Maverick Home Center

CSE 5325-004: Software Engineering: Analysis, Design, and Testing

Group No. 13 - Mavericks

Team Members

S. No.	Student Name	UTA ID
1.	Aditya Vikram Bhat	1002014494
2.	Akash Biswas	1002055500
3.	Amit Gupta	1002066302
4.	Ravi Rajpurohit	1002079916

Maverick Home Center

Maverick Home Center is an app that provides home décor, used furniture, and all other accessories that one needs/wants to study. This app is for students at the University of Texas at Arlington and living close to the campus. This would allow students to have cheap (possibly free) and affordable furniture and would save the time and resources it would take to buy them firsthand or from other sellers. Maverick Home Center is here to help you whether you want to sell things you no longer need or if you're seeking a certain item at a good price.

There are many online marketplaces for buying and selling used goods, including eBay, Facebook Marketplace, Etsy, and Ebid. However, the main issue with the worldwide market is that anyone can pose as a buyer or a seller, which has the potential to defraud other users. This would ultimately result in a loss of faith in the system. Additionally, the user interface and user experience of the current market are so overwhelming that any first-time user could run into problems while attempting to complete fundamental tasks on the application.

The main advantage of having this application is that it is made by students with the aim to provide a seamless and easy-to-access place where one can get all the items one needs to have a convenient and easy journey to studies and have an immersive college life experience. Maverick Home Center, which is only available to students at The University of Texas at Arlington, overcomes all the aforementioned problems by gathering all the products in one location. Here, a buyer and seller can interact to lay the groundwork for negotiation and to have a better understanding of the state of the commodity. There would be no chance of being duped because the application only allows students who have been validated by the university to register. Additionally, the user can sell or buy a thing with just a few clicks because of the interactive design's simple user interface.

Overall Design Approach

The application would be developed by following the Object Oriented Design. Object-oriented programming (OOP) is a programming methodology that emphasizes the use of objects and classes to represent real-world entities and concepts. Object-oriented design works around the entities and their characteristics instead of the functions involved in the software design. There would be two major entities involved in the project -

- Buyer
- Seller
- Platform Maintainer

The application will support certain features which will allow the user to do the following actions -

- Searching for the product
- Placing a buy request
- Communicating with the sellers

Also, the sellers will be able to do certain actions -

- Upload product images
- Insert product descriptions
- Communicate with the buyer

The development would follow a Top-Down design approach by utilizing Agile Methodology. The entire application would be broken down into multiple sub-systems and components. Then, those sub-systems would be analyzed and worked upon until the entire application is built. The tentative sub-systems defined are mentioned below -

- The primary goal is to create a personalized onboarding and login system to collect initial custom data. This data will be utilized to develop the following sub-systems.
- This would be followed by the development of the landing or items screen subsystem.
- The final step is to develop the payment gateway so that the payment record is created and stored.
- At the end, the process of testing would take place to ensure quick deployment of the completed subsystem

System Design

Database Design:

The application would utilize Firebase Database to store all the data. The data would be stored as

JSON objects as a tree. For example:

```
{
        "users": {
                "1002012345": {
                        "firstName": "Adam",
                        "lastName": "Smith",
                        "mavId": 1002012345,
                        "phoneNo": 6821234321,
                        "dob": 929173819738,
                        "phoneExt": "+1"
                },
                "1002012346": {
                                …
                }
        }
}
```

User Interface Aspects:

Inputs:

The system enables both buyers and sellers to interact with it in multiple ways. It allows users to upload media files, including product images, and record geographical data in terms
of latitude and longitude. Additionally, buyers can fill out forms to input product specifications.

Outputs:

Users can search for specific products based on selected filters and search criteria. The application also sends push notifications to both buyers and sellers when a message is received. Buyers have the ability to track their product listings.

User Interface Design:

Login/ Sign-Up Screen:
The login or Sign-up window will enable authentication for the application's user and also allow new users to onboard the application.
Sign-up would require basic details such as First Name, Last Name, UTA Email, UTAID, Date of Birth, and phone number.
Once the user has signed up, they can always log in with their registered details.

Dashboard Screen:
The initial dashboard view for both buyers and sellers will show a list of products available for sale in the local vicinity. Additionally, there will be a floating action button that enables users to access the product selling page.

Profile Screen:
The profile screen would contain all the basic actions such as view profile, edit profile, logout, and settings.

Product Feed Screen:
The purpose of this screen is to enable users to generate a product listing. It begins by allowing the user to upload an image of the product, followed by input fields for product details such as name, listing price, specifications, and pick-up location.

Chats Screen:
On this screen, the user can view a list of all active ongoing communications with other users. Each chat item includes information about the last message time and the number of unread messages. Clicking on an item will lead to the direct chat screen.

Notifications Screen:
This screen would display any push notifications and provide an option for the user to mark all notifications as read.

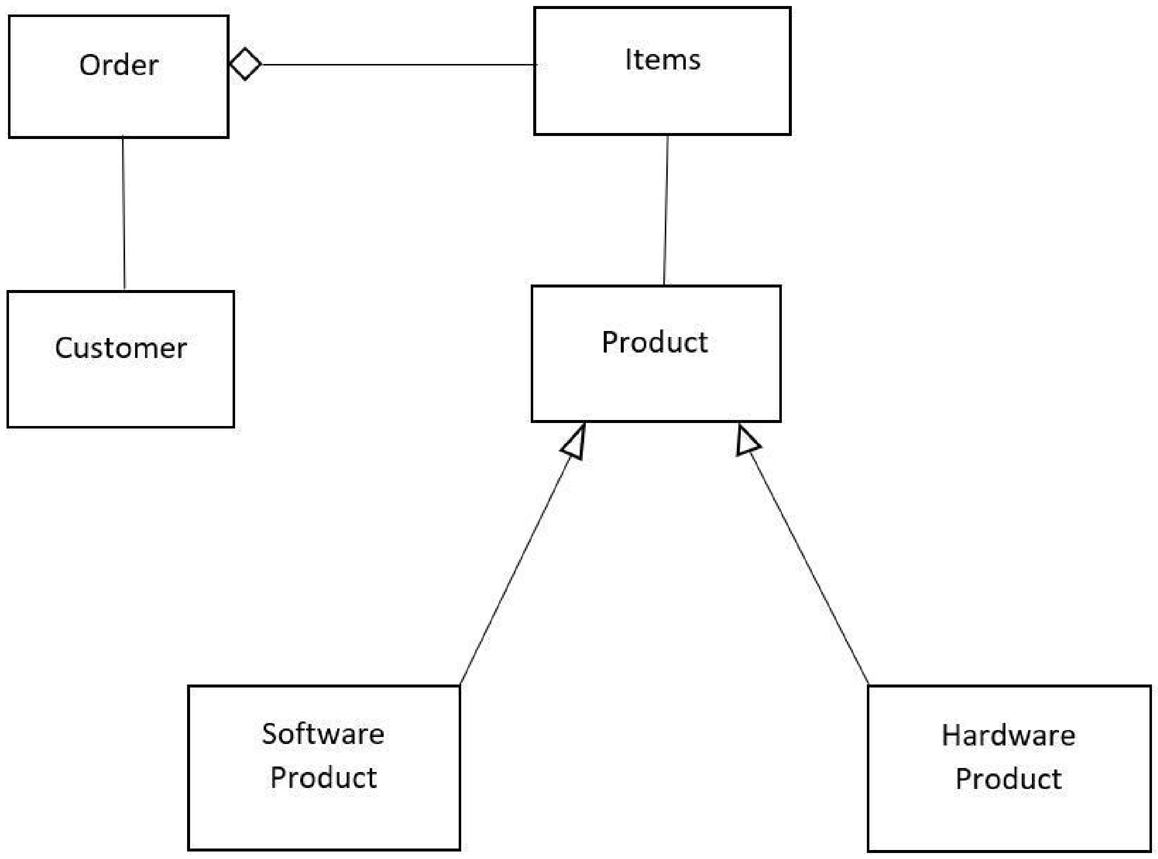
Subsystem Design

In software project management, a subsystem design refers to the process of breaking down a large software system into smaller, more manageable components, or subsystems.

A subsystem is a part of the larger software system that performs a specific function or set of functions. Each subsystem typically has its own set of inputs, outputs, and interfaces with other subsystems.

Subsystem design involves analyzing the requirements of the larger system and identifying the necessary subsystems. The subsystems are then designed, developed, and tested independently, and later integrated into the larger system. The purpose of subsystem design is to improve the modularity, maintainability, and scalability of the software system. It also allows multiple teams to work on different subsystems concurrently, which can speed up the overall development process.

In this project, the software can be divided into the following mentioned subsystems. Once identified, these subsystems will be designed, developed and tested independently.



Requirements

In software project management, requirements refer to the features, functions, and capabilities that a software system must have in order to meet the needs of its users or stakeholders. Requirements can be classified into various types -

Functional requirements:

- **FR1:** Flexible payment gateways like credit/debit cards, UPI payments with financial technological companies like Paypal, Zelle, Venmo etc.
- **FR2:** Should work on different devices like mobiles, tablets, PCs because most users shall try to use it from the portable devices.
- **FR3:** We can also monitor the number of users that interact from different devices to get a rough idea about which device is the most popular among users so that we can work on that more.
- **FR4:** Product features that should be included along with the functioning and uses should be filterable at the start of the site (home page). We can filter out the products based on what they are used for.
- **FR5:** Before buying anything the user should have a login id associated with their account that will be used to track the orders and if they don't have one we should have the feature to make an id.
- **FR6:** We should have a smooth checkout and order tracking flow so that it's easier to buy the things the user wants without any hassle.

Non-Functional requirements:

- **NFR1:** It should include all the social media handles of our e-commerce website so that the users can check out important updates and announcements that might interest them.
- **NFR2:** It should protect user data like their mobile number, address, payment details and personal information.
- **NFR3:** We should try to load up the website quickly even with all the traffic and site congestion. We can also use a benchmark to track the speed of loading up all the functionalities and integrations of the website.

- **NFR4:** The website should be scalable after the first iteration. The performance shouldn’t be affected as we increase the number of functionalities, integrations, features, memory, servers and computations.

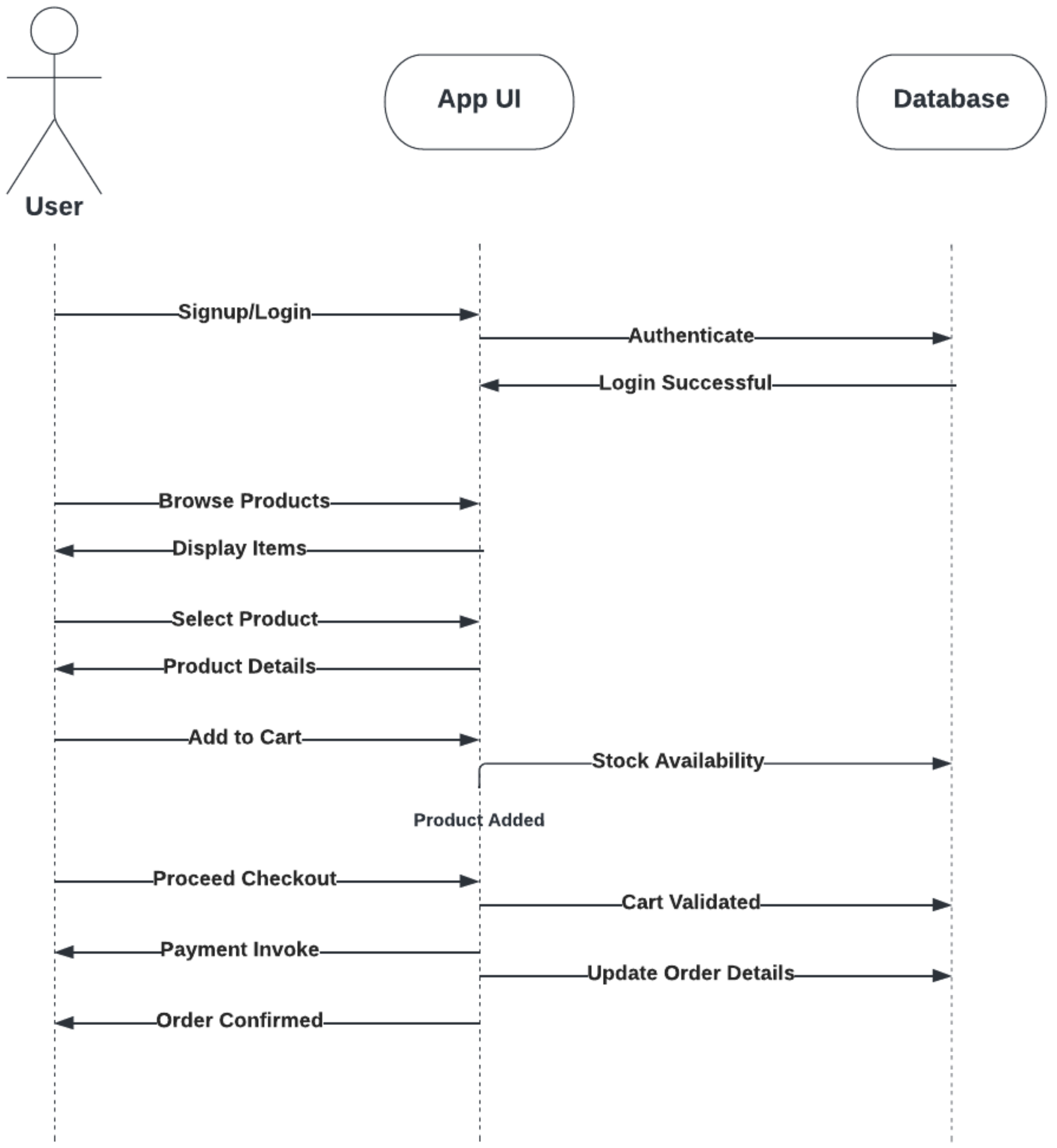
Interface requirements:

- **IR1:** The website should be easy to use for any user whether they have any knowledge about using a website or not (technical knowledge).
- **IR2:** We can also put up a survey after they’ve been at the website for a while for their feedback on the usability, how easy it was to find all the things they were looking for, did they complete what they initially came to the website for, where to go when you don’t know what are you looking for.
- **IR3:** The seller should ideally select at least three images for the product he intends to sell to show its physical condition.
- **IR4:** We can additionally ask the seller for a demo video of the product whether it works, is there any physical or internal damage, how good it’s condition is.

Sequence Diagram

A sequence diagram is a type of UML (Unified Modeling Language) diagram that illustrates how objects in a system interact with each other over time. It is a behavioral diagram that shows the flow of messages or method calls between objects in a specific scenario or use case.

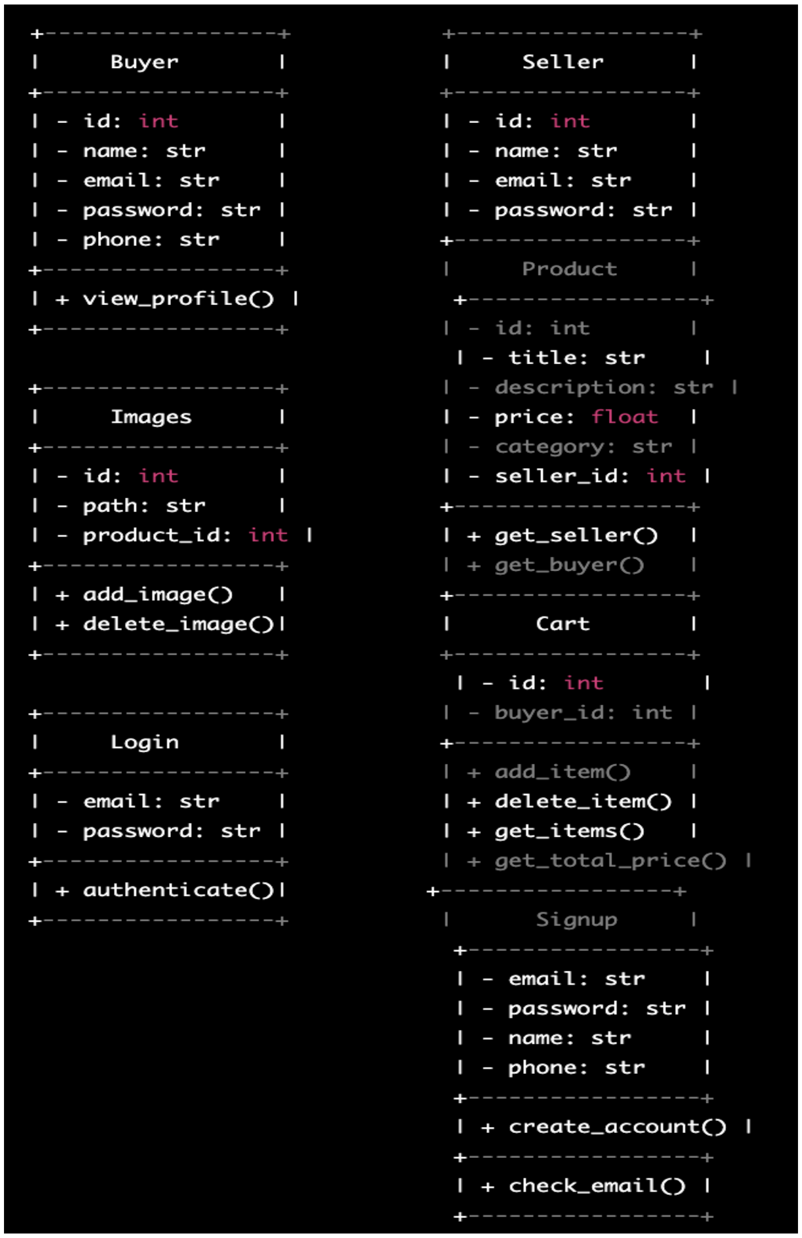
For this project, the sequence diagram covers the interaction between different entities involved in the software. The detailed steps are mentioned in the below diagram -



Design Class Diagram

A class diagram is a type of diagram in software project management that depicts the structure of a system by showing the system's classes, their attributes, methods, and the relationships between them. It is a visual representation of the code and helps developers and stakeholders understand the design and architecture of the system. Class diagrams are a key part of object-oriented design and are often used in the early stages of software development to plan and communicate the system's structure and behavior.

In this project, the class diagram and the software elements are designed as following -

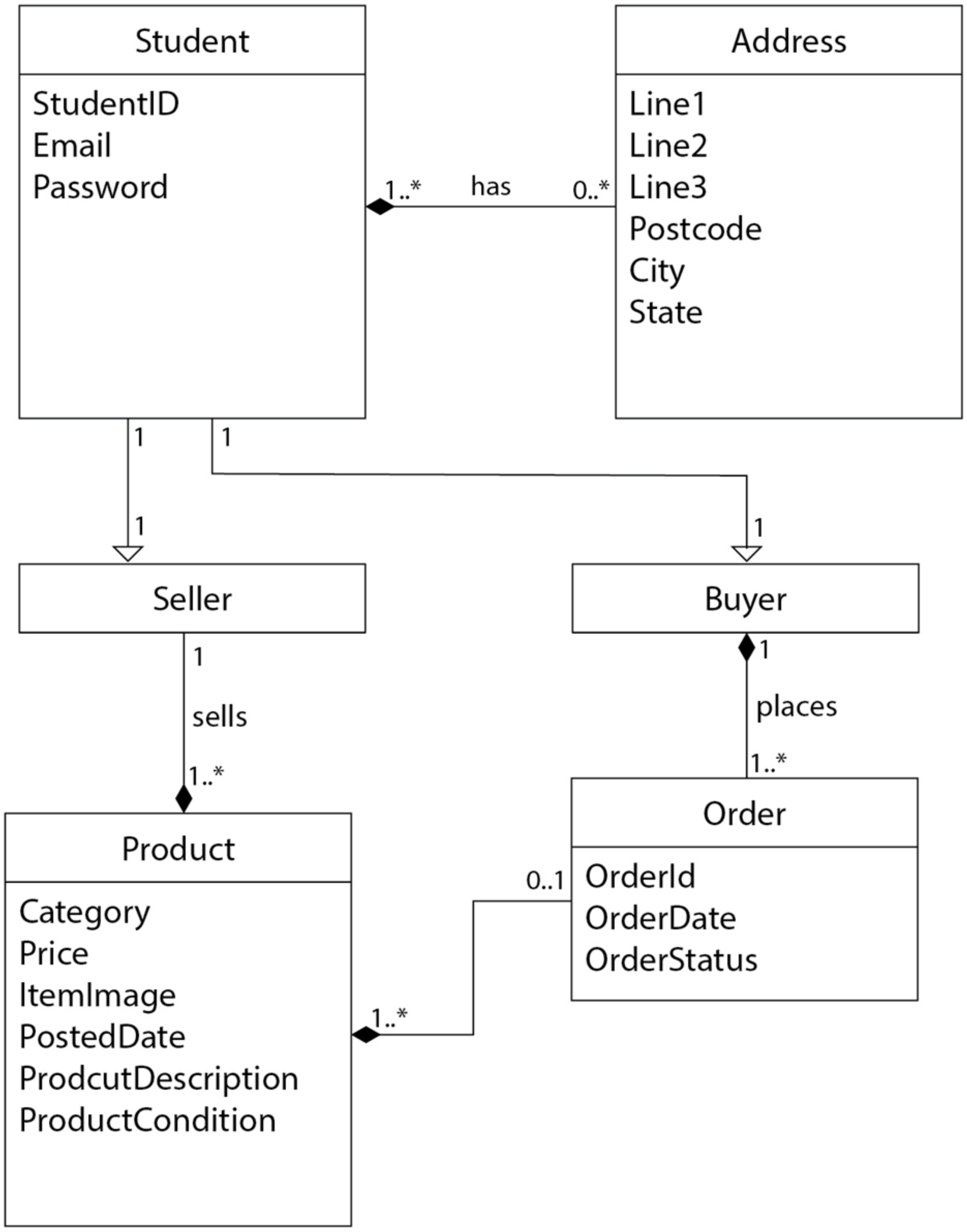


```
+------------------+
|      Buyer       |
+------------------+
| - id: int        |
| - name: str      |
| - email: str     |
| - password: str  |
| - phone: str     |
+------------------+
| + view_profile() |
+------------------+


+------------------+
|      Images      |
+------------------+
| - id: int        |
| - path: str      |
| - product_id: int |
+------------------+
| + add_image()    |
| + delete_image()|
+------------------+


+------------------+
|      Login       |
+------------------+
| - email: str     |
| - password: str  |
+------------------+
| + authenticate()|
+------------------+
```

```
+------------------+
|      Seller      |
+------------------+
| - id: int        |
| - name: str      |
| - email: str     |
| - password: str  |
+------------------+
|      Product     |
 +------------------+
| - id: int        |
 | - title: str     |
| - description: str |
| - price: float   |
| - category: str  |
| - seller_id: int |
+------------------+
| + get_seller()   |
| + get_buyer()    |
+------------------+
|       Cart       |
+------------------+
 | - id: int         |
| - buyer_id: int  |
+------------------+
| + add_item()     |
| + delete_item()  |
| + get_items()    |
| + get_total_price() |
+------------------+
|       Signup     |
 +------------------+
 | - email: str     |
 | - password: str  |
 | - name: str      |
 | - phone: str     |
 +------------------+
 | + create_account() |
 +------------------+
 | + check_email()  |
 +------------------+
```

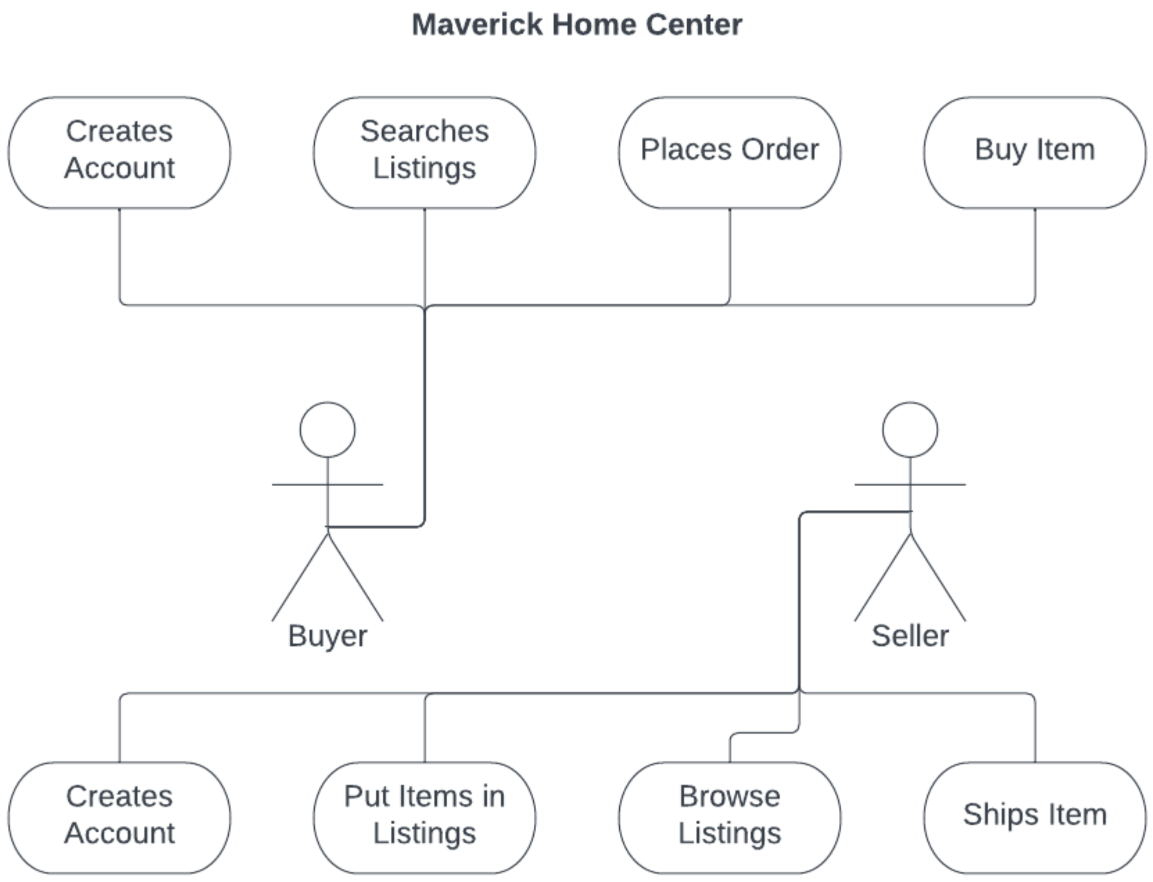
Domain Model

A domain model contains conceptual classes, associations between conceptual classes, and attributes of a conceptual class.
For this project, the associations and flow between various entities is specified in the domain model diagram below -



Actor System Interaction Model

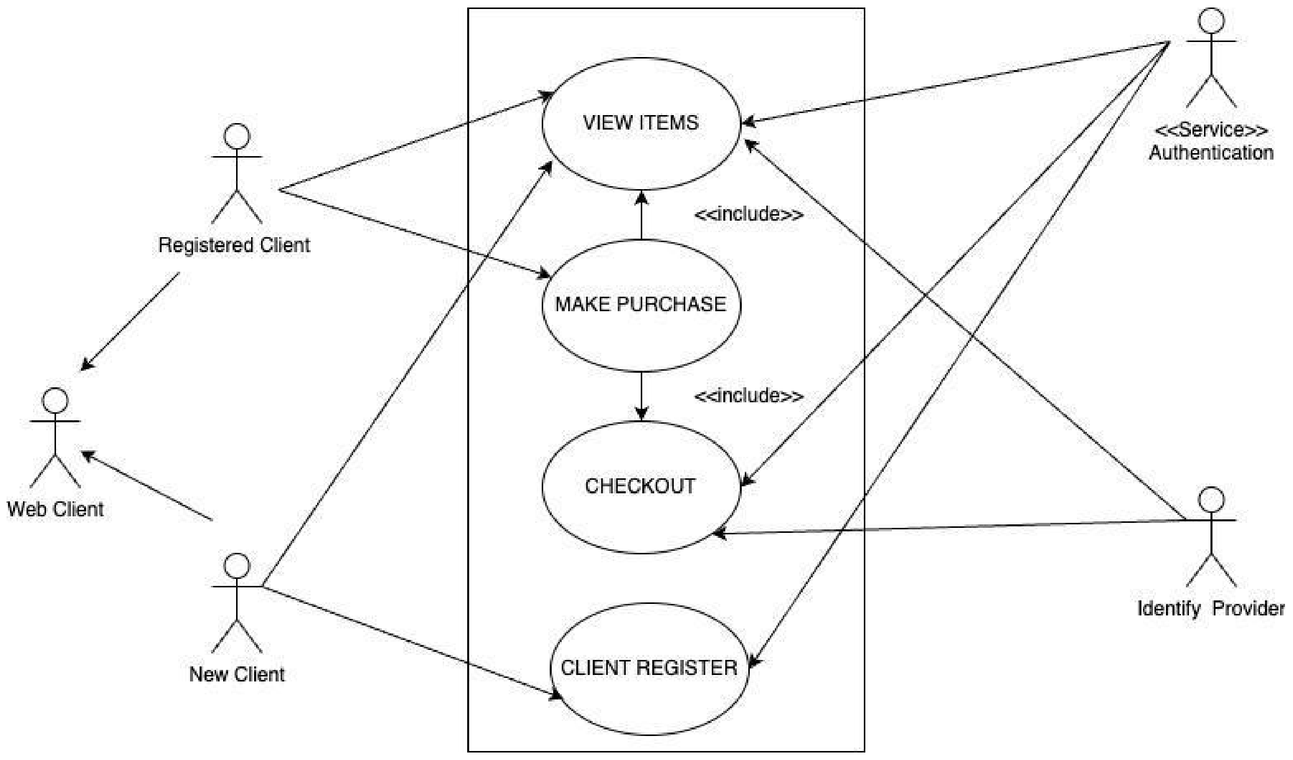
An actor system interaction model is a representation of the interactions between actors and the system under design. It is used to capture the different ways in which actors can interact with the system and the behavior of the system in response to these interactions.
This model helps to identify the functional requirements of the system, refine the system's behavior, and ensure that the system meets the needs of its users. It is often created during the requirements analysis and design phases of the software development life cycle.
For this project, the actor system interaction can be designed as following -



High level Use Cases

High-level use cases typically consist of a brief description of the main interactions between the actors and the system, along with a list of the main functions or features provided by the system. They are often used as a starting point for more detailed use case development and help to ensure that the system meets the needs of its users. High-level use cases are also useful for communicating the system's functionality to stakeholders and for identifying areas of the system that require further analysis and development.

For this project of maverick home center, the high level use cases can be defined as -



- **UC1: View Items**
- **UC2: Make Purchase**
- **UC3: Check Out**
- **UC4: Register**

Expanded Use Cases

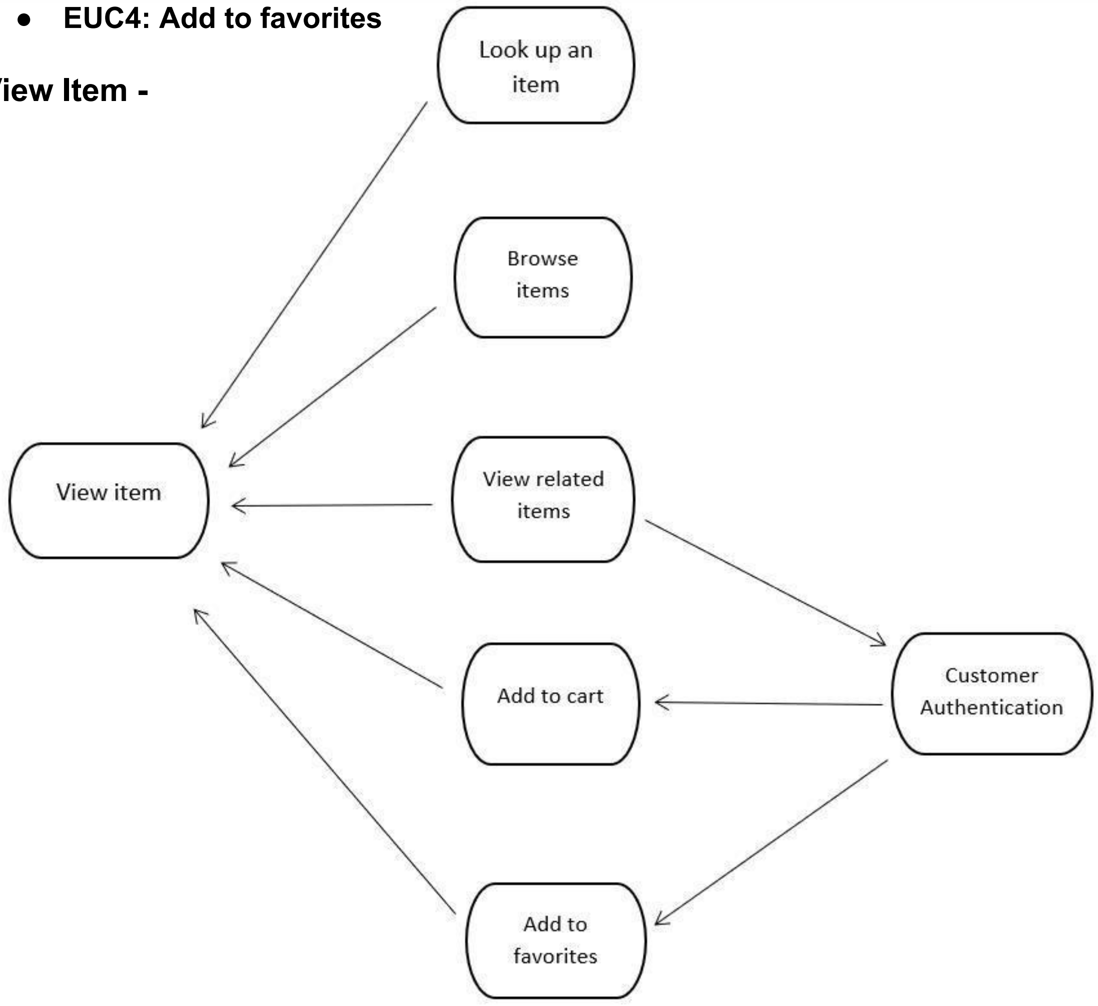
Expanded use cases typically consist of a series of steps or actions that the actor performs, along with the system's responses and any alternate paths that may occur. They may also include preconditions, postconditions, and other details that help to refine the use case and ensure that it is complete and accurate.

Expanded use cases are often created after high-level use cases and are used to further refine the system's functionality and behavior. They are also useful for identifying areas of the system that may require additional development or testing. Expanded use cases can be used to develop test cases, design the system's user interface, and to communicate the system's functionality to stakeholders.

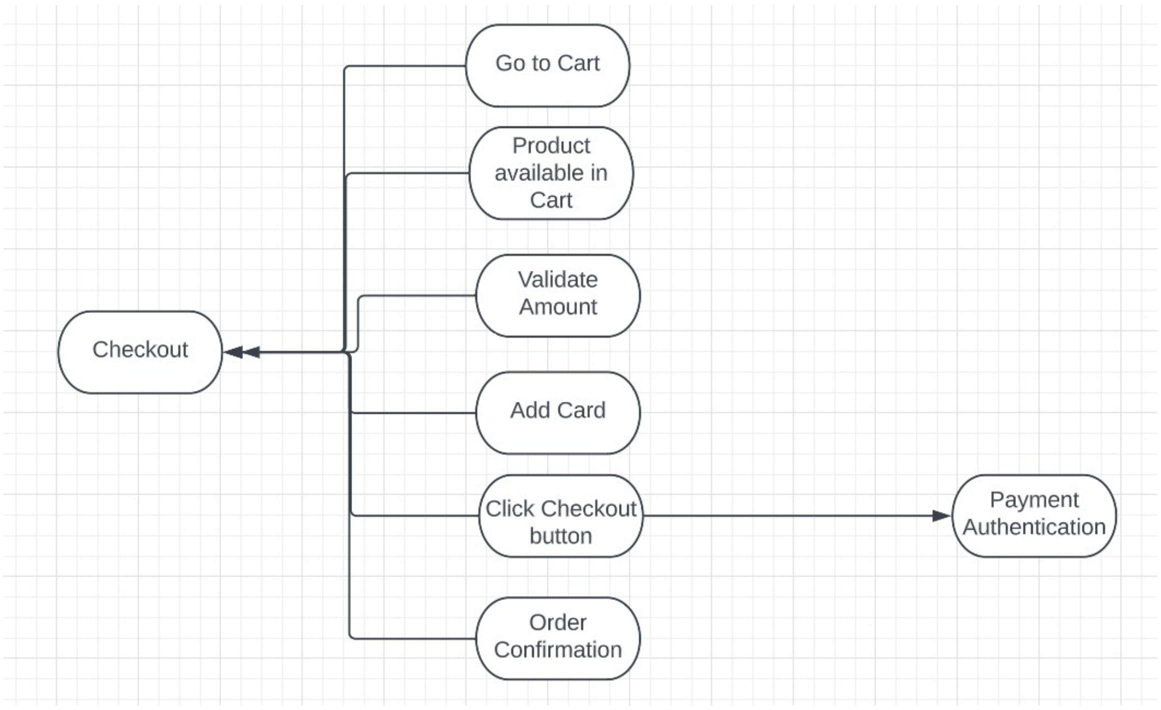
For this project, the expanded use cases cover the detailed aspects for each use case. They are as mentioned below -

- **EUC1: Browse items**
- **EUC2: View related items**
- **EUC3: Add to cart**
- **EUC4: Add to favorites**

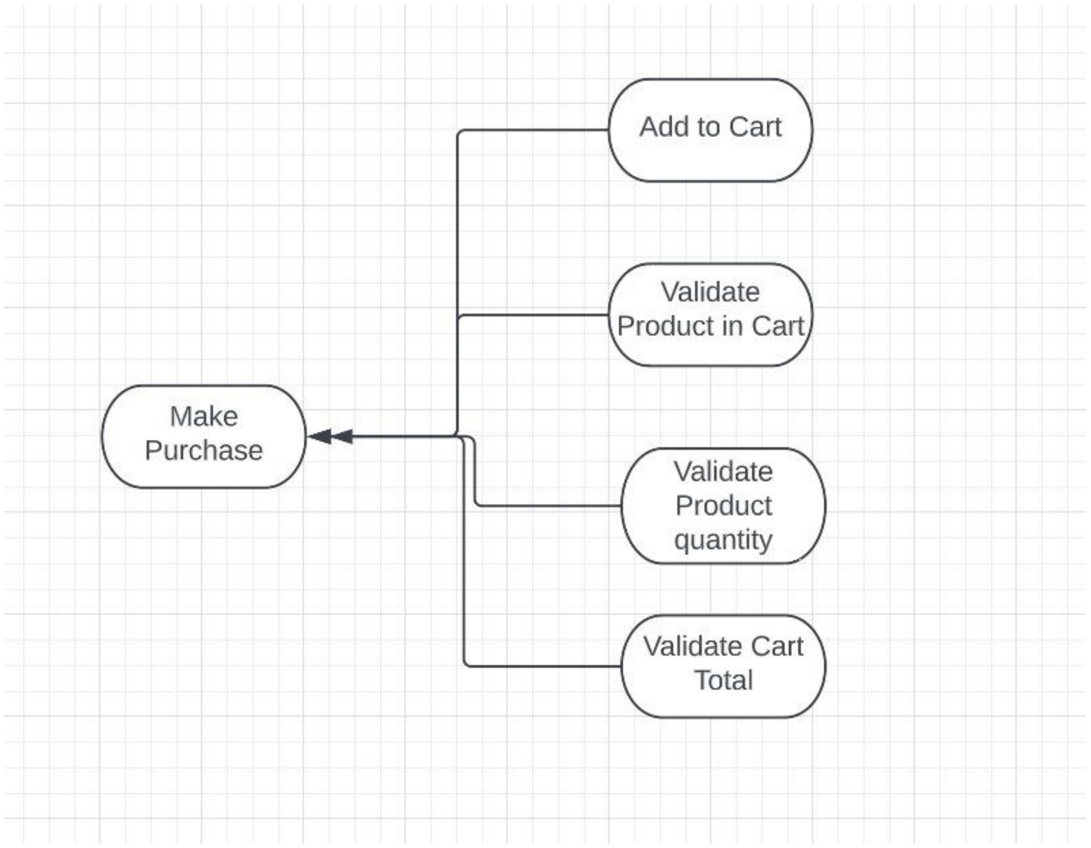
View Item -



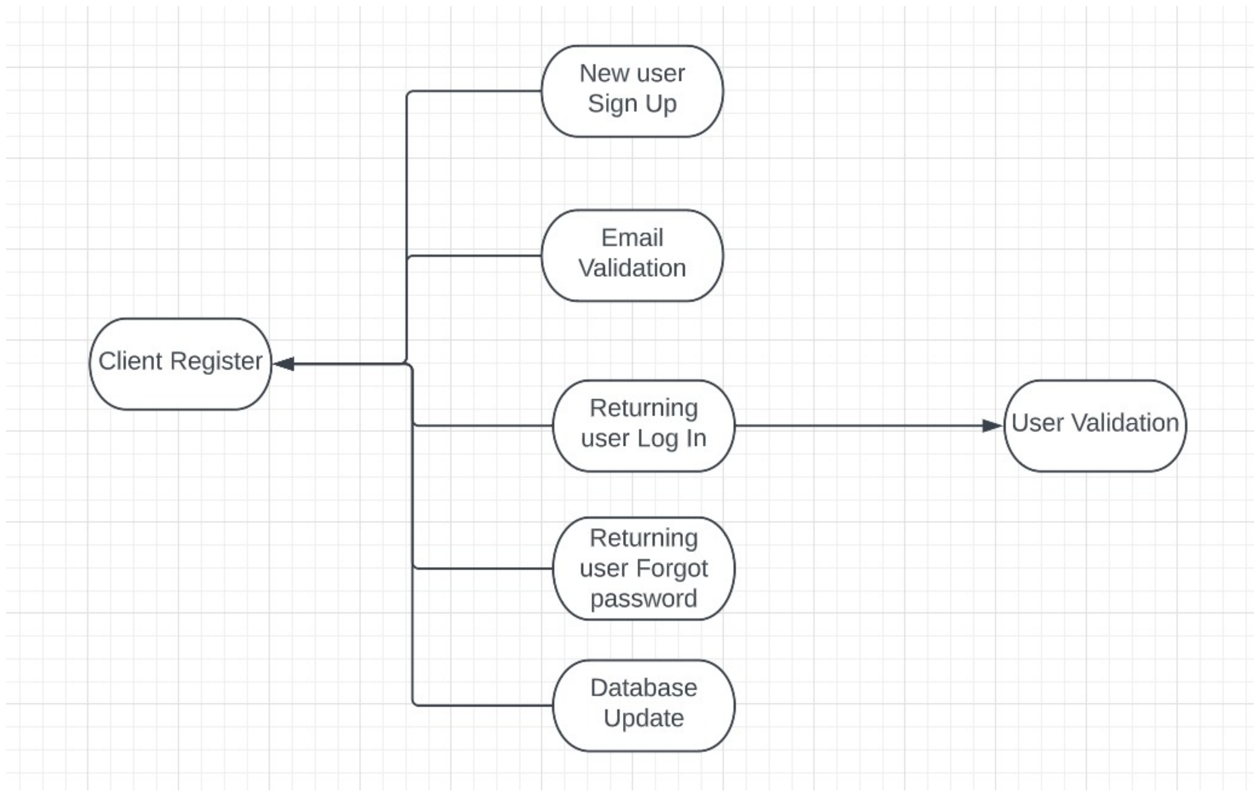
Checkout -



Make Purchase -



Register -


New user Sign Up
Email Validation
Client Register
Returning user Log In
User Validation
Returning user Forgot password
Database Update

Requirements Use Case Traceability Matrix

A Requirements Use Case Traceability Matrix is a tool used in software project management to ensure that all requirements have a corresponding use case, and all use cases have been properly traced to their corresponding requirements. It is a matrix that shows the relationship between the requirements and use cases, and helps to ensure that the system meets the needs of its users by tracing each requirement to the use cases that satisfy it. The matrix can also be used to track changes to the requirements and use cases throughout the software development life cycle.

The matrix also contains the priority level of each requirement, which helps in planning the project properly with a defined timeline. Following is the RUCTM for this project -

Requirement No.	Priority Weight	UC1	UC2	UC3	UC4
FR1	5		x	x	
FR2	5	x			
FR3	5		x		x
FR4	5				
FR5	4				
FR6	4	x			
NFR1	4				
NFR2	3				
NFR3	3			x	x
NFR4	3				
IR1	2				
IR2	2				
IR3	1				
IR4	1				
UC Priority		9	10	8	8

Note: Priority weight: 1 denotes low, 5 denotes highest priority

Increment Matrix

An increment plan is a tool used in Agile software development to outline the work that will be completed in a particular iteration or increment of development. It typically includes a list of user stories or requirements that will be implemented in the upcoming iteration or increment, along with estimates of the time and resources required to complete each task. The iteration plan or increment plan is used to help the development team plan and execute their work, as well as to communicate progress and status to stakeholders.

The iteration plan or increment plan is often updated and revised throughout the software development process as new information is gathered and requirements change. It is a key component of Agile project management and helps to ensure that the software is developed in an iterative and incremental manner, with frequent feedback and adaptation to changing requirements.

The below mentioned increment matrix diagram shows the priority and effort estimated based on the team strength for multiple iterations with respect to this project.

Use Case	Priority	Effort (Person-week)	Depends On	ITR 1 3wks	ITR 2 3wks	ITR 3 3wks	ITR 4 3wks
UC1	9	3	None		12		
UC2	10	3	None			12	
UC3	8	3	None				12
UC4	8	3	None	12			
Total Effort		12		12	12	12	12

Team Size = 4 People

Effort Estimation

First of all we need to know what factors count towards the effort estimation of a software engineering life cycle of an application.

Effort estimation refers to the process of estimating, planning, and monitoring the time and resources required to complete a project. In the context of mobile app development, effort management is critical to ensure that the project is delivered on time, within budget, and with the desired quality.

In order to estimate the cost of the application we are developing, we need to consider several factors that affect the overall performance of the app and the software as a whole.
These can be classified into the following categories:

1. **Platform**: The platform that we have chosen to develop this application if Flutter as it is easier to develop a cross platform application using the same otherwise we would have to build two separate native apps for iOS and Android, which would increase the development time and cost of the overall project and hence would require more efforts to manage, maintain and function.

2. **Features**: The features that we have added are similar to the functional, non-functional and user interface requirements that we have mentioned above. Account creation, browsing and filtering items according to their types (furniture, decor, electronics, etc), payment gateways, data safety, Buyer can search for all the items that are up for sale/ for giveaway, seller can choose whether they want to deliver the items or ask the buyer to pick them up from a certain place. Since we have a number of features to make the user interface and the overall application user friendly, we would require significant development from our team in order to meet these feature requirements.

3. **Overall Design**: The user interface design and overall "look" of the application would also be an essential factor in determining the efforts that it would need in development. This is done keeping in mind that the application home page and the overall flow should be user friendly, that is, it should be easier to use even for people with little to no technical knowledge of the field. If we are delegating the design to a team member, we do that while being mindful of the efforts that it would require, so it would be done using a well defined series of tasks that a

typical user will go through. This would help us target the right areas where the user might get stuck or want to start a new purchase.

4. **Integration**: First and foremost step would be to integrate our app with third-party services like payment gateways (Paypal, Venmo, Zelle, etc) , shipping providers, and inventory management systems. The effort required to integrate these services can vary depending on the complexity of the integration and the availability of APIs. Since we are using the flutter libraries and APIs, this should be easier than your typical android or iOS development integration process.

5. **APIs**: We also plan to integrate with third-party APIs (such as payment gateways or shipping providers), so in order to do that we have to consider the effort required to integrate and test these APIs. This can sometimes be a complex and time-consuming process.

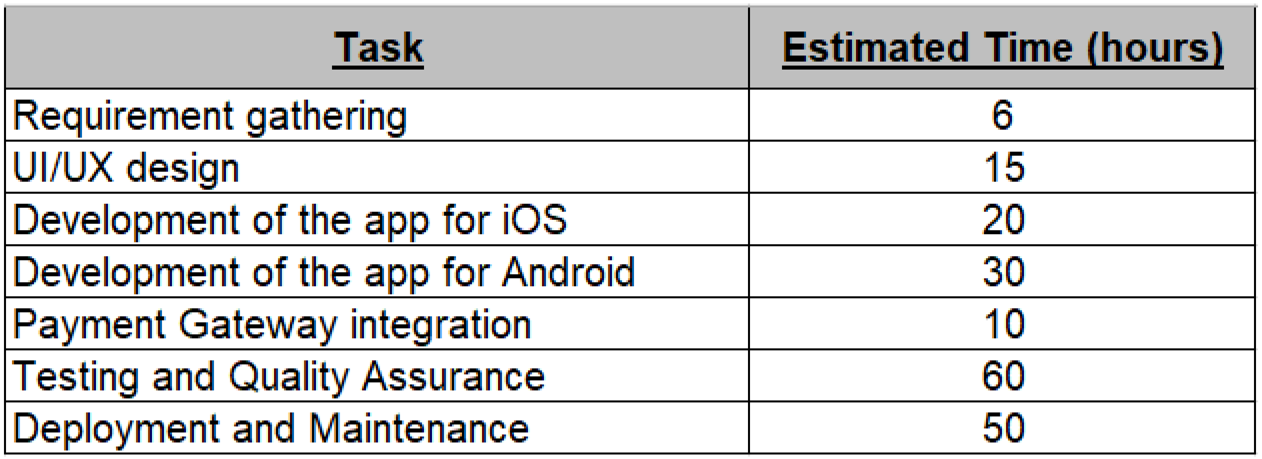
6. **Content management**: We also need to consider how we will manage the content that appears in the app, such as item descriptions and images. This may involve developing a content management system (CMS) or integrating with an existing CMS. This could take a while because we plan on adding additional details and images for every item that we will add to the system including but not limited to the original packaging (if available), item description from where it was originally bought (example: Amazon, eBay, etc) and the most recent pictures of the item

7. **Localization**: We plan to release the app in a singular region (Arlington) since it is for the students and you need a valid UTA ID to login to the application so not much effort would be required to localize the application as it would already be in the ideal state for our purpose. Additional and possibly a bonus feature that we could add in this would be to offer the item descriptions in a language other than English (example: Spanish, Hindi, Arabic) since we have a lot of diversity here in UTA.

8. **Scalability**: If the app becomes popular and attracts a large user base, we'll need to consider the effort required to scale the app to handle the increased traffic and demand. This may involve upgrading the server infrastructure or implementing additional features to support scalability.

9. **Analytics**: We also might integrate analytics tools into the app to track user behavior, measure engagement, and identify areas for improvement. This will require some additional effort to set up and configure the analytics tools.

Using the Work Breakdown Structure (WBS) technique, we have estimated the time required for each task, as shown in the table below:



Task	Estimated Time (hours)
Requirement gathering	6
UI/UX design	15
Development of the app for iOS	20
Development of the app for Android	30
Payment Gateway integration	10
Testing and Quality Assurance	60
Deployment and Maintenance	50

Based on these factors, the effort required to build an app that sells refurbished items to local students could range from a **few weeks** to **several months**. It's important to have a clear project plan and timeline to help manage the effort estimation process. Additionally, we have to consider ongoing maintenance and support costs for the app once it's launched.

People Management

There are a few things to consider if you're thinking about including people management tools in an app that sells reconditioned goods to nearby students. Here are a few things to think about:

- **Purpose:** You must specify the function of the people management features you intend to provide. Do you want users to be able to buy and sell products from other users, for instance? Will you be providing a peer-to-peer rental service? You may decide what functionality is needed by first understanding the role of the feature.
- **User Profiles:** It must be made for each user in order to enable people management functionalities. Users will be able to do this to create and manage their own listings while also viewing those of other users. You must decide what details, such as name, contact information, and a profile photo, to include in the user profiles.
- **Security:** Security is essential for any app that processes financial transactions. In order to make sure that users can only access their own accounts and listings, you must put in place secure authentication and authorization systems.
- **Communication:** You must make it possible for users to communicate with one another through the app. This can include a messaging service or chat function that enables buyers and sellers to discuss prices or schedule item pickup and delivery.
- **Ratings and reviews:** Ratings and reviews: You might want to think about establishing a ratings and review system to help users develop trust. Customers will be able to rate merchants and give feedback in this way, which will enable other users to make better purchasing selections.

An app that sells refurbished items to local students can help to create a user experience that is more dynamic and engaging by adding features for managing people. However, it is essential to take into account the additional effort required to implement these features, which include the creation of user profiles, security, communication, ratings, and reviews.

Quality Management

Quality Management refers to the process of ensuring that the project meets the desired quality standards. In our mobile app development project quality management is very important or rather critical to make sure that the application built is reliable, usable and meets the customer expectations.

Quality Management Steps :

1. **Set Quality Parameters** - The very first step in Quality Management of any project is to clearly set the quality parameters. The quality parameters should be based on customer requirements, industry best practices and regulatory standards. We are using various techniques such as user surveys, customer feedback and competitive analysis to identify the quality parameters.

2. **Develop Test Plan** - Once the quality parameters are set we will be developing an extensive test plan. The test plan will include various types of test such as Unit Testing or whitebox testing, Functional Testing or Spring Testing, Usability Testing, Compatibility Testing, Performance Testing, Security Testing and User Acceptance Testing. The test plan will specify the test criteria, test data and test environment.

3. **Execute Test** - Once the Test Plan is developed the next step will be to execute the tests as per the plan. We will be using various testing tools such as automated testing tools, manual testing tools, load testing tools and performance testing tools to run the tests. The tests will be recorded and any issue or bug found will be tracked and fixed.

4. **Monitor and Improve** - The final step will be to monitor the quality of the app and continuously improve it. We will be vigilant on the app's functionality, usability and user reviews. The quality of the app will be tracked using a variety of methods including user surveys, analytics and feedback tools. Any issues encountered will be fixed right away and the software should be updated regularly depending on user comments.

Best Practices -
Quality management is a vital component of developing mobile apps. Following are some best practices that we are going to use to ensure the app's quality:

1. Use a Test-Driven Development (TDD) strategy: TDD entails developing tests prior to developing code. This strategy will aid in ensuring that the app satisfies the requirements for quality.

2. Use Automated Testing: Using automated testing will help to increase testing's efficacy and efficiency. To conduct tests efficiently and accurately, automated testing technologies will be employed.

3. Performing Usability Testing: Conducting Usability Testing will help us to understand the app's ease of use, learnability and user satisfaction. It will also ensure that the app satisfies the user's needs and expectations.

Testing and Quality Assurance Activities:

The following testing and quality assurance activities will be conducted:

1. Functional Testing - Functional testing will be conducted to ensure that the app functions as intended. The following tests will be performed:

- User registration
- User login
- Payment processing

2. Usability Testing - Usability testing will be conducted to evaluate the user experience of the app. The following tests will be performed:

- Navigation testing
- Design and layout testing
- User feedback and suggestions

3. Compatibility Testing - Compatibility testing will be conducted to ensure that the app works on different devices and operating systems. The following tests will be performed:

- Device compatibility testing
- OS compatibility testing

4. Performance Testing - Performance testing will be conducted to ensure that the app performs well under different usage scenarios. The following tests will be performed:

- Load testing
- Stress testing
- Performance benchmarking

5. Security Testing - Security testing will be conducted to ensure that the app is secure from potential threats. The following tests will be performed:

- Authentication and authorization testing
- Data protection testing
- Network security testing

6. User Acceptance Testing - User acceptance testing will be conducted to ensure that the app meets user requirements and expectations. The following tests will be performed:

- User testing with a group of representative users
- Feedback and suggestions from users

Test Cases

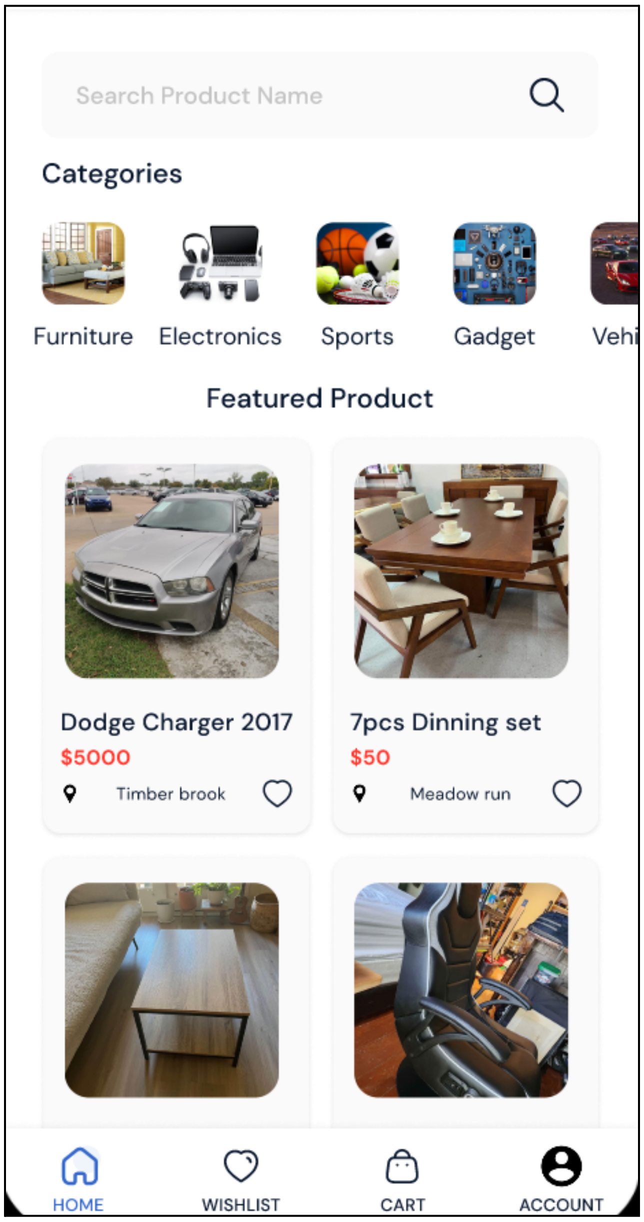
Project Name	*MAVERICK HOME CENTRE*						
Created by	Ravi Rajpurohit Akash Biswas Aditya Bhat Amit Gupta						
Creation Date							
Test cases-ID	Description	Test Steps	Pre-conditions	Test data	Post- condition	Expected Result	Test Pass/Fail
USER INTERFACE TESTING							
UI01	Verify the login screen has rendered correctly with all items appearing as expected on the User Interface (UI)	1.Enter valid username 2.Enter valid password 3.Click on login button	Need valid username and password for login and Valid URL	Username: Password:	You should able to see the Home Centre application’s homepage	Successful Login	Pass
UI02	Verify the login screen throws error if username or password do not match	1. Enter invalid username 2. Enter invalid password 3. Click on login button	Username and password should not have registered Or Username or password should not match	Username: Password:	You should see an error for logging in with incorrect credentials	Unsuccessful login	Pass

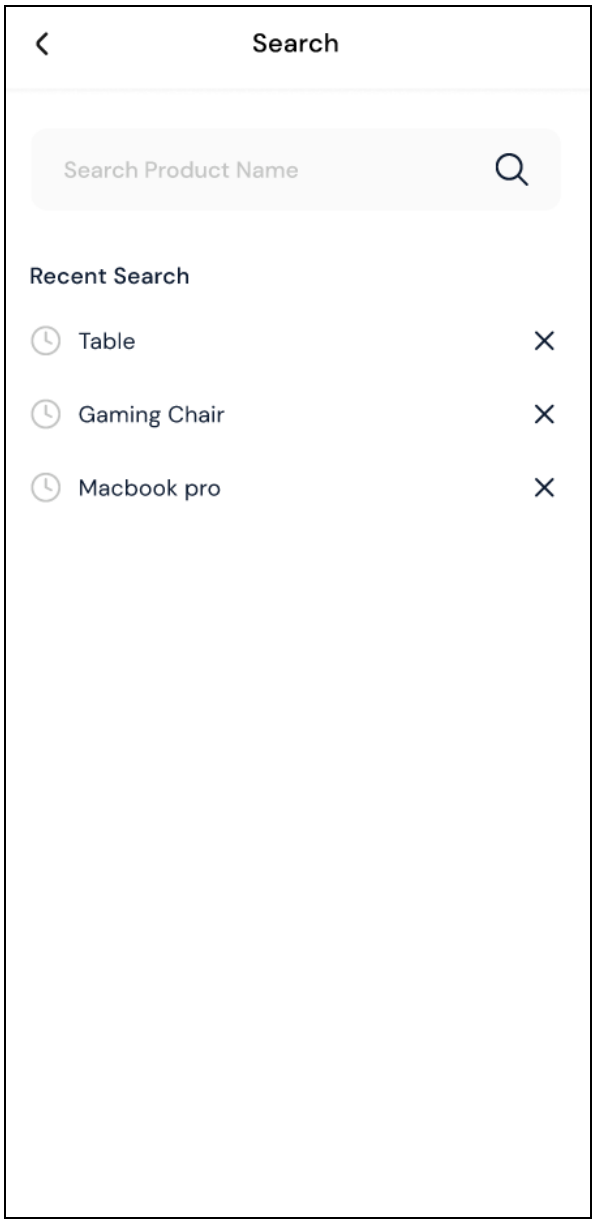
UI03	Verify if Account tab shows the user logged in	1. Go to the account tab 2. Check the user details	User should have logged in	Username: Password:	You should see the user details on the account tab	User can see the login details	Pass
UI04	Verify if the homepage renders the images correctly	1. Login to the applicatio n 2. Get connected to internet	User should have logged in	Products should have been uploaded for display	You should see the products and images successfully	User can see the images rendered well	Pass
UI05	Verify if the user can see recent searches below the search bar	1. Search for an item in the search bar	User should have searched a few things first to create a search history	Product searches	You should see the recent searches as text below the search bar	User can see the recent searches	Pass
FUNCTIONAL TESTING							
FT01	Verify if the user is able to enter their credentials	1. Try putting text in username and password fields	User should open the application	-	User should be able to type text as well as password	User can type text and password	Pass
FT02	Verify if the Login button works	1. Try logging in using any username and password	User should type in their credentials	-	The login button should be active	The login button works to login or throw error	Pass
FT03	Verify registration section is not displayed when the user enters valid	1. Signup using valid credential s	User should sign up	Username Password	The user should not see register if already signed up	The user sees login button	Pass

	credentials for an existing account	2. Login using credential s					
DATABASE TESTING							
DB01	Check the table is available in database	available in database	Users should have signed up	Username Password	Data entry should be happening in DB	Database updates user data	Pass
DB02	Check the product database is updated properly	The database is updated when user buys/sells products	Products should be there to load/render	Product information - Description and images	Database should keep track of products	Database updates product details after transaction	TBD

Features Implemented

- Home Page.
- Search Feature.





Conclusion

In conclusion, building an app that sells refurbished items to local students requires careful consideration of several key factors, including the purpose of the **app**, **user profiles**, **security**, **communication**, and **ratings** and **reviews**. Additionally, implementing quality management in the app is critical to ensure a high-quality user experience, including **testing**, **user feedback**, **continuous improvement**, **security**, and compliance with **legal and regulatory requirements**.

When estimating the effort required to build such an app, it's important to consider additional factors such as **API integration**, **content management**, **localization**, **scalability**, and **analytics**. Being thorough and detailed in the estimation process can help avoid underestimating the time and resources required to build a successful app that meets the needs of its users.

Overall, building an app that sells refurbished items to local students can be a complex and time-consuming process, but with careful planning and execution, it can provide an engaging and valuable service to students and the local community.