGate1;
  } else if ((location == "Passport" &&
```

```
          locationselected == "Security Office") ||
      (location == "Security Office" && locationselected == "Passport F1")) {
    return passportToSecurity;
  } else if ((location == "Security Office" &&
          locationselected == "Gate 1") ||
      (location == "Gate 1" && locationselected == "Security Office")) {
    return securityToGate1;
  } else {
    return defaultRoute;
  }
}

Future<void> setLocation() async {
  timerL = Timer.periodic(const Duration(seconds: 20), (timer) async {
    try {
      if (spots.isNotEmpty) {
        String currentUserUid = FirebaseAuth.instance.currentUser!.uid;
        await FirebaseFirestore.instance
            .collection('passenger')
            .doc(currentUserUid)
            .update({
          "currentlocation": locationRanges(pointX, pointY),
          "x": pointX,
          "y": pointY
        });
        DateTime time = DateTime.now();
        String timestamp = DateFormat('dd/MM/yyyy hh:mm a').format(time);
        await FirebaseFirestore.instance
            .collection('passenger')
            .doc(currentUserUid)
            .collection('locations')
            .doc()
            .set({
          "location": locationRanges(pointX, pointY),
          "time": timestamp,
          "x": pointX,
          "y": pointY
        }, SetOptions(merge: true));
        print("Document updated successfully!");
      }
    } catch (e) {
      print("Error: $e");
    }
  });
}
```

```
void ss(ScanResult r) {
  if (scanResultList.contains(r)) {
    if (r.device.remoteId.str == '40:22:D8:77:2E:26') {
      setState(() {
        double d1 = double.parse(
            calculateDistance(mp, r.rssi.toDouble()).toStringAsFixed(2));
        myController1.text = d1.toString();
      });
    }

    if (r.device.remoteId.str == '40:22:D8:77:26:E6') {
      setState(() {
        double d2 = double.parse(
            calculateDistance(mp, r.rssi.toDouble()).toStringAsFixed(2));
        myController2.text = d2.toString();
      });
    }

    if (r.device.remoteId.str == 'A0:B7:65:61:74:2A') {
      setState(() {
        double d3 = double.parse(
            calculateDistance(mp, r.rssi.toDouble()).toStringAsFixed(2));
        myController3.text = d3.toString();
      });
    }

    if (myController1.text.isNotEmpty &&
        myController2.text.isNotEmpty &&
        myController3.text.isNotEmpty) {
      double d1 = double.parse(myController1.text);
      double d1near = double.parse(d1.toStringAsFixed(2));
      double d2 = double.parse(myController2.text);
      double d2near = double.parse(d2.toStringAsFixed(2));
      double d3 = double.parse(myController3.text);
      double d3near = double.parse(d3.toStringAsFixed(2));
      Point p = trilateration(d1, d2, d3);
      if (p.x.toDouble() <= 5 &&
          p.x.toDouble() >= 0 &&
          p.y.toDouble() <= 5 &&
          p.y.toDouble() >= 0) {
        setState(() {
```

```
          spots.clear();
          spots.add(FlSpot(p.x.toDouble(), p.y.toDouble()));
          pointX = p.x.toDouble();
          pointY = p.y.toDouble();
        });
      }
      myController4.text = p.x.toStringAsFixed(2);
      myController5.text = p.y.toStringAsFixed(2);
      double x = p.x.toDouble();
      double y = p.y.toDouble();
    }
  }
}

void calculaterealtimedist() {
  if (!isCalculating) {
    _timer = Timer.periodic(const Duration(seconds: 2), (timer) {
      if (scanResultList.isNotEmpty) {
        for (ScanResult scanResult in scanResultList) {
          ss(scanResult);
        }
      }
    });
    isCalculating = true;
  } else {
    _timer?.cancel();
    isCalculating = false;
  }
}

/* ble item widget */
Widget? listItem(ScanResult r) {
  // List of MAC addresses you want to display
  var allowedMacAddresses = [
    '40:22:D8:77:2E:26',
    '40:22:D8:77:26:E6',
    'A0:B7:65:61:74:2A'
  ];

  // Check if the MAC address is in the allowed list
  if (allowedMacAddresses.contains(r.device.remoteId.str)) {
    return ListTile(
      onTap: () => onTap(r),
      leading: leading(r),
      title: deviceName(r),
```

```
      subtitle: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          deviceMacAddress(r),
          deviceTX(r),
        ],
      ),
      trailing: deviceSignal(r),
    );
  } else {
    return Container();
  }
}

void navigationSwitch(String value) {
  switch (value) {
    case 'Gate 1':
      setState(() {
        spotsLocations.clear();
        spotsLocations.add(const FlSpot(4, 4.52));
        imagepath = 'images/airport.png';
      });
      break;
    case 'Passport F1':
      setState(() {
        spotsLocations.clear();
        spotsLocations.add(const FlSpot(1.5, 2.79));
        imagepath = 'images/airport.png';
      });
      break;
    case 'Luggage Office':
      setState(() {
        spotsLocations.clear();
        spotsLocations.add(const FlSpot(0.5, 1.2));
        imagepath = 'images/airport.png';
      });
      break;
    case 'WC F1':
      setState(() {
        spotsLocations.clear();
        spotsLocations.add(const FlSpot(4.5, 1.2));
        imagepath = 'images/airport.png';
      });
      break;
```

```
case 'Security Office F1':
  setState(() {
    spotsLocations.clear();
    spotsLocations.add(const FlSpot(0.7, 4.52));
    imagepath = 'images/airport.png';
  });
case 'Gate 2':
  setState(() {
    spotsLocations.clear();
    spotsLocations.add(const FlSpot(4, 4.52));
    imagepath = 'images/airport2.png';
  });
  break;
case 'Passport F2':
  setState(() {
    spotsLocations.clear();
    spotsLocations.add(const FlSpot(1.5, 2.79));
    imagepath = 'images/airport2.png';
  });
  break;
case 'Food Court':
  setState(() {
    spotsLocations.clear();
    spotsLocations.add(const FlSpot(0.7, 1.2));
    imagepath = 'images/airport2.png';
  });
  break;
case 'Security Office F2':
  setState(() {
    spotsLocations.clear();
    spotsLocations.add(const FlSpot(0.7, 4.52));
    imagepath = 'images/airport2.png';
  });
  break;
case 'Information Office':
  setState(() {
    spotsLocations.clear();
    spotsLocations.add(const FlSpot(3.5, 0.6));
    imagepath = 'images/airport2.png';
  });
case 'WC F2':
  setState(() {
    spotsLocations.clear();
    spotsLocations.add(const FlSpot(4.5, 1.2));
    imagepath = 'images/airport2.png';
```

```
      });
      break;
  }
}
/* UI */
@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: const Text('Navigation',
          style: TextStyle(
              color: Colors.white,
              fontWeight: FontWeight.bold,
              fontFamily: 'ADLaMDisplay')),
      backgroundColor: const Color(0xFF01497C),
      shape: const RoundedRectangleBorder(
        borderRadius: BorderRadius.only(
          bottomLeft: Radius.circular(0),
          bottomRight: Radius.circular(0),
        ),
      ),
      actions: [
        IconButton(
          onPressed: () {
            Navigator.pushReplacementNamed(context, '/menu');
          },
          icon: const Icon(
            Icons.arrow_back_ios_new_outlined,
            color: Colors.white,
          ),
        ),
      ],
    ),
    body: Container(
      decoration: BoxDecoration(
        image: DecorationImage(
          image: AssetImage(
              imagepath),
          fit: BoxFit.fill,
        ),
      ),
      child: LineChart(
        LineChartData(
          gridData: FlGridData(
            show: true,
```

```
    horizontalInterval: 0.5,
    verticalInterval: 0.5,
    drawVerticalLine: true,
    drawHorizontalLine: true,
    checkToShowHorizontalLine: (value) => true,
    checkToShowVerticalLine: (value) => true,
    getDrawingHorizontalLine: (value) {
      return const FlLine(
        color: Colors.blue,
        strokeWidth: 0.2,
      );
    },
    getDrawingVerticalLine: (value) {
      return const FlLine(
        color: Colors.blue,
        strokeWidth: 0.2,
      );
    },
  ),
  titlesData: const FlTitlesData(show: false),
  borderData: FlBorderData(show: true),
  minX: 0,
  maxX: 5,
  minY: 0,
  maxY: 5,
  lineBarsData: [
    LineChartBarData(
      spots: spots,
      isCurved: false,
      color: Colors.blue,
      dotData: const FlDotData(show: true),
      belowBarData: BarAreaData(show: false),
    ),
    LineChartBarData(
      spots: route,
      isCurved: false,
      color: Colors.green,
      dotData: const FlDotData(show: false),
      belowBarData: BarAreaData(show: false),
    ),
    LineChartBarData(
      spots: spotsLocations,
      isCurved: false,
      color: Colors.red,
      barWidth: 3,
```

```
            belowBarData: BarAreaData(show: false),
            isStrokeCapRound: true,
            preventCurveOverShooting: true,
          ),
        ],
      ),
    ),
  ),
),
floatingActionButtonLocation: FloatingActionButtonLocation.startFloat,
floatingActionButton: Stack(
  children: [
    Positioned(
      bottom: 20,
      left: 15,
      child: FloatingActionButton(
        heroTag: UniqueKey(),
        onPressed: () {
          if (BluetoothAdapterState.unavailable != true) {
            toggleState();
            calculaterealtimedist();
          } else {
            Fluttertoast.showToast(msg: "Please Enable Bluetooth");
          }
        },
        child: const Icon(Icons.navigation),
      ),
    ),
    Positioned(
      bottom: 20,
      right: 45,
      child: FloatingActionButton(
        heroTag: UniqueKey(),
        onPressed: () {
          showDialog(
            context: context,
            builder: (BuildContext context) {
              return AlertDialog(
                backgroundColor: const Color(0xFF01497C),
                content: SizedBox(
                  height: MediaQuery.of(context).size.height * 0.7,
                  width: MediaQuery.of(context).size.width * 0.7,
                  child: Padding(
                    padding: const EdgeInsets.all(10),
                    child: Column(
                      crossAxisAlignment: CrossAxisAlignment.start,
```

```
children: [
  Container(
    alignment: Alignment.topLeft,
    margin: const EdgeInsets.only(top: 20),
    child: Text(
      'Where to',
      style: TextStyle(
          color: Colors.orange[300],
          fontSize: 26,
          fontWeight: FontWeight.bold,
          fontFamily: 'ADLaMDisplay'),
    ),
  ),
  Expanded(
    child: ListView.builder(
      itemCount: locations.length,
      itemBuilder: (context, index) {
        return Column(
          children: <Widget>[
            GestureDetector(
              onTap: () {
                Navigator.pop(context);
                locationselected = locations[index];
                navigationSwitch(locationselected);
                if (pointX != 0 && pointY != 0) {
                  route = getDirection()!;
                }
              },
              child: ListTile(
                title: Text(
                  locations[index],
                  style: const TextStyle(
                    color: Colors
                        .white,
                    fontSize: 18,
                    fontFamily: 'ADLaMDisplay',
                  ),
                ),
              ),
            ),
            Container(
              margin: const EdgeInsets.symmetric(
                  horizontal: 16.0),
              height: 1.0,
              color: Colors.white.withOpacity(
```

```
                                        0.4),
                                  ),
                                ],
                              );
                            },
                          ),
                        ),
                      ],
                    ),
                  ),
                ),
              );
            },
          );
        },
        child: const Icon(Icons.place),
      ),
    ),
  ],
),
persistentFooterButtons: [
  Row(
    children: [
      Expanded(
        child: TextField(
          controller: myController1,
          readOnly: true,
          decoration: const InputDecoration(
            border: OutlineInputBorder(),
          ),
        ),
      ),
      const SizedBox(width: 10),
      Expanded(
        child: TextField(
          controller: myController2,
          readOnly: true,
          decoration: const InputDecoration(
            border: OutlineInputBorder(),
          ),
        ),
      ),
      const SizedBox(width: 10),
      Expanded(
        child: TextField(
```

```
                controller: myController3,
                readOnly: true,
                decoration: const InputDecoration(
                  border: OutlineInputBorder(),
                ),
              ),
            ),
            const SizedBox(width: 10),
            Expanded(
              child: TextField(
                controller: myController4,
                readOnly: true,
                decoration: const InputDecoration(border: OutlineInputBorder()),
              ),
            ),
            const SizedBox(width: 10),
            Expanded(
              child: TextField(
                controller: myController5,
                readOnly: true,
                decoration: const InputDecoration(border: OutlineInputBorder()),
              ),
            ),
          ],
        ),
      ],
    );
  }
}
```

Rounded TextField

```
  Widget _roundedTextField(String hintText, IconData icon,
      {bool obscureText = false, TextEditingController? controller}) {
    return Container(
      margin: const EdgeInsets.symmetric(vertical: 5),
      decoration: BoxDecoration(
        borderRadius: BorderRadius.circular(10),
        color: Colors.grey[200],
      ),
      child: Padding(
        padding: const EdgeInsets.symmetric(horizontal: 16),
        child: Row(
          children: [
            Expanded(
              child: TextField(
                controller: controller,
                obscureText: obscureText,
                decoration: InputDecoration(
                  hintText: hintText,
                  border: InputBorder.none,
                ),
              ),
            ),
            Icon(icon),
          ],
        ),
      ),
    );
  }
}
```

Backward FloatingButton

```dart
import 'package:flutter/material.dart';

class BackFButton extends StatelessWidget {
  final VoidCallback onPressed;
  final Color backgroundColor;
  final IconData icon;
  final Color iconColor;
  final double bottom;
  final double left;

  const BackFButton({
    Key? key,
    required this.onPressed,
    this.backgroundColor = Colors.black,
    this.icon = Icons.arrow_back_ios_new,
    this.iconColor = Colors.white,
    this.bottom = 16.0,
    this.left = 16.0,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Positioned(
      bottom: bottom,
      left: left,
      child: FloatingActionButton(
        onPressed: onPressed,
        backgroundColor: backgroundColor,
        shape: const CircleBorder(),
        child: Icon(
          icon,
          color: iconColor,
        ),
      ),
    );
  }
}
```

TextField

```dart
import 'package:flutter/material.dart';

class CustomTextWidget extends StatelessWidget {
  final String text;
  final Color color;
  final double fontSize;
  final FontWeight fontWeight;
  final String fontFamily;
  final double letterSpacing;
  final EdgeInsets margin;

  const CustomTextWidget({
    Key? key,
    required this.text,
    this.color = Colors.black,
    this.fontSize = 16,
    this.fontWeight = FontWeight.normal,
    this.fontFamily = 'ADLaMDisplay',
    this.letterSpacing = 0,
    this.margin = const EdgeInsets.all(0),
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Container(
      margin: margin,
      child: Text(
        text,
        style: TextStyle(
          color: color,
          fontSize: fontSize,
          fontWeight: fontWeight,
          fontFamily: fontFamily,
          letterSpacing: letterSpacing,
        ),
      ),
    );
  }
}
```

Menu Buttons

```
import 'package:flutter/material.dart';

class MenuButton extends StatelessWidget {
  final VoidCallback onPressed;
  final Color backgroundColor;
  final String text;
  final double fontSize;

  const MenuButton({
    Key? key,
    required this.onPressed,
    this.backgroundColor = Colors.black,
    this.text = '',
    this.fontSize = 16,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Container(
      width: double.infinity,
      child: ElevatedButton(
        onPressed: onPressed,
        style: ElevatedButton.styleFrom(
          backgroundColor: const Color(0xFF01497C),
          shape: RoundedRectangleBorder(
            borderRadius:
                BorderRadius.circular(10.0),
          ),
        ),
        child: Text(
          text,
          style: TextStyle(fontSize: fontSize, color: Colors.white,fontFamily:
'ADLaMDisplay'),
        ),
      ),
    );
  }
}
```

Next Floating Buttton

```dart
import 'package:flutter/material.dart';

class NextFButton extends StatelessWidget {
  final VoidCallback onPressed;
  final Color backgroundColor;
  final IconData icon;
  final Color iconColor;
  final double bottom;
  final double right;

  const NextFButton({
    Key? key,
    required this.onPressed,
    this.backgroundColor = Colors.black,
    this.icon = Icons.arrow_forward_ios,
    this.iconColor = Colors.white,
    this.bottom = 16.0,
    this.right = 16.0,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Positioned(
      bottom: bottom,
      right: right,
      child: FloatingActionButton(
        onPressed: onPressed,
        backgroundColor: backgroundColor,
        shape: const CircleBorder(),
        child: Icon(
          icon,
          color: iconColor,
        ),
      ),
    );
  }
}
```

Rounded Rectangle

```dart
import 'package:flutter/material.dart';

class RoundedRectangle extends StatelessWidget {
  final Color color;
  final double width;
  final double height;
  final double borderRadius;
  final EdgeInsets margin;
  final VoidCallback? onPressed;

  RoundedRectangle({
    this.color = Colors.blue,
    this.width = 20.0,
    this.height = 10.0,
    this.borderRadius = 10.0,
    this.margin = const EdgeInsets.all(0),
    this.onPressed,
  });

  @override
  Widget build(BuildContext context) {
    return Align(
      alignment: Alignment.bottomCenter,
      child: GestureDetector(
        onTap: onPressed,
        child: Container(
          margin: margin,
          width: width,
          height: height,
          decoration: BoxDecoration(
            color: color,
            borderRadius: BorderRadius.circular(borderRadius),
          ),
        ),
      ),
    );
  }
}
```

Popup Messages

Bluetooth Check Message

```
import 'dart:async';
import 'dart:io';
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';
import 'package:flutter_blue_plus/flutter_blue_plus.dart';
import 'package:permission_handler/permission_handler.dart';

class BluetoothCheck {
  static bool isDialogShown = false;
  static void startBluetoothCheck(BuildContext context) {
    _checkBluetooth(context);
  }

  static void _checkBluetooth(BuildContext context) async {
    const Duration checkInterval = Duration(seconds: 3);

    final BluetoothAdapterState state =
        await FlutterBluePlus.adapterState.first;

    if (!isDialogShown &&
        (state == BluetoothAdapterState.off ||
            state == BluetoothAdapterState.unknown ||
            state == BluetoothAdapterState.unavailable)) {
      isDialogShown = true;

      showDialog(
        context: context,
        barrierDismissible: false,
        builder: (BuildContext dialogContext) {
          return WillPopScope(
            onWillPop: () async => false,
            child: AlertDialog(
              backgroundColor: const Color(0xFF01497C),
              shape: RoundedRectangleBorder(
                borderRadius: BorderRadius.circular(16.0),
              ),
              content: SingleChildScrollView(
                child: Padding(
                  padding: const EdgeInsets.all(0),
                  child: Column(
                    children: [
```

```
const Text(
  'Bluetooth is turned off.\nTurn on Bluetooth to use the app.',
  style: TextStyle(
      color: Colors.white,
      fontSize: 18,
      fontWeight: FontWeight.bold,
      fontFamily: 'ADLaMDisplay'),
),
const SizedBox(height: 20),
Row(
  mainAxisAlignment: MainAxisAlignment.spaceBetween,
  children: [
    Container(
      margin: EdgeInsets.only(left: 10),
      child: TextButton(
        style: TextButton.styleFrom(
          backgroundColor: const Color(0xFF61A5C2),
          shape: RoundedRectangleBorder(
            borderRadius: BorderRadius.circular(8.0),
          ),
        ),
        child: const Text(
          'Cancel',
          style: TextStyle(
              color: Colors.white,
              fontSize: 14,
              fontFamily: 'ADLaMDisplay'),
        ),
        onPressed: () {
          SystemNavigator.pop();
        },
      ),
    ),
    Container(
      margin: EdgeInsets.only(right: 10),
      child: TextButton(
        style: TextButton.styleFrom(
          backgroundColor: const Color(0xFF61A5C2),
          shape: RoundedRectangleBorder(
            borderRadius: BorderRadius.circular(8.0),
          ),
        ),
        child: const Text(
          'Done',
          style: TextStyle(
```

```
                                color: Colors.white,
                                fontSize: 14,
                                fontFamily: 'ADLaMDisplay'),
                          ),
                          onPressed: () async {
                             final BluetoothAdapterState state1 =

                                await FlutterBluePlus.adapterState.first;
                            if (state1 == BluetoothAdapterState.off ||
                                state1 ==
                                    BluetoothAdapterState.unavailable ||
                                state1 == BluetoothAdapterState.unknown) {
                              Fluttertoast.showToast(
                                  msg: 'Please Enable Bluetooth');
                            } else {
                              Navigator.pop(context);
                              isDialogShown = false;
                            }

                            Navigator.pop(context);
                          },
                        ),
                      ),
                    ],
                  ),
                ],
              ),
            ),
          ),
        ),
      );
    }

    Future.delayed(checkInterval, () => _checkBluetooth(context));
  }
}
```

Internet Connection Check Message

```
import 'dart:async';
import 'package:flutter/material.dart';
import 'package:http/http.dart' as http;

class InternetCheck {
  static bool isDialogShown = false;

  Future<bool> checkInternetConnection() async {
    try {
      final response = await http
          .get(Uri.parse('https://www.google.com'))
          .timeout(Duration(seconds: 5));
      if (response.statusCode == 200) {
        return true;
      } else {
        return false;
      }
    } catch (e) {
      return false;
    }
  }
  void checkConnection(BuildContext context) async {
    const Duration checkInterval = Duration(seconds: 3);
    bool isConnected = await checkInternetConnection();
    if (isConnected) {
      print("Connected to the Internet");
    } else {
      print("No Internet Connection");
      if (!isDialogShown) {
        isDialogShown = true;
        showDialog(
          context: context,
          barrierDismissible: false,
          builder: (BuildContext dialogContext) {
            return WillPopScope(
              onWillPop: () async => false,
              child: AlertDialog(
                backgroundColor: const Color(0xFF01497C),
                shape: RoundedRectangleBorder(
                  borderRadius: BorderRadius.circular(16.0),
                ),
                content: SingleChildScrollView(
                  child: Padding(
```

```
                    padding: const EdgeInsets.all(0),
                    child: Column(
                      children: [
                        const Text(
                          'No Internet Connection.',
                          style: TextStyle(color: Colors.white, fontSize: 18,
                              fontWeight: FontWeight.bold,
                              fontFamily: 'ADLaMDisplay'),
                        ),
                        const SizedBox(height: 20),
                        Row(
                          mainAxisAlignment: MainAxisAlignment.center,
                          children: [
                            Container(
                              margin: EdgeInsets.only(right: 10),
                              child: TextButton(
                                style: TextButton.styleFrom(
                                  backgroundColor: const Color(0xFF61A5C2),
                                  shape: RoundedRectangleBorder(
                                    borderRadius: BorderRadius.circular(8.0),
                                  ),
                                ),
                                child: const Text(
                                  'Retry',
                                  style: TextStyle(color: Colors.white,fontSize: 14,
                                      fontFamily: 'ADLaMDisplay'),
                                ),
                                onPressed: () async {
                                  Navigator.pop(context);
                                  isDialogShown = false;
                                },
                              ),
                            ),
                          ],
                        ),
                      ],
                    ),
                  ),
                ),
              ),
            ),
          );
        }
    Future.delayed(checkInterval, () => checkConnection(context));
  }
}
```

7.2. Staff Implementation

Staff Menu

```
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:Travello/widget/staffmenubuttons.dart';

class StaffMenu extends StatefulWidget {
  const StaffMenu({Key? key}) : super(key: key);

  @override
  _StaffMenuState createState() => _StaffMenuState();
}
class _StaffMenuState extends State<StaffMenu> {
  @override
  void initState() {
    super.initState();
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        backgroundColor: const Color(0xFF01497C),
        shape: const RoundedRectangleBorder(
          borderRadius: BorderRadius.only(
            bottomLeft: Radius.circular(20.0),
            bottomRight: Radius.circular(20.0),
          ),
        ),
        title: const Text("Staff", style: TextStyle(color: Colors.white)),
        actions: [
          IconButton(
            onPressed: () async {
              await FirebaseAuth.instance.signOut();
              Navigator.pushReplacementNamed(context, '/Login');
            },
            icon: const Icon(
              Icons.logout,
              color: Colors.white,
            ),
          ),
        ],
      ),
```

```
      body: Container(
        margin: const EdgeInsets.only(left: 10, top: 30, right: 10),
        child: GridView(
          gridDelegate: const SliverGridDelegateWithFixedCrossAxisCount(
            crossAxisCount: 2,
            mainAxisExtent: 220,
            crossAxisSpacing: 20,
            mainAxisSpacing: 20,
          ),
          children: [
            SMenuButton(
              onPressed: () {
                Navigator.pushReplacementNamed(context, '/qrgen');
              },
              buttonText: "Qr Generator",
              iconData: Icons.qr_code,
            ),
            SMenuButton(
              onPressed: () {
                Navigator.pushReplacementNamed(context, '/staffscanner');
              },
              buttonText: "Bag Scanning",
              iconData: Icons.luggage,
            ),
            SMenuButton(
              onPressed: () {
                Navigator.pushReplacementNamed(context, '/agent');
              },
              buttonText: "Support",
              iconData: Icons.support_agent_rounded,
            ),
          ],
        ),
      ),
    );
  }
}
```

Qr Code Generator Screen

```
import 'package:flutter/material.dart';
import 'package:fluttertoast/fluttertoast.dart';
import 'package:Travello/widget/menubuttons.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:qr_flutter/qr_flutter.dart';

class QRGen extends StatefulWidget {
  const QRGen({Key? key}) : super(key: key);
  @override
  State<QRGen> createState() => _QRGenState();
}
class _QRGenState extends State<QRGen> {
  FirebaseFirestore firestore = FirebaseFirestore.instance;
  String docId = '';
  List<QueryDocumentSnapshot> data = [];
  TextEditingController ticketno = TextEditingController();
  String qrData = ''; // Variable to hold QR data

  Future<void> getBags() async {
    try {
      DocumentSnapshot flightDoc = await firestore
          .collection('passengerflights').doc(ticketno.text).get();
      if (flightDoc.exists) {
        docId = flightDoc.id;
        DocumentReference flightRef =
            firestore.collection('passengerflights').doc(ticketno.text);
        QuerySnapshot bagsQuery = await flightRef.collection('bags').get();
        setState(() {
          data = bagsQuery.docs;
        });
      } else {
        Fluttertoast.showToast(msg: "Incorrect Ticket or not exists");
      }
    } catch (e) {
      Fluttertoast.showToast(msg: "Incorrect Ticket or not exists");
    }
  }
  @override
  Widget build(BuildContext context) {
    final screenSize = MediaQuery.of(context).size;
    return Scaffold(
      appBar: AppBar(
        backgroundColor: const Color(0xFF01497C),
```

```
    shape: const RoundedRectangleBorder(
      borderRadius: BorderRadius.only(
        bottomLeft: Radius.circular(20.0),
        bottomRight: Radius.circular(20.0),
      ),
    ),
    title: Text(
      "Qr Generator",
      style: TextStyle(
        color: Colors.white,
      ),
    ),
    actions: [
      IconButton(
        onPressed: () {
          Navigator.pushReplacementNamed(context, '/staffmenu');
        },
        icon: const Icon(
          Icons.arrow_back_ios_new_outlined,
          color: Colors.white,
        ),
      ),
    ],
  ),
  body: SingleChildScrollView(
    child: Padding(
      padding: const EdgeInsets.all(8.0),
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.stretch,
        children: [
          _roundedTextField(
            "Enter Ticket Number",
            Icons.airplane_ticket,
            controller: ticketno,
          ),
          SizedBox(height: 10),
          Container(
            width: screenSize.width * 0.8,
            height: screenSize.height * 0.3,
            decoration: BoxDecoration(
              border: Border.all(color: Colors.black),
              borderRadius: BorderRadius.circular(10),
            ),
            child: ListView.builder(
              itemCount: data.length,
```

```
    itemBuilder: (context, index) {
      int itemNumber = index + 1;
      String description = data[index]['description'];
      String docid = data[index].id.toString();
      String qrid = '$docid,$docId';
      return ListTile(
        title: Text('Bag $itemNumber: $description'),
        trailing: IconButton(
          icon: Icon(Icons.print, color: Colors.blue),
          onPressed: () {
            setState(() {
              qrData = qrid;
            });
            showDialog(
              context: context,
              builder: (BuildContext context) {
                return AlertDialog(
                  title: Text('QR Code'),
                  content: Container(
                    height: 320,
                    width: 320,
                    child: QrImageView(
                      data: qrData,
                      version: QrVersions.auto,
                      size: 320,
                      gapless: false,
                    ),
                  ),
                );
              },
            );
          },
        ),
      );
    },
  ),
),
const SizedBox(
  height: 20,
),
Container(
  child: MenuButton(
      onPressed: () {
        if (ticketno.text != "") {
          getBags();
```

```
                    } else {
                      Fluttertoast.showToast(
                          msg: "Please Enter Ticket Number");
                    }
                  },
                  text: 'Get Bags'),
              )
            ],
          ),
        ),
      ),
    );
  }
```

Support Screen

```
import 'dart:async';
import 'package:flutter/material.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:intl/intl.dart';
import 'package:fluttertoast/fluttertoast.dart';

class AgentScreen extends StatefulWidget {
  const AgentScreen({super.key});

  @override
  _AgentScreenState createState() => _AgentScreenState();
}

class _AgentScreenState extends State<AgentScreen> {
  final TextEditingController messageController = TextEditingController();
  final List<Map<String, String>> clientMessages = [];
  final List<Map<String, String>> agentMessages = [];
  List<String> docIds = [];
  List<String> usernames = [];
  final ScrollController _scrollController = ScrollController();
  String docId = "";
  late Timer timer;
  int selectedIndex = -1;

  @override
  void initState() {
    super.initState();
    _getUsernames();
    timer = Timer.periodic(Duration(seconds: 2), (Timer t) => _getUsernames());
  }
```

```dart
@override
void dispose() {
  timer.cancel();
  super.dispose();
}

Future<void> sendMessage() async {
  try {
    FirebaseFirestore firestore = FirebaseFirestore.instance;
    DateTime now = DateTime.now();
    String timestamp = DateFormat('dd/MM/yyyy hh:mm:ss.SSS a').format(now);
    DocumentReference agentMessagesRef = firestore
        .collection('messages').doc(docId).collection('agent').doc('$docId a');
    await agentMessagesRef.set({
      timestamp: messageController.text,
    }, SetOptions(merge: true));
    messageController.clear();
  } catch (e) {
    print(e);
    Fluttertoast.showToast(msg: "Please select chat first");
  }
}

Future<List<String>> _getUsernames() async {
  final FirebaseFirestore _firestore = FirebaseFirestore.instance;
  try {
    QuerySnapshot querySnapshot =
        await _firestore.collection('messages').get();

    if (querySnapshot.docs.isEmpty) {
      print('No documents found in the "messages" collection.');
    } else {
      for (var doc in querySnapshot.docs) {
        String doci = doc.id;
        if (!docIds.contains(doci)) {
          docIds.add(doc.id);
        }
      }

      for (var docId in docIds) {
        DocumentSnapshot docSnapshot =
            await _firestore.collection('passenger').doc(docId).get();
        if (docSnapshot.exists) {
          var data = docSnapshot.data() as Map<String, dynamic>;
```

```
          if (data.containsKey('username')) {
            String user = data['username'];
            if (!usernames.contains(user)) {
              usernames.add(data['username']);
            }
          }
        }
      }
      setState(() {});
    }
  } catch (e) {
    print('Error getting document IDs or usernames: $e');
  }
  return usernames;
}

void fetchMessages(String docId) {
  try {
    FirebaseFirestore firestore = FirebaseFirestore.instance;
    firestore
        .collection('messages').doc(docId).collection('client').snapshots()
        .listen((snapshot) {
      if (!mounted) return;
      setState(() {
        clientMessages.clear();
        for (var doc in snapshot.docs) {
          Map<String, dynamic> data = doc.data();
          data.forEach((key, value) {
            clientMessages.add({key: value});
          });
        }
      });
    });
    firestore.collection('messages').doc(docId).collection('agent').snapshots()
        .listen((snapshot) {
      if (!mounted) return;
      setState(() {
        agentMessages.clear();
        for (var doc in snapshot.docs) {
          Map<String, dynamic> data = doc.data();
          data.forEach((key, value) {
            agentMessages.add({key: value});
          });
        }
      });
```

```
    });
  } catch (e) {
    print(e);
    Fluttertoast.showToast(msg: "An error occurred, please try again");
  }
}

Future<String> getDocIdByUsername(String username) async {
  FirebaseFirestore firestore = FirebaseFirestore.instance;
  QuerySnapshot querySnapshot = await firestore
      .collection('passenger').where('username', isEqualTo: username).get();
  if (querySnapshot.docs.isNotEmpty) {
    docId = querySnapshot.docs.first.id;
  } else {
    Fluttertoast.showToast(msg: "not found");
  }
  return docId;
}

Widget _roundedTextField(
    {TextEditingController? controller, required String hintText}) {
  return Container(
    margin: const EdgeInsets.symmetric(vertical: 5),
    decoration: BoxDecoration(
      borderRadius: BorderRadius.circular(10),
      color: Colors.grey[200],
    ),
    child: Padding(
      padding: const EdgeInsets.symmetric(horizontal: 16),
      child: Row(
        children: [
          Expanded(
            child: TextField(
              controller: controller,
              decoration: InputDecoration(
                hintText: hintText,
                border: InputBorder.none,
              ),
            ),
          ),
        ],
      ),
    ),
  );
}
```

```
@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      backgroundColor: Colors.green[700],
      shape: const RoundedRectangleBorder(
        borderRadius: BorderRadius.only(
          bottomLeft: Radius.circular(0.0),
          bottomRight: Radius.circular(20.0),
        ),
      ),
      title: const Text('Support'),
      titleTextStyle:
          const TextStyle(fontSize: 24, fontWeight: FontWeight.bold),
      actions: [
        IconButton(
          onPressed: () {
            Navigator.pushReplacementNamed(context, '/staffmenu');
          },
          icon: const Icon(
            Icons.arrow_back_ios_new,
            color: Colors.white,
          ),
        )
      ],
    ),

    body: Row(
      children: [
        Container(
          width: MediaQuery.of(context).size.width * 0.35,
          color: Colors.grey[300],
          child: ListView.separated(
            itemCount: usernames.length,
            separatorBuilder: (context, index) => Divider(color: Colors.grey),
            itemBuilder: (context, index) {
              return ListTile(
                title: Text(
                  usernames[index],
                  style: TextStyle(
                    fontFamily: 'ADLaMDisplay',
                    color:
                        index == selectedIndex ? Colors.blue : Colors.black,
                  ),
```

```
        ),
        onTap: () async {
          String selectedDocId =
              await getDocIdByUsername(usernames[index]);
          setState(() {
            docId = selectedDocId;
            selectedIndex = index;
          });
          fetchMessages(docId);
        },
      );
    },
  ),
),

Expanded(
  child: Column(
    children: [
      Expanded(
        child: ListView.builder(
          controller: _scrollController,
          itemCount: clientMessages.length + agentMessages.length,
          itemBuilder: (context, index) {
            List<Map<String, String>> allMessages = [
              ...clientMessages,
              ...agentMessages
            ];
            allMessages
                .sort((a, b) => a.keys.first.compareTo(b.keys.first));

            bool isClientMessage = clientMessages
                .map((msg) => msg.keys.first)
                .contains(allMessages[index].keys.first);

            return Container(
              alignment: isClientMessage
                  ? Alignment.centerRight
                  : Alignment.centerLeft,
              padding: const EdgeInsets.symmetric(
                  horizontal: 10, vertical: 5),
              child: Container(
                decoration: BoxDecoration(
                  color: isClientMessage ? Colors.blue : Colors.grey,
                  borderRadius: BorderRadius.circular(10),
                ),
```

```
                        padding: const EdgeInsets.all(10),
                        child: Text(
                          allMessages[index].values.first,
                          style: const TextStyle(color: Colors.white),
                        ),
                      ),
                    );
                  },
                ),
              ),
              Padding(
                padding: const EdgeInsets.all(8.0),
                child: Row(
                  children: [
                    Expanded(
                      child: _roundedTextField(
                        hintText: 'Type your message',
                        controller: messageController,
                      ),
                    ),
                    IconButton(
                      icon: Icon(Icons.send, color: Colors.green[700]),
                      onPressed: () {
                        if (messageController.text != "") {
                          sendMessage();
                        }
                      },
                    ),
                  ],
                ),
              ),
            ],
          ),
        ),
      ],
    ),
  );
 }
}
```

Scanning Screen

To be able to scan the qr attached to each bag we used a package offering this service flutter_barcode_scanner, We also setup configurations of firebase messaging service to send notification to the bag owner once it is scanned.

```
import 'dart:convert';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';
import 'package:flutter_barcode_scanner/flutter_barcode_scanner.dart';
import 'package:Travello/apis/access_token.dart';
import 'package:Travello/widget/staffmenubuttons.dart';
import 'package:intl/intl.dart';
import 'package:http/http.dart' as http;

class StaffScanner extends StatefulWidget {
  const StaffScanner({Key? key}) : super(key: key);

  @override
  _StaffScannerState createState() => _StaffScannerState();
}

class _StaffScannerState extends State<StaffScanner> {
  String? _selectedItem; // variable to store selected item in dropdown
  String? _scanResult;
  String? partOne;
  String? partTwo;
  FirebaseFirestore firestore = FirebaseFirestore.instance;

  Future<void> setLocation() async {
    try {
      if (_selectedItem != null &&
          _scanResult != null &&
          partOne != null &&
          partTwo != null) {
        DocumentReference partTwoDocRef = firestore
            .collection('passengerflights')
            .doc(partTwo)
            .collection('bags')
            .doc(partOne);

        DateTime now = DateTime.now();
        String timestamp = DateFormat('dd/MM/yyyy hh:mm a').format(now);

        await partTwoDocRef.collection('status').doc().set({
```

```
        "location": _selectedItem,
        "time":timestamp
      }, SetOptions(merge: true));

      // Fetching passengerId
      DocumentSnapshot partTwoDoc =
          await firestore.collection('passengerflights').doc(partTwo).get();
      String passengerId =
          (partTwoDoc.data() as Map<String, dynamic>)['passengerid'];
      print(passengerId);

      // Fetching bag description
      DocumentSnapshot partOneDoc = await firestore
          .collection('passengerflights')
          .doc(partTwo)
          .collection('bags')
          .doc(partOne)
          .get();
      String bagDescription =
          (partOneDoc.data() as Map<String, dynamic>)['description'];

      String deviceToken= await getToken(passengerId);
      print(deviceToken);

      AccessToken accessToken = AccessToken();
      String token = await accessToken.getaccesstoken();
      print(token);
      // Constructing notification content
      String notificationTitle = 'Luggage Scanned';
      String notificationContent =
          'Luggage: $bagDescription scanned at $_selectedItem';

      // Sending notification
      await sendNotification(
          token, deviceToken, notificationTitle,notificationContent);

      print('Location set successfully!');
    } else {
      print(
          'Error: _selectedItem, _scanResult, partOne, or partTwo is null.');
    }
  } catch (e) {
    print('Error setting location: $e');
  }
}
```

```
Future<void> sendNotification(
    String accessToken, String deviceToken, String title, String message) async {
  final url = Uri.parse(
      'https://fcm.googleapis.com/v1/projects/travello-2ed88/messages:send');
  final headers = {
    'Authorization': 'Bearer $accessToken',
    'Content-Type': 'application/json',
  };
  final body = {
    "message": {
      "token": deviceToken,
      "notification": {
        "title": title,
        "body": message,
      },
    },
  };

  final response =
      await http.post(url, headers: headers, body: jsonEncode(body));
  if (response.statusCode == 200) {
    print('Notification sent successfully');
  } else {
    print('Failed to send notification: ${response.body}');
  }
}

 Future<String> getToken(String uid) async {
  try {
    DocumentSnapshot documentSnapshot = await FirebaseFirestore.instance
        .collection('passenger')
        .doc(uid)
        .get();

    if (documentSnapshot.exists) {
      Map<String, dynamic> data =
          documentSnapshot.data() as Map<String, dynamic>;
      final token = data['token'];
      return token;
    } else {
      throw Exception("Document does not exist");
    }
  } catch (e) {
    print("Error: $e");
    throw e;
```

```
  }
}
Future<void> scanCode() async {
  String barcodeScanRes;
  try {
    barcodeScanRes = await FlutterBarcodeScanner.scanBarcode(
        "#ffffff", "cancel", true, ScanMode.QR);
  } on PlatformException {
    barcodeScanRes = "Failed";
  }
  setState(() {
    _scanResult = barcodeScanRes;
  });
  List<String> parts = barcodeScanRes.split(',');
  partOne = parts.isNotEmpty ? parts[0] : '';
  partTwo = parts.length > 1 ? parts[1] : '';
  setLocation();
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      backgroundColor: const Color(0xFF01497C),
      shape: const RoundedRectangleBorder(
        borderRadius: BorderRadius.only(
          bottomLeft: Radius.circular(20.0),
          bottomRight: Radius.circular(20.0),
        ),
      ),
      title: const Text("Bag Scanner", style: TextStyle(color: Colors.white,)),
      actions: [
        IconButton(
          onPressed: () {
            Navigator.pushReplacementNamed(context, '/staffmenu');
          },
          icon: const Icon(
            Icons.arrow_back_ios_new_outlined,
            color: Colors.white,
          ),
        ),
      ],
    ),
    body: Column(
      children: [
```

```
            // Dropdown list
            Container(
              padding: const EdgeInsets.all(16.0),
              child: DropdownButtonFormField<String>(
                value: _selectedItem,
                onChanged: (String? newValue) {
                  setState(() {
                    _selectedItem = newValue;
                  });
                },
                items: <String>[
                  'Security Checking',
                  'Sorting Area',
                  'Loading Area',

                ].map<DropdownMenuItem<String>>((String value) {
                  return DropdownMenuItem<String>(
                    value: value,
                    child: Text(value),
                  );
                }).toList(),
                decoration: const InputDecoration(
                  labelText: 'Select Area',
                  border: OutlineInputBorder(),
                ),
              ),
            ),
            Center(
              child: Container(
                alignment: Alignment.center,
                child: SMenuButton(
                  onPressed: () {
                    scanCode();
                  },
                  buttonText: "SCAN",
                  iconData: Icons.qr_code, // No icon provided
                ),
              ),
            ),
          ],
        ),
      );
    }
}
```

7.3. Enhancement

Machine learning

KNN (K Nearest Neighbor)

Import data and libraries

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import rc
from sklearn.neighbors import KNeighborsClassifier
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import make_blobs
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import StandardScaler
```

Read data

```
data = pd.read_csv("points_train.csv")
```

extract feture from data

```
y=data[['x', 'y']]
x=data.drop(['x','y'],axis=1)
```

call train_test_split function

```
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)
```

Using KNN predicted xy

```
model=KNeighborsClassifier(n_neighbors=3)
model.fit(X_train,y_train)
inp=input()
inputlist=[float(value) for value in inp.split(",")]
results=model.predict([inputlist])
print({results})
```

Apply for any experiments

do this code with change the data file

```
import pandas as pd
from sklearn.preprocessing import StandardScaler
```

```
from sklearn.neighbors import KNeighborsRegressor
# Load your training and test data
test_data = pd.read_csv("Estimated_dataset.csv")
train_data = pd.read_csv("modified_dataset.csv")

# Separate features (X_train) and target variables (y_train) for training data
X_train = train_data.drop(['x', 'y'], axis=1)  # Features (drop 'X' and 'Y')
y_train = train_data[['x', 'y']]  # Target variables are 'X' and 'Y'

# Separate features (X_test) for testing data
X_test = test_data.drop(['X', 'Y'], axis=1)  # Test features (drop 'x' and 'y')

# Scale the features using StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Train the K-Nearest Neighbors (KNN) Regressor model
knn_regressor = KNeighborsRegressor(n_neighbors=3)
knn_regressor.fit(X_train_scaled, y_train)

# Predict using the trained model on the test data
y_pred = knn_regressor.predict(X_test_scaled)

# Create a DataFrame for the predictions with 'x' and 'y' values
solution_df = pd.DataFrame({
    'X': y_pred[:, 0],  # Predicted 'X' values
    'Y': y_pred[:, 1],  # Predicted 'Y' values
})

# Save the solution DataFrame to a CSV file
solution_df.to_csv("sol_expermint3333.csv", index=False)

# Display the first few rows of the solution DataFrame
print("Solution DataFrame:")
print(solution_df.head())
```

clean the output

```
df = pd.read_csv("outputfile.csv")

# Function to round decimal numbers to two decimal places in each cell
def round_to_two_decimals(value):
    if isinstance(value, float):
```

```
        return round(value, 2)
    else:
        return value

# Apply rounding function to each cell in the DataFrame
df_rounded = df.applymap(round_to_two_decimals)

# Save the rounded dataset to a new CSV file
output_file_path = "newname_clean.csv"
df_rounded.to_csv(output_file_path, index=False)

print(f"Rounded dataset saved to '{output_file_path}'.")

# Display the first few rows of the rounded dataset
print("\nFirst few rows of Rounded Dataset:")
print(df_rounded.head())
```

calculate

`MSE` & `RMSE` for any exp

```
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import mean_squared_error

# Calculate Mean Squared Error (MSE) for both 'X' and 'Y'
mse_x = mean_squared_error(y_test['X'], y_pred[:, 0])
mse_y = mean_squared_error(y_test['Y'], y_pred[:, 1])

# Calculate Root Mean Squared Error (RMSE) for both 'X' and 'Y'
rmse_x = mse_x ** 0.5
round(rmse_x, 2)
rmse_y = mse_y ** 0.5
round(rmse_y, 2)

# Print the accuracy metrics
print(f"RMSE for 'X': {rmse_x}")
print(f"RMSE for 'Y': {rmse_y}")
```

Convert code to DLL and connected with flutter

Step 1: Create a Cython File

```
import numpy as np
from sklearn.neighbors import KNeighborsClassifier
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Load and train the model when the module is loaded
data = pd.read_csv("points_train.csv")
y = data[['x', 'y']]
x = data.drop(['x', 'y'], axis=1)
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2,
random_state=42)
model = KNeighborsClassifier(n_neighbors=3)
model.fit(X_train, y_train)

def predict_points(values):
    inputlist = [float(value) for value in values.split(",")]
    results = model.predict([inputlist])
    return results[0][0], results[0][1]
```

Save this as example.pyx.
Step 2: Create the Setup Script

Create setup.py to build the DLL:

```
from distutils.core import setup
from Cython.Build import cythonize

setup(
    ext_modules=cythonize("example.pyx")
)
```

Build the DLL:

```
Open bash
python setup.py build_ext –inplace
```

Step 3: the Windows Plugin Code

the C++ code in the Flutter plugin to call the updated function with parameters.

Install `flutter_dll_plugin`

```
#include <windows.h>
#include <flutter/method_channel.h>
#include <flutter/plugin_registrar_windows.h>
#include <memory>
#include <sstream>
#include <vector>
namespace flutter_dll_plugin {

typedef std::pair<double, double> (*PredictPointsFunction)(const char*);

class FlutterDllPlugin : public flutter::Plugin {
```

```
 public:
  static void RegisterWithRegistrar(flutter::PluginRegistrarWindows *registrar);

  FlutterDllPlugin();
  virtual ~FlutterDllPlugin();

 private:
  HMODULE dll_handle_;
  PredictPointsFunction predict_points_;

  // Called when a method is called on this plugin's channel from Dart.
  void HandleMethodCall(
      const flutter::MethodCall<flutter::EncodableValue>& method_call,
      std::unique_ptr<flutter::MethodResult<flutter::EncodableValue>> result);
};

// static
void FlutterDllPlugin::RegisterWithRegistrar(
    flutter::PluginRegistrarWindows *registrar) {
  auto channel = std::make_unique<flutter::MethodChannel<flutter::EncodableValue>>(
      registrar->messenger(), "flutter_dll_plugin",
      &flutter::StandardMethodCodec::GetInstance());

  auto plugin = std::make_unique<FlutterDllPlugin>();

  channel->SetMethodCallHandler(
      [plugin_pointer = plugin.get()](const auto& call, auto result) {
        plugin_pointer->HandleMethodCall(call, std::move(result));
      });

  registrar->AddPlugin(std::move(plugin));
}

FlutterDllPlugin::FlutterDllPlugin() {
  // Load the DLL
  dll_handle_ = LoadLibrary(L"example.pyd");
  if (dll_handle_) {
    // Get the function address
    predict_points_ = (PredictPointsFunction)GetProcAddress(dll_handle_,
"predict_points");
  }
}

FlutterDllPlugin::~FlutterDllPlugin() {
  if (dll_handle_) {
    FreeLibrary(dll_handle_);
  }
}

void FlutterDllPlugin::HandleMethodCall(
    const flutter::MethodCall<flutter::EncodableValue>& method_call,
    std::unique_ptr<flutter::MethodResult<flutter::EncodableValue>> result) {
  if (method_call.method_name().compare("predictPoints") == 0) {
    if (predict_points_) {
      const auto* arguments =
std::get_if<flutter::EncodableMap>(method_call.arguments());
      if (!arguments) {
```

```
        result->Error("BAD_ARGS", "Expected map with one string value.");
        return;
      }

      std::string values =
std::get<std::string>((*arguments).at(flutter::EncodableValue("values")));

      auto response = predict_points_(values.c_str());
      std::vector<double> result_vector = { response.first, response.second };
      result->Success(flutter::EncodableValue(result_vector));
    } else {
      result->Error("DLL_ERROR", "Could not find function in DLL");
    }
  } else {
    result->NotImplemented();
  }
}

}  // namespace flutter_dll_plugin

void FlutterDllPluginRegisterWithRegistrar(
    FlutterDesktopPluginRegistrarRef registrar) {
  flutter_dll_plugin::FlutterDllPlugin::RegisterWithRegistrar(
      flutter::PluginRegistrarManager::GetInstance()
          ->GetRegistrar<flutter::PluginRegistrarWindows>(registrar));
}
```

Step 4: Use the Plugin in Your Flutter Application

1. **Modify the Dart code to handle the updated response:**

Here's the Dart code to call the plugin method with the required parameter and handle the two values returned:

```
import 'package:flutter/material.dart';
import 'package:flutter_dll_plugin/flutter_dll_plugin.dart';

void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: MyHomePage(),
    );
  }
}

class MyHomePage extends StatefulWidget {
  @override
  _MyHomePageState createState() => _MyHomePageState();
}
```

```
class _MyHomePageState extends State<MyHomePage> {
  String _response = 'Response will appear here';
  final TextEditingController _controller = TextEditingController();

  void _callDll() async {
    final response = await FlutterDllPlugin.predictPoints(_controller.text);
    setState(() {
      _response = 'Predicted coordinates: ${response[0]}, ${response[1]}';
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Flutter DLL Example'),
      ),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            TextField(
              controller: _controller,
              decoration: InputDecoration(
                hintText: 'Enter values separated by commas',
              ),
            ),
            SizedBox(height: 20),
            Text(_response),
            SizedBox(height: 20),
            ElevatedButton(
              onPressed: _callDll,
              child: Text('Call DLL'),
            ),
          ],
        ),
      ),
    );
  }
}
```

2. Update the plugin Dart code to handle the parameter and return the two values:

Add a method in `flutter_dll_plugin.dart` to call the native method with the required parameter:

```
import 'dart:async';
import 'package:flutter/services.dart';

class FlutterDllPlugin {
  static const MethodChannel _channel = MethodChannel('flutter_dll_plugin');

  static Future<List<double>> predictPoints(String values) async {
    final List response = await _channel.invokeMethod('predictPoints', {
      'values': values,
    });
    return response.cast<double>();}}
```

Pubspec.yaml File

```
name: Travello
description: "A new Flutter project."
publish_to: 'none'
version: 0.1.0
environment:
  sdk: '>=3.3.0 <4.0.0'
dependencies:
  cloud_firestore: ^4.15.5
  firebase_auth: ^4.17.5
  firebase_core: ^2.25.4
  firebase_messaging: ^14.9.4
  fl_chart: ^0.66.2
  flutter:
    sdk: flutter
  flutter_barcode_scanner: ^2.0.0
  flutter_blue_plus: ^1.31.15
  fluttertoast: ^8.2.4
  http: ^1.2.1
  intl: ^0.19.0
  lottie: ^3.1.0
  permission_handler: ^11.3.0
  qr_flutter: ^4.1.0

dev_dependencies:
  flutter_test:
    sdk: flutter
  flutter_lints: ^3.0.0
flutter:
  uses-material-design: true
  assets:
    - lib/raw/splash2.json
    - images/Plane.png
    - images/preview.jpg
    - images/People.png
    - images/Luggage.png
    - images/avatar.png
    - images/airport.png
    - images/airport2.png
    - images/Travello.png
  fonts:
    - family: ADLaMDisplay
      fonts:
   7.3.1.  asset: fonts/ADLaMDisplay-Regular.ttf
```

7.4. Control Room Implementation

NuGet Packages Used:
FirebaseAdmin v3.0.0
Google.Cloud.Firestore v3.7.0
ScottPlot.WPF v5.0.35
The libraries that used:

```
using System;
using System.Collections.Generic;
using System.Windows;
using System.Windows.Controls;
using Google.Cloud.Firestore;
using FirebaseAdmin;
using Google.Apis.Auth.OAuth2;
using System.Windows.Media;
using System.Threading.Tasks;
```

When the window is built, the function initializes necessary components and sets up the initial state of the interface, ensuring all elements are ready and functional upon display.

```
    public partial class MainWindow : Window
    {
        public FirestoreChangeListener listener;
        DispatcherTimer timer = new DispatcherTimer();

        public MainWindow()
        {
            InitializeComponent();
            InitializeFirestore();
            LoadInitialData();
            WatchForChanges();
            InitializeMap();
            InitializeMap2();
        }
```

Initialize Passengers Map

```
        public void InitializeMap()
        {
            AddMapArea2(new double[] { 0, 5, 5, 0, 0 }, new double[] { 0, 0, 5, 5, 0 });
// All map Corners
            AddMapArea2(new double[] { 0, 2, 2, 0, 0 }, new double[] { 0, 0, 2, 2, 0 });
// Luggage Area
            AddMapArea2(new double[] { 4, 5, 5, 4, 4 }, new double[] { 0, 0, 3, 3, 0 });
// WC (Restroom) Area
            AddMapArea2(new double[] { 2.5, 5, 5, 2.5, 2.5 }, new double[] { 3.1, 3.1, 5,
5, 3.1 }); // Gate1
            AddMapArea2(new double[] { 0, 2.5, 2.5, 0, 0 }, new double[] { 2.1, 2.1, 4, 4,
2.1 }); // Passport Area
            AddMapArea2(new double[] { 0, 2.5, 2.5, 0, 0 }, new double[] { 4.1, 4.1, 5, 5,
4.1 }); // Security Office Area
            AddMapArea2(new double[] { 2.5, 3.9, 3.9, 2.5, 2.5 }, new double[] { 0, 0, 3,
3, 0 }); // Rast Area
        }
        private void AddMapArea2(double[] dataX, double[] dataY)
        {
```

```
            var myScatter = WpfPlot1.Plot.Add.Scatter(dataX, dataY,
ScottPlot.Colors.LightBlue);
            myScatter.LineWidth = 2;
            myScatter.MarkerSize = 0;
        }
```

Initialize Staff Map

```
        public void InitializeMap2()
        {
AddMapArea(new double[] { 0, 5, 5, 0, 0 }, new double[] { 0, 0, 5, 5, 0 }); // All map
Corners
 AddMapArea(new double[] { 0, 2, 2, 0, 0 }, new double[] { 0, 0, 2, 2, 0 }); // Luggage
Area
 AddMapArea(new double[] { 4, 5, 5, 4, 4 }, new double[] { 0, 0, 3, 3, 0 }); // WC
(Restroom) Area
 AddMapArea(new double[] { 2.5, 5, 5, 2.5, 2.5 }, new double[] { 3.1, 3.1, 5, 5, 3.1 });
// Gate1
 AddMapArea(new double[] { 0, 2.5, 2.5, 0, 0 }, new double[] { 2.1, 2.1, 4, 4, 2.1 }); //
Passport Area
 AddMapArea(new double[] { 0, 2.5, 2.5, 0, 0 }, new double[] { 4.1, 4.1, 5, 5, 4.1 }); //
Security Office Area
 AddMapArea(new double[] { 2.5, 3.9, 3.9, 2.5, 2.5 }, new double[] { 0, 0, 3, 3, 0 }); //
Rast Area
        }


        private void AddMapArea(double[] dataX, double[] dataY)
        {
                var myScatter = WpfPlot2.Plot.Add.Scatter(dataX, dataY,
ScottPlot.Colors.LightBlue);
                myScatter.LineWidth = 2;
                myScatter.MarkerSize = 0;
        }
```

"Functions that listen for changes in the 'passengers' collection to retrieve real-time updates and plot each passenger's new location on the map."

```
        private void WatchForChanges()
        {
            CollectionReference pass = _database.Collection("passenger");
            Query query = _database.Collection("passenger");

            listener = query.Listen(snapshot =>
            {
                Dispatcher.Invoke(() =>
                {
                    LoadInitialData();
                });
            });
        }
```

"Buttons that toggle visibility of grids, enabling users to show or hide specific grids with a single click for enhanced display customization."

```
        private void Button_Click_1(object sender, RoutedEventArgs e)
        {
            LiveView.Visibility = Visibility.Visible;
            FlightsView.Visibility = Visibility.Hidden;
            NotiSenderViwe.Visibility = Visibility.Hidden;
            BagsView.Visibility = Visibility.Hidden;
            StaffView.Visibility = Visibility.Hidden;
            InitializeMap();
        }

        private void Button_Click_2(object sender, RoutedEventArgs e)
        {
            StaffView.Visibility = Visibility.Hidden;
            LiveView.Visibility = Visibility.Hidden;
            FlightsView.Visibility = Visibility.Visible;
            NotiSenderViwe.Visibility = Visibility.Hidden;
            BagsView.Visibility = Visibility.Hidden;
            LoadFlightData();
        }

        private void BagsViewToggle(object sender, RoutedEventArgs e)
        {
            LiveView.Visibility = Visibility.Hidden;
            FlightsView.Visibility = Visibility.Hidden;
            NotiSenderViwe.Visibility = Visibility.Hidden;
            BagsView.Visibility = Visibility.Visible;
            StaffView.Visibility = Visibility.Hidden;
        }
```

Staff View Grid

```
        private void PlotStaff(Staff staff)
        {

            WpfPlot2.Plot.Add.Scatter(staff.x, staff.y, ScottPlot.Colors.Green);
            WpfPlot2.Refresh();
        }

        private void PlotStaff(Location loc)
        {
            var myScatter = WpfPlot2.Plot.Add.Scatter(loc.x,
loc.y,ScottPlot.Colors.Black);
            myScatter.MarkerSize = 10;
            WpfPlot2.Refresh();
        }

        private async void LoadStaffData()
        {
            List<Staff> ShowStaffToGrid = new();
            InitializeMap2();

            Query StaffQuery = _database.Collection("staff");
            QuerySnapshot snapshot = await StaffQuery.GetSnapshotAsync();

            foreach (var document in snapshot.Documents)
            {
                if (document.Exists)
```

```
            {
                Staff staff = document.ConvertTo<Staff>();
                ShowStaffToGrid.Add(staff);
                PlotStaff(staff);
            }
        }
        StaffDataGrid.ItemsSource = ShowStaffToGrid;
    }


    private void LoadStaffDatabtn(object sender, RoutedEventArgs e)
    {
        BagsView.Visibility = Visibility.Hidden;
        LiveView.Visibility = Visibility.Hidden;
        FlightsView.Visibility = Visibility.Hidden;
        NotiSenderViwe.Visibility = Visibility.Hidden;
        StaffView.Visibility = Visibility.Visible;

        StaffTime.Content = "Time : " + System.DateTime.Now.ToString("dd/MM/yyyy
HH:mm:ss");
        StaffArea.Content = "";
        timer.Stop();
        WpfPlot2.Plot.Clear();
        LoadStaffData();
        WatchStaff();
    }



    private void WatchStaff()
    {
        CollectionReference pass = _database.Collection("staff");
        Query query = _database.Collection("staff");

        listener = query.Listen(snapshot =>
        {
            Dispatcher.Invoke(() =>
            {
                LoadStaffData();
            });
        });
    }


    private async void RollBack(Staff staff , string StartDate , string EndDate)
    {
        List<Location> locs = new List<Location>();

        Query staffQuery = _database.Collection("staff");
        QuerySnapshot snapshot = await staffQuery.GetSnapshotAsync();

        bool Found = false;
        foreach (DocumentSnapshot document in snapshot.Documents)
        {
            if (document.Exists)
            {
                Staff sta = document.ConvertTo<Staff>();
                if (sta.userid == staff.userid)
```

```
                    {
                        Query staffQ =
_database.Collection("staff").Document($"{document.Id}").Collection("locations");
                        QuerySnapshot StaffSnap = await staffQ.GetSnapshotAsync();

                        foreach (DocumentSnapshot documentt in StaffSnap.Documents)
                        {
                            if (document.Exists)
                            {
                                Location loc = documentt.ConvertTo<Location>()!;
                                System.DateTime locafter =
System.DateTime.ParseExact($"{loc.time}", "M/d/yyyy h:mm:ss tt",
                                      CultureInfo.InvariantCulture, DateTimeStyles.None);

                                if (locafter >= convertTimeDate(StartDate) && locafter <=
convertTimeDate(EndDate))
                                {
                                    locs.Add(loc);
                                    Found = true;
                                }
                            }
                        }
                        if (locs.Count>0) { StartTimer(locs); } else {

                            StaffTime.Content = "No PlayBack Found in This Time Zone";
                            StaffArea.Visibility = Visibility.Hidden;
                            StaffTime.Foreground = Brushes.Red;
                        }

                    }
                    break;
                }
            }
        }

        private System.DateTime convertTimeDate(string date) {

            return System.DateTime.ParseExact($"{date}", "MM/dd/yyyy HH:mm",
                                     System.Globalization.CultureInfo.InvariantCulture);
        }


        private void StartTimer(List<Location> locs )
        {
            int i = 0;
            WpfPlot2.Refresh();

            timer.Interval = TimeSpan.FromSeconds(1); // Set the interval to 1 second
            timer.Tick += (sender, e) =>
            {
                // This method will run every second
                // Update your UI or perform any other periodic tasks here
                if (i < locs.Count) {
                    WpfPlot2.Plot.Clear();
                    InitializeMap2();
                    PlotStaff(locs[i]);
                    WpfPlot2.Refresh();
                    StaffTime.Content = $"Time :{locs[i].time}";
```

```
                    StaffArea.Content = $"Location :{locs[i].location}";
                    StopPlayBack.Visibility = Visibility.Visible;

                    i++;
                }
            };
            timer.Start();
        }

        private  void WatchRollback(List<Location> locs , string Date) {

            foreach (var item in locs)
            {
                    PlotStaff(item);
                    StaffTime.Content = $"Time :{item.time}";
                    StaffArea.Content = $"Location :{item.location}";
            }
        }

private void ViewStaff (object sender, RoutedEventArgs e)
 {
     BagsView.Visibility = Visibility.Hidden;
     LiveView.Visibility = Visibility.Hidden;
     FlightsView.Visibility = Visibility.Hidden;
     NotiSenderViwe.Visibility = Visibility.Hidden;
     StaffView.Visibility = Visibility.Visible;

     StaffTime.Content = "Time : " + System.DateTime.Now.ToString("dd/MM/yyyy HH:mm:ss");
     StaffArea.Content = "";
     timer.Stop();
     WpfPlot2.Plot.Clear();
     LoadStaffData();
     WatchStaff();
     StopPlayBack.Visibility = Visibility.Hidden;
 }
```

Notification Sender View C#

```
public partial class MainWindow : Window
 {
     List<string> AllRegistrationTokens = new List<string>();
     private void RadioButton_Checked(object sender, RoutedEventArgs e)
     {
         NotiUsername.IsEnabled = false;
     }


     private void RadioButton_Checked_1(object sender, RoutedEventArgs e)
     {
         NotiUsername.IsEnabled = true;
     }

     private async void ViewNoti(object sender, RoutedEventArgs e)
     {
         BagsView.Visibility = Visibility.Hidden;

         LiveView.Visibility = Visibility.Hidden;
         FlightsView.Visibility = Visibility.Hidden;
```

```
        NotiSenderViwe.Visibility = Visibility.Visible;
        StaffView.Visibility = Visibility.Hidden;

        Query passengerQuery = _database.Collection("passenger");
        QuerySnapshot snapshot = await passengerQuery.GetSnapshotAsync();

        PassengerDataGridSender.ItemsSource = new List<Passenger>();
        List<Passenger> passengers = new List<Passenger>();
        foreach (var document in snapshot.Documents)
        {
            if (document.Exists)
            {
                var passenger = document.ConvertTo<Passenger>();
                AllRegistrationTokens.Add(passenger.token);
                passengers.Add(passenger);
            }
        }
        PassengerDataGridSender.ItemsSource = passengers;
    }

    private async void Send_Noti(object sender, RoutedEventArgs e)
    {
        if (BroadCast.IsChecked == true)
        {
            /*{

"dblkE1e2TI6G4V6h8udYjA:APA91bFJIMOGsScIz_ngApXx_bDN3Trc6hKMS8D-
vB3pORRMbPHMwXoRkW2Wuio46CixR46Sp-tQxZeMzQG2FviNNFcJM5QR-IFm-Zqrion8rxJuTK-
9isePfekO2fZi1GTSxmmjFFxP",
                            };*/

            var message = new MulticastMessage()
            {
                Tokens = AllRegistrationTokens,
                Notification = new Notification()
                {

                    Title = "Test",
                    Body = $"{NotiMessage.Text}",
                }
            };
            var response = await
FirebaseMessaging.DefaultInstance.SendEachForMulticastAsync(message);

            MessageBox.Show($"{response.SuccessCount} messages were sent successfully");
        }
        else if (ToPassenger.IsChecked == true)
        {
            if (PassengerDataGridSender.SelectedItem == null)
            {
                Query passengerQuery = _database.Collection("passenger");
                QuerySnapshot snapshot = await passengerQuery.GetSnapshotAsync();

                foreach (var document in snapshot.Documents)
                {
                    if (document.Exists)
                    {
                        var passenger = document.ConvertTo<Passenger>();
```

```
                        if (passenger.username == NotiUsername.Text)
                        {
                            var message = new Message()
                            {
                                Token = passenger.token,
                                Notification = new Notification()
                                {

                                    Title = "Test",
                                    Body = $"{NotiMessage.Text}",
                                }
                            };
                            string response = await
FirebaseMessaging.DefaultInstance.SendAsync(message);
                            break;
                        }
                    }

            }
            else if (PassengerDataGridSender.SelectedItem != null)
            {
                //Passenger Selected
                Passenger? pass = PassengerDataGridSender.SelectedItem as Passenger;
                var message = new Message()
                {
                    Token = pass.token,
                    Notification = new Notification()
                    {
                        Title = "Test",
                        Body = $"{NotiMessage.Text}",
                    }
                };
                string response = await
FirebaseMessaging.DefaultInstance.SendAsync(message);
            }
            else
            {
                MessageBox.Show("Select A Passenger");
            }
        }
    }
}
```

Live tracking View

```
    public partial class MainWindow : Window
    {
        private FirestoreDb _database;

        private async void LoadInitialData()
        {
            WpfPlot1.Plot.Clear();
            InitializeMap();
            WpfPlot1.Refresh();

            try
            {
```

```
        Query passengerQuery = _database.Collection("passenger");
        QuerySnapshot snapshot = await passengerQuery.GetSnapshotAsync();
        Passenger.ShowToGird1 = new List<Passenger>();
        PassengerDataGrid.ItemsSource = Passenger.ShowToGird1;


        foreach (var document in snapshot.Documents)
        {
            if (document.Exists)
            {
                var passenger = document.ConvertTo<Passenger>();
                PassengerDataGrid.ItemsSource = new List<Passenger>{ };
                Passenger.ShowToGird1.Add(passenger);
                PlotPassengerData(passenger);
            }
        }

        PassengerDataGrid.ItemsSource = Passenger.ShowToGird1;
        InitializeMap();
    }
    catch (Exception)
    {
        MessageBox.Show("No Internet Connection");
    }
}

private void SelectedPassChanged()
{
    if (PassengerDataGrid.SelectedItem != null)
    {
        Passenger? pass = PassengerDataGrid.SelectedItem as Passenger;
        FindId.Text = pass.username;
        RefreshButton.Content = $"Unselect Passenger {pass.username}";
        /*          LoadInitialData();*/

}


private void PlotPassengerData(Passenger? passenger)
{
WpfPlot1.Plot.Add.Scatter(passenger.x, passenger.y,Colors.Red);
WpfPlot1.Refresh();
}

private async void FindPassenger(string PassengerUsername)
{
    Query passengerQuery = _database.Collection("passenger");
    QuerySnapshot snapshot = await passengerQuery.GetSnapshotAsync();

    foreach (var document in snapshot.Documents)
    {
        if (document.Exists)
        {
            var item = document.ConvertTo<Passenger>();
            if (item.username == PassengerUsername)
            {
                WpfPlot1.Plot.Clear();
                PlotPassengerData(item);
```

```
                }
                else
                {
                    PassNotFound.Visibility = Visibility.Visible;

                }
            }
        }

        private void FindAndTrackPassenger(object sender, RoutedEventArgs e)
        {
            if (FindId.Text == "")
            {
                PassNotFound.Content = "Select or Enter Passanger Username";
                PassNotFound.Visibility = Visibility.Visible;
            }
            else
            {
                FindPassenger(FindId.Text);
                PassNotFound.Visibility = Visibility.Hidden;
                FindId.Text = "";
                RefreshButton.Content = "Unselect Passenger";
            }

        }

        private void RefreshPassengerData(object sender, RoutedEventArgs e)
        {
            RefreshButton.Content = "Refresh Data";
            WpfPlot1.Plot.Clear();
            InitializeMap();
            LoadInitialData();
        }
    }
```

Flight View

```
public partial class MainWindow : Window
 {
     private async void LoadFlightData()
     {
         try
         {
             Query flightQuery = _database.Collection("flight");
             QuerySnapshot snapshot = await flightQuery.GetSnapshotAsync();

             Flight.ShowToGird1 = new List<Flight>();

             foreach (var document in snapshot.Documents)
             {
                 if (document.Exists)
                 {
                     try
                     {
                         var flight = document.ConvertTo<Flight>();
                         flights_DataGrid.ItemsSource = new List<Flight> { };
                         Flight.ShowToGird1.Add(flight);
```

```
                    }
                    catch (Exception)
                    {
                        MessageBox.Show("invalid Flight info");
                    }
                }
            }
            flights_DataGrid.ItemsSource = Flight.ShowToGird1;
        }
        catch (Exception)
        {
            MessageBox.Show("wala3 al net ya garbo3333");
        }


    private async void Button_Click_4(object sender, RoutedEventArgs e)
    {
        int flightid = new Random().Next(1, 10001);
        String[] dep = DepartureDate_Input.Text.Split('/');
        String[] arr = DepartureDate_Input.Text.Split('/');

        DocumentReference flightAdd =
_database.Collection("flight").Document($"{flightid}");

        try
        {
            Dictionary<string, object> FlightData = new()
            {
            {"FlightId",flightid},
            {"arrival",Arrival_Input.Text.ToString()},
            {"arrival_date",$"{dep[0]}/{dep[1]}/{dep[2]}"},
            {"arrival_time",$"{Arr_Time_input.Text}"},
            {"departure",Dep_Input.Text},
            {"departure_date",$"{arr[0]}/{arr[1]}/{arr[2]}"},
            {"departure_time",$"{Dep_Time_input.Text}"},
            {"gate",int.Parse(Gate_Input.Text)},
            {"terminal",int.Parse(Terminal_Input.Text)},
            };
            await flightAdd.SetAsync(FlightData);
            MessageBox.Show("Flight Added Succ");
        }
        catch (Exception)
        {
            MessageBox.Show("Invalid Flight info \nPlease Enter Flight info again",
"invalid info", MessageBoxButton.OK, MessageBoxImage.Warning);
        }
        LoadFlightData();
    }

    private void DateTextBox_TextChanged(object sender, TextChangedEventArgs e)
    {
        if (sender is TextBox textBox)
        {
            var text = new string(textBox.Text.Where(char.IsDigit).ToArray());

            // Ensure the text is not longer than 4 characters
            if (text.Length > 4)
            {
```

```
                text = text[..4];
            }
            // Format the time input with hours and minutes
            if (text.Length >= 2)
            {
                // Insert colon after the second character (hours)
                text = text.Insert(2, ":");
            }

            // Update the TextBox text with formatted time
            textBox.Text = text;

            // Set the caret position to the end of the text
            textBox.CaretIndex = textBox.Text.Length;
        }
    }
    private void Delete_Flight_Click(object sender, RoutedEventArgs e)
    {
        Flight? SelectedFli = flights_DataGrid.SelectedItem as Flight;

        if (flights_DataGrid.SelectedItem != null)
        {
            MessageBoxResult result = MessageBox.Show("Do you Want to Delete This
Flight", "Delete Flight", MessageBoxButton.YesNo, MessageBoxImage.Warning);
            if (result == MessageBoxResult.Yes)
            {
                DocumentReference docref =
_database.Collection("flight").Document($"{SelectedFli.FlightId}");
                docref.DeleteAsync();
                LoadFlightData();
            }
        }
        else
       {
            MessageBox.Show("Please Select A Flight", "Please", MessageBoxButton.OK,
MessageBoxImage.Warning);

        }
    }
    private void GetFlightTickets(object sender, RoutedEventArgs e)
    {
        MyGetFlightTickets();
    }

    public async void MyGetFlightTickets()
    {
        if (flights_DataGrid.SelectedItem != null)
        {
            Flight? SelectedFli = flights_DataGrid.SelectedItem as Flight;
            int FlightID = SelectedFli.FlightId;

            Query flightQuery = _database.Collection("ticket");
            QuerySnapshot snapshot = await flightQuery.GetSnapshotAsync();

            List<Ticket> ticketList = new List<Ticket>();

            foreach (var document in snapshot.Documents)
            {
```

```
                if (document.Exists)
                {
                    try
                    {
                        Dictionary<string, object> ticket = document.ToDictionary();
                        if (ticket["flightID"].ToString() ==
SelectedFli.FlightId.ToString())
                        {
                            ticketList.Add(document.ConvertTo<Ticket>());
                        }

                    }
                    catch (Exception)
                    {
                        MessageBox.Show("ERROR FROM HERE");
                    }
                }
            }

            TickesDataGrid.ItemsSource = ticketList;
            TicketCount.Foreground = Brushes.LightGreen;
            TicketCount.Content = $"Assigned Tickets Count :{ticketList.Count}";
        }
        else
        {
            TicketCount.Foreground = Brushes.Red;
            TicketCount.Content = "Select Flight From Flight Grid";
        }
    }


    private void GenerateTicket(object sender, RoutedEventArgs e)
    {

        if (flights_DataGrid.SelectedItem != null)
        {
            Flight? SelectedFli = flights_DataGrid.SelectedItem as Flight;
            int ticketid = (SelectedFli.FlightId * 1000) + new Random().Next(1, 10001);
            DocumentReference TicketAdd =
_database.Collection("ticket").Document($"{ticketid}");

            Dictionary<string, object> ticketData = new()
            {
                { "flightID",SelectedFli.FlightId},
                { "ticket_id",ticketid}
            };
            TicketAdd.SetAsync(ticketData);
            MessageBox.Show($"Ticket Generated For This Flight \nTicket Number
{ticketid}");
            MyGetFlightTickets();

        }
        else
        {
            TicketCount.Foreground = Brushes.Red;
            TicketCount.Content = "Select Flight From Flight Grid";
        }
    } }
```

Bags View

```
    public partial class MainWindow : Window
    {
        List<Bag> bagsFound = new();
        private async void GetBag(object sender, RoutedEventArgs e)
        {
            Query bagsQuery = _database.Collection("passengerflights");
            QuerySnapshot snapshots = await bagsQuery.GetSnapshotAsync();

            foreach (var document in snapshots.Documents)
            {
                if (document.Id == FindBag.Text)
                {
                    Query bags =
_database.Collection("passengerflights").Document($"{document.Id}").Collection("bags");
                    QuerySnapshot bagssnapshot = await bags.GetSnapshotAsync();
                    foreach (var documentSnapshot in bagssnapshot.Documents)
                    {
                        Dictionary<string, object> bagggy =
documentSnapshot.ToDictionary();
                        Bag baggg = new($"{bagggy["description"].ToString()}");
                        MessageBox.Show($"{documentSnapshot.Id}");
                        Query statusQ =
_database.Collection("passengerflights").Document($"{document.Id}").Collection("bags").Doc
ument(documentSnapshot.Id).Collection("status");
                        QuerySnapshot statusQsnap = await statusQ.GetSnapshotAsync();

                        foreach (var documentt in statusQsnap.Documents)
                        {
                            Dictionary<string, object> bag = documentt.ToDictionary();
                            baggg.statuses.Add(new status(bag["location"].ToString()!,
bag["time"].ToString()!));

                        }
                        bagsFound.Add(baggg);
                    }
                }
            }
            BagsDataGrid.ItemsSource = bagsFound;
        }
    }
```

Classes

Flight Class

```
    [FirestoreData]
    public class Flight
    {
        [FirestoreProperty]

        public int FlightId { get; set; }

        [FirestoreProperty]
        public string arrival { get; set; }

        [FirestoreProperty]
        public string arrival_date { get; set; }

        [FirestoreProperty]
        public string arrival_time { get; set; }

        [FirestoreProperty]
        public string departure { get; set; }

        [FirestoreProperty]
        public string departure_date { get; set; }


        [FirestoreProperty]
        public string departure_time { get; set; }

        [FirestoreProperty]
        public int terminal { get; set; }

        [FirestoreProperty]
        public int gate { get; set; }

        public static List<Flight> ShowToGird1 { get => ShowToGird; set => ShowToGird =
value; }
        static List<Flight> ShowToGird = new List<Flight>();
   }
```

Passenger Class

```
[FirestoreData]
 public class Passenger
 {
     public static List<Passenger> ShowToGird1 { get => ShowToGird; set => ShowToGird =
value; }
     static List<Passenger> ShowToGird = new();
     [FirestoreProperty]
     public string currentflight { get; set; }

     [FirestoreProperty]
     public string currentlocation { get; set; }

     [FirestoreProperty]
```

```
     public string? username { get; set; }
     [FirestoreProperty]
     public string token { get; set; }

     [FirestoreProperty]
     public string passenger_id { get; set; }
     [FirestoreProperty]
     public double x { get; set; }

     [FirestoreProperty]
     public double y { get; set; }
}
```

Staff Class

```
    [FirestoreData]
    internal class Staff
    {
        [FirestoreProperty]
        public string currentlocation { get; set; } = string.Empty;

        [FirestoreProperty]

        public string userid { get; set; } = string.Empty;

        [FirestoreProperty]

        public double x { get; set; }


        [FirestoreProperty]
        public double y { get; set; }

        public List<Location> locations { get; set; } = new();
    }
```

Bag Status Class

```
    internal class status
    {
        string location;
        string time;
        public status(string location, string time)
        {
            this.Location = location;
            this.Time = time;
        }
        public string Location { get => location; set => location = value; }
        public string Time { get => time; set => time = value; }
    }
```

Bag Class

```
        public string description { get; set; }

        public List<status> statuses  = new List<status>();

        public Bag(string description)
        {
            this.description = description;
        }
```

Ticket Class

```
    [FirestoreData]
    public class Ticket
    {
        [FirestoreProperty]
        public int flightID { get; set; }

        [FirestoreProperty]
        public string? passenger_id { get; set; }

        [FirestoreProperty]
        public int ticket_id { get; set; }
```

Location Class

```
    [FirestoreData]
    internal class Location
    {
        [FirestoreProperty]
        public string location { get; set; }
        [FirestoreProperty]
        public string  time { get; set; }

        [FirestoreProperty]

        public double x { get; set; }

        [FirestoreProperty]
        public double y { get; set; }
    }
```

7.5. ESP32 Sketch

```
#include <BLEDevice.h>
#include <BLEUtils.h>
#include <BLEServer.h>
#define SERVICE_UUID        "4fafc201-1fb5-459e-8fcc-c5c9c331414c"
#define CHARACTERISTIC_UUID "beb5483e-36e1-4688-b7f5-ea07361ba6a4"
void printMACAddress(const char* label, esp_mac_type_t macType) {
  uint8_t baseMac[6];
  esp_read_mac(baseMac, macType);
  Serial.print(label);
  for (int i = 0; i < 5; i++) {
    Serial.printf("%02X:", baseMac[i]);
  }
  Serial.printf("%02X\n", baseMac[5]);
}
void setup() {
  Serial.begin(115200);
  Serial.println("Starting BLE work!");
  BLEDevice::init("Beacon 1");
  BLEServer *pServer = BLEDevice::createServer();
  BLEService *pService = pServer->createService(SERVICE_UUID);
  BLECharacteristic *pCharacteristic = pService->createCharacteristic(
                                         CHARACTERISTIC_UUID,
                                         BLECharacteristic::PROPERTY_READ |
                                         BLECharacteristic::PROPERTY_WRITE
                                       );
  pCharacteristic->setValue("Hello! It's P27");
  pService->start();
  BLEAdvertising *pAdvertising = BLEDevice::getAdvertising();
  pAdvertising->addServiceUUID(SERVICE_UUID);
  pAdvertising->setScanResponse(true);
  pAdvertising->setMinPreferred(0x06);
  pAdvertising->setMinPreferred(0x12);
  BLEDevice::startAdvertising();
  Serial.println("Characteristic defined! Now you can read it on your phone!");
   // Print MAC addresses
  printMACAddress("Station MAC: ", ESP_MAC_WIFI_STA);
  printMACAddress("SoftAP MAC: ", ESP_MAC_WIFI_SOFTAP);
  printMACAddress("Bluetooth MAC: ", ESP_MAC_BT);
  printMACAddress("Ethernet MAC: ", ESP_MAC_ETH);
}

void loop() {}
```

Chapter 8

Testing

8.1. Introduction into testing phase

Welcome to the Testing Phase of our software development lifecycle. This critical stage ensures the robustness, functionality, and reliability of our application before it reaches our users. In this phase, we rigorously examine every aspect of the software to identify and rectify any bugs, errors, or inconsistencies.

Objectives

- **Verify Functionality**: Ensure all features work as intended.
- **Identify Bugs**: Detect and document any defects or issues.
- **Improve Quality**: Enhance the overall quality and performance of the application.
- **User Experience**: Confirm that the user interface is intuitive and user-friendly.

Key Activities:

- **Unit Testing**: Validate individual components for correct operation.
- **Integration Testing**: Ensure that integrated modules function together correctly.
- **System Testing**: Test the complete system to verify it meets the specified requirements.
- **User Acceptance Testing (UAT)**: Conduct testing with actual users to confirm the system meets their needs and expectations.

Expected Outcomes:

- A robust, reliable, and user-friendly application.
- Comprehensive documentation of all testing activities and results.
- A high level of confidence in the software's readiness for release.

8.2. Test Parts

1. Firebase Database Operations:

- **Connection Establishment**: We tested the system's ability to establish a stable connection with the Firebase database. This includes handling authentication and ensuring secure access.
- **CRUD Operations**: We verified that Create, Read, Update, and Delete operations work correctly. This ensures that data can be added, retrieved, modified, and removed seamlessly from the database.
- **Error Handling**: We tested the system's response to various database errors and network issues to ensure robust error handling and user-friendly error messages.

2. Notification System:

- **Send Notifications**: We ensured that notifications can be sent reliably from the server to various endpoints, such as user devices.
- **Receive Notifications**: We verified that notifications are received promptly and correctly by the intended recipients.
- **Notification Content**: We checked that the content of notifications is accurate and relevant to the user's context.

3. Authentication Processes:

- **Registration**: We tested the user registration process to ensure new users can sign up easily and securely.
- **Verification**: We verified the email verification process to ensure users can confirm their accounts.
- **Login**: We tested the login functionality to ensure users can access their accounts with the correct credentials.
- **Security**: We ensured that the authentication process is secure, protecting user data and preventing unauthorized access.

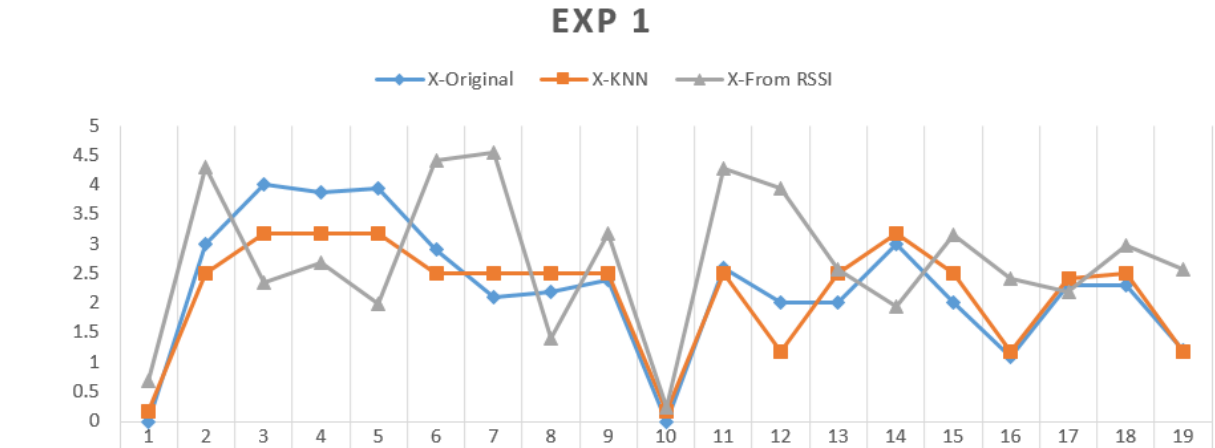
4. Navigation System:

- **Position Accuracy**: We conducted multiple experiments to test and enhance the accuracy of the user's calculated position within the airport.
- **Position Updates**: We ensured that the system provides real-time position updates to users as they navigate the airport.
- **User Experience**: We tested the user interface for displaying position information to ensure it is intuitive and helpful.

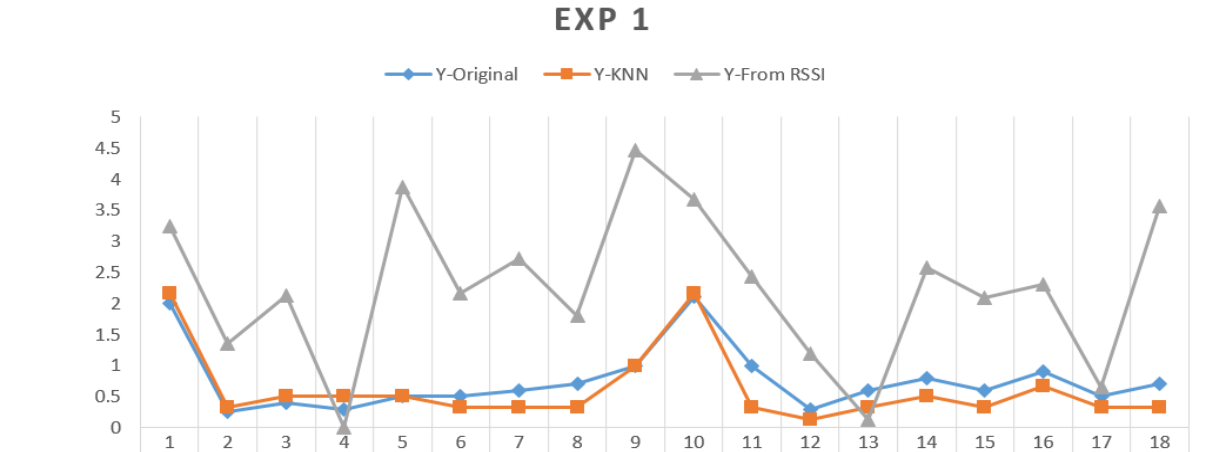
4.1. Experiments

Experiment 1

we recorded 100 movable points in 20*5 m area. The Graph recorded about 20 movable points in our environment based on (RSSI) values received from the three fixed point (BLE) devices, then we applied (KNN) approach to enhance the result as mentioned in the tables and accuracy 93.45%,



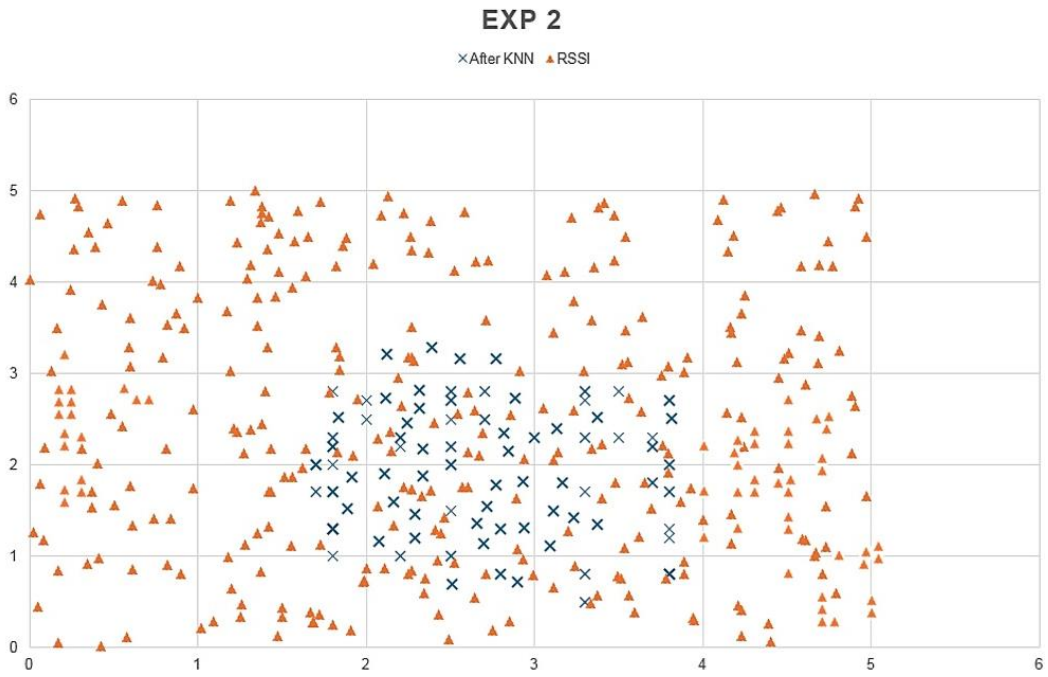
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
X-Original	0	3	4	3.88	3.94	2.9	2.1	2.2	2.4	0	2.6	2	2	3	2	1.1	2.3	2.3	1.2
X-KNN	0.17	2.5	3.17	3.17	3.17	2.5	2.5	2.5	2.5	0.17	2.5	1.17	2.5	3.17	2.5	1.17	2.41	2.5	1.17
X-From RSSI	0.68	4.31	2.35	2.68	1.99	4.42	4.54	1.4	3.17	0.24	4.27	3.94	2.58	1.95	3.15	2.41	2.18	2.97	2.56



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Y-Original	2	0.25	0.4	0.3	0.5	0.5	0.6	0.7	1	2.1	1	0.3	0.6	0.8	0.6	0.9	0.5	0.7
Y-KNN	2.17	0.33	0.5	0.5	0.5	0.33	0.33	0.33	1	2.17	0.33	0.13	0.33	0.5	0.33	0.67	0.33	0.33
Y-From RSSI	3.25	1.35	2.13	0.01	3.88	2.16	2.72	1.81	4.46	3.68	2.43	1.19	0.13	2.58	2.09	2.31	0.65	3.56

Experiment 2

Applying 3 fixed (BLE) beacons in area 5m in width and 5m in length. At coordinates (2.5, 2.5) We recorded 300 points over a duration of 10 minutes for the same point. After classification process using (KNN) and accuracy 93% and approach the error rate decreased



X	Y	RSSI1	RSSI2	RSSI3
3.47	4.24	-67	-77	-62
0.4	2.01	-66	-78	-64
0.78	3.98	-62	-79	-58
4.97	4.5	-66	-79	-63
4.97	4.5	-66	-79	-63
4.18	4.51	-68	-80	-63
3.79	2.13	-64	-75	-64

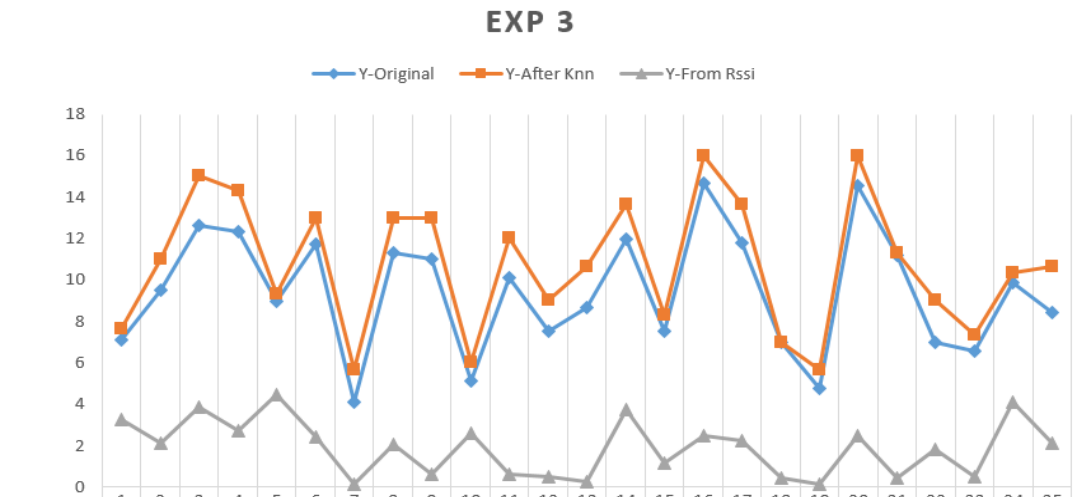
calibrated values

X	Y
2.5	2
2.5	2.5
2.5	2.8
2.7	2.8
3	2.3
3.3	1.7
3.7	1.8

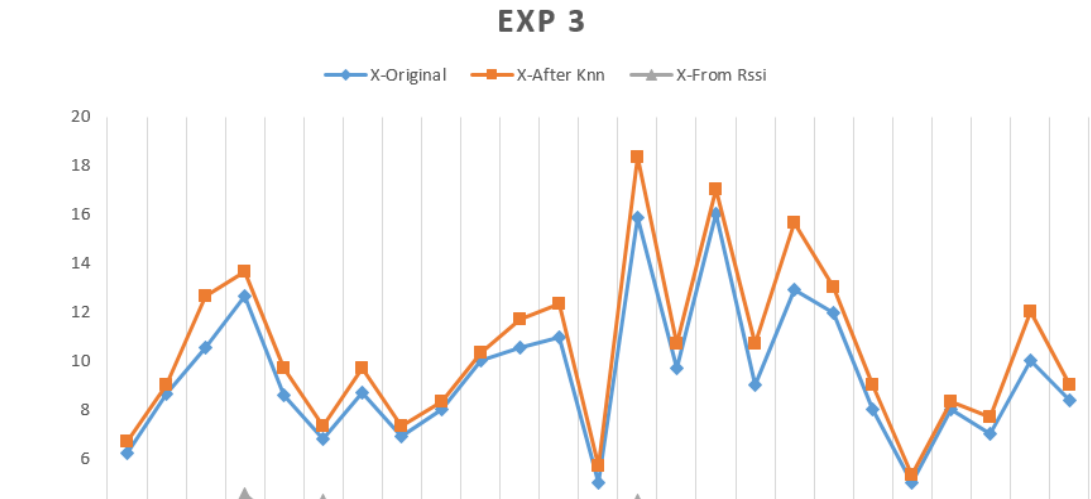
Enhanced values

Experiment 3

The Graph recorded about 500 movable points but show a sample of them in our environment based on (RSSI) values received from the three fixed point (BLE) devices, then we applied (KNN) approach to enhance the result as mentioned in the tables, and the accuracy in this case 93%



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
Y-Original	7.1	9.5	12.6	12.3	8.98	11.7	4.12	11.3	11	5.11	10.1	7.5	8.66	12	7.5	14.7	11.8	6.98	4.78	14.6	11.2	7	6.55	9.89	8.41
Y-After Knn	7.67	11	15	14.3	9.33	13	5.67	13	13	6	12	9	10.7	13.7	8.33	16	13.7	7	5.67	16	11.3	9	7.33	10.3	10.7
Y-From Rssi	3.25	2.13	3.88	2.72	4.46	2.43	0.13	2.09	0.65	2.62	0.65	0.49	0.24	3.77	1.19	2.46	2.27	0.46	0.13	2.48	0.41	1.8	0.52	4.1	2.1



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
X-Original	6.2	8.65	10.6	12.7	8.6	6.78	8.71	6.89	7.98	10	10.5	11	5	15.9	9.69	16	9	12.9	12	8	5	7.98	7	10	8.4
X-After Knn	6.67	9	12.7	13.7	9.67	7.33	9.67	7.33	8.33	10.3	11.7	12.3	5.67	18.3	10.7	17	10.7	15.7	13	9	5.33	8.33	7.67	12	9
X-From Rssi	0.68	2.35	1.99	4.54	3.17	4.27	2.58	3.15	2.18	2.56	0.39	1.68	2.04	4.25	1.14	1	1	3.7	2.35	0.88	3.61	2.28	3.74	3.96	1.57

5. Luggage Tracking Feature:

- **Add Luggage**: We verified that passengers could add their luggage details into the system easily.
- **Track Status**: We tested the tracking functionality to ensure passengers can monitor the status and location of their luggage in real-time.
- **QR Code Assignment**: We ensured that each piece of luggage is assigned a unique QR code.
- **Checkpoint Scanning**: We tested the QR code scanning at predefined checkpoints to ensure passengers are notified when their luggage arrives at these locations.

6. Customer Support Chat:

- **Real-Time Response**: We ensured that the customer support chat system provides real-time responses to user inquiries.
- **Message Handling**: We tested the handling of messages, including sending, receiving, and displaying chat messages correctly.
- **User Interface**: We ensured that the chat interface is user-friendly and accessible.

7. Admin Desktop Application:

- **Data Monitoring**: We tested the admin interface for monitoring all system data, ensuring it is comprehensive and up to date.
- **Data Manipulation**: We verified that admins can effectively manipulate system data, including updating user information, managing notifications, and handling luggage tracking data.

8. Mobile Permissions:

- **Location Services**: We tested the permission request for accessing location services, ensuring it is prompted correctly and is essential for the indoor positioning system.
- **Camera Access**: We verified that the system correctly asks for camera access to scan QR codes for luggage tracking.
- **Notification Permissions**: We ensured that users are prompted to allow notifications, essential for real-time updates and alerts.

Chapter 9

Conclusion & Future Work

9.1. Conclusion

In our project we have developed two applications, the first one is a mobile application with two different interfaces for passengers and staff. The other application is desktop application for administrative purposes.

1. Mobile Application

- **Passengers Interface**
    - Provides real-time navigation inside the airport using IPS BLE, enhancing passenger convenience in navigating complex airport environments.
    - Allows tracking of luggage in real-time, minimizing the risk of loss or mishandling.
    - Helps passengers locate family and friends within the airport, fostering a connected travel experience.
    - Includes customer support features to promptly address passenger inquiries and issues, ensuring a smooth and enjoyable journey.

- **Staff Interface**
    - Generates QR code stickers for luggage, streamlining the check-in and tracking process for airport staff.
    - Includes a QR code scanner to efficiently scan luggage tags and track their movement throughout the airport.
    - Features robust customer support capabilities to handle passenger inquiries promptly and provide assistance as needed, ensuring smooth operations and passenger satisfaction.

2. Desktop Application for Administration

- Manages and monitors system data related to IPS and BLE beacon infrastructure, ensuring optimal performance and reliability.
- Sends notifications to users regarding system updates and relevant information, enhancing communication and operational efficiency.

Main System Idea

- Provides real-time indoor navigation where GPS is unavailable, utilizing an Indoor Positioning System (IPS) based on BLE beacons.
- Helps passengers monitor their luggage to prevent loss or theft, enhancing overall travel security.
- Implements a k-nearest neighbor machine learning algorithm to enhance positioning accuracy, ensuring precise navigation and tracking within airport facilities.

9.2. Future Work

Enhanced Positioning Accuracy

- Focus on further improving the accuracy of the positioning system, especially for indoor navigation using IPS and BLE beacons.
- Explore advanced algorithms or refinements to the existing machine learning models (like KNN) to achieve higher precision.

Integration of Commercial Features

- Introduce commercial features within the mobile application, such as:
  - Sending offers and discounts to passengers based on their location within the airport, leveraging BLE beacon packets for targeted notifications.

Real-Time Luggage Tracking

- Develop a robust method for real-time tracking of luggage within the airport premises.

- Utilize the existing infrastructure (IPS and BLE beacons) to provide accurate location updates of luggage to passengers and staff.

Expand System

- Expand the application beyond navigation and luggage tracking to include functionalities such as:
  - Ticket booking and reservation management for passengers.

  - Integration with airline systems to provide real-time flight updates, gate information, and boarding details.

  - Enhancing customer service features to include support for various airport services (e.g., lounges, restaurants, transportation).