<Love412Forever Group>

MAiMAi mobile application

<Final Assignment>

Yihan Xiao, Zihan Gao, Rongchuan Hai

2022-12-11

Software Requirements Specification and Use Case

Table of Contents

Software Requirements Specification

1. Introduction

This document can be read by anyone, and it will help both developers and instructors better comprehend the project's requirements and identify where to make improvements. This means that when students and staffs leave school, there is a propensity for them to leave many items that are inconvenient to take away, including study books that may have value to lower-grade students.

The mobile application, "MAiMAi," is created to allow students and staff to exchange, sell or buy items they no longer have use for through a standardized and secure platform. There is a limit placed on target users – only students and staff of the schools – which ensures that the items are not only good and cheap but also reduce the chances of students being cheated; while simultaneously increasing the rate of item usage and reducing the waste of academic resources. MAiMAi is typically a medium that facilitates the transfer of unwanted items to students from one university to another. While helping one group avoid the accumulation of waste grants another the opportunity to access resources at a reasonable price.

Other than the value mentioned above concerning business transactions, our software allows the users to create interrelations with their seniors and thus meet more friends.

1.1 Purpose

The project's purpose is to provide a reliable used item trading platform only for students, faculty, and staff at the University of Miami. Users can quickly and easily use this application to post for sale and buy other people's used items. Meanwhile, the personal information on the application will be linked to the school account, which needs to be approved by privacy legislation. This project should be able to run as a mobile application on a mobile phone, programmed primarily in Python as well as C, with subsequent development of a version that can run on Apple devices. Also, this application shall pass through the App Store’s review.

1.2 Scope

The project software primarily consists of a shopping platform entailing login, communication, and payment functions. By focusing on online sales, the sellers can edit information for the items on sale and post it within the application. At the same time, the buyers can browse and find items by viewing their information at will before buying them. The application seeks to acquire the University of Miami's approval, ensuring its support and promotion. Ideally, this application can be strategically utilized to earn revenue through user privileges and commissions.

1.3 Definitions, Acronyms, and Abbreviations

UM	University of Miami
HTML	Hyper Text Markup Language

1.4 References

1.5 Overview

The rest of the document comprises general descriptions, detailed requirements, and the use case model. There are general descriptions in section 2 of the paper, including product functions and constraints. Section 3 gives the project's specific functional requirements, non-functional requirements, and restrictions. The section also discusses the software platform interfaces. Section 4 is for Use-Case Model for the system. Sections 5 and 6 are the list of actors and Use Cases, respectively.

2. Overall Description

This project looks to help UM students, faculty, and staff sell and purchase used items in a quick and easy manner. As such, it has requirements for user identification, a degree of authentication to prevent malicious or illegal activity, and also provides a secure and reliable context for use. A user of MAiMAi must be a student or faculty member of UM, and the software must be available on both Android and Apple phones.

The project includes the ability to browse products, post products, communicate between users, and make payments. The primary function is to help users sell and buy unused items, which includes browsing products, posting requests, posting products, buying products, and user communication. Product constraints allow open-source projects, the number of developers is limited to 3-5 people, and no risky technologies are allowed. Assumptions and dependencies are that if the school does not agree to access the staff and student data system to help authenticate user information, the phone number is used for binding, and a phone number can only be bound to a single account. If the User does not authenticate, the account is publicly marked as an "unauthenticated account."

3. Specific Requirements

Since our initial plan is to design a mobile application, we may use Html, CSS3, and specific programming languages for the UI layout.

3.1 Functionality

This subsection contains our requirements for this on-campus trading platform for used items. These requirements are organized based on functional characteristics. They can then be visualized by envisioning their salient active interaction features to form use case diagrams and sequence diagrams for easy understanding and follow-up.

3.1.1 User registration system

The system shall collect and save the necessary information of users and accessing the school system for authentication requires high security.

3.1.2 User data collection system

The system shall facilitate 7X24 real-time data collection, capturing all user access and end-to-end communication interactions within the application.

3.1.3 Sale information upload system

The system shall provide for users to set prices and transaction information, upload display images of traded items, and convert them into complete visual information.

3.1.4 For-sale item browsing system

The system shall provide for users to browse and search for items for sale, the first uploaded item image is displayed by default, the transaction price and transaction information is thumbnailed, and the complete transaction information is displayed after users click in.

3.1.5 Money Transaction System

The system shall provide for buyers pay for funds via bank card when they are sure they want to buy items, transfer funds to a third-party bank security account, transfer funds to the account submitted by the seller after confirming the transaction is completed and return the original account if the transaction fails.

3.1.6 Communication System

The system shall provide for both sides of the transaction to send messages and pictures for communication, network backup of message records, and high security against external tampering.

3.2 Usability

This section includes all those requirements that affect usability. Those are non-functional requirements, that means that it is not the first goals of the project, but the customer will be very happy and satisfied if we can provide those services.

3.2.1 Fast training time

The required training time for a normal user and a power user to become productive and to know how the application works shall be about 10 minutes. The system shall match the familiarity of the real world. Visibility of the system state shall be more familiar.

3.2.2 Graphical user interface

The system interface shall be consistent with both internal and any existing external standards. The application shall be flexible and efficient. The design shall meet project goals of simplicity and consistency.

3.2.3 Principles of user control and fault tolerance

Principles of user control and fault tolerance will be observed. The system shall detect errors and handle error repair. The system shall have memory of transactions.

3.2.4 Language

The system shall support multiple mainstream languages.

3.2.5 Accessibility for people with disabilities

The system will provide user interface and functions for people with disabilities.

3.3 Reliability

3.3.1 Availability

The preliminary design software availability percentage shall be 99%.

3.3.2 MTBF

The preliminary design Mean Time Between Failures shall be 720 hours, at which point routine system maintenance by operators is required.

3.3.3 MTTR

Preliminary design Mean Time To Repair shall be 3 hours .

3.3.4 Resolution

Preliminary design shall have a minimum resolution of 960X640,326PPI.

3.3.5 Defect Rate

The maximum defect rate shall be one minor defect (a defect that does not affect the program's operation) per 1000 lines of code. No more than one moderate defect (a defect that affects the program process to some extent but does not cause the program to crash) per 3,000 lines of code.

3.3.6 Major Bugs

The system shall not have more than two major bugs (bugs that seriously affect program operation, such as causing program crashes, data loss, etc.) in the entire program.

3.3.7 Maintenance

All the bugs mentioned above shall be patched and fixed during each system maintenance (2 hours).

3.4 Performance

3.4.1 This application shall entail Internet-based interactions, and all communications must be run from a web server.

3.4.2 The initial loading time of the application depends on the strength of the User's network connection and also on the data throughput of this application, which needs to be determined from a base figure of the number of subscribers (initially specified as the number of all students in a school).

3.4.3 The memory disk footprint of the application should be no more than 10% of the client's end users to provide a good user experience.

3.5 Supportability

3.5.1 Naming shall be done using the standard naming convention; refer to the notes in the program.

3.5.2 The system shall require a team of 5 people to perform routine maintenance for 2 hours per month.

3.5.3 Additional server memory and storage space shall be required to store user data and handle program interactions.

3.6 Design Constraints

3.6.1 Standard Development Tools:

The system shall be built using a standard web page development tool that conforms to Html, CSS3, and specific programming languages.

3.6.2 Web-Based Product:

The computers must be equipped with web browsers such as Internet explorer. Response time for loading the product should take no longer than three minutes.

3.7 Online User Documentation and Help System Requirements

Since the program is generally aimed at college students with higher education, it is sufficient to provide clear and concise guidelines for use. A bug feedback mechanism is needed to resolve problems not found during system maintenance.

3.8 Purchased Components

Not Applicable

3.9 Interfaces

The MaiMai software system supports most interfaces, such as user interfaces, software interfaces, hardware interfaces, etc. The HTTP protocol will be used to facilitate communications between the client and server. The protocol used should be HTTP with port number 80.

3.9.1 User Interfaces

The software's user interface shall be initially designed to be compatible with all cell phone systems, whether Android or iOS. The user interface should be implemented using any tool or software package.

3.9.2 Hardware Interfaces

Since the application is based on Internet interaction, all the hardware needed to connect to the Internet shall be the same hardware interface as the system. Such as modems, Wi-Fi, Ethernet crossover cables, etc.

3.9.3 Software Interfaces

The program needs to establish interaction with buyers and sellers to ensure smooth transactions. Use the Readable interface that has a single read.

3.9.4 Communications Interfaces

The MaiMai system shall use the HTTP protocol for communication over the Internet, and the intranet communication will be through the TCP/IP protocol suite.

3.10 Licensing Requirements

MAiMAi grants the User the right to install and use the Software Product on a device that shall run an operating system obtained lawfully.

MAiMAi will grant the User a non-exclusive, non-sublicensable, limited license to install and use one copy of the Software Product on a single Device, with the utility being subject to the terms and conditions of this License Agreement. To the extent permitted by applicable law, the Software Product may not be used, copied, modified, or distributed in any other way, and the Software Product may not be rented. Following applicable law, updated versions of the Software Product may be automatically downloaded and installed. Update versions are intended to improve, enhance, and further expand MAiMAi's Services by fixing defects, enhancing functionality, adding new software modules and new versions, and may include amendments to the terms and conditions of this Software License Agreement. The user agrees to receive such updated versions when using the Services. This License Agreement shall apply to all updated versions, upgraded versions, and additional features not distributed under separate licenses or other agreements.

3.11 Legal, Copyright, and Other Notices

The Software Product is protected by copyright laws, international copyright treaties, and other intellectual property laws and treaties. MAiMAi reserves all rights to the Software Product. The Software Product is owned by MAiMAi and is protected by applicable copyright, trademark, trade secret, patent, and other intellectual property laws. As between User and MAiMAi, MAiMAi owns and shall continue to hold all rights, titles, and interests in and to the Software Product, including related intellectual property rights, under applicable intellectual property laws. This Agreement does not grant the User any ownership interest in the Software Product, but only a revocable, limited right to use it under this Agreement. The final right of interpretation shall remain with the copyright owner.

3.12 Applicable Standards

It shall be as per the industry standard. HTML, TCP/IP, SMTP, POP and FTP protocols are followed. This license agreement shall be governed by and construed under the laws of the courts of the State of Florida, USA.

4. Use-Case Model for the system

See the Example Use Case for a Bank ATM attached to this exercise. Also, check the slides uploaded to Microsoft teams.

The sample file is also attached to the assignment instructions.

You can use any tool to create a use case diagram.

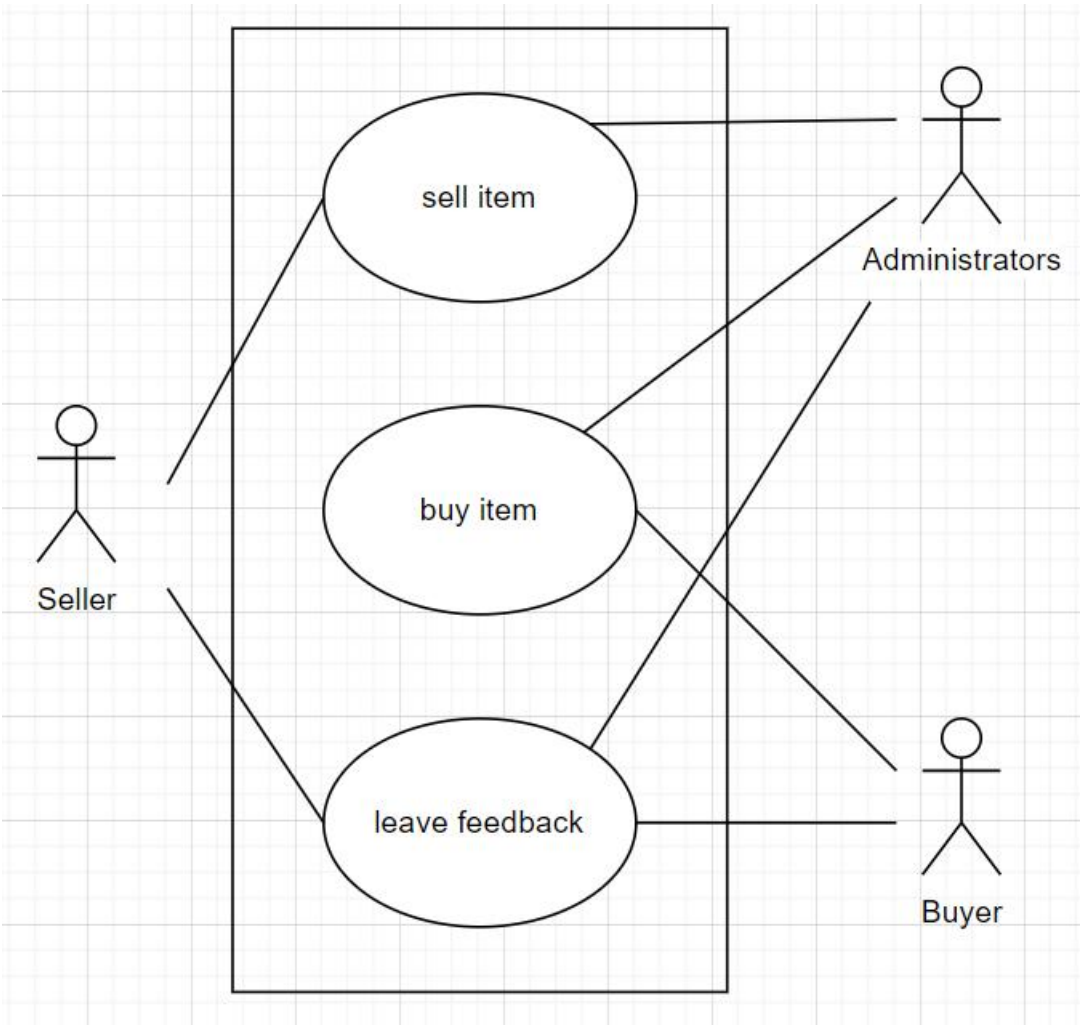
4.1 Introduction

This is a use-case model for the application. The model helps to identify the primary actors and use cases.

4.2 Overview

MaiMai is a used item trading platform mobile application. The application aims to help users sell and buy used items easily, cheaply, and safely. It also helps in the protection of the environment and reduction of waste generation.

4.3 Use-case Diagram



5. List of Actors

5.1 Seller

This actor represents a person with a MaiMai account and has items on the application that need to be sold.

5.2 Buyer

This actor represents a person with a MaiMai account who needs to buy items on the application.

5.3 Administrator

This actor represents a person with a MaiMai management account who can maintain the system.

6. List of Use Cases

6.1 Sell item

This use case describes how the MaiMai user leverages the application's framework to sell their item to other people.

6.2 Buy item

This use case describes how the MaiMai user utilizes the application to buy items from other people.

6.3 Leave feedback

This use case describes how the MaiMai user, after finishing the trade for an item, uses the application to leave feedback regarding their satisfaction levels.

Use-Case: Sell products

7. Brief Description

This usage example will describe how sellers can place their unused items on the MAiMAi platform.

8. Actor Brief Descriptions

8.1 Seller

8.2 Buyer

8.3 Administrators

9. Preconditions

The user's phone has the full MAiMAi application installed and connected to the network.

The server is not under maintenance.

Users need to link a bank card or other payment method with the MAiMAi payment function.

10. Basic Flow of Events

1. The use case begins when the Seller opens the MAiMAi application.
2. Use Case: Validate User is performed.
3. MAiMAi displays the client UI, showing the most recently traded items and feature options. In this case, the User would select "Sell Item."
4. The client displays a form for the Seller to fill out with information about the item to be traded.
5. The Seller fills in the transaction text information and selects the item image.
6. The Seller's account information, the item(s) to be traded and the image of the item(s) are uploaded to MAiMAi's server.

7. The server generates the item UI based on the information uploaded by the Seller and displays it to the MAiMAi client.
8. The Seller is informed that the item is successfully listed.
9. The use case ends successfully.

11. Alternative Flows

11.1 Invalid User

If, in step 2 of the basic flow, the Seller does not complete the form successfully, then the use case ends with a failure condition.

11.2 Missing text information/images for uploaded transactions

If the Seller fails to fill in all necessary information in the form (i.e. in step 5 of the basic flow), for instance, the Seller has only uploaded pictures or has only filled in text information, then;

1. Client prompts, "Missing text information/picture, please be sure to fill in the complete."

2. The use case resumes at step 5

12. Key Scenarios

12.1 MAiMAi server maintenance in progress, no response.

13. Post-conditions

13.1 Successful Completion

Users can successfully upload their trade items and display them to other users on the MAiMAi app.

13.2 Failure Condition

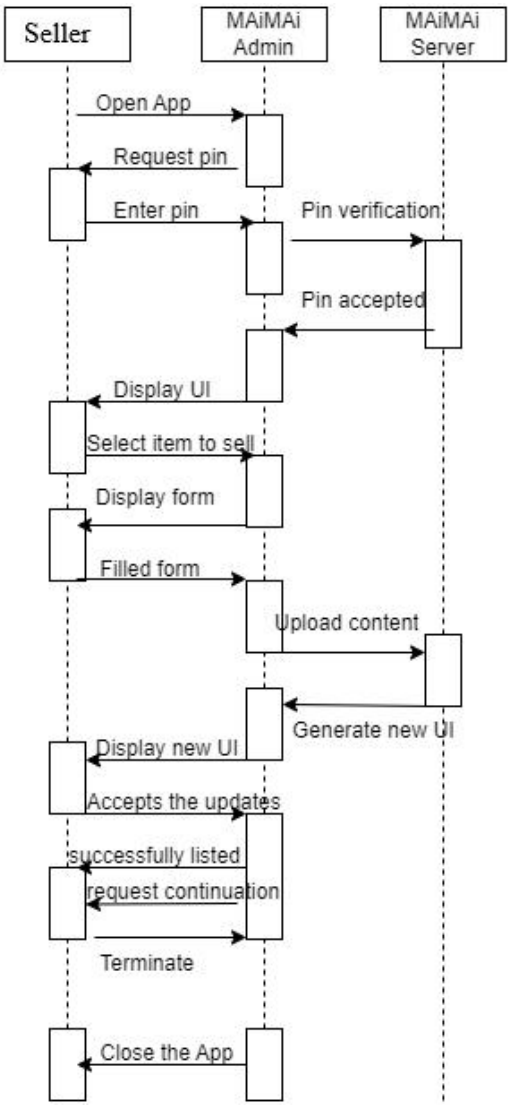
The logs have been updated accordingly.

14. Special Requirements

[SpReq:WC-1] MAiMAi server will record and backup the information of the transaction items uploaded by users, including the date and time of uploading text images, etc.

[SpReq:WC-2] Each User cannot list more than ten items per day.

15. Sequence Diagram for the Basic Flow



Use-Case: Buy products

16. Brief Description

This usage example will describe how buyers can buy the products they want on the MAiMAi platform.

17. Actor Brief Descriptions

17.1 Buyer

17.2 Administrators

18. Preconditions

The User's phone has the full MAiMAi application installed and connected to the network.

MAiMAi server is not under maintenance.

User needs to link their bank card with the payment function or other payment methods.

19. Basic Flow of Events

1. Buyer open maimai app.

2. The user logs in to their account.

3. MAiMAi displays the client UI, showing the most recently traded items and feature options. In this case, the User would select "Buy Item".

4. Buyers can search for their desired items using keywords, categories, usage time, etc.

5. The Buyer confirms the purchase, providing address and account information to the MAiMAi server.

6. Maimai app notifies the Seller and ships the item.

7. Buyer gets the logistics information, receives the goods, and successful transaction.

20. Alternative Flows

20.1 Invalid User

If in step 2 of the basic flow, the Buyer does not complete the form successfully, then the use case ends with a failure condition.

20.2 Missing text information/images for uploaded transactions

Buyer: If there is a problem with the payment method and payment cannot be complete

d because of missing information on the shipping address.

1. The client prompted, "missing text information/picture, please be sure to fill in the complete".

2. The use case resumes at step 5

3. Client prompt: "Please provide a payment method that can be used".

21. Key Scenarios

21.1 MAiMAi server maintenance is in progress. No response.

22. Post-conditions

22.1 Successful Completion

The Buyer successfully receives the purchased item, indicating that the transaction is complete, and can rate the purchase.

22.2 Failure Condition

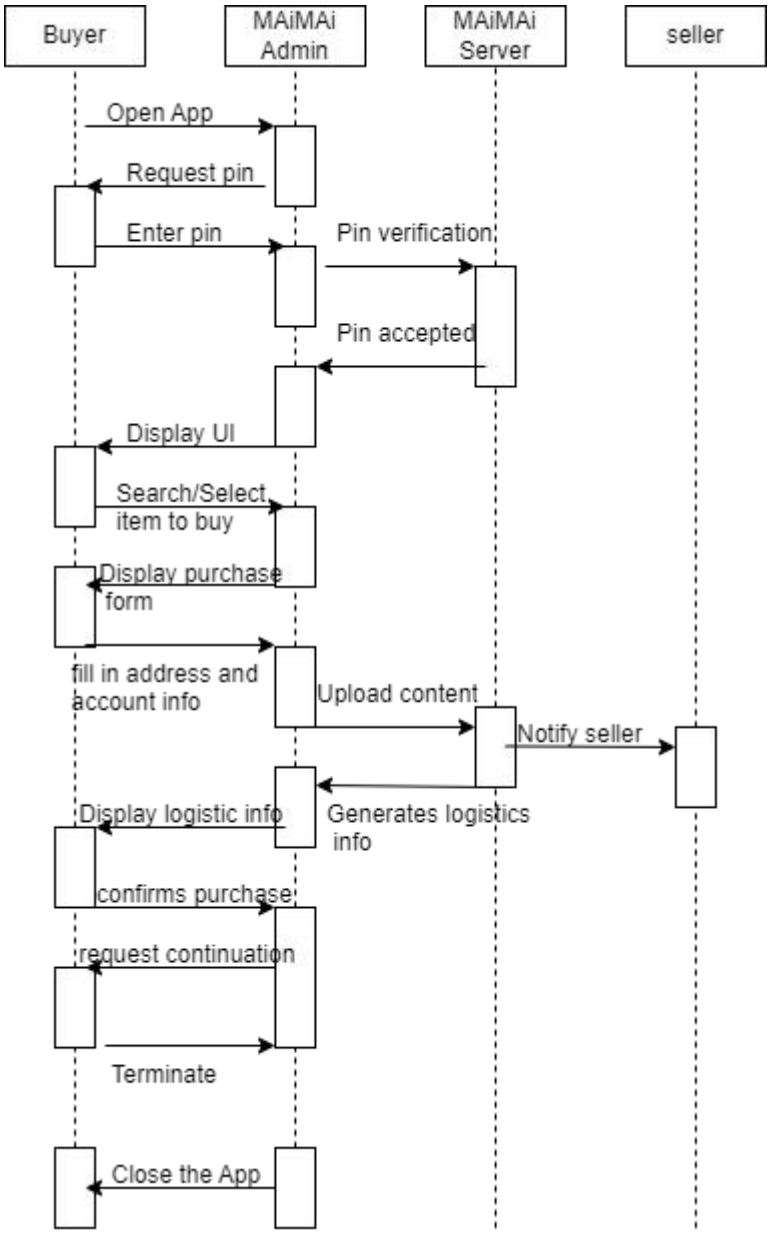
The logs have been updated accordingly.

23. Special Requirements

[SpReq:WC-1] MAiMAi server will record and backup the information of the transaction items uploaded by the User, including the date and time of uploading text images, etc.

[SpReq:WC-2] Buyers who need to return items will need to have their items inspected by MAiMAi officials for completeness to avoid damage.

24. Sequence Diagram for the Basic Flow



25. Software Architecture and Database

25.1 Conceptual Architecture:

Four conceptual architectures that apply to the MaiMai app include Client-Server Style, Blackboard Style, Publish-Subscribe Style, and Event-Based Style.

Client-Server Style is a layered style that sees each layer as a server for the layer above and as a client for the layer below and only has two layers (de Lima Fontao, dos Santos, & Dias-Neto, 2015). For the MaiMai mobile app, the buyer and seller will utilize linear layering when searching

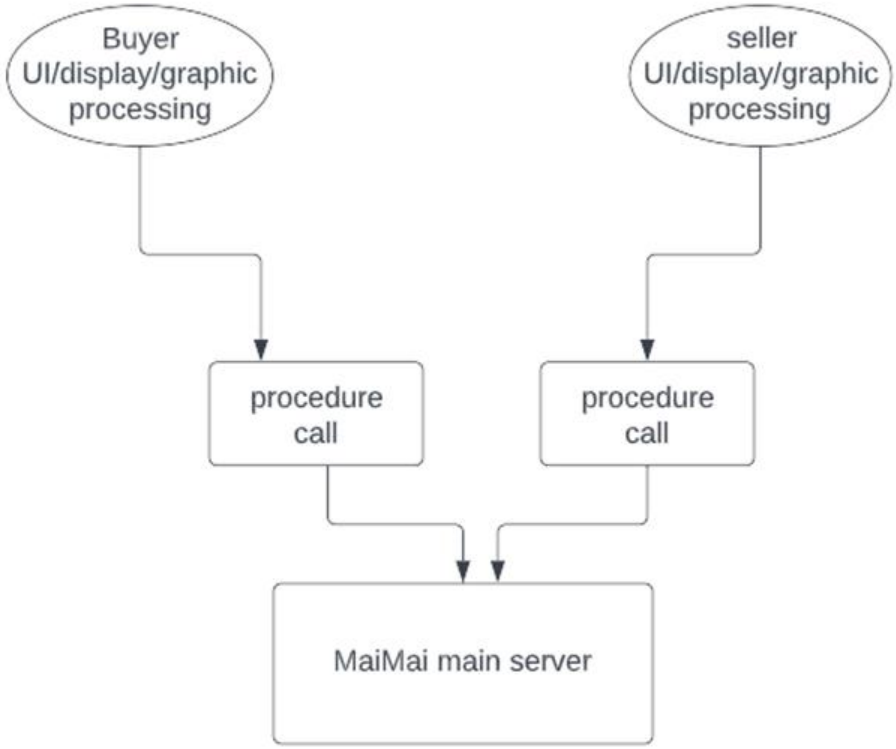
for products, placing orders, uploading products, and receiving orders from the seller. It is powerful to serve many clients.

Blackboard Style applies to MaiMai app as it offers a centralized data structure where all clients, sellers and buyers can easily access it.

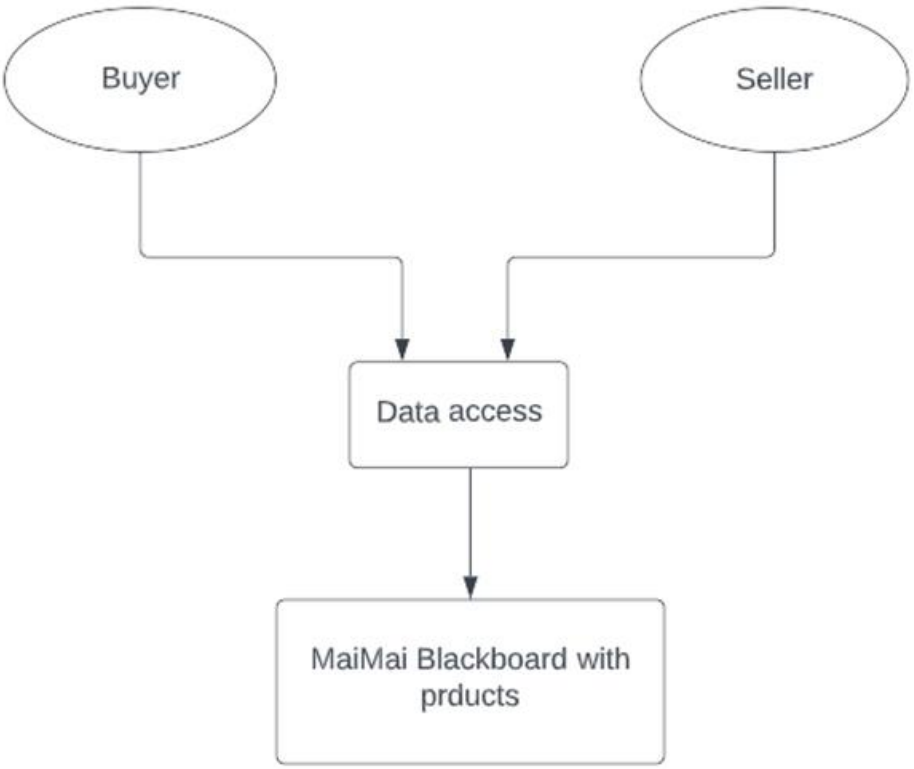
Publish-Subscribe Style allows the MaiMai system to decide what to display to the user. For instance, the buyer will be registered as a subscriber enabling the system to display products that suit their character, and the buyer will be registered as the publisher. The system will show orders for delivery to buyers only to sellers.

The event-Based Style involves various components communicated via the event bus, allowing asynchronous events emission. Since there are multiple events in the MaiMai app, such as searching for products and placing orders, this Style can be easy to use due to reduced complexity (Gruhn, & Köhler, 2006).

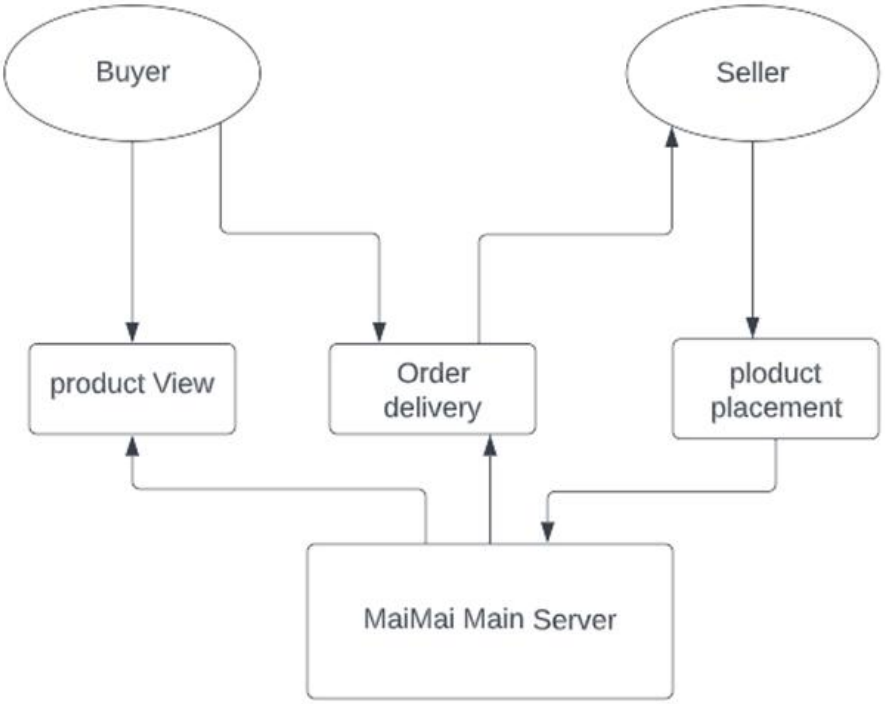
i. **Client-Server**



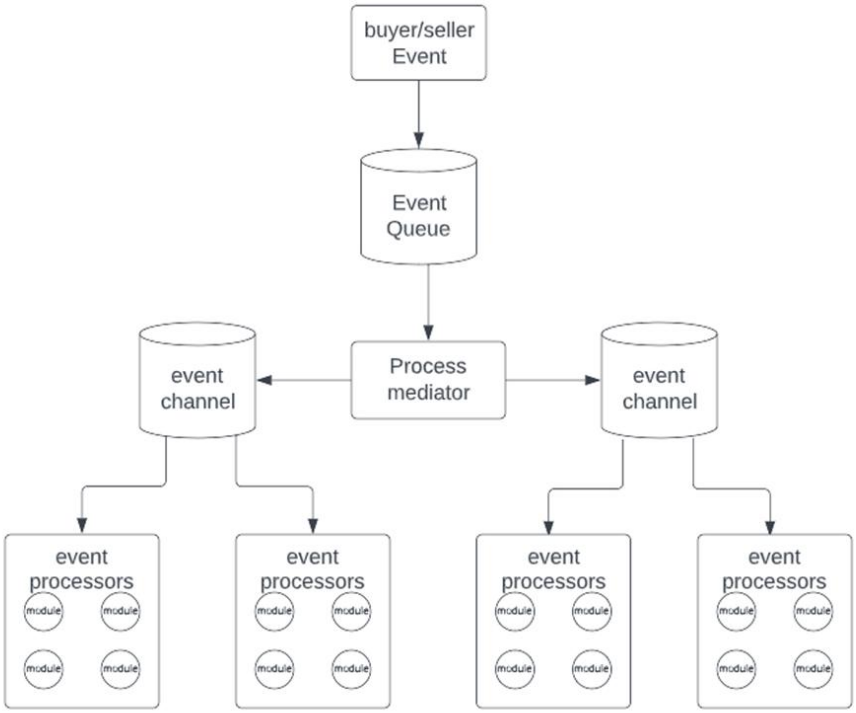
ii. **Blackboard Style**



iii. Publish-Subscribe Style



iv. **Event-Based Style**



25.2 Cloud Architecture.

MaiMai allows sellers to post their products on the site. Buyers can purchase the merchant's items from the site. The Basic Service Model we may choose is PaaS, so sellers can also create their applications on MaiMai to display their items and products. Sellers can use MaiMai's API to develop their applications.

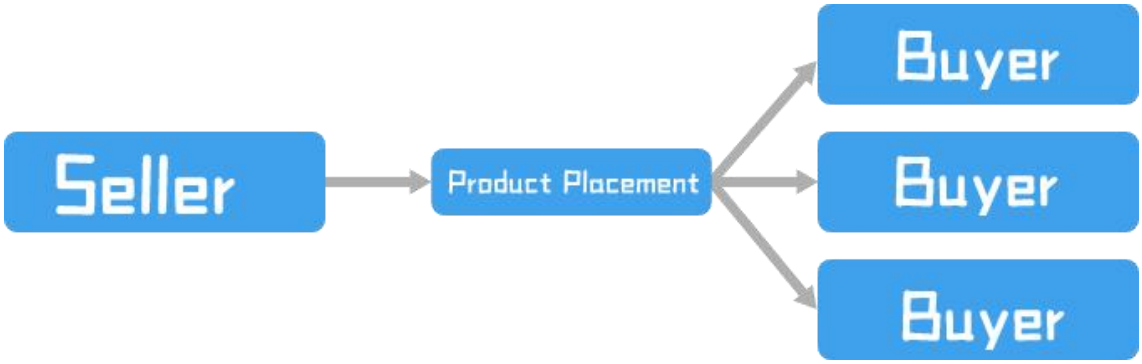
The Cloud Application Architecture we chose is:

i. **(Possible) Big data**

Big Data cloud application architecture is convenient for use by MaiMai system. It will help in collection of item transaction information, conduct real-time data analysis or use machine learning for predictive analysis, understand buyers' shopping preferences, help sellers better sell their items, and increase the platform's trading volume (Capilla et al. 2020).

ii. **(Main) Event Driven**

Similar to pub-sub, when a seller (pub) posts a second-hand item for trade, the buyer (sub) can receive the latest information about the item immediately. Because there is a lower time delay, it allows items to be posted in real-time and updates the transaction's status in real-time, preventing the error of duplicate transactions. This architecture has good scalability, therefore, it can handle many products posted simultaneously and it has the capability to store any product information. It also enables new users to be easily added.



25.3 Database

A transaction is a group of operations with the following properties: atomic, consistent, isolated, and durable (ACID). The support of transactions enables new applications to be developed while simplifying the development process and making the application more robust (Weber, Gabriel, Lux, & Menke, 2022). Transaction data is usually related to the organization's transactions and includes data captured, for example, when a product is sold or purchased. Master data is referred to in different transactions, examples are customer, product, or seller data.

Product master data

Product ID	Product name	Product description	Price
1982SP	Lamp	Metal floor-standing lamp	56.95
454YH	Chair	Oak dining chair	70.99

Customer master data

Customer ID	Customer name	Customer address
67	J. Smith	56 Quy Road, Chester, UK
68	R. Hurst	14 Back Lane, Norwich, UK

Transactions

Sale ID	Product ID	Customer ID	Actual Sale price	Sale time
002	1982SP	67	56.95	1/3/13 15.34.12
003	454YH	67	65.99	1/3/13 15.34.25
004	454YH	68	70.99	4/3/13 12.05.43

For our application, MaiMai, the characteristics of transactions can be reflected in many functions. For example, when a user needs to purchase an item, the transaction process goes through the following steps: Check that the account has enough money to cover the fee. If there is enough money in the account, the amount is debited from the account and credited to the MaiMai platform. The seller sends the goods, and the platform records the data. Maimai sends the money to the seller.

Transactional data will be stored in the Search Engine Database because the key characteristics of a search engine database are the ability to store and index information very quickly and provide fast response times for search requests Launio, E. (2019. Indexes can be multi-dimensional and support free-text searches across large volumes of text data. Indexing can be performed using a pull model triggered by the search engine database or an external application code initiated by a push model (Weber, Gabriel, Lux, & Menke, 2022).

User data for our application

Key/Value Stores. Users can find the items they need through the keyword index. A single key/value store can be highly scalable, as the data store can efficiently distribute data across multiple nodes on separate machines.

26. High Fidelity Wireframe

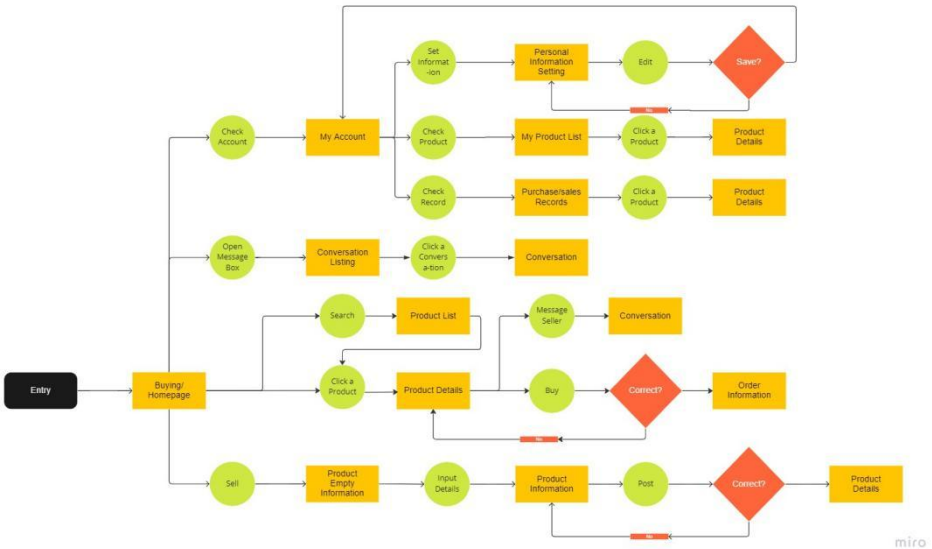


Figure 1: UX flow Chart

After entering the application, the user will see the homepage which is also the buying page. In this page, all the products posted by the user will be list in the selected order and category, so that the user can easily select and buy them. After entering this page, the user can choose to search or go to a specific product page and make a purchase. Before making a purchase, the program will confirm with the user before confirming the order. The user can also choose to communicate with the merchant and enter the conversation page.

Users can also visit the selling page, my account page and message box pages through the homepage.

For the selling page, users can quickly post the items they want to sell by filling in the information of the items, the application will also confirm with the user before posting to ensure the accuracy of the published information.

For the account page, users can easily access the personal information page and modify it from this page. Users can also check their selling or buying history and the items are selling.

For the message box page, users can continue communication and view chat history on this page.

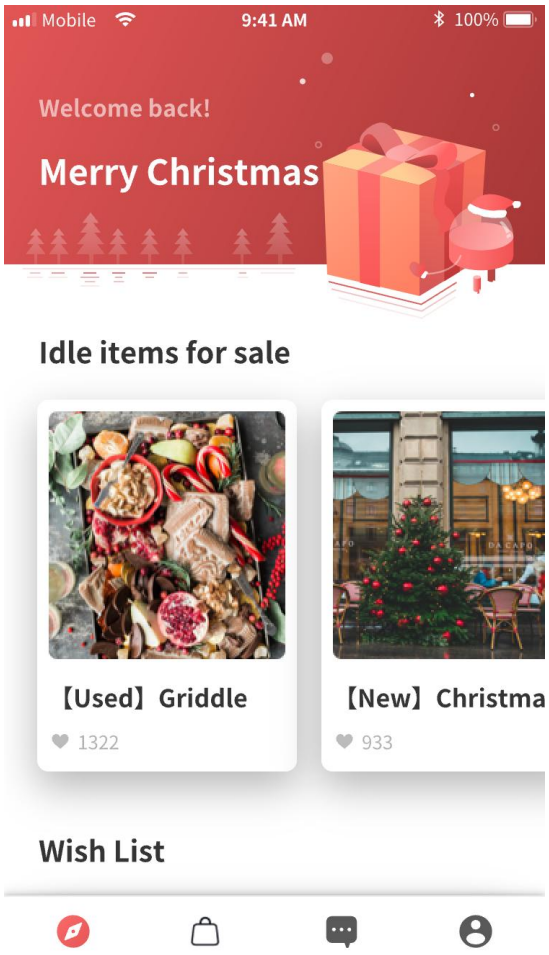


Figure 2: Homepage

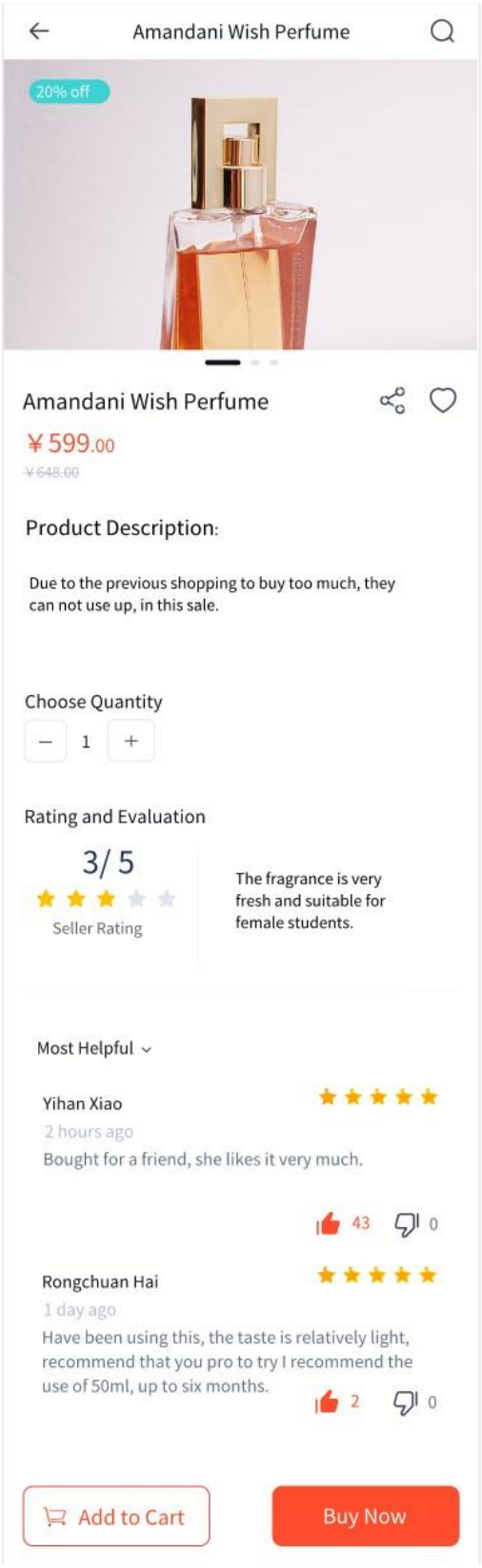


Figure 3: Product Information

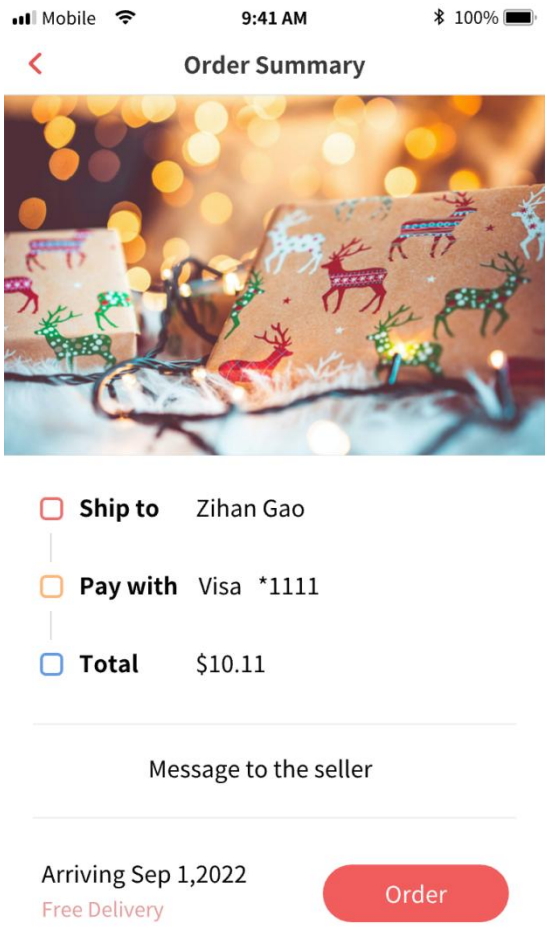


Figure 4: Order Summary

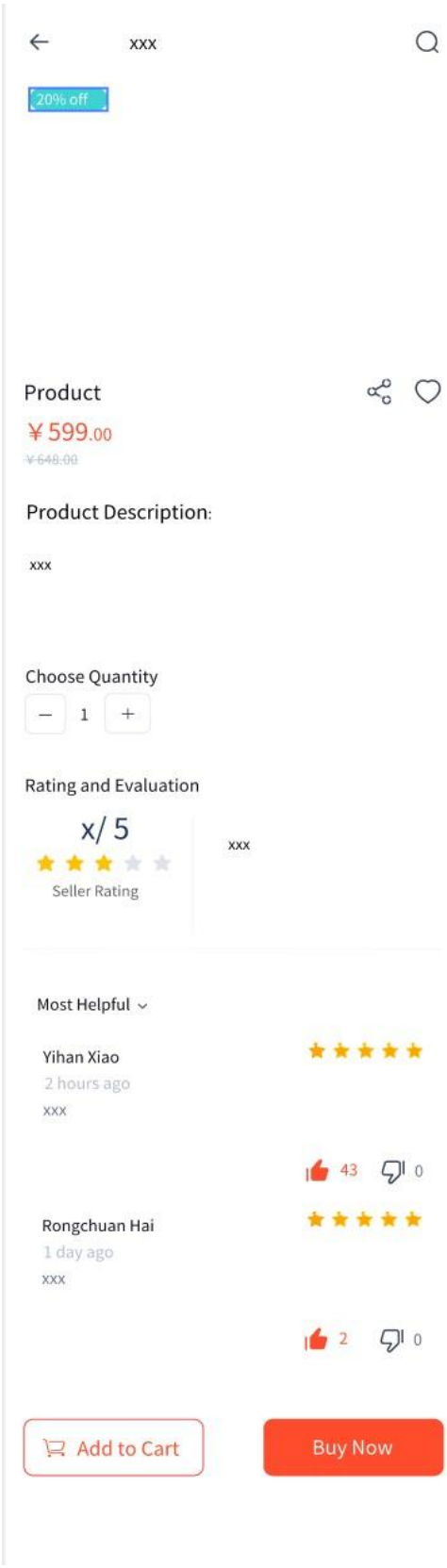


Figure 5: Product Empty Information

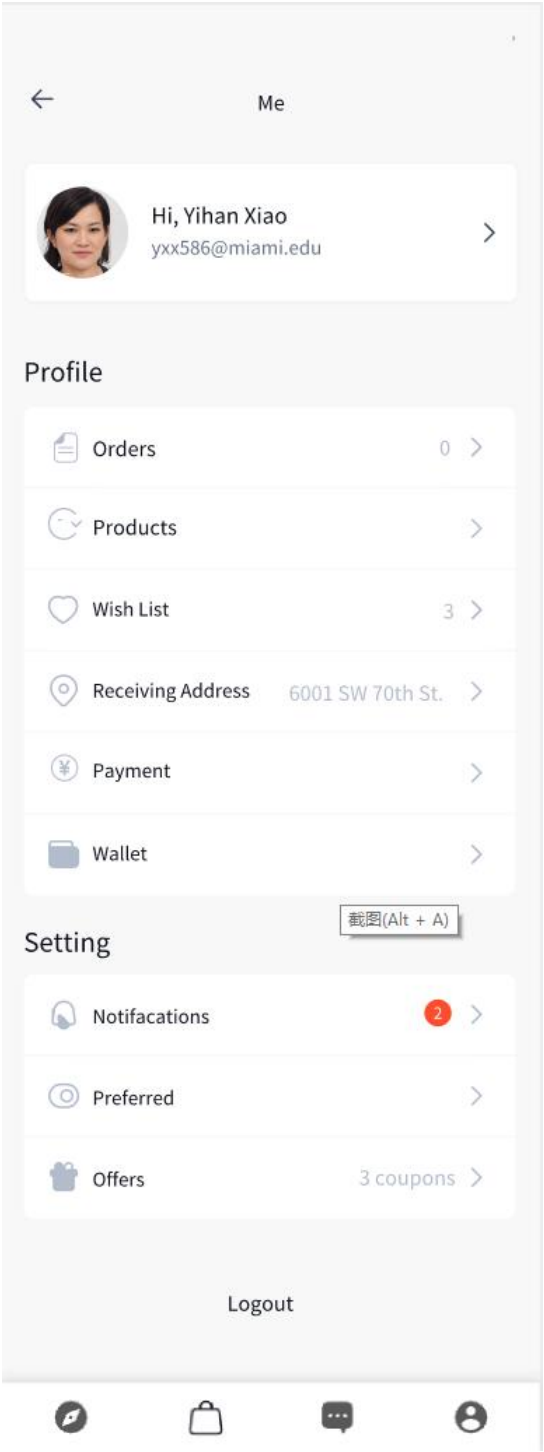


Figure 6: My Account

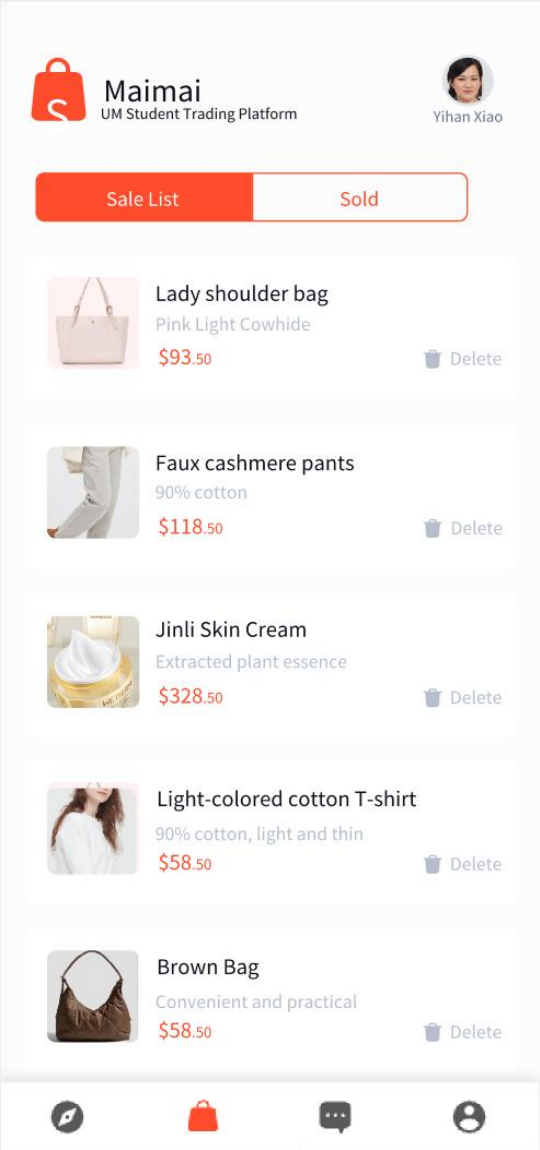


Figure 7: Selling status

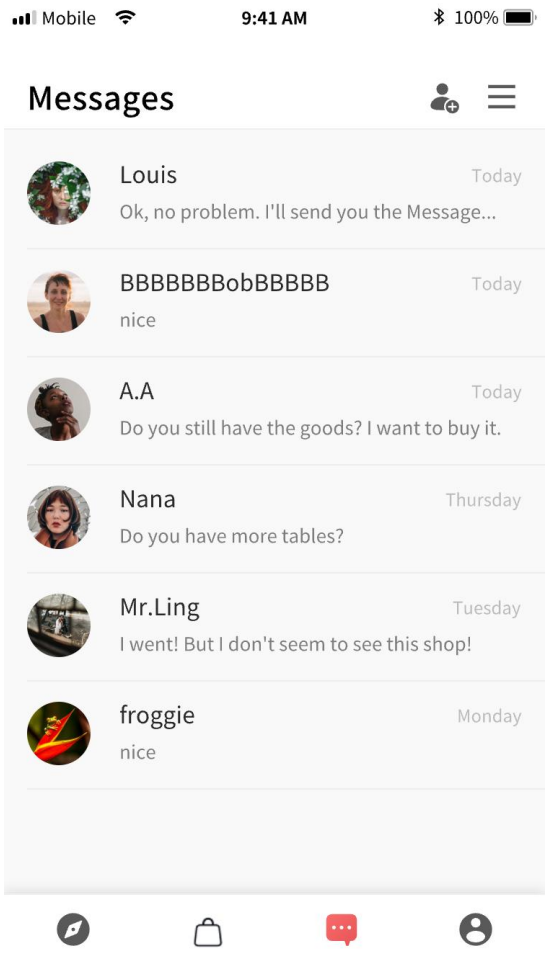


Figure 8: Message box

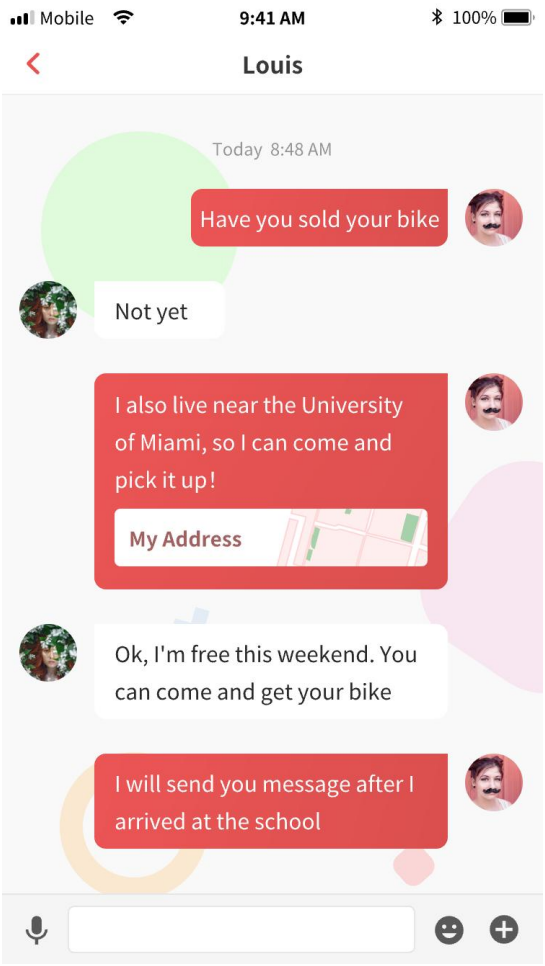


Figure 9: Conversation

27. Tech stack

27.1 Front End

a. JavaScript: Lightweight, interpreted, object-oriented language with first-class functions

b. jQuery: jQuery is a cross-platform JavaScript library designed to simplify the client-side scripting of HTML. It is a tool in the JavaScript UI Libraries category of a tech stack.

c. PHP: Fast, flexible and pragmatic, PHP powers everything from your blog to the most popular websites in the world. PHP is a tool in the Languages category of a tech stack.

d. HTML5: HTML5 is a core technology markup language of the Internet used for structuring and presenting content for the World Wide Web.

e. NGINX: nginx [engine x] is an HTTP and reverse proxy server, as well as a mail proxy server, written by Igor Sysoev. NGINX is a tool in the Web Servers category of a tech stack.

f. Bootstrap: Bootstrap is the most popular HTML, CSS, and JS framework for developing responsive, mobile first projects on the web.

g. Git: Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency. Git is a tool in the Version Control System category of a tech stack.

27.2 Backend Logic

h. JavaScript: Lightweight, interpreted, object-oriented language with first-class functions.

i. ES6: Goals for ECMAScript 2015 include providing better support for large applications, library creation, and for use of ECMAScript as a compilation target for other languages. Some of its major enhancements include modules, class declarations, lexical block scoping, iterators and generators, promises for asynchronous programming, destructuring patterns, and proper tail calls.

j. Redis: Redis is an open source (BSD licensed), in-memory data structure store, used as a database, cache, and message broker. Redis provides data structures such as strings, hashes, lists, sets, sorted sets with range queries, bitmaps, hyperloglogs, geospatial indexes, and streams.

k. ExpressJS: Express is a minimal and flexible node.js web application framework, providing a robust set of features for building single and multi-page, and hybrid web applications.

l. AWS Lambda: AWS Lambda is a compute service that runs your code in response to events and automatically manages the underlying compute resources for you. You can use AWS Lambda to extend other AWS services with custom logic, or create your own back-end services that operate at AWS scale, performance, and security.

27.3 Database

m. sql-server: Terraform module for Azure where it creates a SQL server with initial database and Azure AD login. sql-server is a tool in the Terraform Packages category of a tech stack.

n. MySQL: The MySQL software delivers a very fast, multi-threaded, multi-user, and robust SQL (Structured Query Language) database server. MySQL Server is intended for mission-critical, heavy-load production systems as well as for embedding into mass-deployed software.

27.4 DevOps

o. Docker: Docker Platform is the industry-leading container platform for continuous, high-velocity innovation, enabling organizations to seamlessly build and share any application — from legacy to what comes next — and securely run them anywhere. Docker is a tool in the Virtual Machine Platforms & Containers category of a tech stack.

p. Jenkins: An extendable open source continuous integration server. In a nutshell Jenkins CI is the leading open-source continuous integration server. Built with Java, it provides over 300 plugins to support building and testing virtually any project. Jenkins is a tool in the Continuous Integration category of a tech stack. Jenkins is an

open source tool with 20K GitHub stars and 7.8K GitHub forks.

q. TeamCity: TeamCity is an ultimate Continuous Integration tool for professionals. TeamCity is a user-friendly continuous integration (CI) server for professional developers, build engineers, and DevOps. It is trivial to setup and free for small teams and open-source projects. TeamCity is a tool in the Continuous Integration category of a tech stack.

28. Work Breakdown Structure

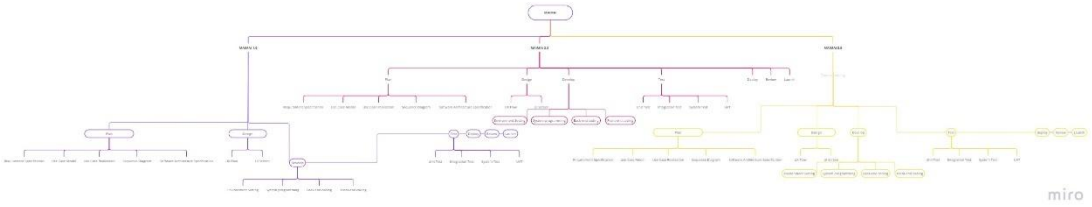


Figure 1: Work Breakdown Structure Chart

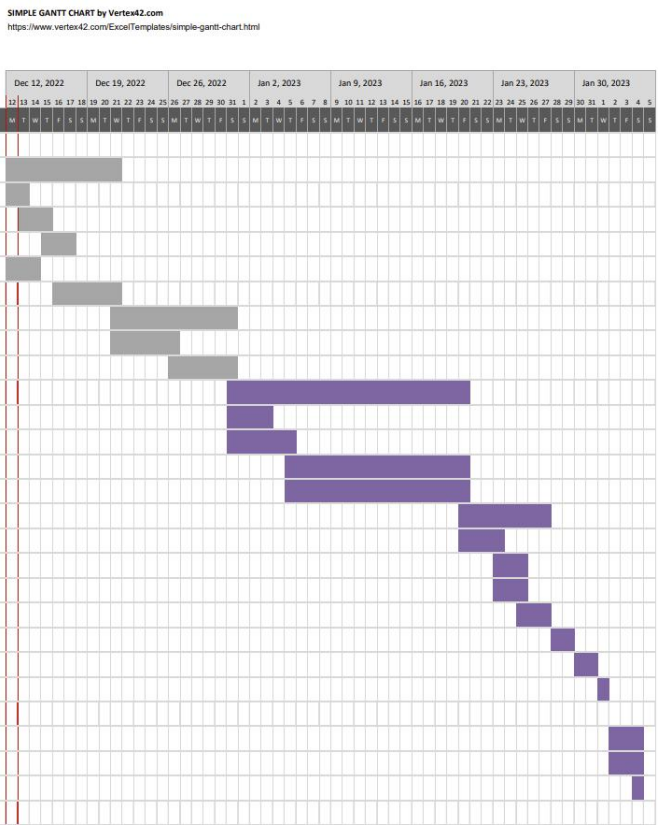
29. Project Timeline Plan

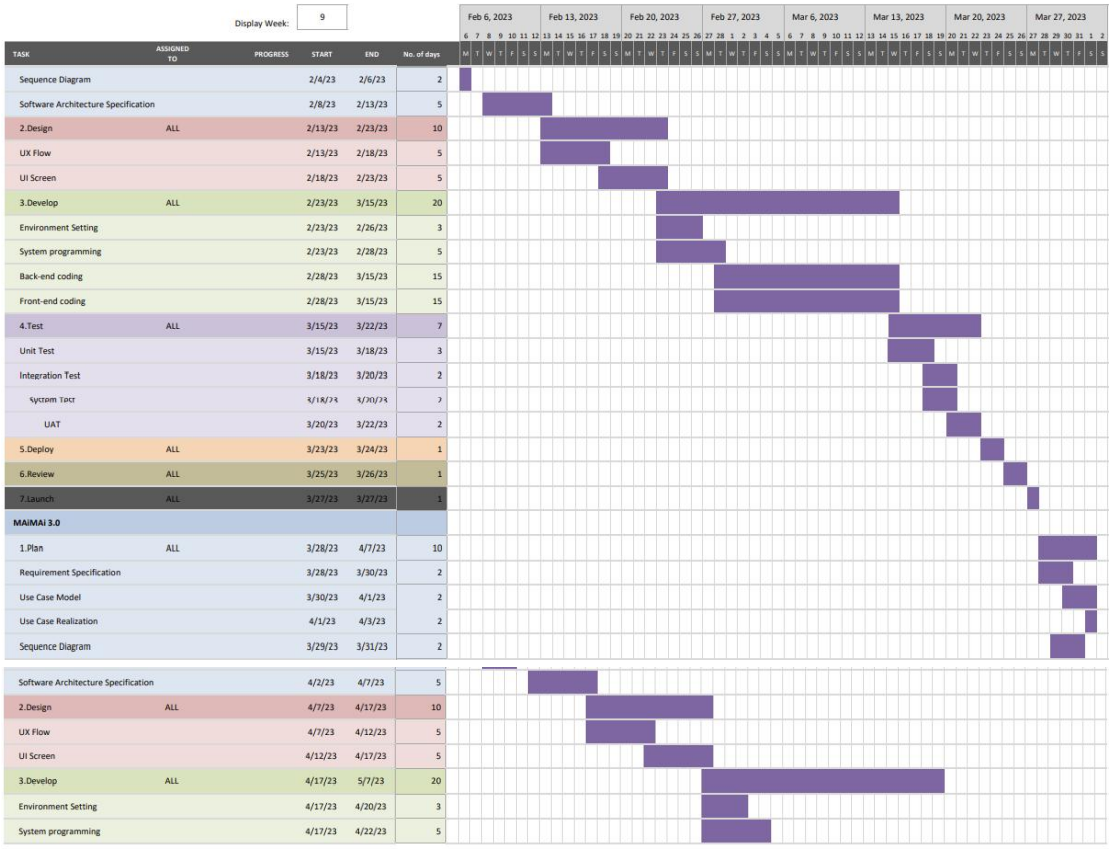
MAiMAi
Love412Forever
Zihan Gao

SIMPLE GANTT CHART by Vertex42.com
https://www.vertex42.com/ExcelTemplates/simple-gantt-chart.html

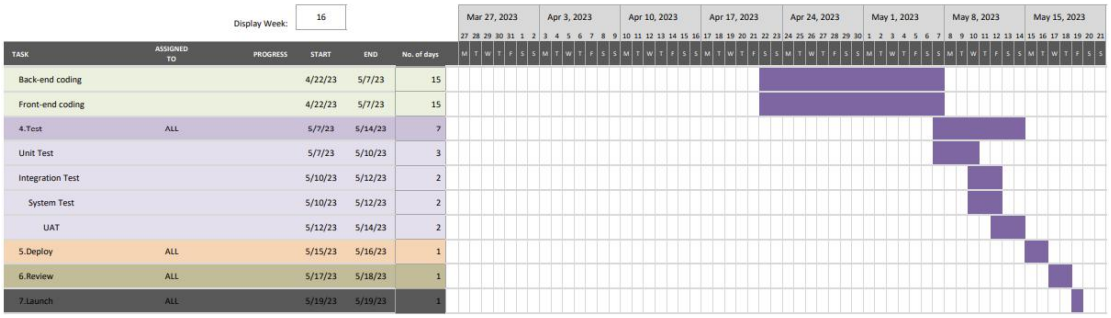
Project Start: Sun, 12/11/2022
Display Week: 1

TASK	ASSIGNED TO	PROGRESS	START	END	No. of days
MAiMAi 1.0					
1.Plan	All	100%	12/11/22	12/21/22	10
Requirement Specification		100%	12/11/22	12/13/22	2
Use Case Model		100%	12/13/22	12/15/22	2
Use Case Realization		100%	12/15/22	12/17/22	2
Sequence Diagram		100%	12/12/22	12/14/22	2
Software Architecture Specification		100%	12/16/22	12/21/22	5
2.Design	ALL	100%	12/21/22	12/31/22	10
UX Flow		100%	12/21/22	12/26/22	5
UI Screen		100%	12/26/22	12/31/22	5
3.Develop	ALL		12/31/22	1/20/23	20
Environment Setting			12/31/22	1/3/23	3
System programming			12/31/22	1/5/23	5
Back-end coding			1/5/23	1/20/23	15
Front-end coding			1/5/23	1/20/23	15
4.Test	ALL		1/20/23	1/27/23	7
Unit Test			1/20/23	1/23/23	3
Integration Test			1/23/23	1/25/23	2
System Test			1/23/23	1/25/23	2
UAT			1/25/23	1/27/23	2
5.Deploy	ALL		1/28/23	1/29/23	1
6.Review	ALL		1/30/23	1/31/23	1
7.Launch	ALL		2/1/23	2/1/23	1
MAiMAi 2.0					
1.Plan	ALL		2/2/23	2/12/23	10
Requirement Specification			2/2/23	2/4/23	2
Use Case Model			2/4/23	2/6/23	2
Use Case Realization			2/6/23	2/8/23	2





Display Week:	9				
TASK	ASSIGNED TO	PROGRESS	START	END	No. of days
Sequence Diagram			2/4/23	2/6/23	2
Software Architecture Specification			2/8/23	2/13/23	5
2.Design	ALL		2/13/23	2/23/23	10
UX Flow			2/13/23	2/18/23	5
UI Screen			2/18/23	2/23/23	5
3.Develop	ALL		2/23/23	3/15/23	20
Environment Setting			2/23/23	2/26/23	3
System programming			2/23/23	2/28/23	5
Back-end coding			2/28/23	3/15/23	15
Front-end coding			2/28/23	3/15/23	15
4.Test	ALL		3/15/23	3/22/23	7
Unit Test			3/15/23	3/18/23	3
Integration Test			3/18/23	3/20/23	2
System Test			3/18/23	3/20/23	2
UAT			3/20/23	3/22/23	2
5.Deploy	ALL		3/23/23	3/24/23	1
6.Review	ALL		3/25/23	3/26/23	1
7.Launch	ALL		3/27/23	3/27/23	1
MAIMAi 3.0					
1.Plan	ALL		3/28/23	4/7/23	10
Requirement Specification			3/28/23	3/30/23	2
Use Case Model			3/30/23	4/1/23	2
Use Case Realization			4/1/23	4/3/23	2
Sequence Diagram			3/29/23	3/31/23	2
Software Architecture Specification			4/2/23	4/7/23	5
2.Design	ALL		4/7/23	4/17/23	10
UX Flow			4/7/23	4/12/23	5
UI Screen			4/12/23	4/17/23	5
3.Develop	ALL		4/17/23	5/7/23	20
Environment Setting			4/17/23	4/20/23	3
System programming			4/17/23	4/22/23	5



Display Week:	16				
TASK	ASSIGNED TO	PROGRESS	START	END	No. of days
Back-end coding			4/22/23	5/7/23	15
Front-end coding			4/22/23	5/7/23	15
4.Test	ALL		5/7/23	5/14/23	7
Unit Test			5/7/23	5/10/23	3
Integration Test			5/10/23	5/12/23	2
System Test			5/10/23	5/12/23	2
UAT			5/12/23	5/14/23	2
5.Deploy	ALL		5/15/23	5/16/23	1
6.Review	ALL		5/17/23	5/18/23	1
7.Launch	ALL		5/19/23	5/19/23	1

Figure 2: Project Timeline Plan

ECE412-GanttChart (1) (1).pdf

30. Project Cost Plan

Project Lead: Zihan Gao
Start Date: Sun, 12/11/2022

		BUDGET	ACTUAL	Under(Over)
	Total	$ 7,484.31	$ 7,465.00	$ 19.31

		Labor		Materials		Fixed			
WBS	Task	Hrs	Rate	Units	$/Unit	Costs	Budget	Actual	Under(Over)
1	MAiMai 1.0						$ 2,177.69	$ 2,205.00	$ (27.31)
1.1	Plan	1.2	$12.50	5.0	$2.89	$51.06	80.37	150.00	(69.63)
1.1.1	Requirement Specification	10.6	$11.25	6.3	$1.79	$250.60	381.16	400.00	(18.84)
1.1.2	Use Case Model	9.1	$11.35	4.8	$2.30	$48.99	114.23	100.00	14.23
1.1.3	Use Case Realization	9.9	$12.31	5.5	$1.70	$50.31	131.15	140.00	(8.85)
1.1.4	Sequence Diagram	8.3	$11.76	6.0	$2.45	$20.30	112.31	120.00	(7.69)
1.1.5	Software Architecture Specification	15.4	$13.45	3.9	$2.14	$20.30	215.44	230.00	(14.56)
1.2	Design	1.0	$14.87	9.9	$4.95	$4.50	63.73	65.00	(1.27)
1.2.1	UX Flow	12.6	$14.45	10.5	$5.06	$44.98	280.38	250.00	30.38
1.2.2	UI Screen	13.5	$13.65	10.7	$5.73	$46.80	292.10	300.00	(7.90)
1.3	Develop	1.1	$17.64	11.1	$5.36	$47.00	125.90	100.00	25.90
1.3.1	Environment setting	18.4	$16.35	9.9	$8.09	$43.00	380.93	350.00	30.93
1.3.2	System programming	31.0	$15.89	10.0	$7.77	$44.50	570.29	500.00	70.29
1.3.3	Back-end coding	91.4	$17.01	10.3	$8.03	$45.50	1,637.42	1,700.00	(62.58)
1.3.4	Front-end coding	89.7	$16.57	11.4	$8.30	$44.25	1,580.95	1,500.00	80.95
1.4	Test	1.2	$15.37	5.5	$11.97	$61.75	146.03	100.00	46.03
1.4.1	Unit Test	17.9	$15.80	13.4	$3.68	$60.70	392.83	400.00	(7.17)
1.4.2	Integration Test	12.1	$17.78	15.6	$4.03	$58.00	336.01	330.00	6.01
1.4.3	System Test	12.4	$16.75	14.6	$4.40	$62.00	333.94	300.00	33.94
1.4.4	UAR	11.9	$16.00	14.3	$4.84	$61.50	321.11	300.00	21.11
1.5	S.Deploy	7.9	$15.05	13.9	$31.66	$49.70	608.67	500.00	108.67
1.6	6.Review	8.2	$13.96	15.0	$33.04	$40.00	650.07	500.00	150.07
2	MAiMai 2.0						$ 2,177.69	$ 2,205.00	$ (27.31)
2.1	Plan	1.2	$12.50	5.0	$2.89	$51.06	80.37	150.00	(69.63)
2.1.1	Requirement Specification	10.6	$11.25	6.3	$1.79	$250.60	381.16	400.00	(18.84)
2.1.2	Use Case Model	9.1	$11.35	4.8	$2.30	$48.99	114.23	100.00	14.23
2.1.3	Use Case Realization	9.9	$12.31	5.5	$1.70	$50.31	131.15	140.00	(8.85)
2.1.4	Sequence Diagram	8.3	$11.76	6.0	$2.45	$20.30	112.31	120.00	(7.69)
2.1.5	Software Architecture Specification	15.4	$13.45	3.9	$2.14	$20.30	215.44	230.00	(14.56)
2.2	Design	1.0	$14.87	9.9	$4.95	$4.50	63.73	65.00	(1.27)
2.2.1	UX Flow	12.6	$14.45	10.5	$5.06	$44.98	280.38	250.00	30.38
2.2.2	UI Screen	13.5	$13.65	10.7	$5.73	$46.80	292.10	300.00	(7.90)
2.3	Develop	1.1	$17.64	11.1	$5.36	$47.00	125.90	100.00	25.90
2.3.1	Environment setting	18.4	$16.35	9.9	$8.09	$43.00	380.93	350.00	30.93
2.3.2	System programming	31.0	$15.89	10.0	$7.77	$44.50	570.29	500.00	70.29
2.3.3	Back-end coding	91.4	$17.01	10.3	$8.03	$45.50	1,637.42	1,700.00	(62.58)
2.3.4	Front-end coding	89.7	$16.57	11.4	$8.30	$44.25	1,580.95	1,500.00	80.95
2.4	Test	1.2	$15.37	5.5	$11.97	$61.75	146.03	100.00	46.03
2.4.1	Unit Test	17.9	$15.80	13.4	$3.68	$60.70	392.83	400.00	(7.17)
2.4.2	Integration Test	12.1	$17.78	15.6	$4.03	$58.00	336.01	330.00	6.01
2.4.3	System Test	12.4	$16.75	14.6	$4.40	$62.00	333.94	300.00	33.94
2.4.4	UAR	11.9	$16.00	14.3	$4.84	$61.50	321.11	300.00	21.11
2.5	S.Deploy	7.9	$15.05	13.9	$31.66	$49.70	608.67	500.00	108.67
2.6	6.Review	8.2	$13.96	15.0	$33.04	$40.00	650.07	500.00	150.07
3	MAiMai 3.0						$ 2,177.69	$ 2,205.00	$ (27.31)
3.1	Plan	1.2	$12.50	5.0	$2.89	$51.06	80.37	150.00	(69.63)
3.1.1	Requirement Specification	10.6	$11.25	6.3	$1.79	$250.60	381.16	400.00	(18.84)
3.1.2	Use Case Model	9.1	$11.35	4.8	$2.30	$48.99	114.23	100.00	14.23
3.1.3	Use Case Realization	9.9	$12.31	5.5	$1.70	$50.31	131.15	140.00	(8.85)
3.1.4	Sequence Diagram	8.3	$11.76	6.0	$2.45	$20.30	112.31	120.00	(7.69)
3.1.5	Software Architecture Specification	15.4	$13.45	3.9	$2.14	$20.30	215.44	230.00	(14.56)
3.2	Design	1.0	$14.87	9.9	$4.95	$4.50	63.73	65.00	(1.27)
3.2.1	UX Flow	12.6	$14.45	10.5	$5.06	$44.98	280.38	250.00	30.38
3.2.2	UI Screen	13.5	$13.65	10.7	$5.73	$46.80	292.10	300.00	(7.90)
3.3	Develop	1.1	$17.64	11.1	$5.36	$47.00	125.90	100.00	25.90
3.3.1	Environment setting	18.4	$16.35	9.9	$8.09	$43.00	380.93	350.00	30.93

Figure 3: Project Cost Plan

References

Capilla, R., Kazman, R., Romera, C., & Carrillo, C. (2020). Usability implications in software architecture: The case study of a mobile app. *Software: Practice and Experience*, *50*(12), 2145-2168.

Gruhn, V., & Köhler, A. (2006). Aligning Software Architectures of Mobile Applications on Business Requirements. In *WISM*.

De Lima Fontao, A., dos Santos, R. P., & Dias-Neto, A. C. (2015, July). Mobile software ecosystem (mseco): a systematic mapping study. In *2015 IEEE 39th Annual Computer Software and Applications Conference* (Vol. 2, pp. 653-658). IEEE.

Launio, E. (2019). Digitalization & Transactional Information Flow Analysis of a Pulp Supply Chain.

Weber, P., Gabriel, R., Lux, T., & Menke, K. (2022). Database Systems. In *Basics in Business Informatics* (pp. 123-150). Springer Vieweg, Wiesbaden.

Alexander Borek, ... Philip Woodall, in Total Information Risk Management, 2014.

Tools for our flow design: https://miro.com

Tools for our Wireframing: https://js.design
https://js.design/f/cHQPii?p=9C797875
https://js.design/f/wMe8ji?p=ksmNJUCfgt

Tech stack refer: https://stackshare.io/stacks