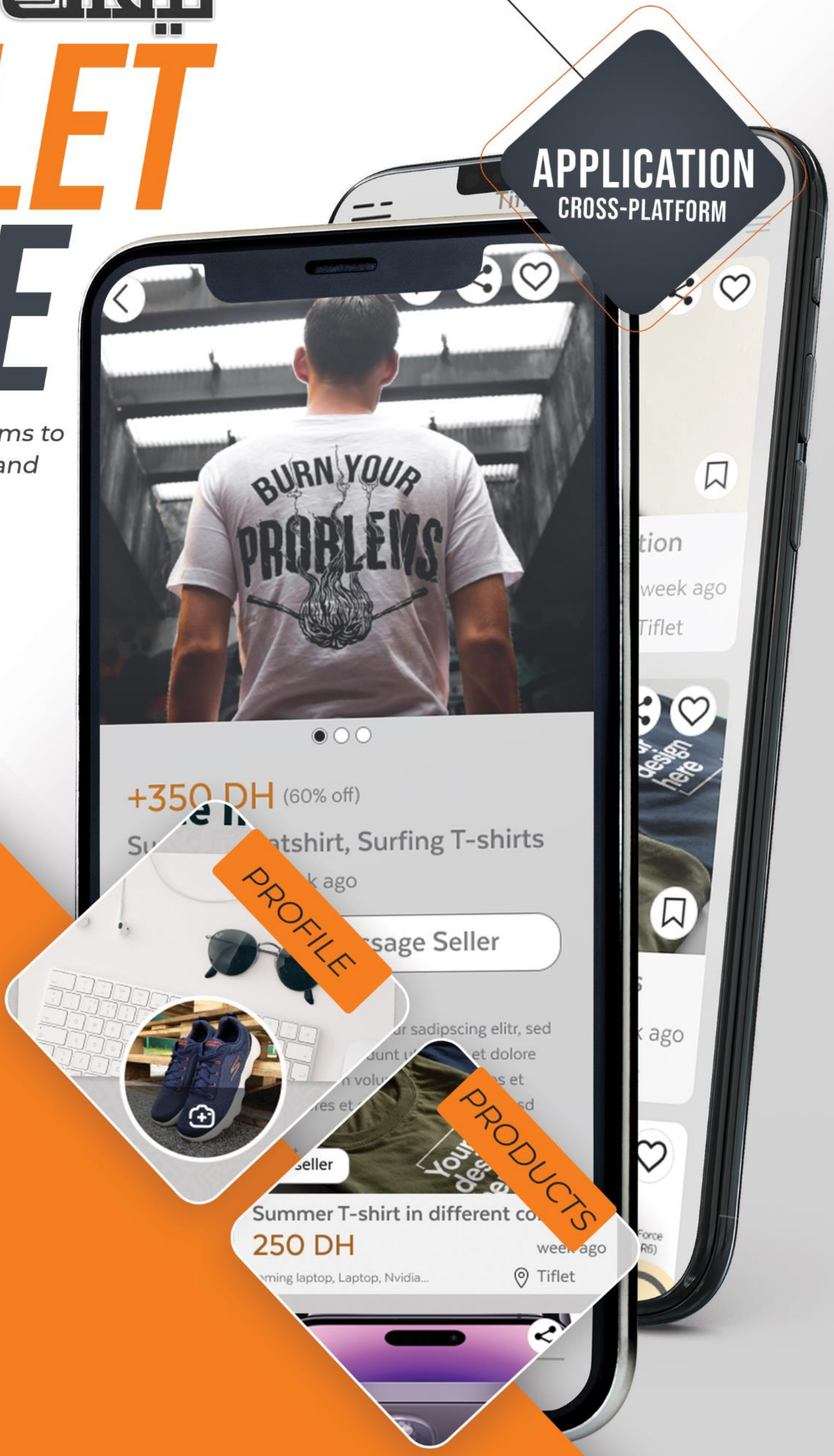
TIF PRESS
ALL ABOUT TIFLET PRESS

FOLLOW US NOW

تيفلت MEDIA

TIFLET STORE

cross-platform store application aims to bridge the gap between sellers and buyers



PROJECT GRADUATION

- Supervisor
  ERRAIS MOHAMMED
- External Supervisor
  BOUARFI ABDELALI
- Directed By
  ELOURAINI ANAS

VINCI
Ecole Supérieure

ACADEMIC YEAR 2022 - 2023

WWW.TIFELTPRESS.COM

ACKNOWLEDGMENT

I would like to express my deep gratitude to all the individuals who played a role, directly or indirectly, in making my internship project a reality. Their support and collaboration have been of immeasurable importance and have greatly contributed to the success of my professional experience.

I want to extend my heartfelt thanks to the administration of my university, especially to [Mr. El Mehdi BENDRISS] and [Mr. Karim RACHDI], for their guidance, availability, and support throughout my academic journey. Their expertise and insightful guidance have significantly enriched my education and played an essential role in the success of my internship.

I also wish to convey my profound appreciation to my university professors, who imparted valuable knowledge to me and encouraged me to excel. Their inspiring lectures and constructive feedback played a pivotal role in my understanding of key concepts and the development of my skills.

I want to give special thanks to my mentors, starting with [Mohammed ERRAIS], my internal university mentor. His patience, expertise, and enlightened advice were invaluable throughout my project. I am grateful for his unwavering support and for sharing his in-depth knowledge with me.

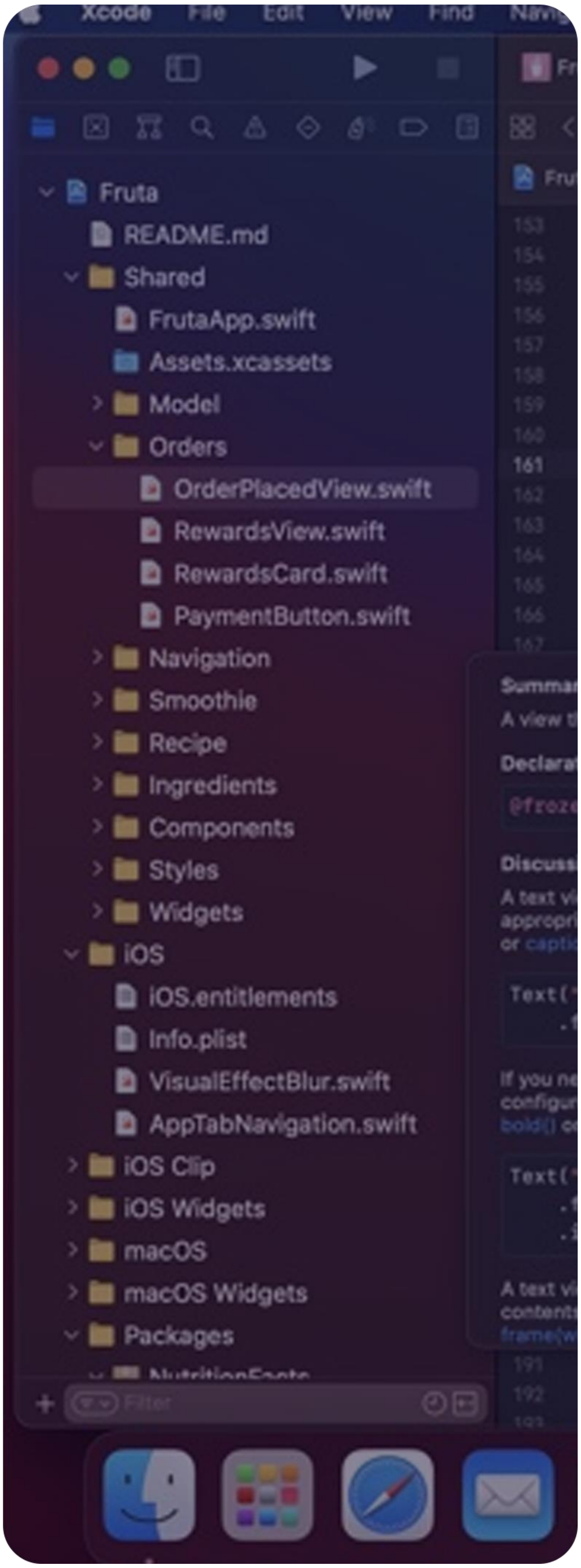


TABLE OF CONTENTS

INTRODUCTION

In the heart of Tiflet City, a new era of commerce is dawning with the arrival of Tiflet Store—an intuitive cross-platform application designed to redefine the way products are bought and sold. This app is set to revolutionize local trade by making it accessible, user-friendly, and more convenient than ever before.

Driven by the overwhelming demand from the Tiflet Press community, the visionary behind this initiative has taken a strategic step towards creating a digital marketplace. Tiflet Store empowers individuals and businesses alike, providing them with a straightforward platform to effortlessly list their products for sale, connecting sellers with buyers right within the local community.

At its core, Tiflet Store aims to bridge the gap between sellers and buyers, fostering a sense of community commerce. It simplifies the process of showcasing and purchasing products, putting a wide range of offerings at your fingertips.

With a stylish and intuitive interface, Tiflet Store's Home page offers a visually captivating experience, highlighting the latest additions to its ever-expanding inventory. As users delve further into the app, they discover a curated selection of trending products, ensuring they stay in the loop and never miss out on the hottest items in town.

Tiflet Store's search feature empowers users to easily find products of interest, making it simple to locate similar-named items listed by various sellers.

The product page offers a wealth of information, including multiple images, ratings with feedback, listing date, stock availability, location, detailed descriptions, and insights into the seller. Users can even navigate to the seller's profile page by clicking on their image.

Tiflet Store not only connects buyers and sellers but also showcases seller profiles. These profiles display public information, including images, names, and follower counts, enhancing user interaction. Users can follow sellers, view contact information, and explore the seller's product listings.

Tiflet Store introduces two distinct user roles: sellers and buyers. Sellers have dedicated profiles to showcase their products, while buyers enjoy access to a curated selection of products within the Tiflet community.

Tiflet Store isn't just an application; it's a transformative force that brings ease, accessibility, and a sense of community to commerce in Tiflet City. It redefines the way products are bought and sold, making it a powerful addition to the city's digital landscape.

PROJECT FRAMEWORK AND OBJECTIVES

The Tiflet Store project is a visionary cross-platform application developed to transform the way products are showcased and sold in Tiflet City. Leveraging a robust technology stack, the application aims to seamlessly connect sellers and buyers within the local community, offering a user-friendly platform for showcasing a diverse range of products.

Developing this new application involved several key components. Firstly, Qt and QML were the primary technologies used to establish the application's logic and structure. Qt, along with QML, offers powerful capabilities and a user-friendly approach, enabling the creation of clear, modular, and maintainable code for cross-platform development.

C++ stands out as the primary development language for crafting the core functionality and features of the Tiflet Store application. With its performance-oriented nature and extensive libraries, C++ allows for the creation of a responsive and efficient cross-platform app. This language choice ensures that the application operates smoothly and responsively on various devices and platforms.

At the heart of Tiflet Store's backend, PHP plays a crucial role, serving as the engine that drives data management and communication between the application's frontend and backend components. PHP provides a versatile and scalable foundation for handling user authentication, database interactions with MySQL, and the dynamic generation of content. It ensures a smooth and reliable experience for both sellers and buyers.

To facilitate seamless data exchange between the frontend and backend, JavaScript and JSON are employed. JavaScript, a versatile scripting language, is utilized for dynamic interactions and enhancing the user experience. JSON (JavaScript Object Notation) serves as a lightweight data interchange format, enabling efficient communication between the application's frontend and PHP-based backend. This combination ensures that data flows smoothly and consistently, allowing users to access product listings, make purchases, and interact with sellers effortlessly.

Data management played a pivotal role in this application, with MySQL serving as the selected database management system. MySQL offers a robust and secure platform for storing, managing, and retrieving the data required for effective store management. It also provides advanced features such as SQL support, index management, stored procedures, and transaction management.

In addition to the core store management functionalities, this new application included extra features to enhance the user experience. These included a user-friendly interface, access rights management, report generation, and data visualization. These features were implemented using web technologies such as HTML and CSS, seamlessly integrated with Qt and QML, to ensure a consistent and user-friendly experience across all supported platforms.

In summary, the project aimed to develop a cross-platform store application using Qt and QML as the primary frameworks. It was designed to provide modern and efficient solutions to assist businesses in effectively managing their stores. Through the utilization of these technologies and the implementation of a robust architecture, the project aimed to deliver a reliable, user-friendly application tailored to the specific needs of businesses in their daily store management operations.

WORK METHODOLOGY

The successful execution of any project hinges on effective organization and a well-structured work methodology, ensuring optimal results while adhering to the stipulated deadlines outlined in the project's specifications.

My final-year project encompasses four key dimensions:

1. **Functional**: Addressing specific needs and requirements.
2. **Technical**: Ensuring compliance with implementation constraints and specifications.
3. **Organizational**: Adhering to the operational framework of the target structure.
4. **Timelines**: Meeting project deadlines and schedules.

In this context, I will outline the development process followed and provide justifications for this chosen approach.

Agile Approach

The Agile approach is a project management methodology that emphasizes flexibility, adaptability to change, and continuous collaboration with project stakeholders. In contrast to traditional project management approaches that prioritize extensive planning and task prioritization, the Agile approach encourages the project team to work iteratively and incrementally. It focuses on delivering small, incremental features at regular intervals. This approach aligns with my project as it ensures responsiveness to evolving requirements and efficient collaboration throughout the development process.

- Figure 1 Agile Methodology

Scrum Methodology

Scrum is an agile software development methodology designed to deliver results incrementally and iteratively. It relies on the use of short and iterative development cycles known as "sprints," typically lasting between one and four weeks.

The Scrum methodology is founded on the principles of a self-organizing, cross-functional, and autonomous team, comprising developers, a Scrum Master, and a Product Owner. The Scrum Master facilitates the development process and resolves impediments, while the Product Owner is responsible for defining business objectives and product requirements.

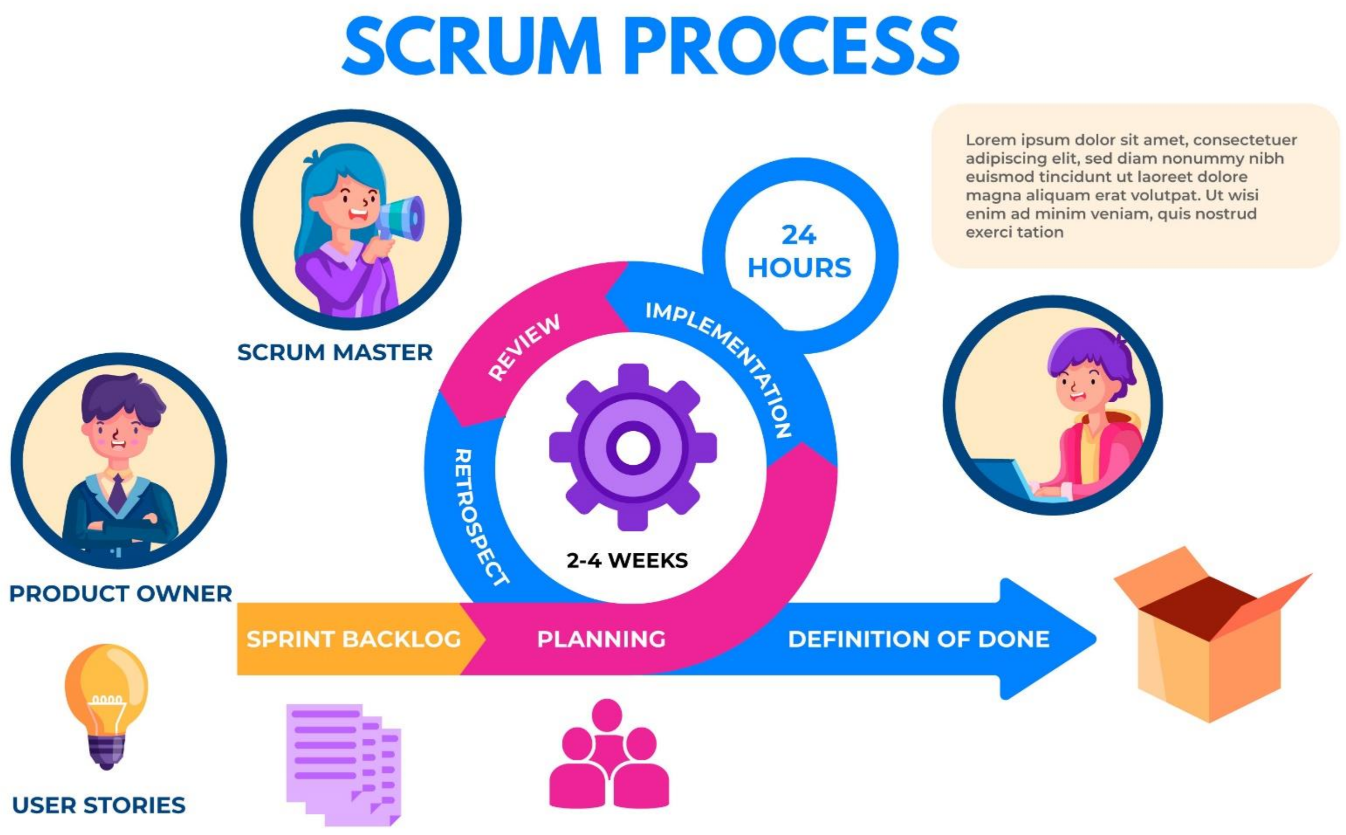

SCRUM PROCESS
Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation
24 HOURS
SCRUM MASTER
REVIEW
IMPLEMENTATION
RETROSPECT
2-4 WEEKS
PRODUCT OWNER
SPRINT BACKLOG
PLANNING
DEFINITION OF DONE
USER STORIES

Figure 2 Scrum Methodology

THE SCRUM PROCESS

- ***Sprint Planning***

The Scrum team initiates sprint planning, a period of work lasting two to four weeks during which the team will create a set of features known as an "increment." In this meeting, the team determines what can be accomplished during the sprint and sets a sprint goal.

- ***Sprint Backlog***

The Scrum team establishes a sprint backlog, which is a detailed list of tasks for the increment's elements. Each task must be clearly defined and assigned to a team member.

- ***Daily Scrum***

The Scrum team gathers daily for a meeting known as the Daily Scrum. During this brief meeting (lasting a maximum of 15 minutes), each team member answers three questions: What have I accomplished since the last meeting? What will I do before the next meeting? What obstacles am I encountering?

- ***Sprint Review***

At the end of the sprint, the Scrum team conducts a sprint review where they showcase the produced increment and collect feedback from stakeholders.

- ***Sprint Retrospective***

Following the sprint review, the Scrum team holds a sprint retrospective to discuss the successes and failures of the previous sprint and establish an action plan to enhance the process for the next sprint.

- ***Product Backlog***

The Scrum team collaborates with the Product Owner to create and maintain a Product Backlog, a prioritized list of all the features the product must include.

- ***Next Sprint***

The Scrum team repeats the cycle of planning, development, review, and retrospective for each subsequent sprint, relying on stakeholder feedback and continuously improving its process.

- ***Roles***

The SCRUM team comprises the following roles:

- ***SCRUM Master:***

The SCRUM Master is responsible for ensuring the smooth operation of the SCRUM agile methodology within the team. They strive to ensure that their team operates at 100% capacity to deliver a superior product within the specified timeframe. They are considered the team's coach.

- ***Product Owner:***

The Product Owner is the "owner" of the product and serves as the client's representative. They collaborate with the development team, providing instructions that the team must follow. They validate completed features.

- ***Proxy Product Owner:***

The Proxy Product Owner acts as the right hand of the Product Owner when the latter is unavailable due to other commitments. This role aims to bridge gaps that might hinder project progress. The Proxy Product Owner ensures smooth communication and decision-making.

Figure 3 <PPO> Proxy Product Owner

- ***QA Team (Quality Assurance Team):***

The QA team is responsible for testing the software throughout its development, delivering reports that highlight any regressions to ensure the product functions correctly in each iteration.

- ***Development Team:***

This team is responsible for product development. They must deliver a product increment at the end of each iteration, following the guidance of the Product Owner and the SCRUM Master. The team is often self-organized and multidisciplinary to cover various aspects of development.

1. FUNCTIONAL ANALYSIS

Functional specifications play a vital role in shaping the Tiflet Store application's development and functionality. These specifications outline the specific features and expected behaviors of the application, providing a comprehensive overview of what the application must accomplish to meet the needs of its users and the Tiflet City community.

- Product Listing and Management (Stock): Tiflet Store facilitates the listing and management of products by sellers. Sellers can add, edit, hide, or delete their product listings. Each product listing includes essential details such as images, pricing in local currency, product name, description, and the option to express interest.

- User Management (Customers): The application allows users to create and manage their profiles. Users can customize their profiles with images, display essential information like their name and location, and interact with other users by following and contacting them. Special recognition is given to profiles with substantial follower counts.

- Seller Features (Sellers): Tiflet Store distinguishes between sellers and buyers. Sellers have dedicated profiles showcasing their products. They can manage their store descriptions, opening dates, and product listings. Additional features allow sellers to update, hide, or delete their products directly from their profiles.

- Favorite Products (Management): The "Favorite" functionality provides users with quick access to all the products they have saved for future reference.

These functional specifications provide a solid foundation for the development of Tiflet Store, ensuring that it meets the expectations and needs of its users and serves as a seamless platform for the Tiflet City community to showcase and discover a wide range of products.

Case Study

- **Data Dictionary**

#	Property	Signification	Type	Note
1	idProduct	Product identifier	Int	Automatic number that increments after each addition
2	nameProduct	Product name	String	Ex: "Lenovo Legion 5"
3	descProduct	Product Description	String	
4	imgProduct	Product Images	List<Image>[3]	Each product can have one to three images
5	priceProduct	Product Price	Double	
6	idSeller	Seller Identifier	Int	references the user who listed the produc
7	~~stockAvailability~~	~~Avible on stock or not~~	~~Bool~~	~~if Avible show stock number if not show out of stock~~
8	locationProduit	the city the product listed in	String	The name of the city the product selller in

9	ListedDate		DateTime	The date the product listed
~~10~~	~~allRating~~	~~Rating List~~		~~Rating List related to the Product~~
~~11~~	~~countRating~~	~~count~~	~~Int~~	~~count of rating related to this product~~
~~12~~	~~averageRating~~		~~Double~~	~~The average rating of product~~
13	idRating	Rating Identifier	Int	Primary key Automatic number that increments after each addition
	idUser	User Identifier	Int	Foreign Key of the user
14	idProduit	produit Identifier to specific the product rating Rating	Int	Foreign Key of the product
15	descRating	Rating Feedback	String	
16	numRating	Number of start	Double	Ex: "4.3/5"
17	imgRating	Rating with Image	Image	
18	idUser	User Identifier	Int	Primary key Automatic number that increments after each addition
19	nameUser	the user name of the user	String	Unique key
20	emailUser	the user email	String	unique key

21	phoneNumberUser			
22	passwordUser	Hashed and stored securely	String	The password will be Hashed and Stored as a String
22	roleUser	Even the user is a seller or a buyer	bool	The user can be even a seller or a buyer
23	profileImageUser	user profile image	List<Image>[]	Profile image can be null and it will hold a list of all the images the user has updated by time
24	backfroundImageUser	User Profile background image	List<Image>[]	Background Profile image the same as Profile main image
~~25~~	~~bioUser~~	~~The user Bio~~	~~String~~	
~~26~~	~~locationUser~~	~~User Location~~	~~String~~	~~The user location and by that I mean the city of the user~~
27	followerCountUser	The count of the profile followers	int	
28	Valid Badge		Bool	This Badge determine if you are a valid account and you get this Badge wen you have more than 20,000 followers
29	contactInfoUser	The phone number that people will call you on	String	The phone number will display in the profile and in every product

30	registrationDateUser	The date you Register in the app	DateTime	

- **Relationships**

1. User-Seller Relationship:
    - A user can be a seller and have multiple listed products.
    - A seller can have multiple products for sale.
    - Seller ID in the Product entity references the User entity.
2. User-Buyer Relationship:
    - A user can be a buyer and save multiple products.
    - A buyer can have multiple saved products.
    - Buyer information is stored in the User entity.

Analysis and Design

In a software development project, the stages of analysis and design lay the foundation and define the objectives of the system development process. They provide an overarching view of the project and outline the steps and principles that will be followed to conceive and implement the solution.

In this initial phase, analysis involves understanding the project's needs, constraints, and objectives, as well as the expectations of stakeholders. It entails collecting and analyzing information related to the project's context, users, required functionalities, and specific requirements.

Design, on the other hand, focuses on crafting a technical solution to address the identified needs. This entails creating models, diagrams, and detailed specifications to describe the system's architecture, data structure, user interfaces, workflows, and more. Design considerations encompass aspects such as modularity, reusability, performance, security, and user-friendliness.

The goal of analysis and design is to establish a solid foundation for system development, providing a clear vision of both functional and technical requirements. This helps mitigate the risks of misunderstanding requirements, design errors, and long-term maintenance issues.

Furthermore, analysis and design also offer valuable insights for project planning, estimating necessary resources, and developing a realistic timeline.

In summary, introducing analysis and design into a software development project is crucial for establishing a clear understanding of project needs and defining an appropriate technical solution. It marks the beginning of the development process and ensures that all stakeholders share a common vision of the system to be developed.

I. UML2 Modeling Language

The Unified Modeling Language (UML) is a set of integrated diagrams used by software developers for visually representing objects, states, and processes within software or systems. This modeling language serves as a blueprint for a project, ensuring a structured information architecture and helping developers convey system descriptions in an understandable manner to external specialists. UML finds significant use in object-oriented software development, and enhancements made in version 2.0 have also made it suitable for representing management processes.

- **UML and Object-Oriented Programming (OOP)**

Object-Oriented Programming (OOP) is one of the primary programming paradigms in use today. In OOP, algorithms are formulated by defining "objects" and managing their interactions. Objects typically represent real-world entities and can span from physical objects like buildings to software elements such as desktop widgets or even abstract concepts.

UML (Unified Modeling Language) is closely intertwined with OOP. It emerged from the amalgamation of several object-oriented notations, including Object-Oriented Design (OOD), Object Modeling Technique (OMT), and Object-Oriented Software Engineering (OOSE). UML combines the strengths of these approaches to provide a coherent, unified, and user-friendly methodology for modeling software and enterprise systems.

UML offers a standardized and expressive graphical notation, enabling the representation of various modeling aspects, such as structure, behavior, relationships, and interactions between objects. It enables developers and software designers to communicate effectively, document systems and processes, and facilitate understanding and collaboration among development team members.

In summary, OOP and UML are closely intertwined concepts. OOP provides a programming paradigm based on object definition and manipulation, while UML offers a standardized graphical notation to represent and document object-oriented modeling aspects. Together, they facilitate the development of robust, modular, and scalable software by promoting a systematic approach and a better understanding of the systems being designed.

- **Advantage**

The Object Management Group (OMG), the organization that standardized UML, has clearly defined the language's purpose. UML was designed to provide system architects, software engineers, and application developers with tools for analyzing, designing, and implementing software systems. It is also used for modeling business processes and similar processes.

OMG views UML as a means to enhance the industry by promoting interoperability of object-oriented visual modeling tools. Achieving this goal requires consensus on the semantics and notation used in UML.

UML meets several requirements, including defining a common meta-model based on the Meta-Object Facility (MOF). This meta-model specifies UML's abstract syntax, defining the set of UML modeling concepts, their attributes, relationships, and the rules for combining these concepts to construct partial or complete UML models.

Furthermore, UML provides a detailed explanation of the semantics of each UML modeling concept, specifying how UML tools can conform to this specification. It also specifies human-readable notation elements to represent individual UML modeling concepts.

In summary, UML is a powerful tool offering flexibility and richness in modeling and visually representing various aspects of a software system. It is essential for our project and will be extensively used in the analysis and design phases of our mobile e-commerce application, as illustrated in the following sections of this chapter.

II. System Actors

In the context of the Tiflet Store application, system actors represent entities that directly interact with our system. They can be categorized as primary actors (those performing essential tasks) or secondary actors (those assisting primary actors or interacting with the system indirectly).

In the case of our Stock Management application, the primary actors are Sellers and Buyers.

Here is a detailed table outlining their roles and interactions with the system:

Actor	Role	Interactions with the System
Seller	Manages products, including adding, modifying, and deleting listings.	Sellers have full access to their store and product management. They can add, update, or remove product listings, manage orders, and respond to customer inquiries.
Buyer	Browses products, makes purchases, and interacts with sellers.	Buyers can explore product listings, make purchases, save favorite items, and communicate with sellers for inquiries or support.

In the Tiflet Store system, Sellers and Buyers play crucial roles in the operation of the application, enabling the buying and selling of products within the local community. Sellers are responsible for listing and managing products, while Buyers interact with the platform to discover, purchase, and engage with sellers as needed.

III. Use Case Diagram

A use case diagram is a type of behavioral diagram used in UML (Unified Modeling Language). It graphically represents the functionalities of a system from the user's perspective. In other words, it illustrates how users (or actors) interact with a system and the primary actions or processes they perform.

Here is a description of the elements typically found in a use case diagram:

- **Use Case**

It represents a sequence of actions performed by the system to generate a result of interest for an actor. Each use case is depicted as an ellipse in the diagram.

- **Actor**

This is an entity that interacts with the system. It can be a person, another system, or an organization. Each actor is represented as a human silhouette.

- **Associations**

These are the relationships between actors and use cases. They are represented by straight lines connecting actors to their corresponding use cases.

- **System**

It represents the entity you are working on, i.e., the system being modeled. In the diagram, it is depicted as a large box containing the use cases.

The main objective of a use case diagram is to clarify what the system should do and for whom, without worrying about how these functions will be achieved. It provides a high-level view of the system's functionalities and helps in understanding the system's requirements.

In summary, a use case diagram is a graphical tool used to represent interactions between actors and the system, focusing on actions and outcomes of interest to users. It facilitates communication and understanding of the system's needs and functionalities to be developed.

- **Use Cases:**

The image below represents the Use Cases Diagram of The Application TIflet Store

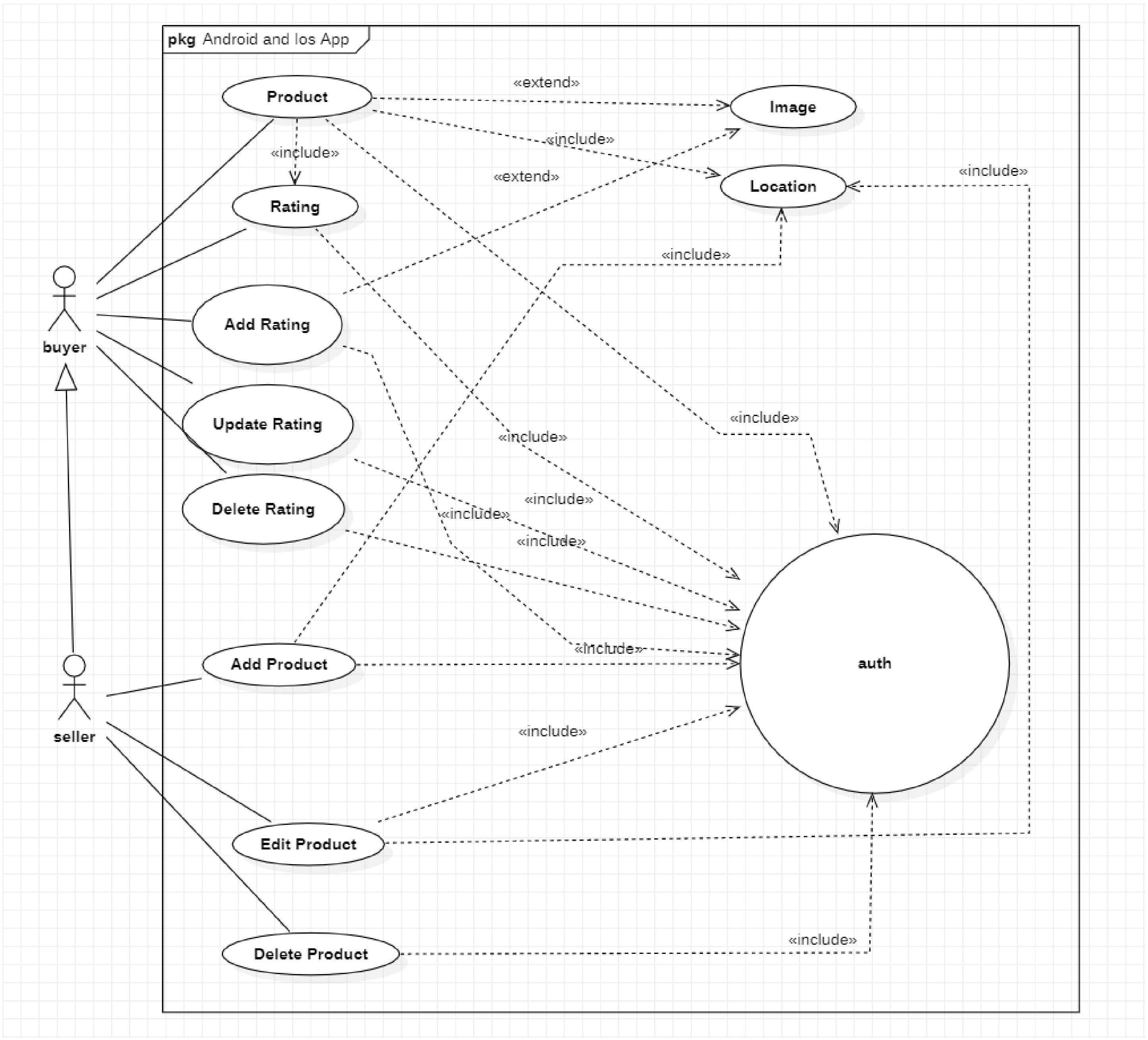


Figure 4 Use Cases Diagram

- **Sequence Diagram**

Sequence diagrams are essential tools in the analysis and design of computer systems. They allow modeling how objects interact in a system, highlighting the chronological sequence of messages exchanged between these objects. In this presentation, we will explore the key concepts of sequence diagrams, their components, and their use in software development.

IV. Entity Relationship Diagram (ERD)

Entity Relationship Diagram (ERD) is a visual representation of the entities (objects or concepts), attributes (properties or characteristics), and relationships between entities within a database or information system. ERDs are commonly used in database design and software engineering to model the structure and organization of data.

Key components of an ERD include:

- **Entities:** These are the main objects or concepts in the system, such as "Customer," "Product," "Employee," etc. Entities are typically represented as rectangles in the diagram.

- **Attributes:** Attributes are properties or characteristics of entities. For example, a "Customer" entity might have attributes like "CustomerID," "FirstName," "LastName," and "Email." Attributes are usually depicted as ovals or ellipses connected to their respective entities.

- **Relationships:** Relationships illustrate how entities are related or connected to each other. Relationships are represented as lines connecting entities, and they describe how data is associated between entities. Common types of relationships include "one-to-one," "one-to-many," and "many-to-many."
    - One-to-One (1:1): Each entity in the relationship is associated with exactly one entity in the other.
    - One-to-Many (1:N): Each entity in one side of the relationship can be associated with multiple entities on the other side.
    - Many-to-Many (N:M): Entities on both sides of the relationship can be associated with multiple entities on the other side.

- **Primary Keys:** Primary keys are attributes that uniquely identify each instance (record) of an entity. In ERDs, they are often underlined or marked to indicate their uniqueness.

- Cardinality: Cardinality notations in an ERD describe the minimum and maximum number of instances of one entity that can be associated with another entity in a relationship. Common cardinality notations include "0..1" (zero or one), "0..N" (zero or many), "1..1" (exactly one), "1..N" (one or many), etc.

- **Use ERD:**

The image below represents the Entity Relationship Diagram of The Application TIflet Store

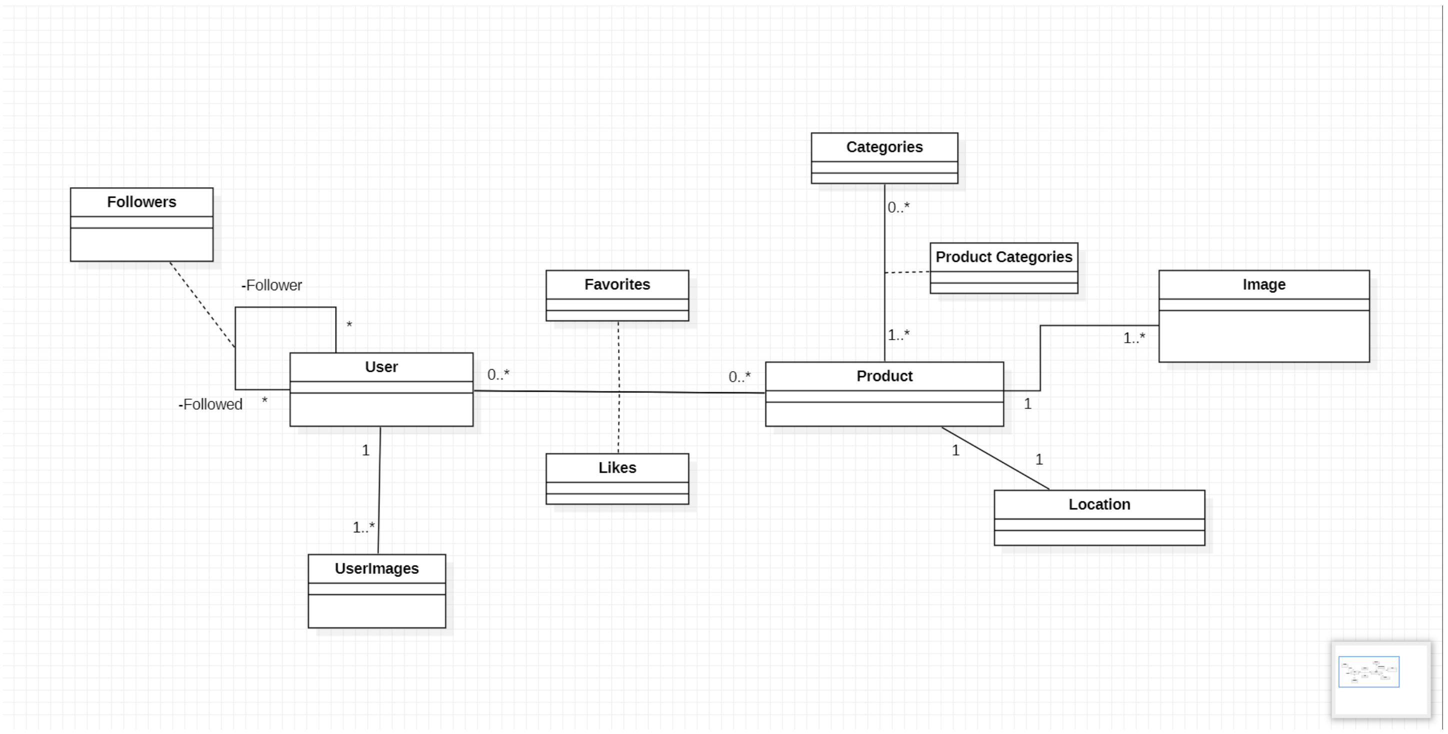


Figure 5 Entity Relationship Diagram

V. Class Diagram

Class Diagram is a type of UML (Unified Modeling Language) diagram used in software engineering and system design to visually represent the classes, attributes, operations (methods), and their relationships within a system or software application. Class diagrams are a fundamental part of object-oriented modeling and serve as a blueprint for the structure of the software.

Key components and concepts of a class diagram include:

- **Class:** A class is a blueprint or template for creating objects. It represents a category of objects with shared properties and behaviors. In a class diagram, classes are typically depicted as rectangles with three compartments: the top compartment contains the class name, the middle compartment lists the attributes (data members or properties), and the bottom compartment lists the operations (methods or functions) associated with the class.

- Attributes: Attributes are the data members or properties of a class, representing the characteristics or state of objects created from that class. Attributes are typically shown in the middle compartment of the class rectangle and include their names and data types.

- Operations: Operations are the methods or functions associated with a class, representing the behaviors or **actions** that objects of that class can perform. Operations are usually listed in the bottom compartment of the class rectangle and include their names, parameters, and return types.

- Relationships: Class diagrams show relationships between classes, which describe how classes are connected or interact with each other. Common types of relationships include:

  - Association: Represents a simple connection between classes. It shows that objects of one class are related to objects of another class.
  - Aggregation: Indicates a "whole-part" relationship between classes, where one class (the whole) contains or is composed of other classes (the parts).
  - Composition: Similar to aggregation but with a stronger association, indicating that the parts are tightly coupled to the whole and have a lifecycle dependent on the whole.

- Inheritance (Generalization): Represents an "is-a" relationship, where one class (subclass or derived class) inherits attributes and operations from another class (superclass or base class).
- Realization: Denotes that a class implements an interface, meaning it provides concrete implementations for all the methods defined by the interface.

- **Use Class Diagram:**

The image below represents the Class Diagram of The Application TIflet Store

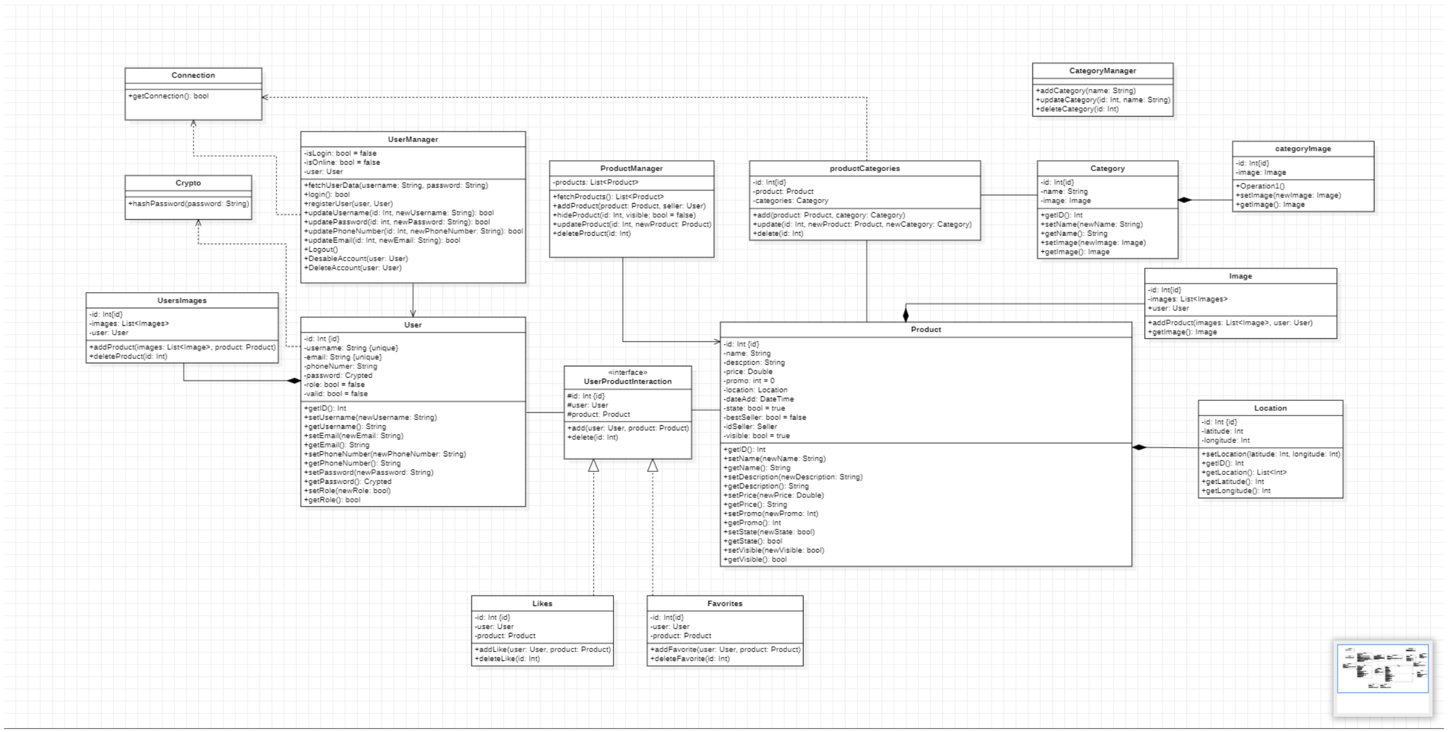


Figure 6 Class Diagram

2. REALIZATION

This project represents a significant milestone in my professional journey as an application developer during my six-month internship. In this chapter, I will describe the various technologies I have used to design and develop the application. Additionally, I will provide screenshots illustrating the application in action to provide a better visualization of the outcomes of my work.

Before delving into the core of the matter, it is important to underscore the critical aspect of my training during the internship and how my school and studies contributed to acquiring my skills.

Throughout this extensive six-month internship period, I devoted a substantial amount of time and effort to my training. This prolonged duration was made possible thanks to the support and resources provided by my school and the internship program itself, along with my supervisor's guidance.

The training I underwent was highly comprehensive, covering a wide range of technologies and essential concepts for any application developer today. I learned to utilize C++, Qt 6, QML, CMake, JavaScript (JS), JSON, Felgo, PHP, MySQL, Apache, DigitalOcean, and Python in the development of my application.

I. Technology Used In The Project Development

- **StarUML software modeling and diagramming tool**

StarUML
The Open Source UML/MDA Platform

StarUML is a popular software modeling and diagramming tool used in software engineering and design. It is designed to help software developers, architects, and designers create UML (Unified Modeling Language) diagrams and models for software systems. UML diagrams are graphical representations used to visualize and document various aspects of software systems, such as their structure, behavior, and interactions.

Some of the key features of StarUML include:

- UML Diagram Support: StarUML supports various types of UML diagrams, including class diagrams, use case diagrams, sequence diagrams, activity diagrams, and more. These diagrams help in visualizing different aspects of a software system's design.

- Code Generation: It can generate code from UML diagrams, making it easier to transition from design to actual implementation.

- Reverse Engineering: StarUML can also perform reverse engineering, allowing developers to create UML diagrams from existing source code, which is useful for understanding and documenting legacy systems.

- Extensibility: It supports extensions and plugins that can enhance its functionality and support additional modeling languages or features.

- User-Friendly Interface: It provides a user-friendly interface with drag-and-drop functionality, making it easier to create and edit UML diagrams.

- Collaboration: It supports team collaboration features, allowing multiple users to work on the same project simultaneously.

- Cross-Platform: StarUML is available for multiple platforms, including Windows, macOS, and Linux.

- **Microsoft Visual Studio Code (IDE)**

Microsoft Visual Studio Code, commonly known as VS Code, is a free and open-source integrated development environment (IDE) developed by Microsoft. It is widely used by developers for writing, editing, and debugging code across various programming languages. VS Code is known for its flexibility, extensibility, and a wide range of features that make it a popular choice among developers for various programming tasks.

Key features of Microsoft Visual Studio Code include:

- Multi-Language Support: VS Code supports a wide range of programming languages out of the box, and developers can easily add extensions to support additional languages.

- IntelliSense: It provides intelligent code completion and suggestions, helping developers write code more efficiently.

- Debugging: VS Code includes a powerful debugging feature with support for various languages and platforms.
- Version Control: It integrates seamlessly with version control systems like Git, making it easier to manage and track code changes.
- Extensions: VS Code has a rich ecosystem of extensions created by the community, which can be used to add functionality and support for specific programming languages, frameworks, and tools.
- Customization: Developers can customize the appearance and behavior of VS Code through themes and settings.
- Integrated Terminal: It includes a built-in terminal, allowing developers to run commands and scripts without leaving the IDE.
- Task Automation: VS Code supports the creation and execution of tasks and build processes, streamlining development workflows.
- Live Share: This feature enables collaborative coding sessions, allowing multiple developers to work on the same codebase in real-time, even if they are in different locations.
- Extensions Marketplace: Developers can find and install extensions from the Visual Studio Code Marketplace to tailor the IDE to their specific needs.

- **Qt Framework**

Qt is a popular cross-platform framework used for developing graphical user interface (GUI) applications, as well as non-GUI applications, across various platforms. It provides a comprehensive set of libraries and tools for creating software applications that are compatible with different operating systems, such as Windows, macOS, Linux, and even embedded systems.

Key features of the Qt framework include:

- Cross-Platform Development: Qt allows developers to write code once and deploy it on multiple platforms without significant modifications, saving time and effort in application development.

- Rich Set of Libraries: It offers a wide range of libraries for tasks like GUI creation, networking, file handling, and more. These libraries are well-documented and help streamline development.

- Qt Widgets: Qt provides a set of GUI widgets that can be used to create desktop applications with native-looking user interfaces. It follows the traditional widget-based approach for creating UIs.

- Qt Quick: Qt Quick is a part of Qt that introduces a more modern approach to GUI development called QML (Qt Meta-Object Language). It's designed for creating fluid, dynamic, and touch-friendly user interfaces.

- Open Source: Qt is available under both open-source and commercial licenses, making it accessible to a wide range of developers.

- Community and Ecosystem: Qt has a thriving community of developers and a rich ecosystem of third-party libraries, tools, and extensions.

- Integration: Qt can be integrated with various IDEs and development tools, including Qt Creator, Visual Studio, and others.

- **Qt Meta-Object Language (QML)**

QML (Qt Meta-Object Language) is a user interface markup language within the Qt framework. It is designed to create rich and interactive user interfaces, particularly for applications that require dynamic and visually appealing displays. QML allows developers to define user interfaces using a declarative syntax, which is different from the traditional imperative code used in many programming languages.

Key features of QML include:

  - Declarative Syntax: QML allows developers to describe the UI and its behavior in a declarative manner. This means specifying what the UI should look like and how it should respond to user interactions without specifying step-by-step instructions.

- JavaScript Integration: QML integrates seamlessly with JavaScript, enabling developers to add interactivity to their UIs. JavaScript code can be embedded directly within QML files.

- Animations and Transitions: QML supports animations and transitions, making it easy to create fluid and visually appealing user interfaces.

- Component-Based Development: QML encourages a modular and component-based approach to UI development, allowing developers to reuse UI elements across different parts of their application.

- Cross-Platform: QML is designed for cross-platform development and is suitable for applications targeting various platforms, including desktop, mobile, and embedded systems.

- **C++**

C++ is a high-level programming language known for its performance, flexibility, and versatility. It was developed as an extension of the C programming language and includes features like object-oriented programming (OOP), which allows developers to structure code in a more organized and modular way. C++ is widely used in various domains, including system programming, game development, embedded systems, and more.

Here are some key features and concepts of C++:

- Object-Oriented Programming (OOP): C++ supports OOP principles, such as encapsulation, inheritance, and polymorphism. This allows developers to create reusable and maintainable code by organizing data and functions into classes and objects.

- Performance: C++ is known for its performance, as it allows low-level memory manipulation and fine-grained control over system resources. This makes it suitable for applications where speed is crucial.

- Standard Template Library (STL): C++ includes the STL, which provides a collection of template classes and functions for common data structures (e.g., vectors, lists, maps) and algorithms (e.g., sorting, searching). The STL promotes code reusability.

- Compatibility: C++ is backward-compatible with C, meaning C code can be integrated into C++ programs. This makes it easy to leverage existing C libraries and code.

- Rich Ecosystem: C++ has a vast ecosystem of libraries and frameworks, making it suitable for various application domains, including game development (e.g., Unreal Engine), scientific computing (e.g., ROOT), and more.

- **CMake**

CMake is an open-source build system and project management tool designed to simplify the build process for software projects. It helps developers generate build files, such as Makefiles or project files for IDEs (Integrated Development Environments), in a platform-independent and efficient manner. CMake is particularly popular in the C++ development community but can be used for building projects in various programming languages.

Key features and concepts of CMake include:

- Platform Independence: CMake is designed to be platform-independent. Developers can write CMakeLists.txt files that specify how the project should be built, and CMake generates platform-specific build files for various operating systems, such as Makefiles for Unix-like systems or project files for Visual Studio on Windows.

- CMakeLists.txt: CMake projects are configured using CMakeLists.txt files, which are written in a simple scripting language. These files define the project's source files, dependencies, build options, and other project-specific settings.

- Generators: CMake supports various "generators" that produce build files for different build systems and IDEs. Common generators include Makefile generators, Visual Studio generators, Xcode generators, and more. Developers can choose the generator that suits their development environment.

- Configurable Build Options: CMake allows developers to define build options and variables in CMakeLists.txt files. These options can be configured by developers or users to customize the build process, enable or disable features, and set compiler flags.

- Dependency Management: CMake simplifies the management of project dependencies. It can locate libraries and packages installed on the system or provide options to specify custom paths for required dependencies.

- Out-of-Source Builds: CMake promotes the practice of out-of-source builds, where build files and build artifacts are kept separate from the source code. This ensures that the source directory remains clean and allows for easy clean-up and reproducibility of builds.

- Integration with IDEs: CMake integrates with popular integrated development environments (IDEs) like Microsoft Visual Studio, CLion, Xcode, and others. This means developers can use their preferred IDE while CMake manages the underlying build system.

- Extensibility: CMake is highly extensible, and developers can write custom CMake modules or scripts to support specific project requirements or to integrate with third-party tools.

- **Java Script (JS)**

JS stands for JavaScript, which is a widely used programming language primarily used for front-end web development. JavaScript is a versatile and dynamic language that allows developers to add interactive and dynamic features to websites and web applications.

Here are some key characteristics and uses of JavaScript:

- Client-Side Scripting: JavaScript is primarily used as a client-side scripting language, meaning it runs in the user's web browser, not on the server. It enables web developers to create interactive and responsive user interfaces.

- Web Development: JavaScript is a core technology for web development and is used to enhance the functionality and behavior of web pages. It can manipulate HTML and CSS, handle user input, and make asynchronous requests to servers (AJAX).

- Cross-Platform Compatibility: JavaScript is supported by all major web browsers, including Chrome, Firefox, Safari, and Edge. This makes it a cross-platform language for web development.

- Libraries and Frameworks: JavaScript has a rich ecosystem of libraries and frameworks, such as jQuery, React, Angular, and Vue.js, that simplify common web development tasks and promote best practices.

- Server-Side Development: While JavaScript is primarily known for front-end development, it can also be used for server-side development. Node.js, a runtime environment for JavaScript, allows developers to build server-side applications using JavaScript.

- Event-Driven: JavaScript is event-driven, meaning it responds to events like user interactions (clicks, mouse movements, etc.) and executes code accordingly. This makes it suitable for creating interactive and real-time applications.

- Dynamic Typing: JavaScript is dynamically typed, which means you don't need to specify data types explicitly. Variables can change their data type during runtime.

- Object-Oriented: JavaScript supports object-oriented programming (OOP) concepts, including objects, classes (with the introduction of ES6), inheritance, and encapsulation.

- Community and Resources: JavaScript has a large and active developer community, which means there are plenty of online resources, tutorials, and forums available for learning and getting help with JavaScript development.

- **Java Script Object Notation (JSON)**

JSON (JavaScript Object Notation) is a lightweight data interchange format that is easy for humans to read and write and easy for machines to parse and generate. JSON is often used to transmit data between a server and a web application or between different parts of an application.

Key characteristics of JSON include:

- Text-Based: JSON data is represented as plain text, making it easy for both humans and machines to understand. It uses a simple syntax of key-value pairs.

- Data Structure: JSON supports various data types, including objects, arrays, strings, numbers, booleans, and null values. It allows you to represent complex data structures and hierarchies.

- Readability: JSON is designed to be human-readable and easy to write. It uses a syntax that resembles JavaScript object literal notation, making it familiar to programmers.
- Language Agnostic: JSON is not tied to any particular programming language and can be used with many programming languages, including JavaScript, Python, Java, C#, and more.
- Interchangeable: JSON is commonly used for data interchange between a client and a server in web applications. It is often used in APIs (Application Programming Interfaces) to send and receive data.

- **Php**

PHP (Hypertext Preprocessor) is a widely-used, server-side scripting language designed for web development but also used as a general-purpose programming language. It is known for its ability to embed directly into HTML code, making it a popular choice for creating dynamic web pages and web applications.

Key characteristics and features of PHP include:

- Server-Side Scripting: PHP is primarily executed on the web server, which means it can generate dynamic web content and interact with databases and other server resources before sending the results to the client's web browser.

- Open Source: PHP is an open-source language, and its interpreter is freely available. This has led to a large and active community of developers who contribute to its development and create numerous libraries and frameworks.

- Cross-Platform: PHP is available on various operating systems, including Windows, macOS, Linux, and others. This cross-platform compatibility makes it suitable for a wide range of web hosting environments.

- Database Connectivity: PHP has built-in support for connecting to various database systems, including MySQL, PostgreSQL, SQLite, and more. It can be used to create, read, update, and delete data in databases.

- Embedding in HTML: PHP code can be embedded directly within HTML, allowing developers to mix dynamic server-side logic with static web content. PHP code is typically enclosed within <?php and ?> tags.

- **MySql**

MySQL is an open-source relational database management system (RDBMS) that is widely used for managing and storing structured data. It is known for its reliability, scalability, and ease of use, making it one of the most popular database systems in the world. Here are some key characteristics and features of MySQL:

- Relational Database: MySQL follows the principles of a relational database, which means it stores data in tables with rows and columns. It supports complex queries, transactions, and relationships between tables.

- Open Source: MySQL is open-source software, which means it is freely available for use, modification, and distribution. This open nature has contributed to its widespread adoption and development by a large community of users and contributors.

- Cross-Platform: MySQL is available for various operating systems, including Windows, macOS, Linux, and more. This cross-platform compatibility makes it suitable for a wide range of applications and environments.

- High Performance: MySQL is designed for high performance and can handle large volumes of data and concurrent users efficiently. It includes features like indexing, caching, and query optimization to improve performance.

- Scalability: MySQL supports both vertical and horizontal scalability. Vertical scalability involves increasing the capacity of a single server, while horizontal scalability involves distributing data across multiple servers (sharding).

- Security: MySQL offers robust security features, including user authentication, access control, encryption, and auditing capabilities. It allows administrators to define fine-grained access permissions for users and applications.

- ACID Compliance: MySQL follows the ACID (Atomicity, Consistency, Isolation, Durability) properties, ensuring that database transactions are processed reliably and consistently.

- Data Types: MySQL supports a wide range of data types, including numeric, string, date and time, spatial, and more. This versatility allows it to handle diverse data requirements.

- **Apache**

Apache is a widely used open-source web server software that plays a central role in serving web content and facilitating the delivery of web pages to users' browsers. It is known for its reliability, performance, and extensibility. Here are some key aspects of Apache:

- Web Server: At its core, Apache is a web server software designed to handle HTTP requests from clients (typically web browsers) and respond with the requested web content, such as HTML files, images, and other resources.

- Open Source: Apache is open-source software, which means it is freely available for anyone to download, use, and modify. The open-source nature has contributed to its widespread adoption and continuous development by a large community of contributors.

- Cross-Platform: Apache is cross-platform, meaning it can run on various operating systems, including Linux, Windows, macOS, and more. This versatility makes it a popular choice for hosting websites on diverse server environments.

- Modules and Extensions: One of Apache's strengths is its extensibility through modules. Users can add functionality to Apache by enabling or creating modules.

There are modules for handling different types of content, security features, authentication methods, and more.

- Performance: Apache is designed to handle high volumes of web traffic efficiently. It includes features like multi-processing and multi-threading to serve multiple requests simultaneously, which helps maintain good performance even under heavy loads.

- Security: Apache provides various security features to protect web servers and websites. It supports SSL/TLS encryption for secure data transmission, access control based on IP addresses or usernames, and logging to monitor server activity.

- Virtual Hosting: Apache allows for virtual hosting, meaning it can serve multiple websites on a single physical server. This is useful for hosting providers and organizations that manage multiple domains.

- **DigitalOcean**

DigitalOcean is a cloud infrastructure provider that offers cloud computing services to developers, businesses, and organizations. It enables users to deploy and manage cloud-based virtual servers, commonly referred to as "Droplets," along with various other cloud resources like databases, object storage, and networking solutions. Here are some key aspects of DigitalOcean:

- Droplets: Droplets are virtual private servers (VPS) that users can quickly deploy and manage in the cloud. Users can choose from various pre-configured images, including different Linux distributions and application stacks, to create Droplets tailored to their needs.

- Kubernetes: DigitalOcean offers a managed Kubernetes service that simplifies the deployment, scaling, and management of containerized applications using Kubernetes, an open-source container orchestration platform.

- Managed Databases: DigitalOcean provides managed database services for popular databases like PostgreSQL, MySQL, and Redis. These services offer automated backups, scaling, and high availability features.

- Object Storage: Spaces is DigitalOcean's object storage service, similar to Amazon S3. It allows users to store and serve large amounts of unstructured data, such as images, videos, and backups.

- Block Storage: DigitalOcean offers scalable block storage that can be attached to Droplets, providing additional storage capacity that can be resized as needed.

- Load Balancers: Users can set up load balancers to distribute incoming traffic across multiple Droplets, ensuring high availability and improved performance for their applications.

- Developer-Friendly: DigitalOcean is known for its developer-friendly approach, with a straightforward web-based control panel and a user-friendly API. It also provides an active community, extensive documentation, and a marketplace of pre-configured one-click applications.

- Pricing: DigitalOcean is often appreciated for its transparent and competitive pricing, which includes a straightforward billing model with no hidden fees.

- **Github**

GitHub is a web-based platform and service that is primarily used for version control and collaborative software development. It offers a wide range of features for developers and teams, making it one of the most popular and widely used platforms for hosting and sharing code. Here are some key aspects of GitHub:

- Version Control: GitHub is built on top of Git, which is a distributed version control system. It allows developers to track changes in their codebase, collaborate with others, and manage different versions of their software projects. Git is particularly powerful for branching, merging, and resolving conflicts in code.

- Code Hosting: Developers can host their Git repositories on GitHub. These repositories can be public, allowing anyone to access and view the code, or private, limiting access to specific collaborators.

- Collaboration: GitHub provides tools for collaboration among developers and teams. Multiple developers can work on the same project simultaneously, and GitHub offers features like pull requests, code reviews, and issue tracking to facilitate collaboration and communication.

- Workflow Automation: GitHub Actions is a feature that allows users to automate various aspects of the software development workflow, such as building, testing, and deploying code. It enables the creation of custom workflows using code or predefined templates.

- Community and Social Features: GitHub is not just a code repository; it's also a social platform for developers. Users can follow other developers, star repositories, and contribute to open-source projects. It has features like discussions, wikis, and project boards to facilitate community engagement.

- Security: GitHub provides tools and features to enhance the security of code repositories. It includes vulnerability scanning, dependency analysis, and access control to help developers and organizations manage the security of their projects.

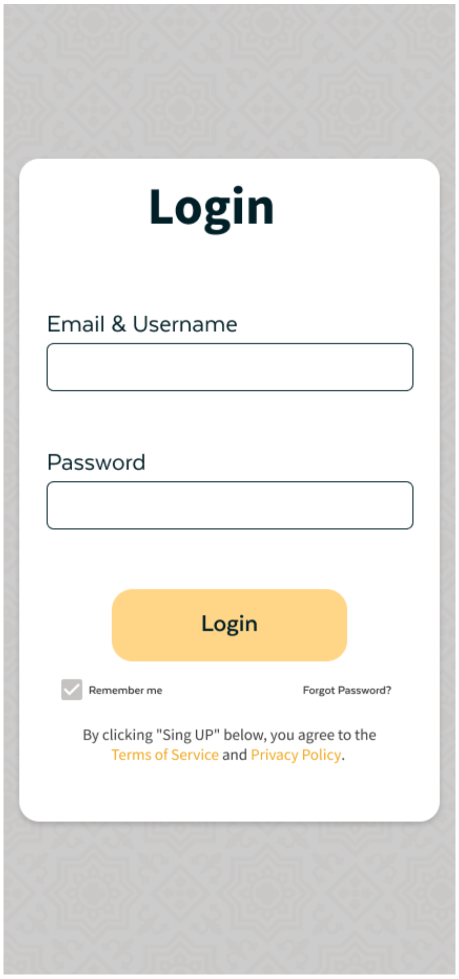
- Integration: GitHub integrates with a wide range of development tools, such as continuous integration/continuous deployment (CI/CD) systems, project management tools, and code editors. This integration ecosystem makes it easier to incorporate GitHub into various parts of the development process.

- Education: GitHub offers education-focused features and programs for students, teachers, and educational institutions. It provides free access to GitHub features for classrooms, helping educators teach coding and collaboration.

II. Application Architecture

- **Login Page**

When you launch the application, your first encounter will be with the Login Page.



By clicking "Sing UP" below, you agree to the Terms of Service and Privacy Policy.

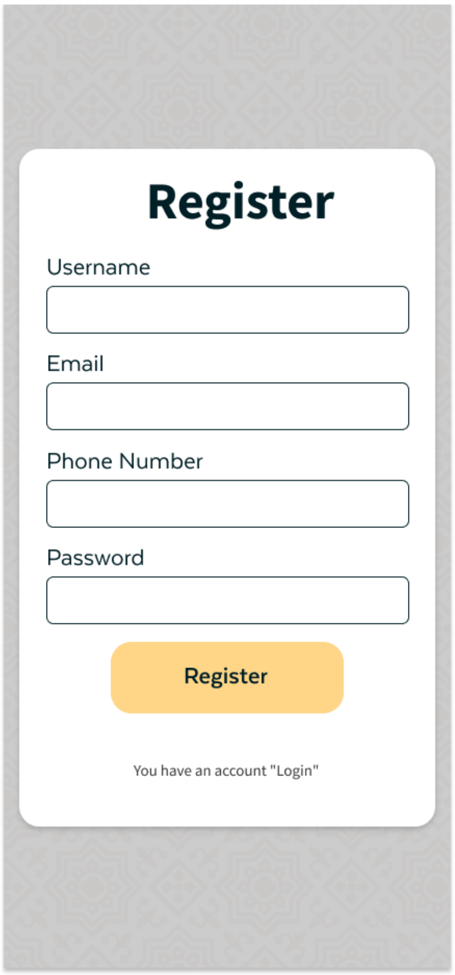
♦ **Inputs: in the Register page we have 4 inputs**
   - Email & Username: A text field for entering the user's email address or username.
   - Password: A password field for entering the user's password.

- Remember me: A checkbox that allows the user to stay logged in for the next time they visit the site.

- **The buttons are:**
  - Login: A button that submits the form data to the server for processing.
  - Forgot Password?: A link that takes the user to a page where they can reset their password.

- **Register Page**

The register page is where you can create an account if you don't already have one.



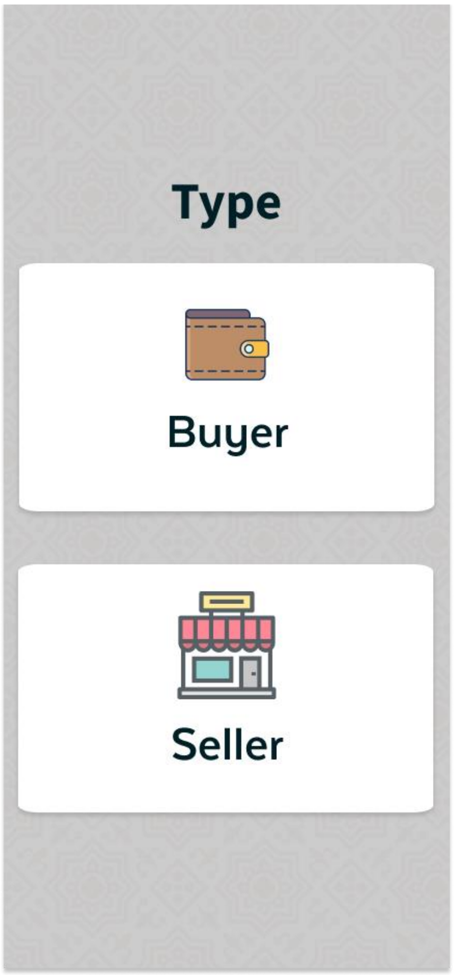
We have white register form with a yellow register button. The input fields are:

- **Inputs: in the Register page we have 4 inputs**
  - Username: A text field for entering the user's username.
  - Email: A text field for entering the user's email address.
  - Phone Number: A text field for entering the user phone number
  - Password: A password field for entering the user's password.

- **The buttons are:**
  - Register: A button that submits the form data to the server for processing.
  - You have an account? Login: A link to the login page.

- **Type Page**

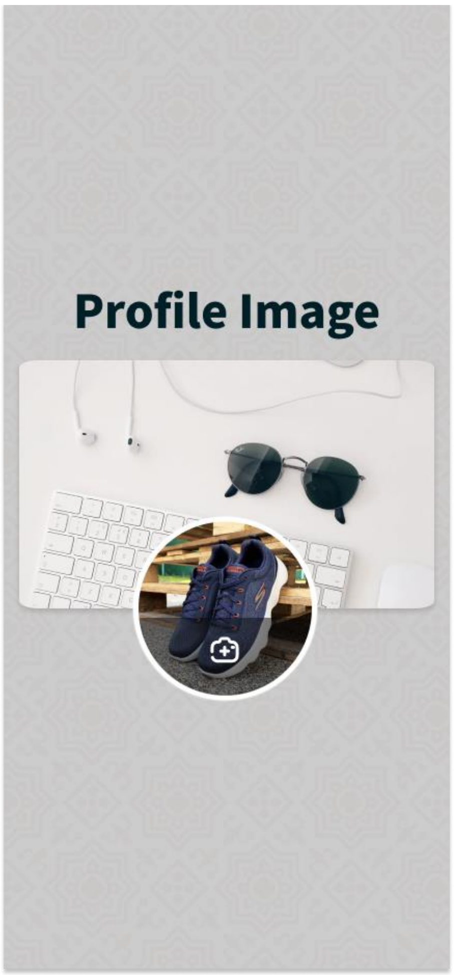
Type page is where the user can pick whether he want to be a seller on the app or only a product buyer.



- ♦ **The buttons are:**
  - Buyer: A button with a pocket icon.
  - Seller: A button with a Shop Icon make It easy for the user to understand the meaning or it.

- **Profile Image Page**

Profile image page is accessible only if you chose to be a seller, to upload you profile image wen you done the registration.

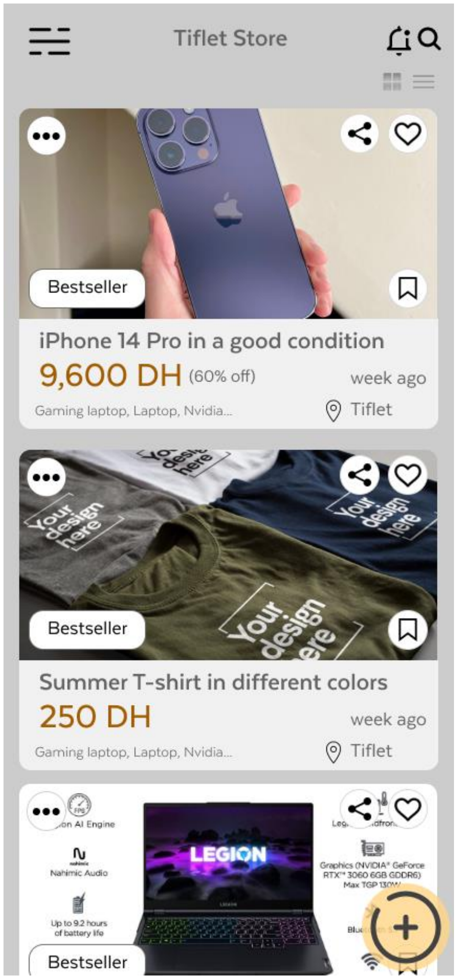


The Profile Image Page is a page related to the seller only and give you the possibility to update your profile image.

- **The buttons are:**
  - Image: A button with a camera icon, update the seller image.

- **Home Page**

Home page is one of the most important pages on the application because the user will spend the most of his time scrolling the home page


Tiflet Store
Bestseller
iPhone 14 Pro in a good condition
9,600 DH (60% off)
week ago
Gaming laptop, Laptop, Nvidia...
Tiflet
Bestseller
Summer T-shirt in different colors
250 DH
week ago
Gaming laptop, Laptop, Nvidia...
Tiflet
Bestseller

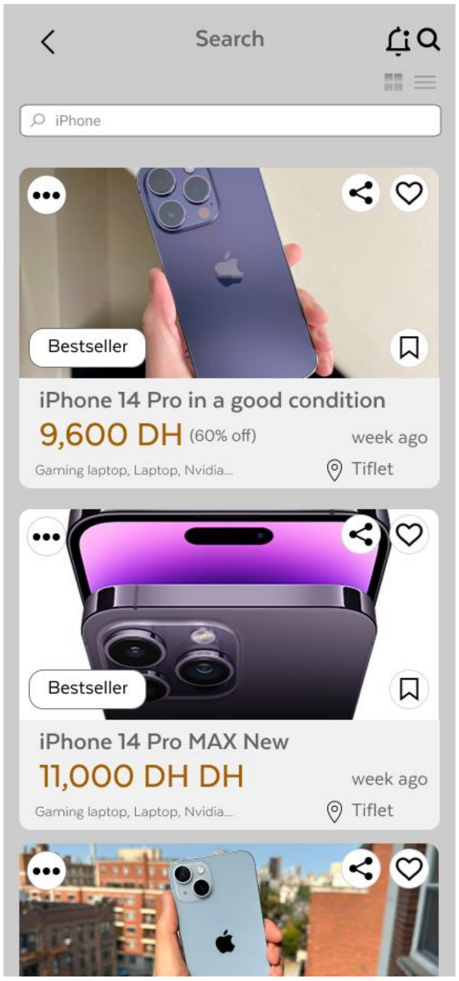
The home page is 2 parts 1th we have the header and the header in a component that will be accessible in the most pages and the 2th part is the main Home Page part because it has a list of card

➢ Header:

- **The buttons are:**
  - Hemberger Menu: A button with a 3 horizontal lines icon, That show the Side Menu.
  - Notification Icon: A button with a notification Icon, take you to the notification page
  - Search Icon: A button with a search icon, Redirect you to the search page
  - Tiflet Store: The page title, and It’s changeable depending on the page you are in

- Boddy:
  - Alignment Type: a button that make you pick the alignment of the product cards
  - List: A List of Product Cards that show the latest uploaded cards

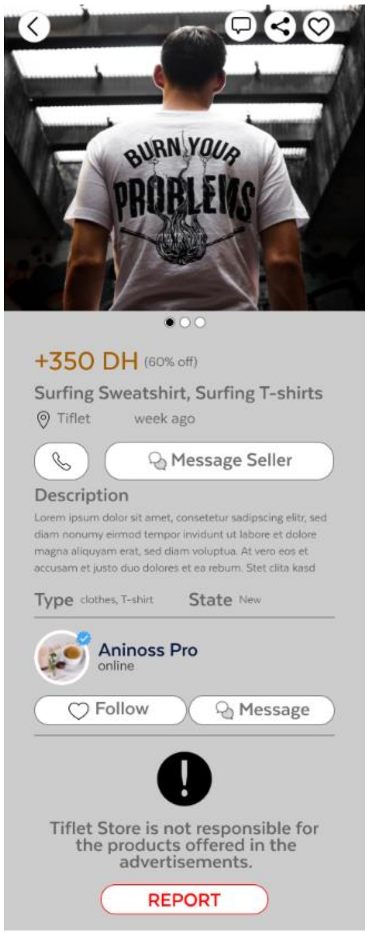
- **Search Page**



The search Page similar to the homepage because they both have the same alignment a Header with a deference Title “Search”, and a small search bar and a list of product cards with the keyword you are searching for

- **Product Page**

Page to display more info about the product and the product seller



+350 DH (60% off)

Surfing Sweatshirt, Surfing T-shirts

Tiflet week ago

Description

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd

Type clothes, T-shirt State New

Aninoss Pro
online

Tiflet Store is not responsible for the products offered in the advertisements.

➢ Header

- **The Image:** The 1th thig the user will see after clicking on the product card Is the card image in full size with a slider if the product has more than one image
- **The buttons are:**
  - Back: A button with a Arow icon icon, That will redirect you to the main image.

- Share: A button with a share icon, display a list of application that you could use to share our product page
- Like: A button with a like icon, Will add the product to your like list

➢ Boddy

- Price: Product Price
- Promo: Will Display if there is a promo on the product
- Title: The title of the product
- Location: The Location the product listed in
- Time: The time the product was listed on
- Description: The Product description
- State: the state of product New, Old or Vantage
- Type: The category of the product
- Aninoss Pro: The Seller Name
- Online: The Seller Status Is he Online or Offline

♦ **The buttons are:**

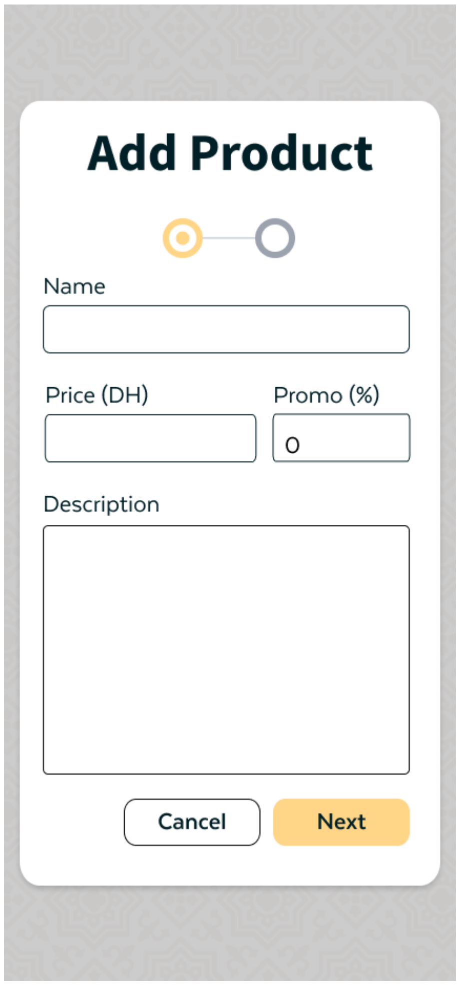
- Display Phone: A button display the phone number
- Seller Image: A Button with the seller profile image that redirect you to the seller profile
- Follow: A Button to follow the seller
- Message: A Button to send message to the seller

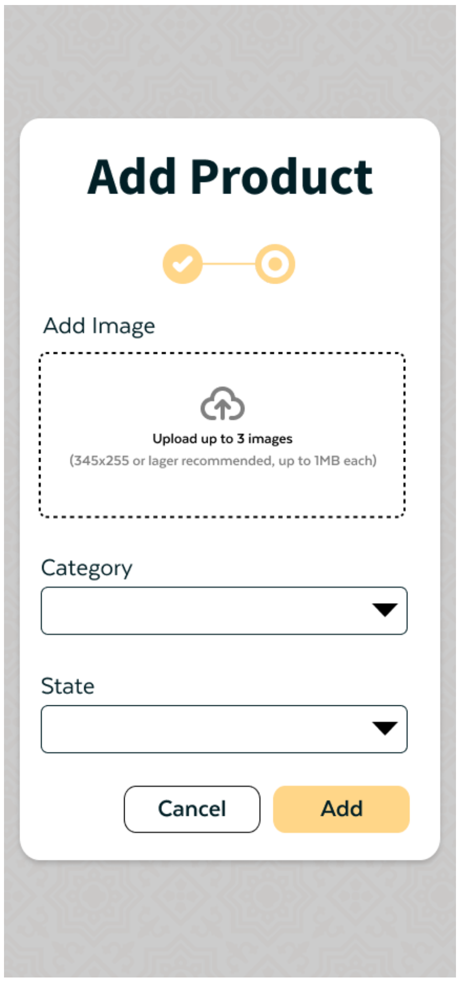
➢ Footer

- Report Icon: An Icon to show that you can report the product
- Report Button: A button with report Text, used to report the product

- **Add Product Pages**

The Add product page is actually 2 pages because there is a lot of information to include in the product and we don’t want to make the user experience (UX) horrible by stacking all the information needed in one page





Add Product page similar to Register and Login pages we have white form with a yellow button.

- **Title:** Add Product

- **Steps:** A Custom component that change the icon in every page base of if you done the step or not yet

- **Inputs:** in the Add Product page we have 4 inputs
    - Name: Product name
    - Price: Product price.
    - Promo: Product promo.
    - Description: Product Description.

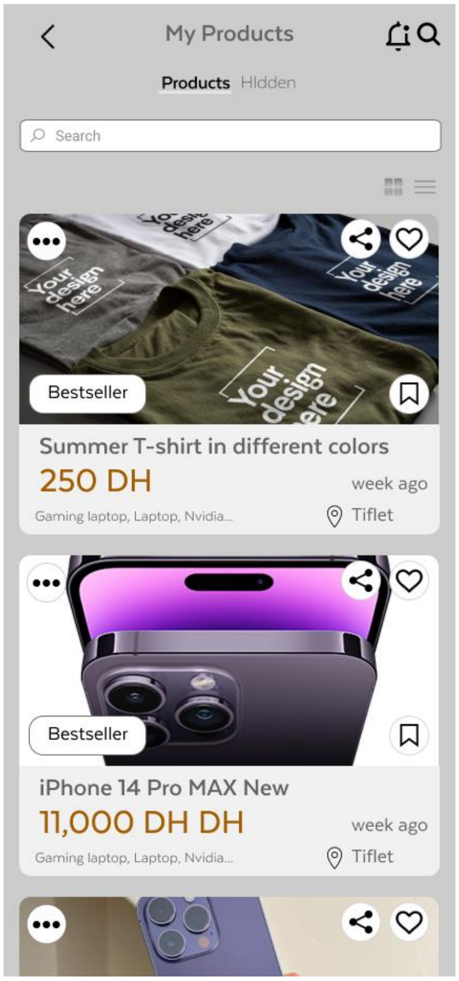
- **Drop Down List:** in the Add Product page we have 2 Drop Down List
    - Category: is a drop-down list that contain a list of Product Categories like, Sport, Car, Phone, Tv, Laptop and more.
    - State: is a drop-down list that contain the stets of the product, New, Old or Vintage.

- **The buttons: :** in the Add Product page we have 4 buttons
    - Next: A button with redirect you to the 2th Add product page.
    - Add: A button to add the product to the product list and redirect you back to the Home page.
    - Upload Images: A Custom button with a Upload icon, Will upload the selected pages from your device (Android / iOS) to the cloud.

- Cancel: A Button redirect you to the Home page

- **My Product List page**

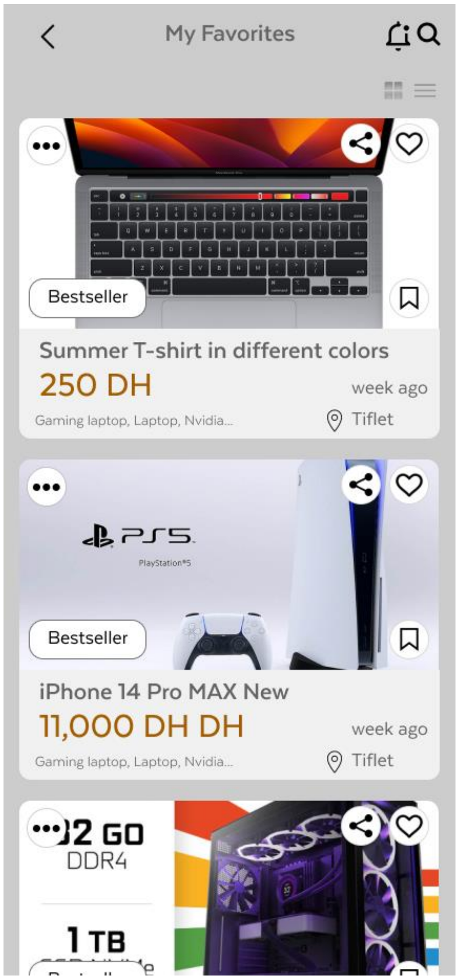
My Product List is a page where you can find your products and hidden products



My Procut List is similar to the home page but instead of a list of products from deference users it shows up only my product that I upload to the application

- **My Favorite Page**

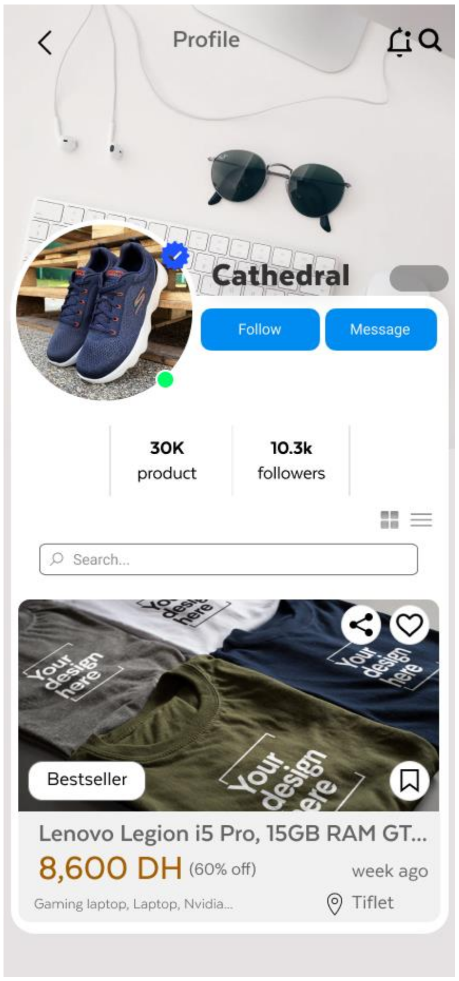
My Favorite is a page where I can store all my favorite products



My Favorite List is similar to the home page but instead of a list of products from deference users it show up only the product I tag as favorite.

- **Profile Page**

My Procut List is similar to the home page but instead of a list of product from deference users it show up only my product that I upload to the application



➢ Header

- **The Background Image:** An Updatable Image that display as a background
- **The Profile Image Container:**
  - ➢ **Profile Image:** An updatable image that display ad the seller image
  - ➢ **Greed Circle:** An icon that display the user state is Online or Offline

- ➢ **Blue Validate Bag:** An Icon on the top left of the profile image that show if the user is a valid user or not

- ♦ **The buttons are:** Profile Page has 2 buttons
  - Follow: A button to follow the seller.
  - Message: A button to send images to the seller.

➢ Boddy

- Products: Display the seller products number
- Followers: Display the seller followers number

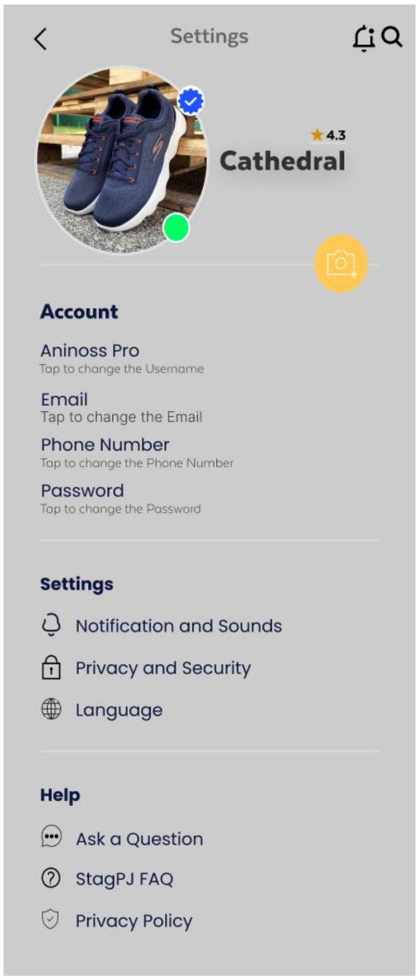
- ♦ **Inputs: in the Profile Page we have only 1 input**
  - Search: an input to search on the seller products

- ♦ **The buttons are:**
  - Alignment Type: a button that make you pick the alignment of the product cards

➢ Footer

- Product List: A list of seller Product cards

- **Settings Page**



➢ Header

♦ **The Profile Image Container:**
  - ➢ **Profile Image:** An updatable image that display ad the seller image
  - ➢ **Greed Circle:** An icon that display the user state is Online or Offline
  - ➢ **Blue Validate Bag:** An Icon on the top left of the profile image that show if the user is a valid user or not

♦ **The buttons are:** Profile Page has 2 buttons

  - Update Profile Image: A button with Camera Icon, that update the profile image.

- Boddy
    - **Account:** Account area to update all the user settings
        - **The buttons are:**
            - Username: A button to update the username
            - Email: A button to update the email
            - Phone Number: A button to update the Phone number
            - Password: A button to update the Password
    - Settings: Settings area to update all the application settings
        - Notification and Sounds: A button to update the Notification an sound settings
        - Privacy and Security: A button to display the privacy and security page
        - Language: A button to switch between language
    - Help: Help area
        - Ask a Question: A button to redirect the user to Ask Question page.
        - TIflet Press FAQ: A button to redirect the user to the FAQ page.
        - Privacy Policy: A button to redirect the user to Privacy Policy page.