

XBCAD7319\_Project

Mzwandile Shinga – ST10210264

Ofentse Mogafe – ST10133493

Leyaine Mungani – ST10091914

Marco Lopes – ST10165142



Contents

[1. INTRODUCTION 3](#_Toc183391821)

[1.1 Company letter 4](#_Toc183391822)

[1.2 Work agreement 5](#_Toc183391823)

[1.3 Definition of “Ready” and “Done” 6](#_Toc183391824)

[2. PROJECT REQUIREMENTS 7](#_Toc183391825)

[2.1 Project Scope 7](#_Toc183391826)

[2.2 User Requirements 7](#_Toc183391827)

[2.3 Functional Requirements 7](#_Toc183391828)

[2.4 Non-Functional Requirements 7](#_Toc183391829)

[2.5 Technical Requirements 8](#_Toc183391830)

[2.6 Security Requirements 8](#_Toc183391831)

[3. USER INTERFACE DESIGN 9](#_Toc183391832)

[4. IMPLEMENTATION DOCUMENTATION 13](#_Toc183391833)

[4.1 UML Sequence Diagram (Critical Flows) 13](#_Toc183391834)

[4.2 Deployment Documentation 14](#_Toc183391835)

[4.3 Security Documentation 14](#_Toc183391836)

[5. DATA SCHEMA DOCUMENTATION 15](#_Toc183391837)

[5.1 Entity Relationship Diagram (ERD) (If using SQL-based storage) 15](#_Toc183391838)

[5.2 JSON Schemas (if using NoSQL) (Example of document-based storage in Firebase): 16](#_Toc183391839)

[6. ARCHITECTURE ARTIFACTS 17](#_Toc183391840)

[6.1 Architecture patterns 17](#_Toc183391841)

[6.2 Design patterns 17](#_Toc183391842)

[6.3 Cloud architecture 17](#_Toc183391843)

[7. SECURITY 19](#_Toc183391844)

[7.1 Authentication and Authorisation 19](#_Toc183391845)

[7.2 Data encryption 19](#_Toc183391846)

[7.3 Network security 20](#_Toc183391847)

[7.4 Vulnerability Mitigation 20](#_Toc183391848)

[7.5 Threat monitoring and incident response 20](#_Toc183391849)

[7.6 Compliance and data privacy 21](#_Toc183391850)

[8. DEV OPS 22](#_Toc183391851)

[8.1 GitHub Action Pipeline 22](#_Toc183391852)

[8.2 Pipeline Diagram: 23](#_Toc183391853)

[9. RUNNING COSTS 24](#_Toc183391854)

[10. CHANGE MANAGEMENT 25](#_Toc183391855)

[11. FAST-TRACK APPLICATION 26](#_Toc183391856)

[11.1 Android app screenshots 26](#_Toc183391857)

[11.2 Web app screenshots 30](#_Toc183391858)

[11.3 Project summary 31](#_Toc183391859)

[12. SELF REFLECTION 33](#_Toc183391860)

[12.1 Leyaine Mungani – ST10091914 33](#_Toc183391861)

[12.2 Mzwandile Shinga – ST10210264 34](#_Toc183391862)

[12.3 Ofentse Mogafe – ST10133493 34](#_Toc183391863)

[12.4 Marco Lopes – ST10165142 34](#_Toc183391864)

[13. REFERENCES 35](#_Toc183391865)

[14. WORKS CITED 36](#_Toc183391866)

# 1. INTRODUCTION

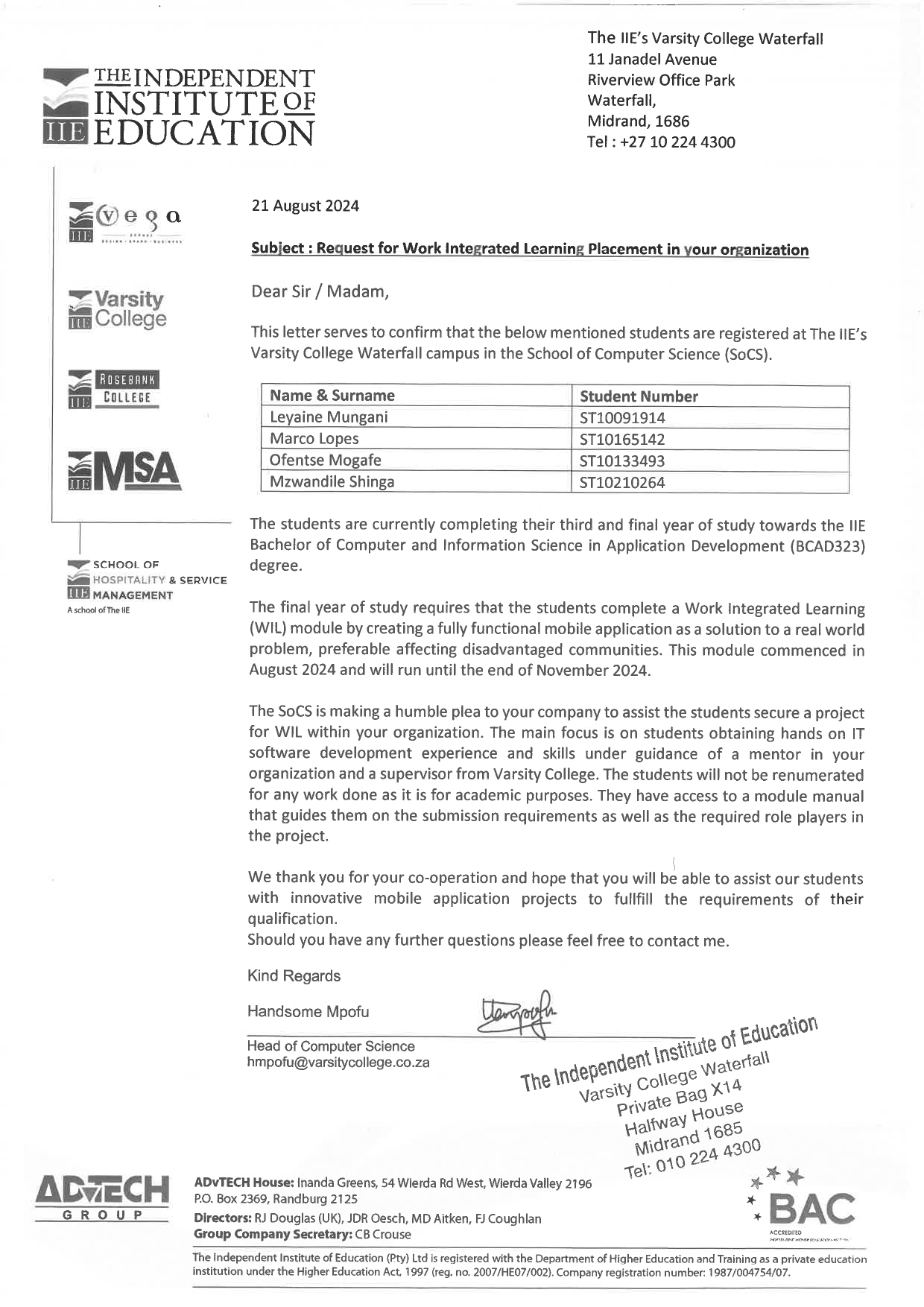
Our team of fours project is to create an order tracking software for TasTech. Using different user interfaces for each user type, this app will cater to both internal administrators (admins) and customers of the business. Using a distinctive tracking number, customers will be able to utilise the app to register, log in, and monitor the status of their service delivery or shipment. Crucially, the user won't have to re-enter the tracking number every time they log in because the app will save their tracking information.

Admins will, however, have access to an admin-only interface and secret logins. They will be able to generate tracking numbers, offer thorough product or service descriptions, and keep track of delivery or service progress updates through their app view.

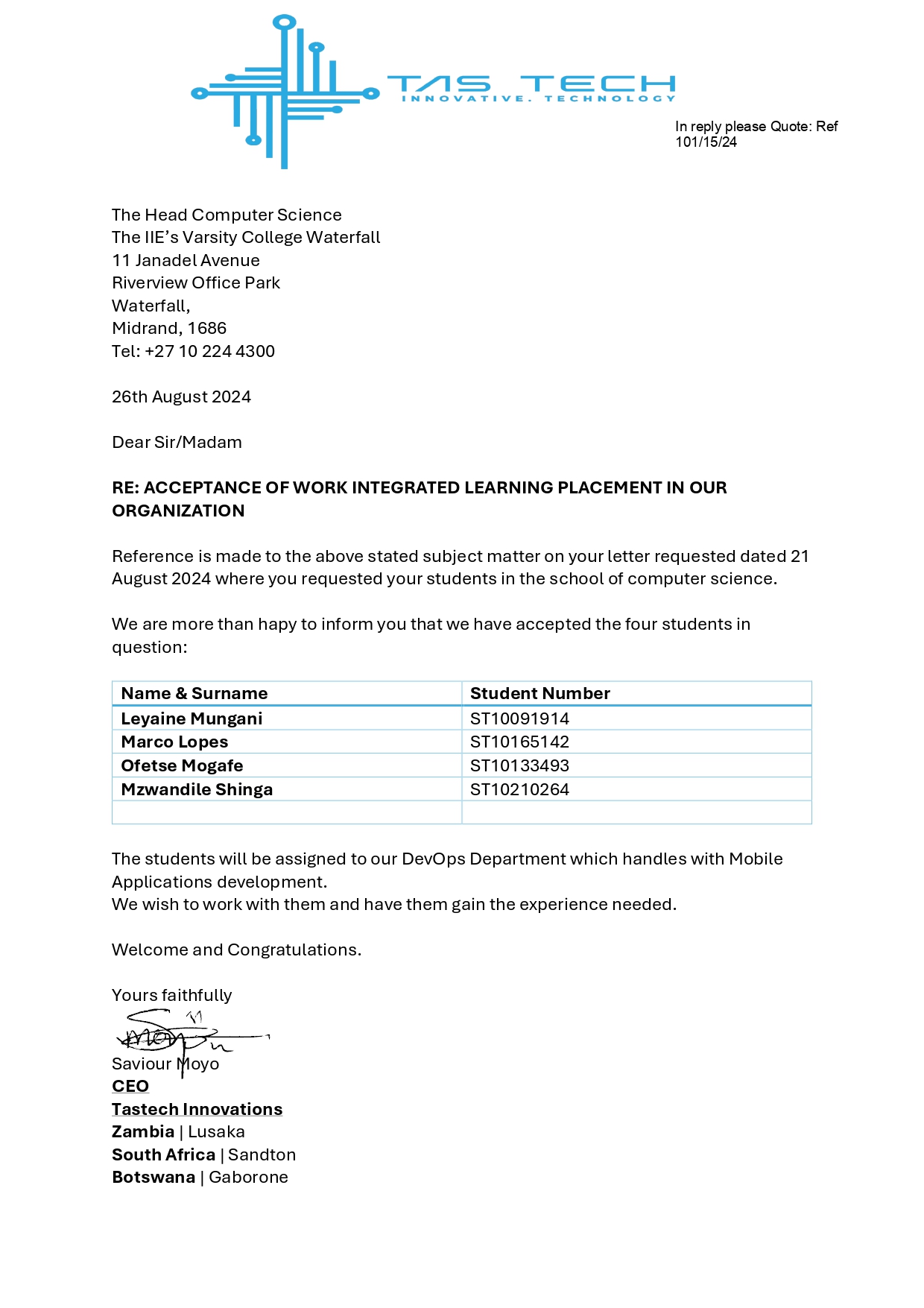
We created a high-level plan with the following milestones:  
**Milestone 1:** Create user authentication systems such as customer and admin logins. Create database schemas and set up the project environment.  
**Milestone 2:** Complete the implementation of the customer interface, including durability and tracking number entry.  
**Milestone 3:** Create an admin interface that allows users to add and modify tracking information.  
**Milestone 4:** Evaluate the app, get user and customer feedback, and improve the user interface and user experience.  
**Milestone 5:** Finalise deployment and produce all paperwork required for handover.

The duration of each sprint is set at 2 weeks, and the team's schedule has been revised appropriately to ensure that we meet the projects deadline. We have made sure that every team member is available for the duration of the project and have altered responsibilities in accordance with personal schedules.

## 1.1 Company letter



## 1.2 Work agreement



## 1.3 Definition of “Ready” and “Done”

Our teams **"Definition of Ready"** means that every team member needs to have a thorough understanding of each user role and story before development can begin. Furthermore, all external dependencies such as databases, third-party services, or APIs must be accessible and operational. Finalising design mock-ups and wireframes will help the team picture the desired user experience. Last but not least, every user story needs clear guidelines for acceptance that specify the particular requirements that must be satisfied in order for the project to be considered complete. With these components in place, we can go forward with development quickly and effectively.

Our teams **"Definition of Done"** means that the Application satisfies all the requirements and is ready for deployment. Peer reviews of the code will be conducted to guarantee quality and correctness, as well as the building and passing of unit tests. User interface usability should be assessed before the app is integrated. Additionally, end-to-end testing is necessary to ensure that the entire user experience for both admins and customers performs as planned. The application should be maintainable and simple for others to understand by providing any necessary documentation to reflect the most recent updates.

# 2. PROJECT REQUIREMENTS

## 2.1 Project Scope

* **Android Application (Customer Side):** Enables customers to track packages and manage their accounts.
* **ASP.NET Core Web Application (Admin Side):** Allows administrators to manage tracking information, admin users, and view reports.

## 2.2 User Requirements

**Customer Side**

* **UR1:** Customers can create and manage personal accounts.
* **UR2:** Customers can securely log in and log out of the application.
* **UR3:** Customers can enter tracking numbers and view real-time tracking information.
* **UR4:** Customers can view previous data of past shipments.

**Admin Side**

* **UR5:** Admins can log in with username and password authentication.
* **UR6:** Admins can manage tracking numbers including creating, updating, and deleting tracking data.
* **UR7:** Super admins can manage other admin users, including adding new admins or removing existing ones..

## 2.3 Functional Requirements

**Android Application**

* **FR1:** Registration and Authentication
* **FR2:** Tracking view for entering and viewing tracking data.
* **FR3:** Account Management for updating personal information and settings.

**ASP.NET Core Web Application**

* **FR4:** Admin Authentication and User Management.
* **FR5:** CRUD operations for tracking number management.

## 2.4 Non-Functional Requirements

**Security**

* **NFR1:** Ensure secure data transmission using HTTPS.
* **NFR2:** Encrypt sensitive user data at REST and in transit.

**Performance**

* **NFR3:** Response time under normal load should be within 2 seconds.
* **NFR4:** System should handle up to 10,000 concurrent users.

**Scalability**

* **NFR5:** System must be scalable to accommodate growth in user accounts and data sizes.

**Maintainability**

* **NFR6:** The system should be modular to allow for easy updates and maintenance.

**Interoperability**

* **NFR7:** Provide a flexible RESTful API for integration with other systems.

**Availability**

* **NFR8:** System should be available 99.9% of the time.

**Usability**

* **NFR9:** User interfaces must be intuitive and accessible on mobile platforms (scaledagileframework.com,2023)

## 2.5 Technical Requirements

**Technologies Used**

* **Android Application:** Developed using Android Studio and Kotlin.
* **Web Application:** Developed using ASP.NET Core.
* **Database:** Options include MySQL (Azure) or Firebase, to be finalized.
* **API:** RESTful API developed using Node.js and Express.
* **Authentication:** Managed through Firebase.

**System Integration**

* **Tech1:** Ensure integration between Android app and ASP.NET Core backend via RESTful API.

## 2.6 Security Requirements

**Data Security**

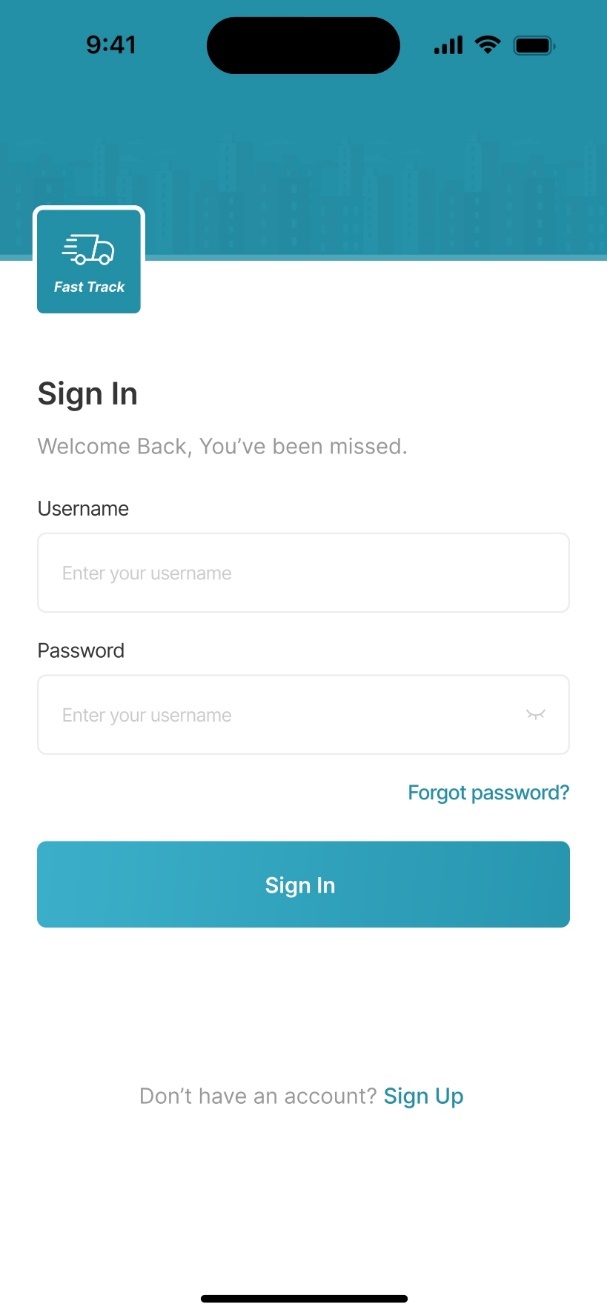
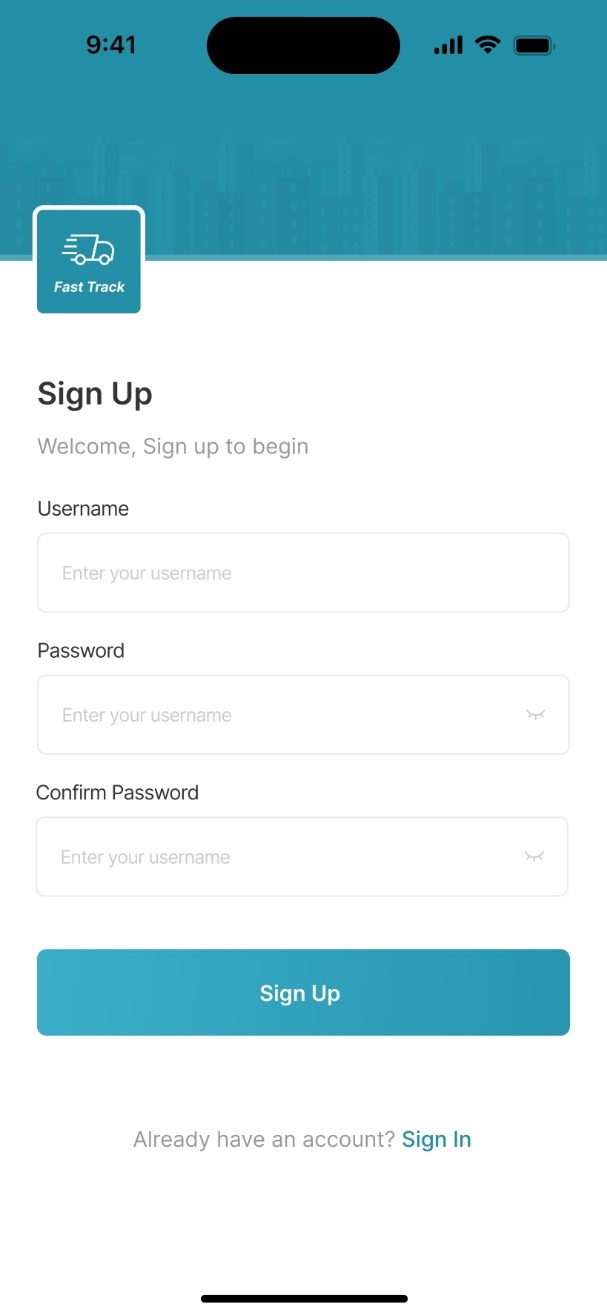
* **Sec1:** Implement role-based access control.
* **Sec2:** Use industry-standard encryption for data in transit and at rest.
* **Sec3:** Regular security audits to identify and mitigate vulnerabilities.

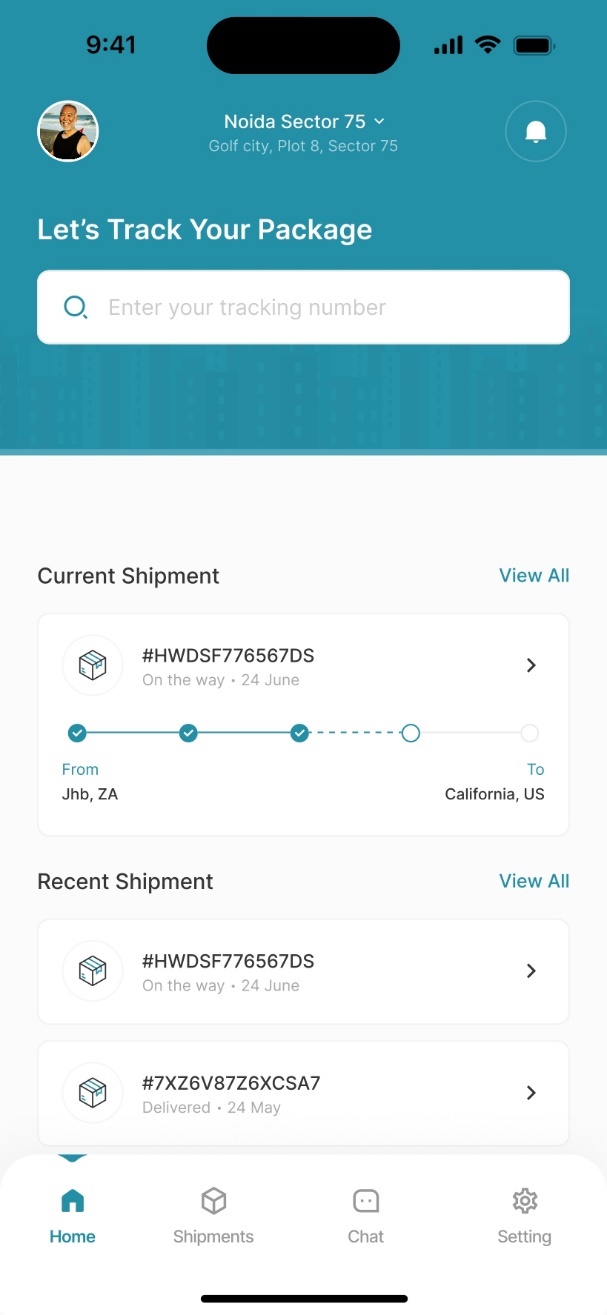
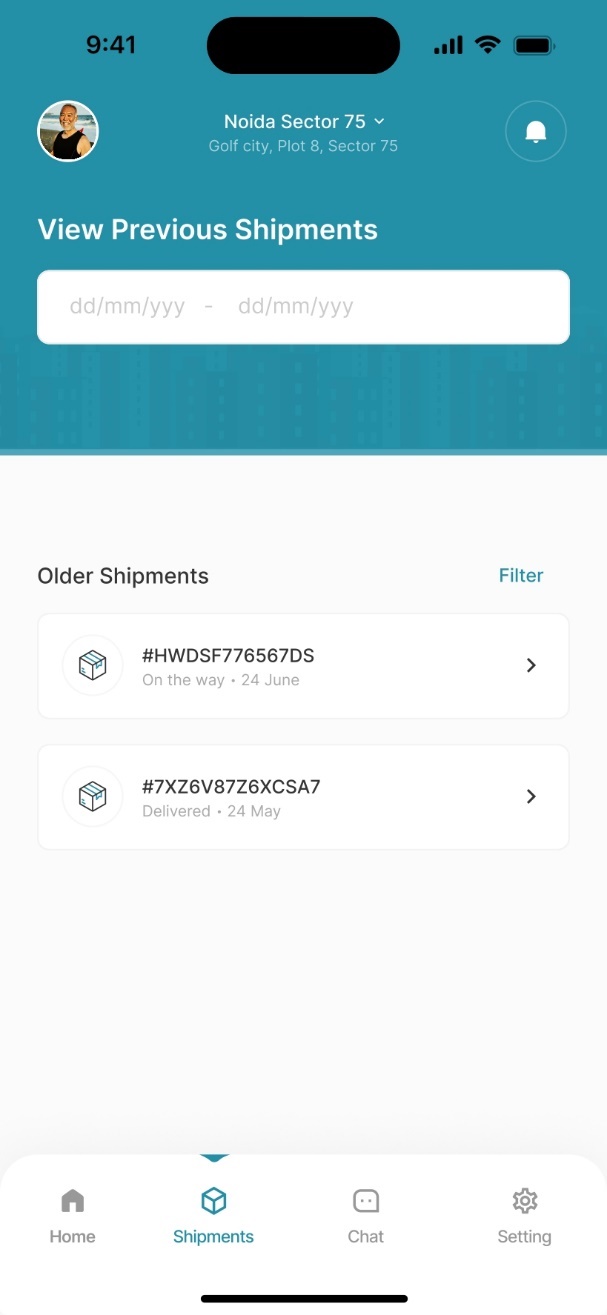
**Application Security**

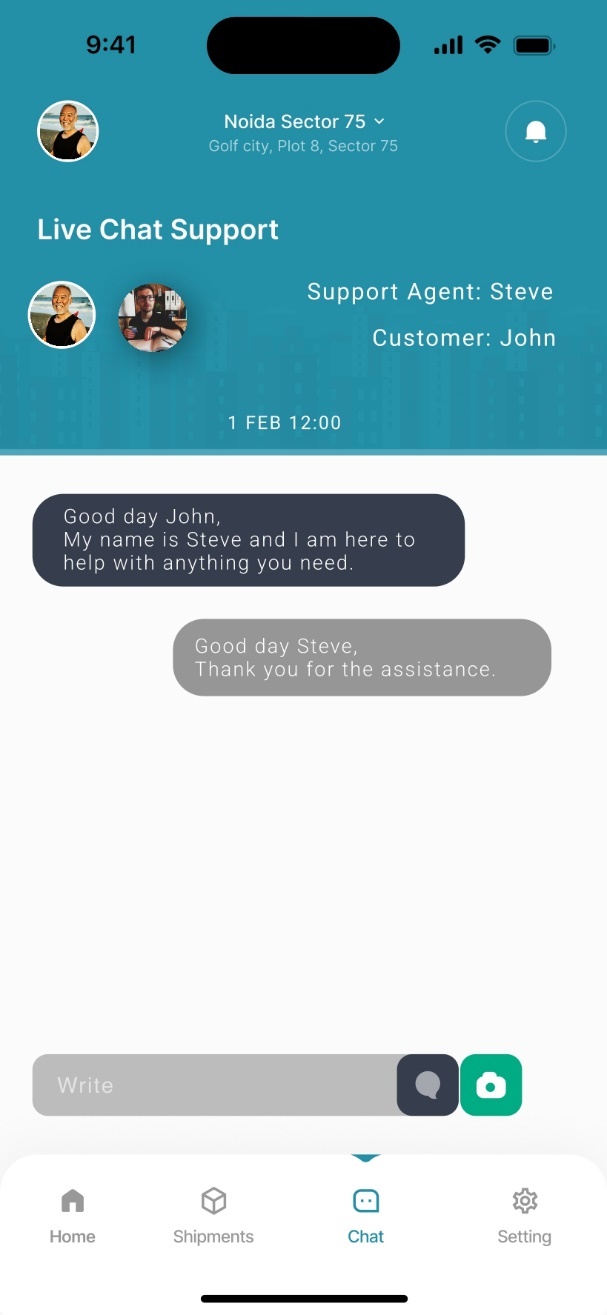
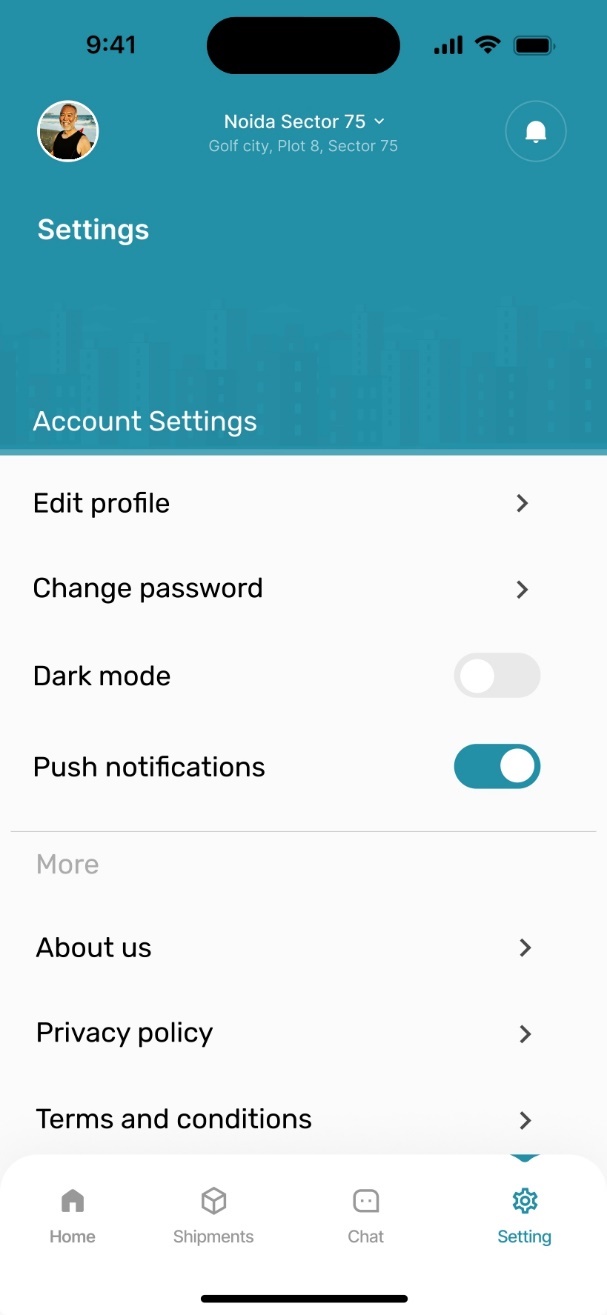
* **Sec4:** Implement stringent authentication measures including options for multi-factor authentication.
* **Sec5:** Ensure all APIs are secured against common vulnerabilities like SQL injection, Cross-Site Scripting (XSS), and Cross-Site Request Forgery (CSRF). (PortSwigger.com)

# 3. USER INTERFACE DESIGN

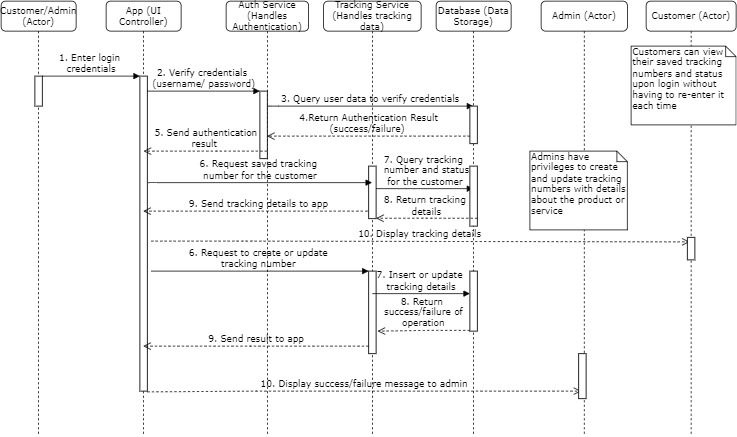
 

# 4. IMPLEMENTATION DOCUMENTATION

## 4.1 UML Sequence Diagram (Critical Flows)



a. **Customer Login and Tracking Flow:**

* **Actors**: Customer, App, Database
* **Flow**:

1. Customer opens the app and inputs their credentials (username/password).
2. The app checks the credentials in the database.
3. If authenticated, the customer is logged in.
4. The app retrieves the customer’s previously saved tracking number.
5. The customer can view the shipment’s status.

b. **Admin Login and Tracking Update Flow**:

* **Actors**: Admin, App, Database
* **Flow**:
  1. Admin logs into the app using secure credentials.
  2. The system checks the admin's credentials.
  3. Once authenticated, the admin accesses the dashboard.
  4. The admin can create a new tracking number and enter product/service details.
  5. Admin updates the status of the tracking information at various stages.

## 4.2 Deployment Documentation

**Hosting**: Firebase as Backend-as-a-Service (BaaS) will be used, providing real-time database, authentication, and cloud functions. (Chauhan, 2024)

**Technology Stack**:

* **Frontend**: Flutter or React Native for cross-platform mobile app development.
* **Backend**: Firebase for data handling and cloud functions.
* **Database**: Firebase Realtime Database or Cloud Firestore for handling orders and tracking information.

(Chauhan, 2024)

**Security**:

* OAuth 2.0 for user authentication (Firebase Authentication).
* SSL/TLS for data in transit.
* AES-256 encryption for sensitive data stored in Firebase.

(Chauhan, 2024)

## 4.3 Security Documentation

* **Role-Based Access Control (RBAC)**: Separate access rights for admins and customers. (Zhang, 2023)
* **Secure Authentication**: Use of bcrypt or PBKDF2 for password hashing and multi-factor authentication (MFA) for admin accounts.

**Data Encryption**:

* SSL/TLS for communication between client and server.
* AES-256 encryption for data at rest in Firebase.

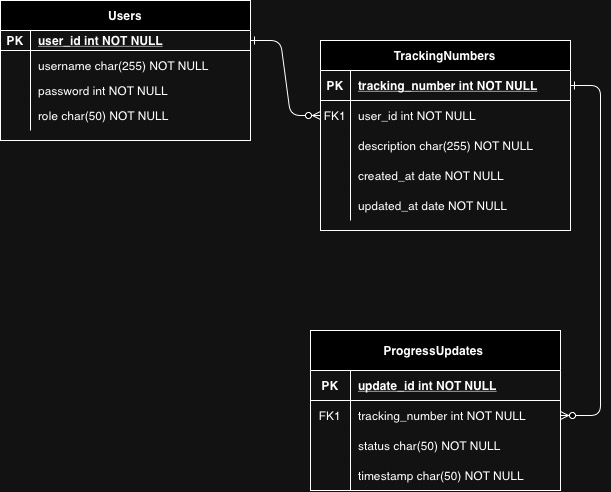
(De Groot, 2024)

* **Firewall and Secure API Gateway**: Token-based authentication for API endpoints to protect against unauthorized access.
* **Input Validation and Dependency Management**: Use input validation to prevent SQL injection and XSS. Use tools like OWASP Dependency Check for managing vulnerabilities in third-party packages.

(Laurent, 2023)

# 5. DATA SCHEMA DOCUMENTATION

## **5.1 Entity Relationship Diagram (ERD)** (If using SQL-based storage)



Key Entities:

* **Users**:

user\_id (PK), username, password, role (admin/customer)

* **TrackingNumbers**:

tracking\_number (PK), user\_id (FK), description, created\_at, updated\_at

* **ProgressUpdates**:

update\_id (PK), tracking\_number (FK), status, timestamp

Relationships:

* One-to-many between Users and TrackingNumbers (a user can have multiple tracking numbers).
* One-to-many between TrackingNumbers and ProgressUpdates (a tracking number can have multiple progress updates).

## **5.2 JSON Schemas (if using NoSQL)** (Example of document-based storage in Firebase):

**Users Collection:**

{

"user\_id": "12345",

"username": "john\_doe",

"password\_hash": "hashed\_password",

"role": "customer",

"tracking\_numbers": ["ABC123", "XYZ456"]

}

**Tracking Numbers Collection:**

{

"tracking\_number": "ABC123",

"user\_id": "12345",

"description": "Package from Store XYZ",

"progress\_updates": [

{

"status": "Dispatched",

"timestamp": "2024-09-24T10:00:00Z"

},

{

"status": "In Transit",

"timestamp": "2024-09-25T14:30:00Z"

}

}

}

# 6. ARCHITECTURE ARTIFACTS

## 6.1 Architecture patterns

For the architecture pattern **Model View View Model (MVVM)** was put to use. According to Gallardo (2023) MVVM is an android architecture pattern that cleanly separates the app's UI logic (View), business logic (View Model), and data (Model). Testability, maintainability, and scalability are all enhanced by this separation, which is crucial for an expanding app like TasTech's order tracking system. The View Model will manage the interaction between the data and the View, making sure that the UI stays responsive and updates immediately with any changes. The Model will handle all data operations, whether from a local database or remote API. The application will be simpler to scale and maintain as it grows due to this division of responsibilities.

## 6.2 Design patterns

In order to handle the various data formats received from various sources and guarantee that the data can be shown consistently throughout the user interface, TasTech used the **Adapter Pattern** in its order tracking application. For example, it could be challenging to directly display data in a consistent user interface if the application interfaces with several other APIs for shipment tracking, as each API may return data in a different format. The data from these various APIs may be converted into a consistent format that the app's components can process with ease by using the Adapter Pattern. Every courier may display a shipment in a different way, including differences in the names or formatting of data like tracking numbers, delivery dates, and statuses. TasTech is able to develop distinct adapter classes for every courier's API by utilizing the Adapter Pattern. The distinct data from each API would be transformed by these adapters into a standard model that the app's user interface elements, such as a RecyclerView showing shipment details, could display consistently. For example another courier could return {"trackingNumber": "5678", "deliveryStatus": "Shipped", and "deliveryDate": "2024-11-21"}, whilst the data returned by one might look like this: {"shipmentId": "1234", "status": "In Transit", "expectedDelivery": "2024-11-21"}. Regardless of the original format, an adapter would convert both into a standardized model with uniform field names like shipmentId, status, and deliveryDate, enabling the app to show the data with ease.

According to Sharma (2023) TasTech's development team can manage discrepancies between data sources, streamline the integration of new data sources, and preserve a clear division of responsibilities between the app's internal logic and external data sources by employing the Adapter Pattern.

## 6.3 Cloud architecture

For this application, **Firebase** is being used as the Backend-as-a-Service (BaaS) platform because of its extensive toolkit, which is perfect for mobile applications. According to Elkan (2018) in order to manage admin and customer operations without having to maintain a complicated backend architecture, Firebase offers real-time databases, authentication, and cloud functions. Real-time syncing is one of the main advantages of using Firebase. Customers can view tracking status changes instantly without having to manually reload the app because of this. Additionally, Guriev (2021) notes that firebase authentication accelerates and secures user login management for administrators and customers equally. Lastly, because Firebase automates a great deal of backend functionality, it requires minimal maintenance. As a result, instead of having to spend time setting up and maintaining servers, the development team was able to concentrate more on creating the essential aspects of the program.

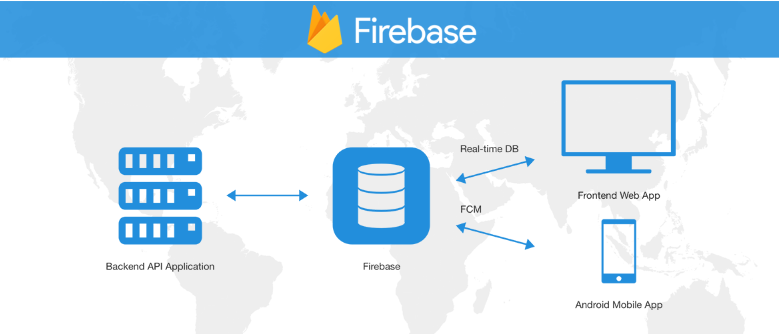


Figure 1: firebase architecture pattern (Elkan, 2018)

# 7. SECURITY

Over the course of development and deployment, a number of security measures were put in place to ensure the security of admin and customer information as well as the general integrity of the application. Significant security issues including networking, vulnerabilities, and potential threats are addressed by these security concerns. The security measures are as follows:

## 7.1 Authentication and Authorisation

* **Role Based Access Control (RBAC)**

Strict role-based access control was implemented by the application to ensure that only authorised users can access certain features. Admins can create and update tracking data as they have more privileges than customers. Customers only have access to the tracking details that are specific to them.

* **Secure Authentication**

To guarantee that passwords are secure even in the event that the database is compromised, industry-standard hashing and salting algorithms such as bcrypt and PBKDF2 can be used. To provide a further level of protection, multi-factor authentication (MFA) will also be taken into consideration for admin logins.

## 7.2 Data encryption

* **In transit encryption**

All data exchanged between the client and the server can be encrypted using SSL/TLS to prevent man-in-the-middle (MITM) attacks. This ensures that sensitive data, such as login credentials and tracking information, cannot be intercepted by hackers.

* **At rest encryption**

Strong encryption techniques such as AES-256 were used to protect sensitive data kept in the database, including tracking information and individual user details. This guarantees that the data will not be readable without the decryption keys, even in the event that an attacker manages to access the database.

## 7.3 Network security

* **Firewall protection**

A firewall can be used to host the application in order to prevent unauthorised users from accessing the server. This firewall can be set up to only let through the traffic that is absolutely required to reach the server, blocking any unusual activities.

* **Secure API gateways**

We'll be using token-based authentication to secure API endpoints. Valid authentication tokens, which are time limited in order to reduce the frame of opportunity for session hijacking or token theft, will be required for each request.

## 7.4 Vulnerability Mitigation

* **Input validation and sanitisation**

In order to stop common web application vulnerabilities like SQL Injection and Cross-Site Scripting (XSS), all user inputs will be rigorously checked and sanitized. Strict input validation is enforced to reduce the likelihood of malicious code injection.

* **Dependency management**

To ensure that third-party libraries and frameworks used in the app are up to date and free from known vulnerabilities, regular audits will be carried out on them. Open Web Application Security Project (OWASP) dependency check and other tools can be used to find possible vulnerabilities in external packages.

## 7.5 Threat monitoring and incident response

* **Logging and monitoring**

All important actions, including the setting up of tracking numbers, data modifications, and successful and unsuccessful login attempts, may be fully logged by the program. Real-time monitoring of these logs can be done to look for unusual activity, including brute force attacks or unauthorised access attempts.

* **Intrusion detection system (IDS)**

Traffic can be monitored using an intrusion detection system (IDS) to look for strange patterns that might point to an attack. The development team can expect alerts so they can look into any potential threats.

* **Regular penetration testing**

We can prepare monthly penetration testing that simulate possible attacks on the system in order to proactively find vulnerabilities. This will assist in identifying weak spots before they are used by attackers.

## 7.6 Compliance and data privacy

* **Compliance with data privacy laws**

We can make sure that the app complies with all relevant privacy rules and regulations, including General Data Privacy Regulation (GDPR) and Personal Information Protection Act (POPIA), since it will manage customer information. For example, giving customers control over their personal data and the option to have their accounts or data deleted upon request.

* **Data minimisation**

Only the most basic information needed for tracking will be gathered and kept. To lessen the impact of any data breaches, as little personal information as possible will be disclosed.

# **8. DEV OPS**

To ensure rapid deployment of a high-quality application for TasTech, the team has integrated DevOps practices throughout the development lifecycle. The goal is to minimize downtime, improve quality, and enable quick, continuous deployment of new features and bug fixes. This includes:

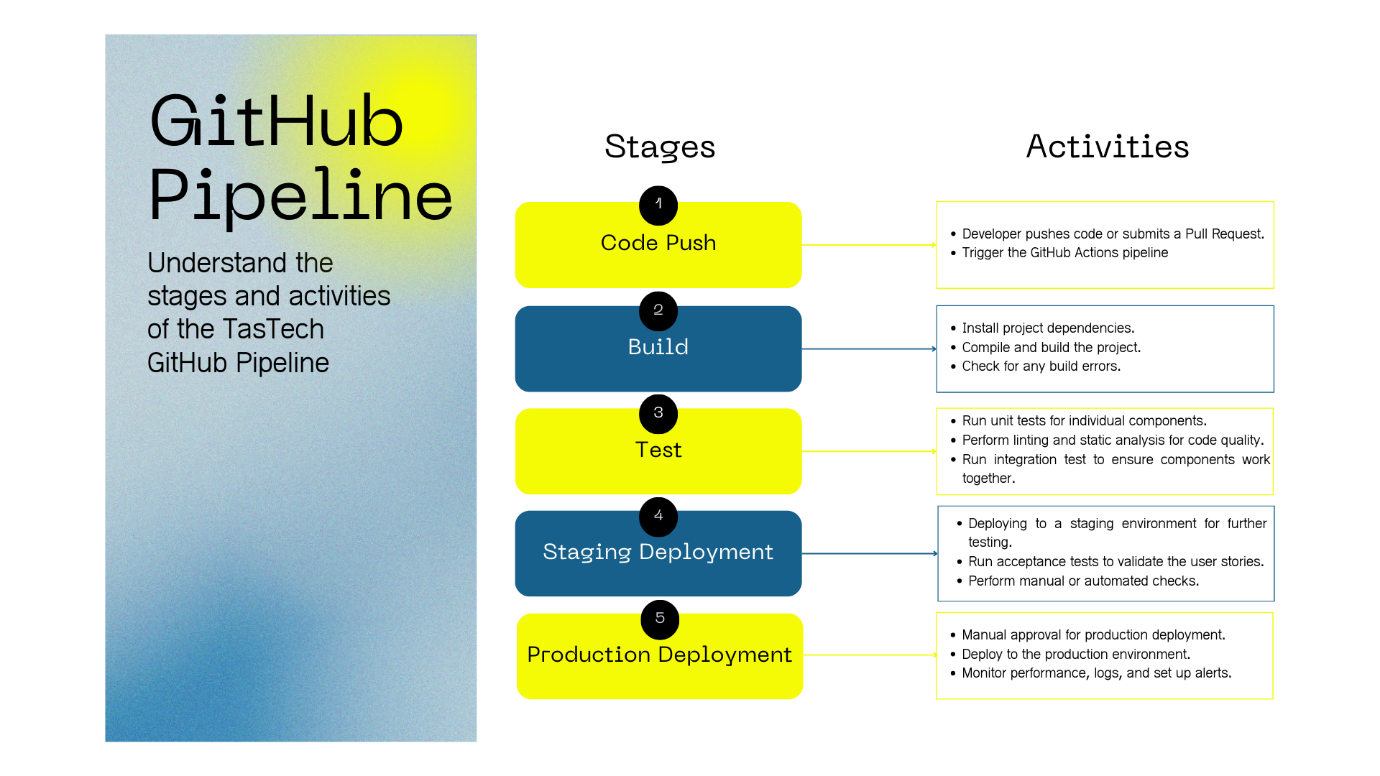
* **Continuous Integration/Continuous Deployment (CI/CD):** By automating the building, testing, and deployment process, we aim to release changes swiftly and with minimal manual intervention.
* **Automated Testing:** Unit, integration, and end-to-end tests are run automatically on every code commit to detect issues early.
* **Monitoring and Feedback Loops:** Application performance and errors will be monitored in real-time, allowing the team to respond to issues quickly and maintain high-quality service delivery.

## 8.1 GitHub Action Pipeline

The GitHub Actions pipeline we set up for our project automates key stages of the software development lifecycle. The process includes:

1. **Code Push**: When a developer pushes code to the repository, the pipeline is triggered.
2. **Build**: The application is built using the project’s dependencies and setup configurations.
3. **Testing**: Automated tests (unit, integration, and end-to-end) are executed to validate that new changes do not break existing features.
4. **Deployment**: If tests pass, the application is deployed to a staging environment. Manual approval triggers the final deployment to production.
5. **Monitoring**: Post-deployment, automated tools monitor the app for performance issues or errors.

## 8.2 Pipeline Diagram:



<https://www.canva.com/design/DAGR5kjsMxg/-5XypfR-XcrKJI8DK0nEAA/edit>

# 9. RUNNING COSTS

The monthly running costs will depend on infrastructure components such as cloud servers, storage, and databases. For TasTech’s order tracking software, we predict the following:

**Monthly Estimated Costs:**

* **Cloud Server** (e.g., AWS EC2 or Azure VM): R1,850 - R3,700 per month (depending on load).
* **Database** (e.g., AWS RDS or Azure SQL): R925 - R1,850 per month.
* **Storage** (e.g., S3, Blob Storage): R185 - R370 per month.
* **Monitoring & Logging** (e.g., CloudWatch or Azure Monitor): R555 - R925 per month.
* **Backup & Recovery**: R370 - R740 per month.
* **Google Play Store Developer Fee**: R370 once-off registration (initial cost).

**Growth Projections (2 Years):**

* **Year 1**: Initial costs (low traffic): ~R5,550 - R9,250/month.
* **Year 2**: With projected growth and increasing user adoption: ~R9,250 - R12,950/month, depending on the number of users, transactions, and storage needs.

# 10. CHANGE MANAGEMENT

For TasTech to successfully adopt the order tracking software, we plan to implement a structured change management strategy:

**Organizational Adoption:**

* **Why and How:** TasTech will adopt the software to streamline operations and improve service delivery. The software will reduce manual work and improve the customer experience. We will involve key stakeholders from the beginning, ensuring the system aligns with business needs. Training sessions and onboarding for admin staff will ensure smooth adoption.

**User Adoption:**

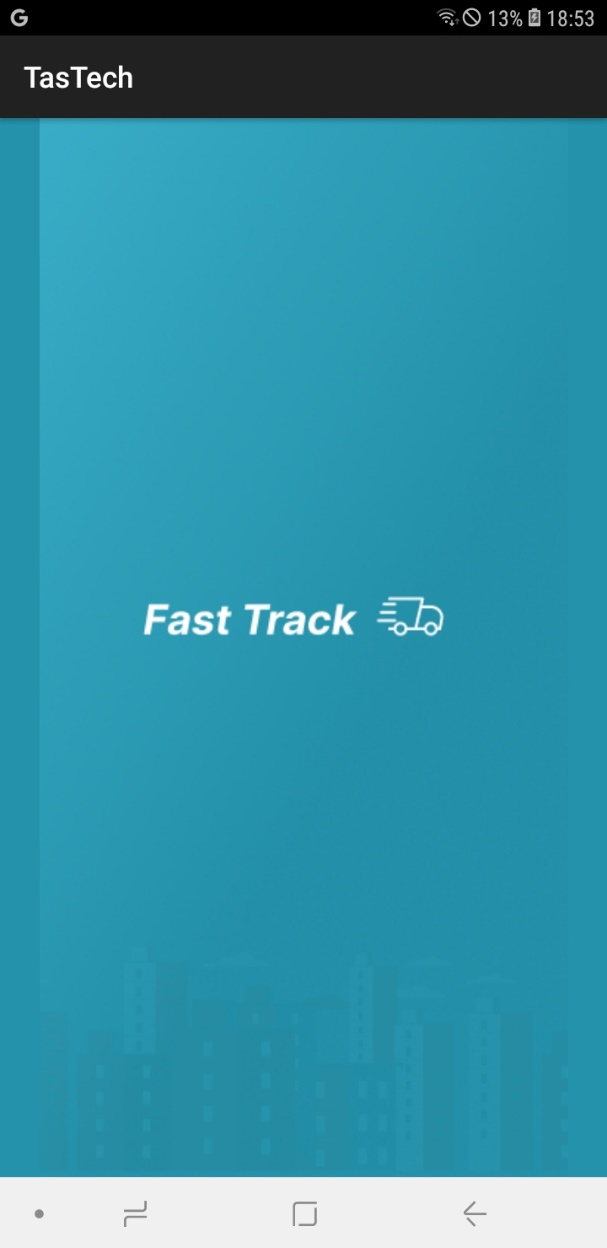
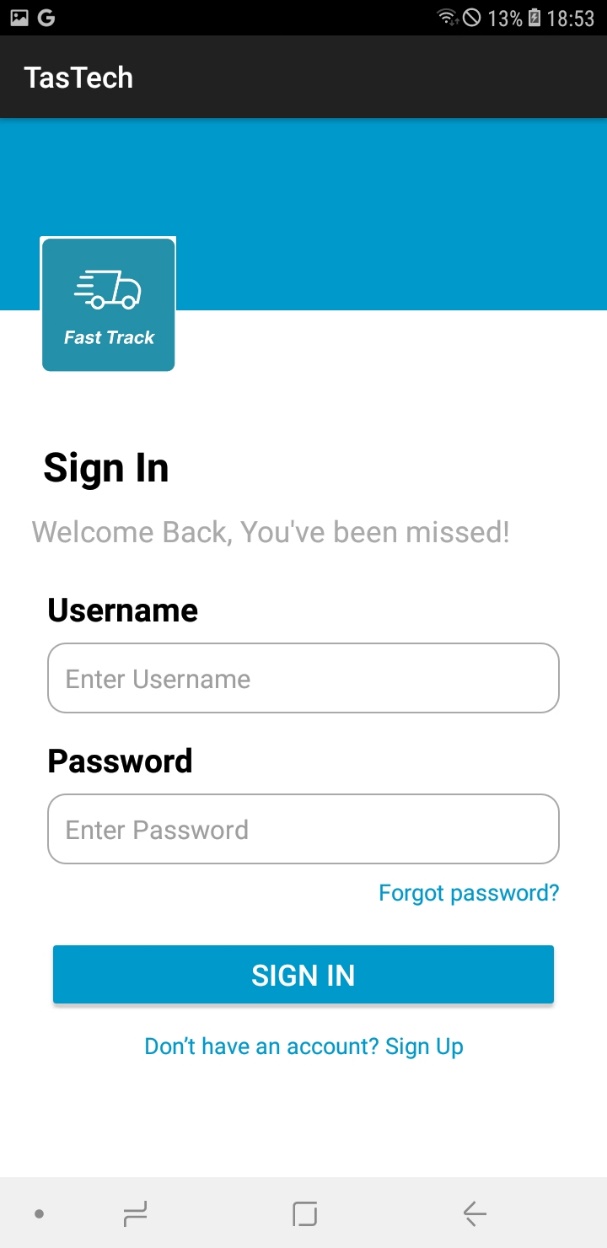
* **Why and How:** Customers will adopt the software because it provides an easy way to track orders without re-entering tracking information. By providing intuitive, user-friendly interfaces, and a smooth onboarding process, we aim to make the user experience seamless. Marketing efforts (e.g., emails, banners) will notify customers of the new system.

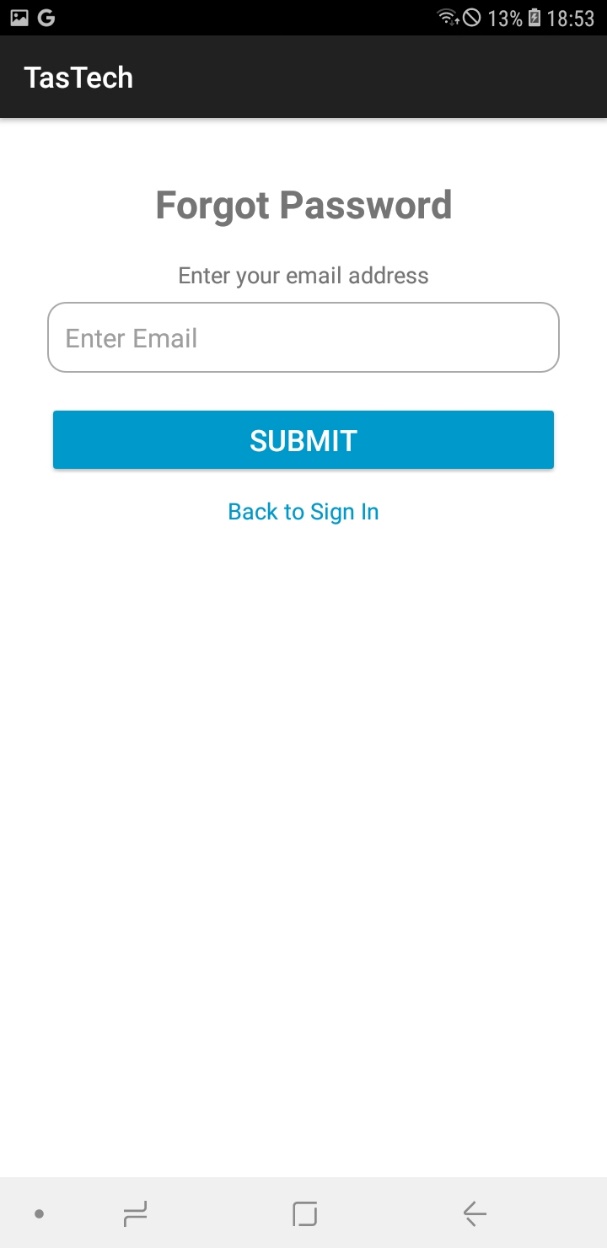
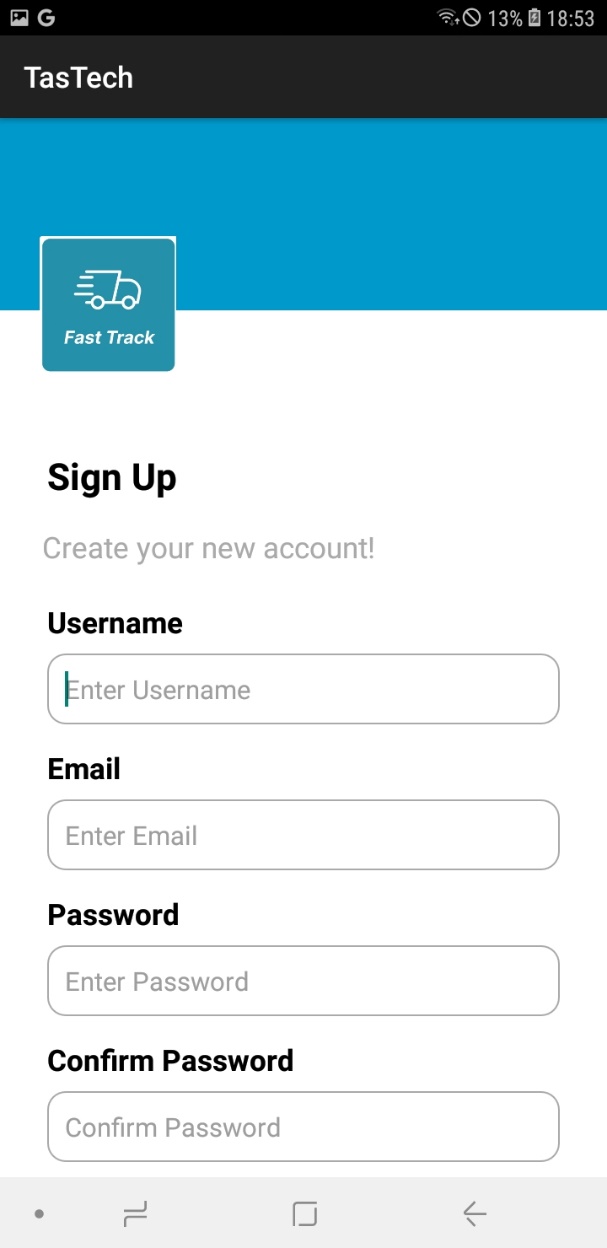
**Adoption Strategy:**

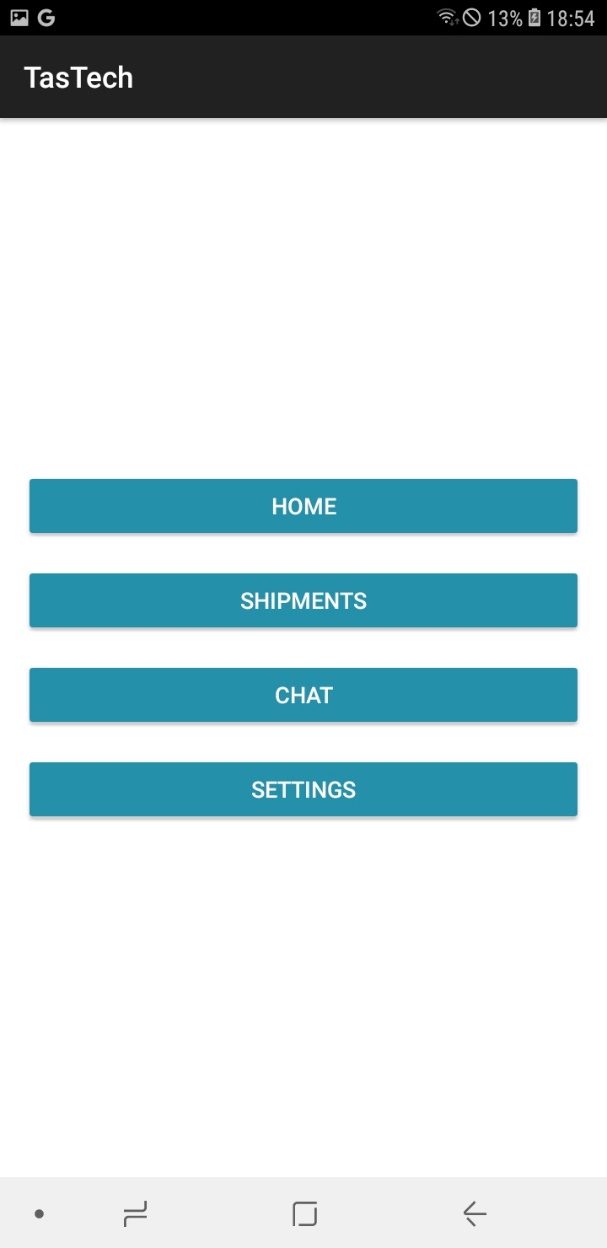
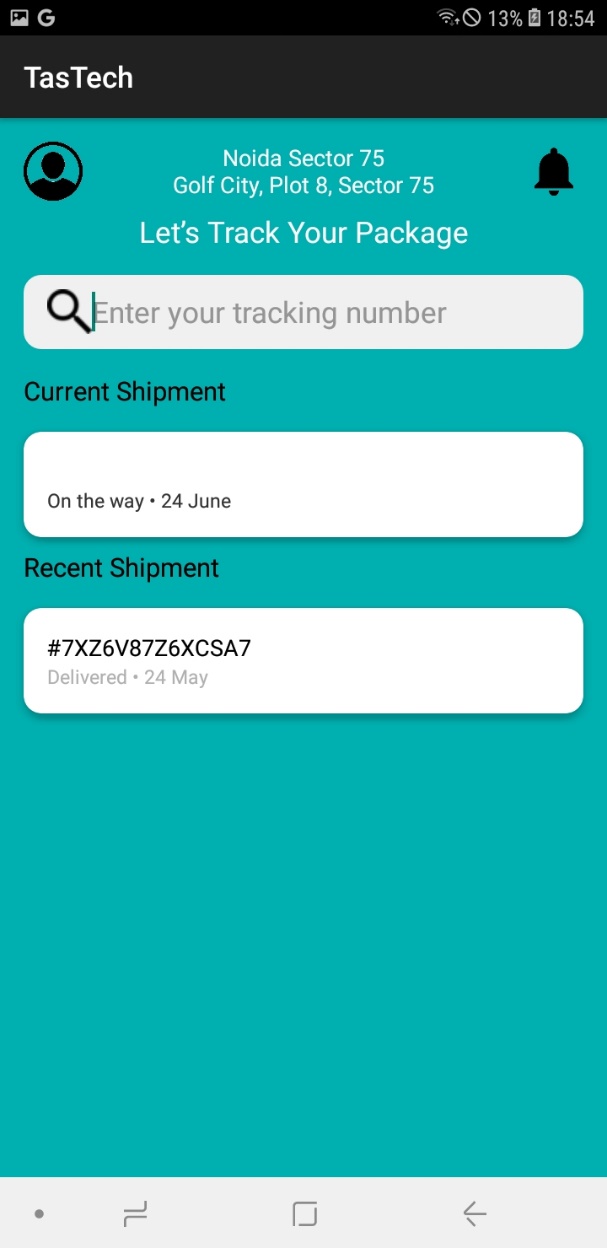
* **Communication:** We will communicate the benefits and changes via emails, announcements, and training sessions for both internal staff and customers.
* **Training and Support:** Admins will receive training on managing orders and generating tracking numbers, while customers will be provided with easy-to-follow guides. A support system will be available to resolve any issues.
* **Feedback Loops:** A feedback mechanism will allow users and admins to report issues, which will be swiftly addressed in future updates. Regular updates based on feedback will keep users engaged.

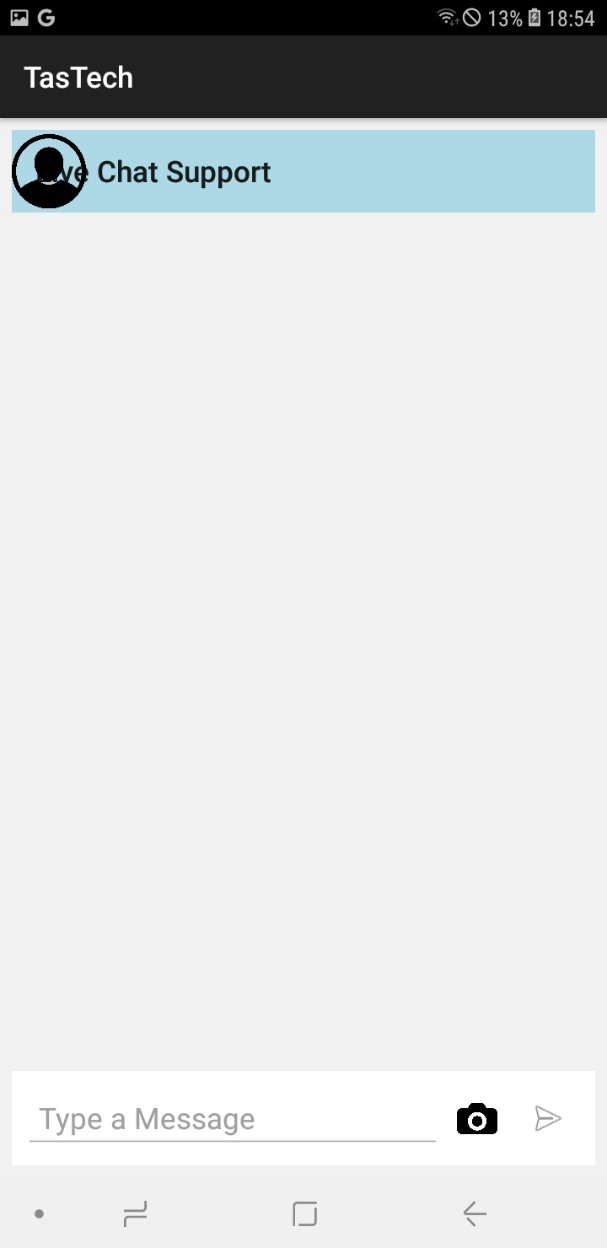
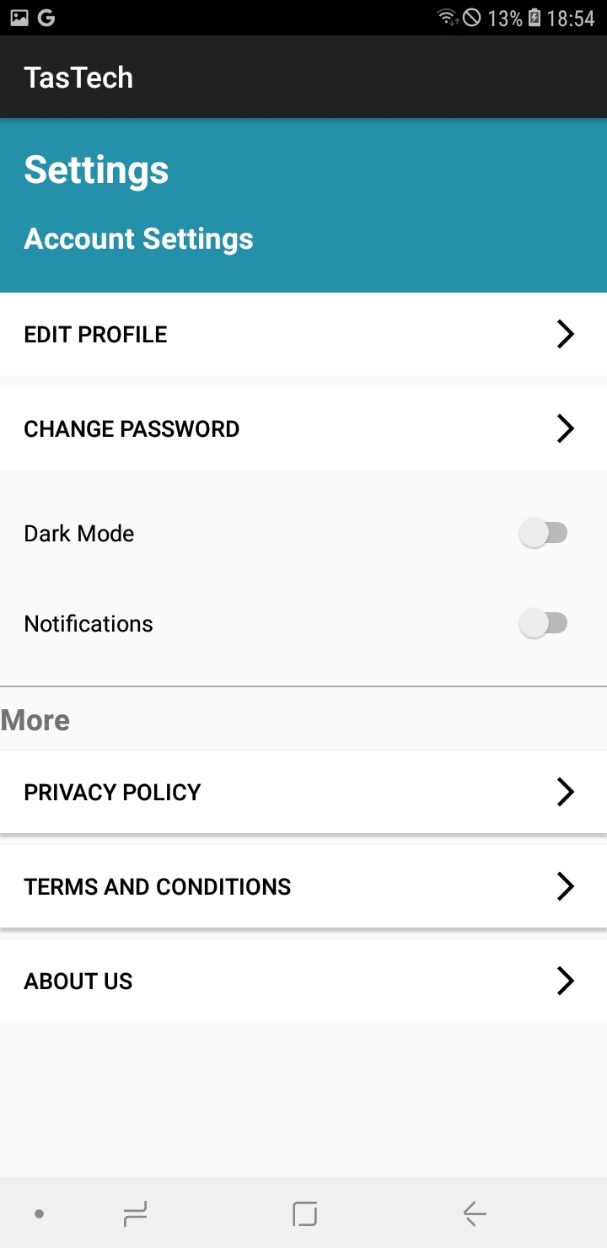
# 11. FAST-TRACK APPLICATION

## 11.1 Android app screenshots

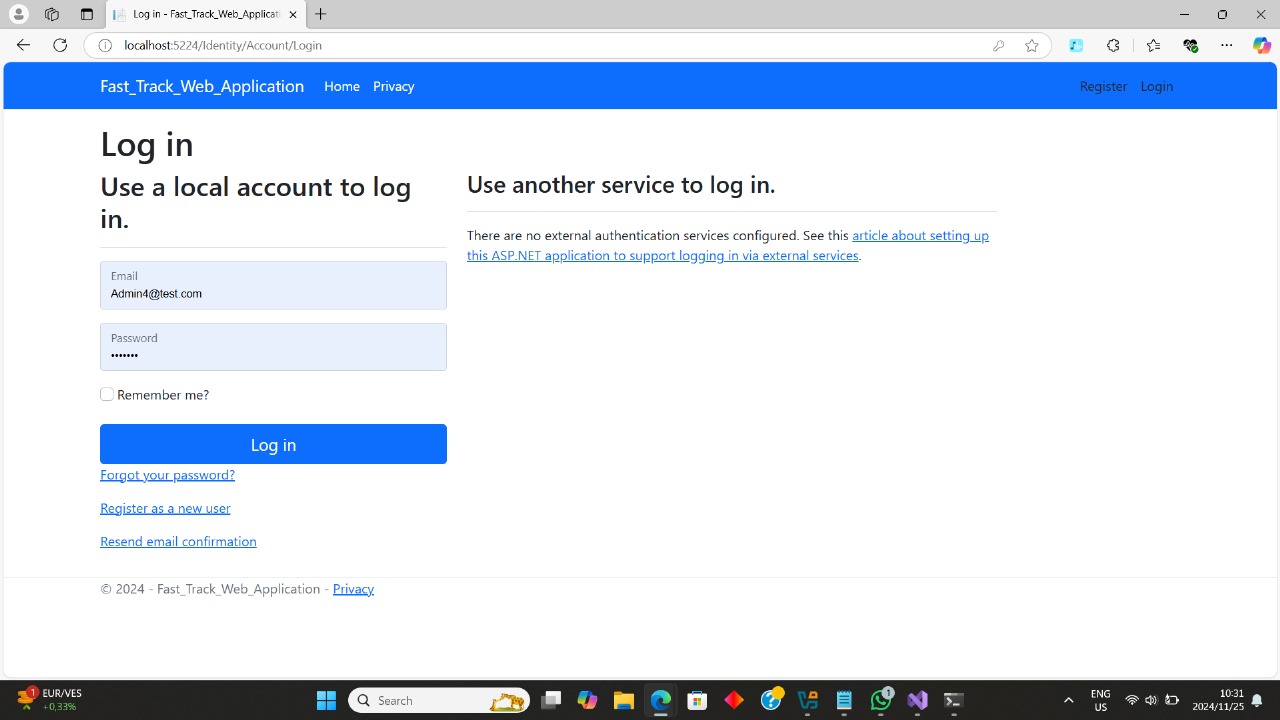
 

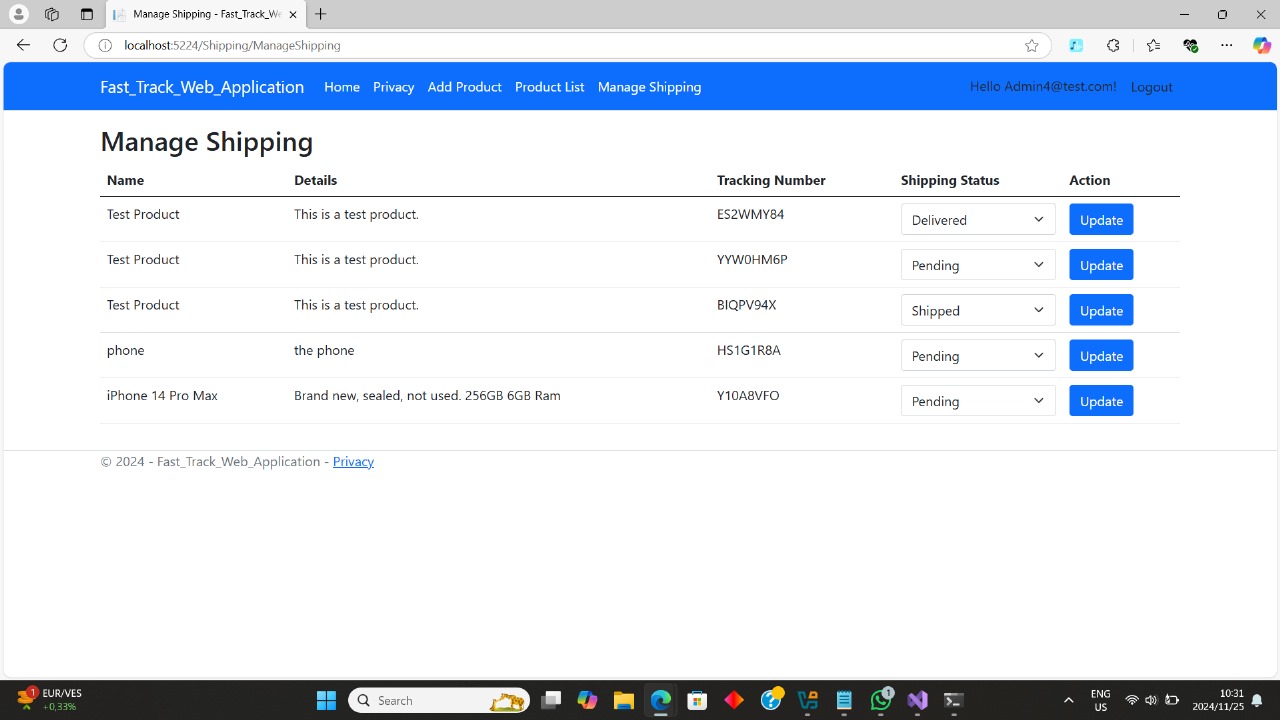
 

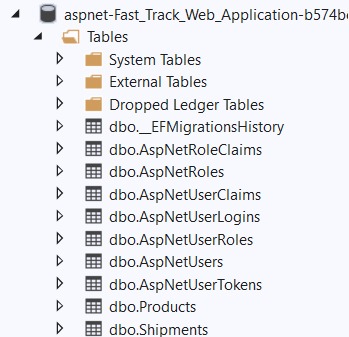
 

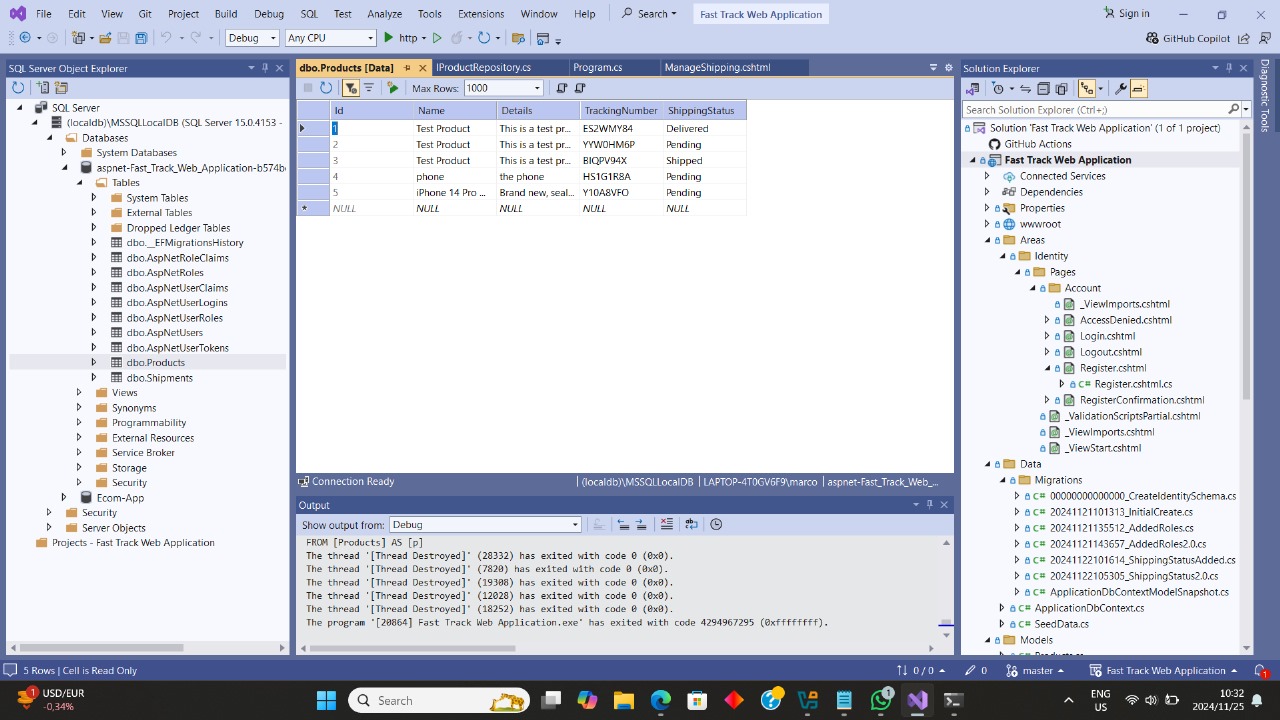
 

## 11.2 Web app screenshots









## 11.3 Project summary

The FastTrack application is a shipment tracking solution for TasTech customers made to make shipment tracking and administration easier. It is made up of an admin web application, a mobile application, and an API that acts as a bridge to connect the two. Users can log in, track current shipments, see previous shipments, and interact with live chat support with the Android mobile app, which was created in Kotlin using Android Studio. Unique Tracking numbers that are sent to users by admin are entered by users to keep track of shipment statuses, which are recorded and shown on the shipments page. The mobile application's backend, Firebase, offers safe user authentication and data storage. A C# web application was developed on the admin side to give administrators the ability to generate unique token numbers monitor user information, update statuses, and manage shipping records through a user-friendly dashboard. A custom API created with Node.js in Visual Studio Code allows the two apps to be smoothly integrated. Because of the safe data transmission provided by this API, the mobile app can display real-time admin-side updates. When combined, each of these components offer an efficient and practical means to track and manage shipments.

# 12. SELF REFLECTION

## 12.1 Leyaine Mungani – ST10091914

Work Integrated Learning is incredibly valuable because it bridges the gap between academic understanding and practical application. It enables students to obtain real-world experience in their field of choice, improving their professional development, technical knowledge, and problem-solving skills. Students can increase their job prospects, develop professional networks, and gain a better understanding of industry standards through practical learning in workplace environments. WIL promotes flexibility and teamwork. It also gives firms access to new talent, creative ideas, and the chance to train potential employees.

I gained a number of technical and soft skills during my work integrated learning experience in XBCAD7311, experiences that were essential to successfully finishing this project. I improved my knowledge of Kotlin and XML in particular, which I used to make useful and intuitive apps for Android. This involved creating responsive layouts with XML for smooth user experiences and implementing effective Kotlin code to implement app functionalities. I also became skilled in Firebase integration, which I utilize for backend services like cloud storage, real-time databases, and user authentication. For example, in order to safely store and synchronize user data in real-time, firebase was integrated into the FastTrack app. By setting up navigation graphs and NavHostFragment for seamless fragment transitions, I also improved my knowledge of the elements of navigation architecture. Projects like Fast Track, where my teammates and I used a BottomNavigationView for dynamic app navigation, showcased this. Using Android Studio tools like Logcat, I was able to fix and debug problems like white screen failures and layout inconsistencies. By assessing apps on BlueStacks and actual devices to make sure they satisfied usability and functionality criteria, I developed crucial skills in testing and deployment.

I worked with a group of talented individuals who brought diverse perspectives and expertise to our project. My team consisted of four members, each with specific responsibilities tailored to their strengths, such as backend development, frontend design, database management, and API integration. We reported to each other through our WhatsApp group chat. When it came to leadership duties, I frequently took the initiative to alert members about deadlines, and starting our next phase in the development process. Getting instruction was another important aspect of the process. My fellow teammates that are more experienced developers gave me advice, especially in areas like complex database operations and security best practices. I was able to apply effective approaches while understanding industry standards thanks to this coaching. I was able to contribute effectively and learn from others because of the balanced dynamic, my technical and social abilities were greatly improved by the collaboration and guidance I received.

I performed exceptionally well on responsibilities requiring technical problem-solving, project coordination, and backend programming during my work-integrated learning experience. By facilitating team conversations, breaking down complex requirements into manageable tasks, and making sure that everyone in the team was on the same page regarding the project's objectives, I also showed that I was a very good organizer. I also discovered that I thrived on front-end modification, which included creating visually beautiful and intuitive user interfaces, especially for apps that needed dark mode functionality and multilingual support. I was able to apply both my technical expertise and my innovative problem-solving skills to these assignments. However I found it quite difficulty mastering certain parts of WIL. For example, as I had little prior expertise with advanced security measures, such as biometric authentication and encryption for sensitive data, their implementation took a great deal of research and testing. Additionally, I initially found it difficult to manage database synchronization in offline modes using RoomDB or SQLite because it turned out to be more complicated than I had expected to ensure data consistency upon reconnection. I also found it very frustrating to debug navigation problems and fix unexpected app crashes in Android Studio because these issues called for a thorough understanding of system architecture and thorough attention to detail. Despite these challenges, I guidance from my teammates, which helped me overcome these obstacles and gain a better grasp of these advanced concepts.

In conclusion my journey of work integrated learning was stimulating and rewarding, offering priceless chances for both professional and personal development. It improved my technical abilities in database administration, system integration, and application development by enabling me to put my theoretical knowledge to use in real-world situations. I developed practical experience in developing user-centric solutions, including order management apps, while becoming proficient with Firebase, Android Studio, and RESTful APIs. Through active participation in team discussions and leadership roles, I also enhanced my communication, project coordination, and problem-solving skills.

## 12.2 Mzwandile Shinga – ST10210264

## 12.3 Ofentse Mogafe – ST10133493

## 12.4 Marco Lopes – ST10165142

# 13. REFERENCES

Elkan, M., 2018. *Firebase with.Net Core and C#.* [Online]   
Available at: <https://dot-net-box.blogspot.com/2018/12/firebase-withnet-core-and-c.html> [Accessed 24 September 2024].

Gallardo, E. G., 2023. *What Is MVVM Architecture?.* [Online]   
Available at: <https://builtin.com/software-engineering-perspectives/mvvm-architecture#:~:text=MVVM%20(Model-View-ViewModel)%20cleanly%20separates%20the%20business%20logic%20of> [Accessed 24 September 2024].

Guriev, B., 2021. *Firebase Pros and Cons: When You Should and Shouldn’t Use Firebase | OSDB.* [Online]   
Available at: <https://osdb.io/firebase-pros-and-cons-when-you-should-and-shouldnt-use-firebase-osdb/#:~:text=The%20extensive%20documentation,%20pre-made%20APIs,%20simple%20UIs,%20and> [Accessed 24 September 2024].

Muzammil, K., 2023. *Advantages of Firebase for Mobile App Development.* [Online]   
Available at: <https://www.aalpha.net/blog/firebase-in-mobile-app-development/#:~:text=Home.%20Advantages%20of%20Firebase%20for%20Mobile%20App%20Development>. [Accessed 24 September 2024].

Smith, A., 2024. *Repository Pattern.* [Online]   
Available at: <https://deviq.com/design-patterns/repository-pattern> [Accessed 24 September 2024].

Chauhan, A. 2024. *Overview of Firebase Backend as a Service Platform.* 04 May 2024. [Online]. Available at: <https://medium.com/@engrshul/overview-of-firebase-aa8e05710542>. [Accessed 25 September 2024]

Zhang, E. 2023. *What is Role-Based Access Control (RBAC)? Examples, Benefits, and More.* 05 May 2023. [Online]. Available at: <https://www.digitalguardian.com/blog/what-role-based-access-control-rbac-examples-benefits-and-more>. [Accessed 25 September 2024]

De Groot, J. 2024. *What Is Data Encryption? (Definition, Best Practices & More).* 25 September 2024. [Online]. Available at: <https://www.digitalguardian.com/blog/what-data-encryption#:~:text=Data%20encryption%20translates%20data%20into,unencrypted%20data%20is%20called%20plaintext>. [Accessed 25 September 2024]

**Gallardo, J. 2023.** Android Architecture: MVVM Design Pattern. [Online]. Available at: <https://developer.android.com>. [Accessed 26 September 2024]

Bergman, P. 2017. *Repository Design Pattern*. 20 April 2017. [Online]. Available at: <https://medium.com/@pererikbergman/repository-design-pattern-e28c0f3e4a30>. [Accessed 26 September 2024].

Kruglyak, O. 2024. Firebase Backend Platform: How to Reduce Expenses for Mobile Development. 14 March 2024. [Online]. Available at: <https://triare.net/insights/firebase-backend-platform-to-reduce-expenses-for-mobile-dev/#:~:text=Firebase%20makes%20app%20development%20convenient,easy%2Dto%2Dlearn%20technology>. [Accessed 26 September 2024].

Laurent, L. 2023. *How to Secure Your App Effectively with Firebase?* 03 November 2023. [Online]. Available at: <https://appmaster.io/blog/how-to-secure-your-app-with-firebase>. [Accessed 26 September 2024].

Non functional requirements, 13 October 2023, <https://scaledagileframework.com/nonfunctional-requirements/#:~:text=Details.%20Nonfunctional%20Requirements%20(NFRs)%20are%20intended%20to%20specify%20%E2%80%98system%20qualities,%E2%80%99>

PortSwigger, https://portswigger.net/web-security/authentication/multi-factor#:~:text=In%20this%20section,%20we'll%20look%20at%20some%20of%20the%20vulnerabilities

Sharma, A., 2023. *Adapter Design Pattern in Java.* [Online]   
Available at: <https://medium.com/@akshatsharma0610/adapter-design-pattern-in-java-fa20d6df25b8> [Accessed 21 November 2024].

# 14. WORKS CITED

* Amazon. “How Can I Avoid Charges on My Account When Using AWS Free Tier Services?” *Amazon Web Services, Inc.*, 2023, aws.amazon.com/free/?gclid=Cj0KCQjwjNS3BhChARIsAOxBM6ppDCSLCGQUZnFUI43VNLOow8oibn1zK3eAwdUlVgF2SoFPQVxifWYaAqVJEALw\_wcB&all-free-tier.sort-by=item.additionalFields.SortRank&all-free-tier.sort-order=asc&awsf.Free%20Tier%20Types=. Accessed 26 Sept. 2024.
* AWS. “Amazon CloudWatch - Application and Infrastructure Monitoring.” *Amazon Web Services, Inc.*, 2024, aws.amazon.com/cloudwatch/. Accessed 27 Sept. 2024.
* ---. “What Is DevOps? - Amazon Web Services (AWS).” *Amazon Web Services, Inc.*, Amazon, 21 May 2022, aws.amazon.com/devops/what-is-devops/#:~:text=DevOps%20is%20the%20combination%20of. Accessed 27 Sept. 2024.
* Douglas, Brian. “How to Build a CI/CD Pipeline with GitHub Actions in Four Simple Steps.” *The GitHub Blog*, 2 Feb. 2022, github.blog/enterprise-software/ci-cd/build-ci-cd-pipeline-github-actions-four-steps/. Accessed 26 Sept. 2024.
* GitHub. “GitHub Actions Documentation - GitHub Docs.” *Docs.github.com*, 2024, docs.github.com/en/actions. Accessed 27 Sept. 2024.
* IBM. “What Is DevOps? | IBM.” *Www.ibm.com*, 27 Sept. 2024, www.ibm.com/topics/devops. Accessed 27 Sept. 2024.
* Lawton, George . “What Is Change Management? - Definition from WhatIs.com.” *SearchCIO*, 2023, www.techtarget.com/searchcio/definition/change-management#:~:text=Change%20management%20is%20a%20systematic. Accessed 27 Sept. 2024.
* microsoft. “Pricing Overview—How Azure Pricing Works | Microsoft Azure.” *Microsoft.com*, Microsoft Azure, 2018, azure.microsoft.com/en-us/pricing/. Accessed 27 Sept. 2024.
* Murphy, Chris. “How to Calculate and Analyze a Company’s Operating Costs.” *Investopedia*, 17 Mar. 2022, www.investopedia.com/terms/o/operating-cost.asp. Accessed 27 Sept. 2024.