POLITECNICO
MILANO 1863

Academic Year 2020-2021

Computer Science and Engineering

Baratto

DD

Design Document

Version: 1.0

Sudarshan De

Stefano De Cillis

Table of Contents

Definitions, acronyms, Abbreviations

Definitions

- **User**: Normal people who could sign into the application with an intent of bartering and posting/selecting a commodity. Once the user is registered, it could be a different entity/actor (registered user or customer)
- **Bartering**: It's an activity of exchange (goods or services) for other goods or services without using money.
- **Imagga**: Image Recognition API provides solutions for image tagging & categorization, visual search, content moderation

Acronyms

- API: Application Programming Interface
- DD: Design Document
- UI: User Interface
- HTTP: Hypertext Transfer Protocol

Scope

Baratto allows users in the application to barter with other users. The application allows users to upload images of new commodities that they are planning to barter with others and select any goods or services that they are interested in for a successful trade. The mobile application gives the possibility to take a picture of the commodity (goods or services) and help users to write details in a fast way using an image recognition model. Apart from classification and categorization, it will also provide us with the details of moderation of the image which will give the user an idea of the social sensitivity level of the image, so that uploading of anti-social images can be avoided. Baratto gives the user the provision of contacting the owner of an item through the chat to arrange an exchange.

This document however will not specify any actual bartering activity or the software testing.

Document Structure

Chapter 1 is an introduction to the design document. Its goal is to explain the purpose of the document which focuses on the details of the functionality and the services being offered by the application.

Chapter 2 aims to describe the architecture design of the system, it is the core section of the document. More precisely, this section is divided into the following parts:

- Overview: High-level components and their interaction
- Deployment view
- Runtime view

Chapter 3 Specifies the user interface design. This part has UX diagrams to better describe the interaction between the customer and the application.

Chapter 4 Includes a description of the implementation plan, the integration plan and the testing plan, specifying how all these phases are thought to be executed.

Chapter 5 Describes the hardware required for Baratto application

Chapter 1

Introduction

Design Document will focus mainly on the technical details providing the application's functional importance as well as the architecture followed to achieve it. Here all the components forming part of the system are described, with the related runtime processes and all the design choices are listed and motivated.

In particular, the following topics are touched by the document:

- The high-level architecture
- The main components, their interfaces and deployment
- The runtime behaviour
- The design patterns
- Additional details about the user interface
- A mapping of the requirements on the architecture's components
- Implementation, integration and testing plan

Chapter 2

Architectural Design

2.1 Overview: High-level components and their interaction

The Baratto is a distributed application and the three logic software layers of Presentation (P), that manages the user/customers interactions with the system, Application (A), that handles the business logic of the application and its functionalities, and Data access (D), that manages the information with access to the database. This architecture is thought to guarantee the system characteristics of scalability and flexibility and to lighten the server side-splitting it into two nodes. In particular, the second tier is thought to contain only the business logic to physically separate clients and data to guarantee more safety in accessing data since the system deals with sensitive data. Also, the application exposes/exploits api call in this tier to get/post data to/from external entities like Imagga.

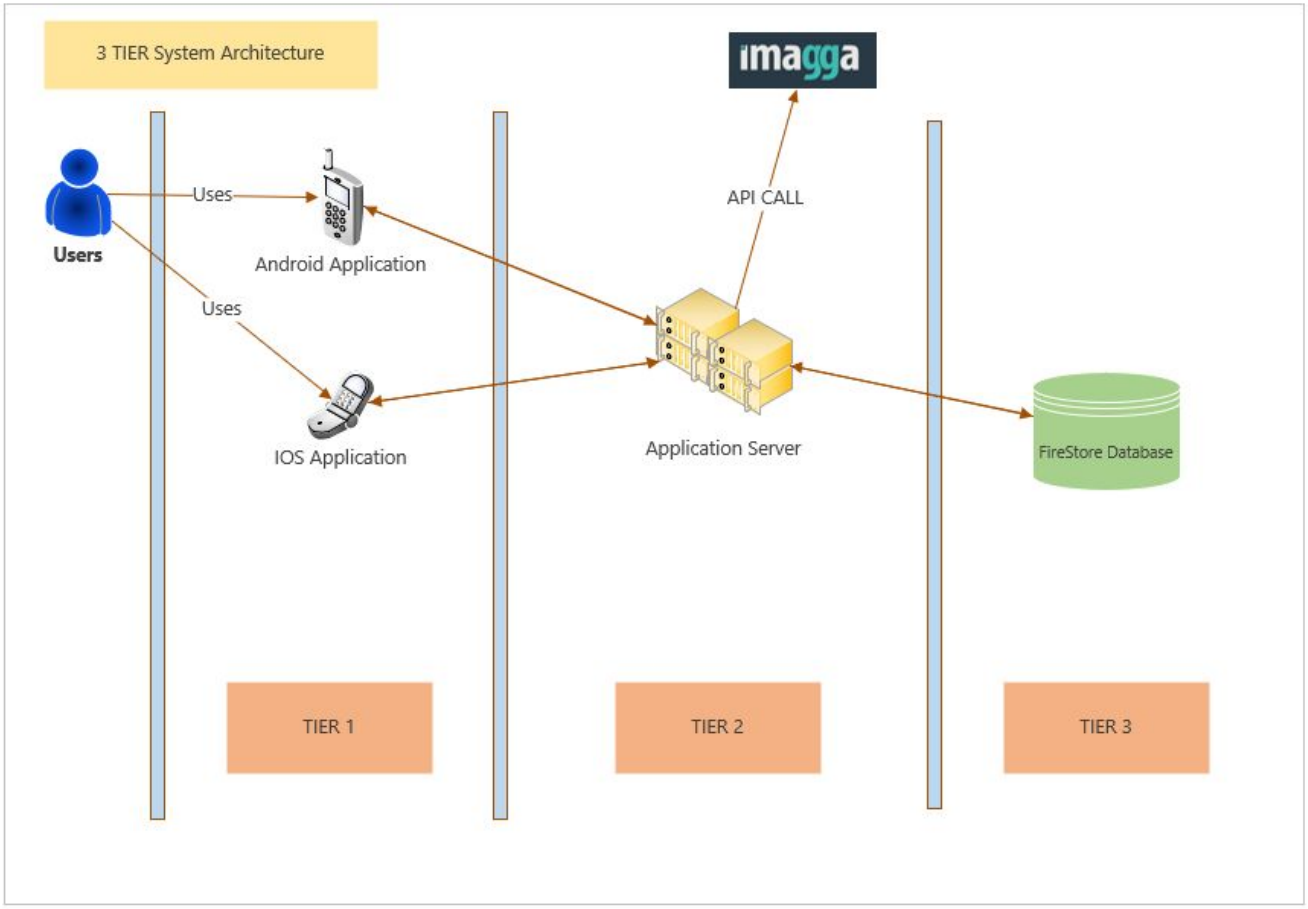


Figure 2.1: High--level system architecture

2.2 Deployment View

In the following image, the Baratto deployment diagram is represented: it shows the execution architecture of the system and represents the distribution (deployment) of software artefacts to deployment targets (nodes). Artefacts, in general, represent pieces of information that are used or are produced by a software development process and are deployed on nodes that can represent either hardware devices or software execution environments.

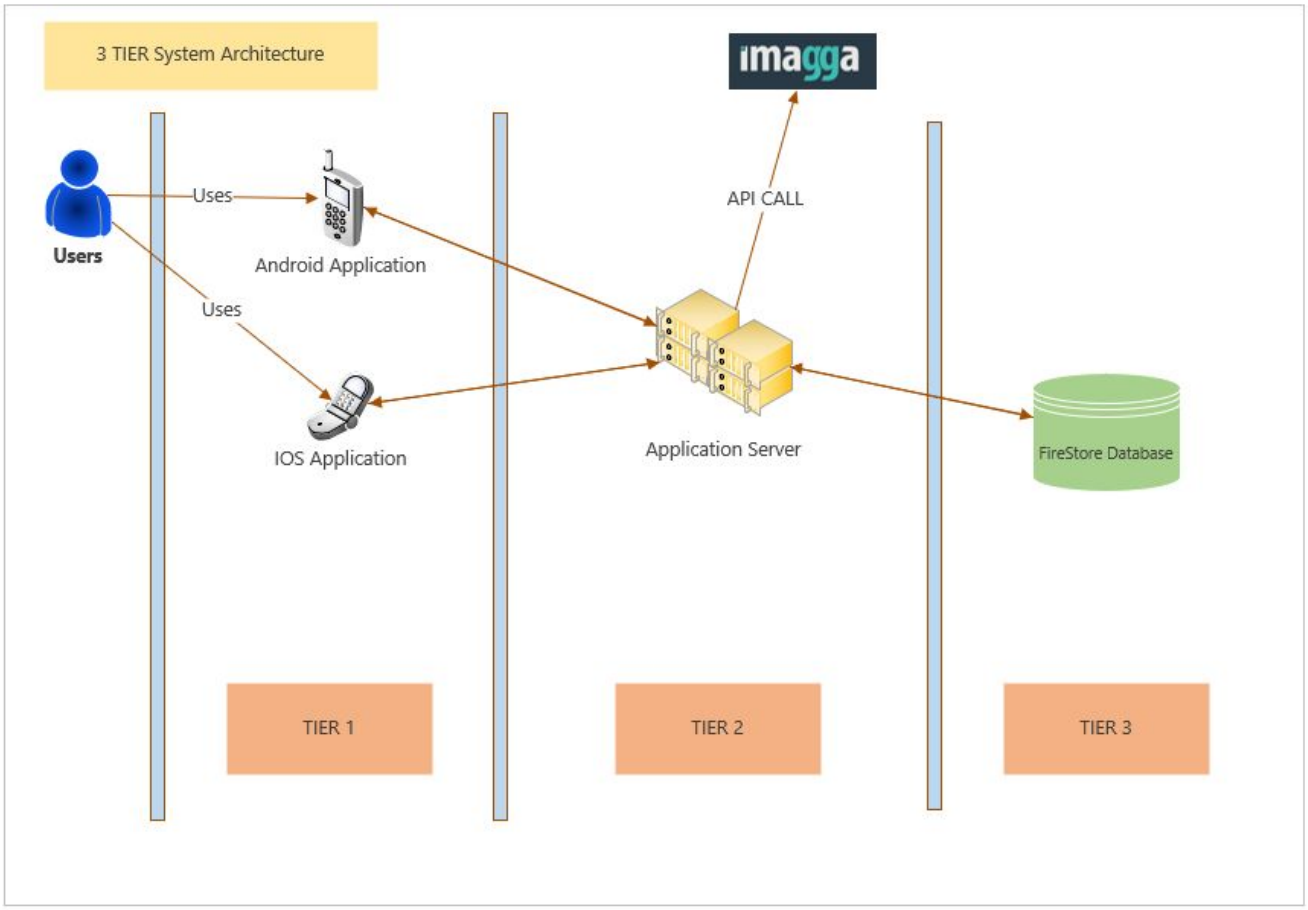


Figure 2.2: Deployment View for Baratto Application

In the diagram, external systems, as well as some other components (like load balancers and firewalls), are not represented to focus only on the components that host the core functionalities of the application or on the components for which the deployment is effectively executed. The three tiers respectively contain:

- **Tier 1:** Presentational logic is deployed in this layer. Users must be provided with a mobile application on their smartphone the mobile application is the most comfortable way for a user to have access to the services. The mobile application must be available for both Android and iOS to make it available on most of the devices.
- **Tier 2:** Application logic is deployed in this layer. The application server implements all the business logic and API call (Imagga API), handles all the requests and provide the appropriate answers for all the offered services. It is directly addressed by the mobile application.
- **Tier 3:** Data access is deployed in this layer. We will be accessing Cloud FireStore database in our application which is a NoSQL document database that lets you easily store, sync, and query data.

2.3 Runtime View

2.3.1 Users Uploading/Posting new commodity

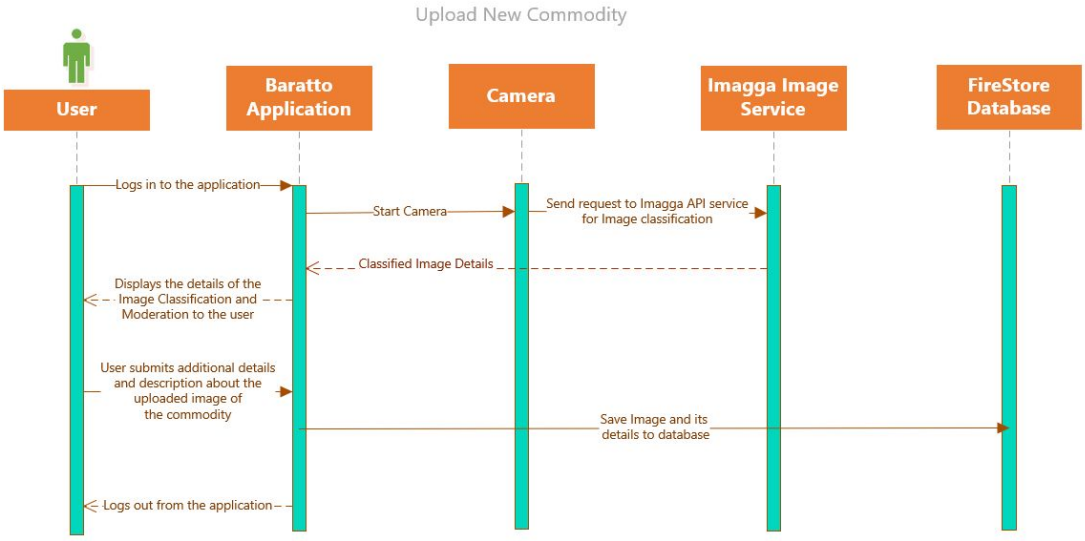


Figure 2.3: Upload/Post Commodity sequence diagram

The sequence diagram in 2.3 briefly describes the process through which a user can successfully Upload/Post new commodity which they are planning to trade with other registered users in the application. As clearly shown in the sequence diagram user here after successful login to the application has the possibility of uploading an image of a new commodity. During this case the Baratto application allows the user to open the camera of the mobile phone which captures the image of the goods or services. On successful capture of the image, the Imagga Image recognition service is called which returns with the different classification category of the image back to the user. This helps the user to reduce the time taken to fill up the description of the image posted and also to maintain a proper tagging of the images in the application. The details of the commodity i.e the image and description are then saved into the FireStore database upon confirmation from the user.

2.3.2 User making selection of commodity request

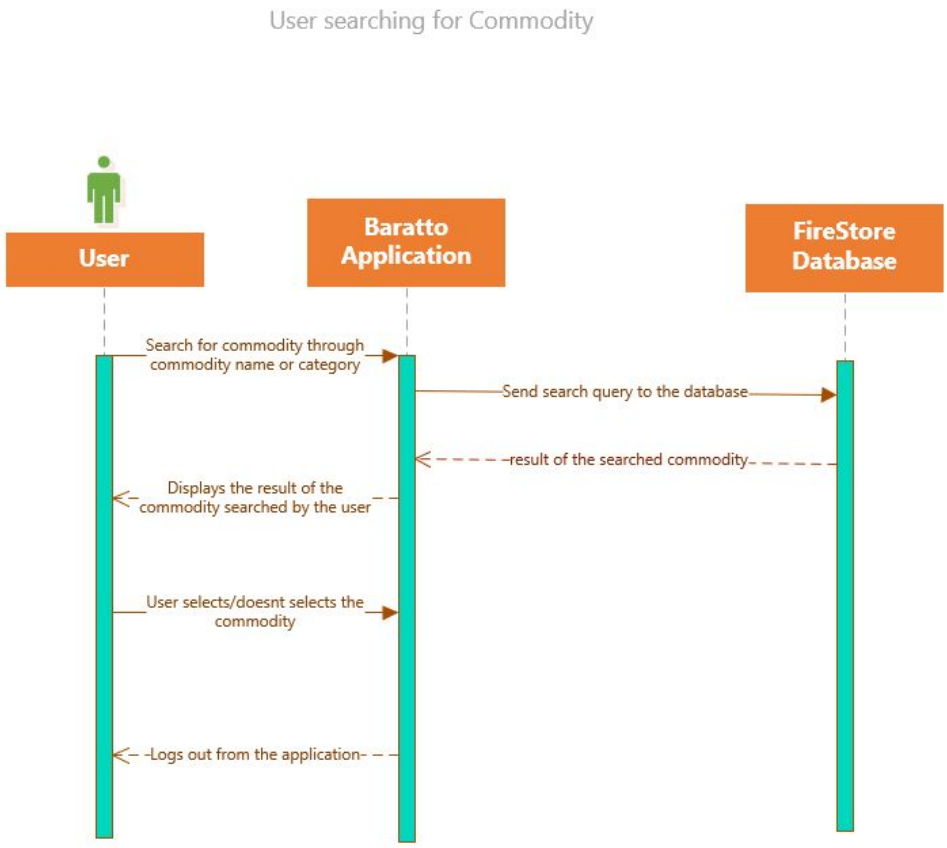


Figure 2.4: Selection of Commodity sequence diagram

The sequence diagram in 2.4 describes the process through which a user can select the available commodity posted by different registered users in the application. In this case, the user upon successful login to the application can search for commodity. Upon search, a request is been sent to the cloud FireStore database, which returns with the result of matching commodity in the database to the application and thus giving an option to the user to choose between different goods or services among the available options.

2.3.3 User communicating with the owner of the commodity

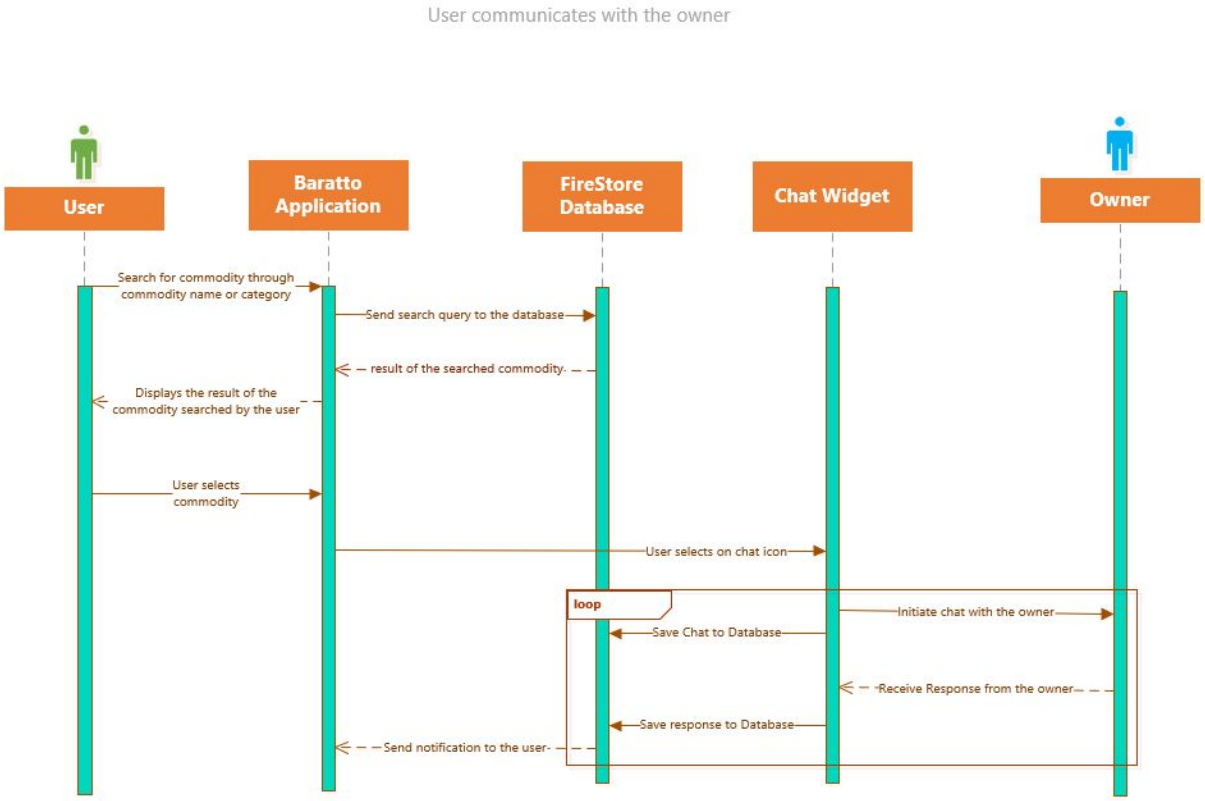


Figure 2.5: User-Owner Chat sequence diagram

The sequence diagram in 2.5 describes the process through which users can initiate a conversation with the owner of the commodity. This functionality helps the user as well as the owner to come down to a personal agreement that helps them carry forward with the bartering activity. In this sequence diagram, the user upon selection of the commodity has the option to initiate a conversation with the owner on selecting the chat option in the application. The initiated conversation in the chat window between the user and the owner then gets recorded in the FireStore database to enable the application to display Chat history in case the user is interested in it.

2.3.4 Chat History

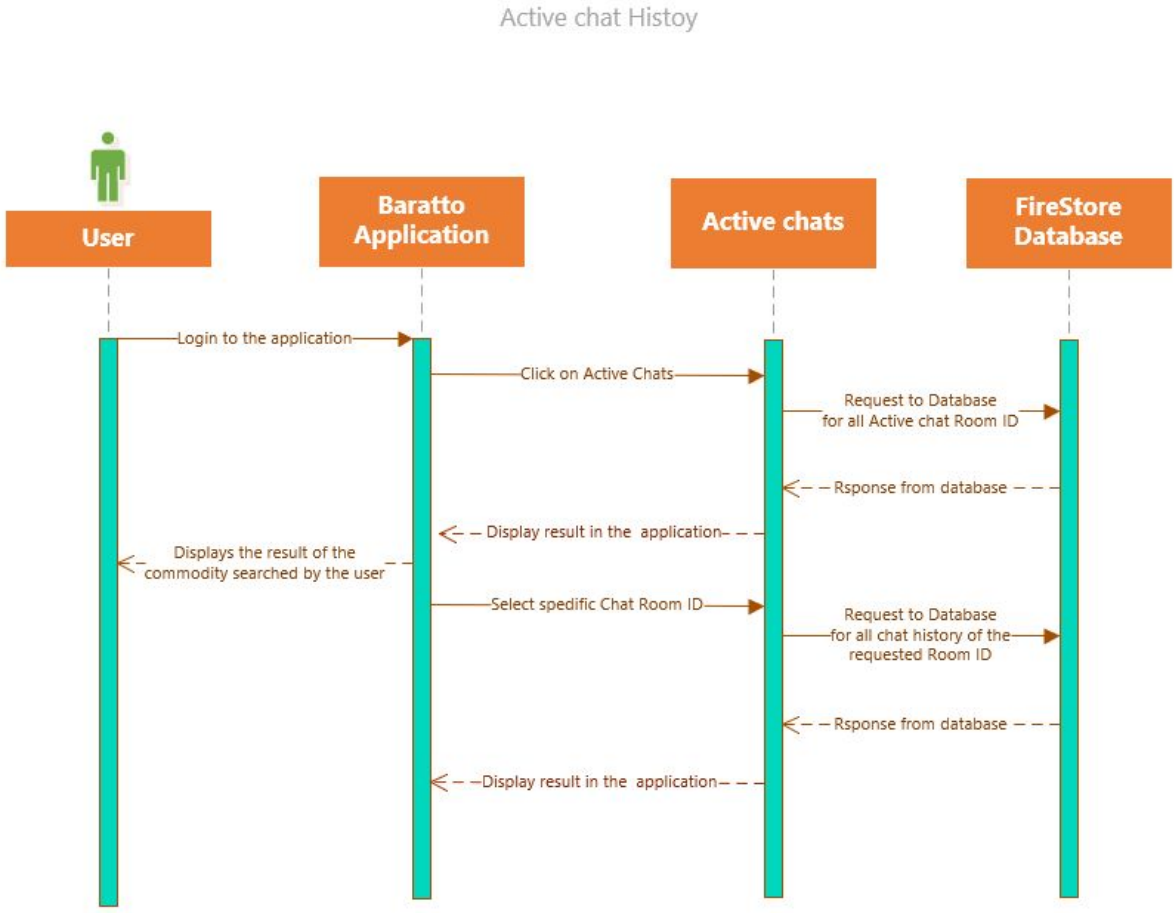


Figure 2.6: User Chat History sequence diagram

The sequence diagram in 2.6 describes the process through which a user can view the conversation that they have made with other registered users. The registered user in the application has the option to view active chats which contains the details of all the chats made by him. On selection of this option, a request is sent to the FireStore database which returns a response with all the chatroom id created against each chat initiated by the user. The user can further select any specific chatroom id to view the history of the chat or further continue the conversation with the user.

2.3.5 Database Diagram

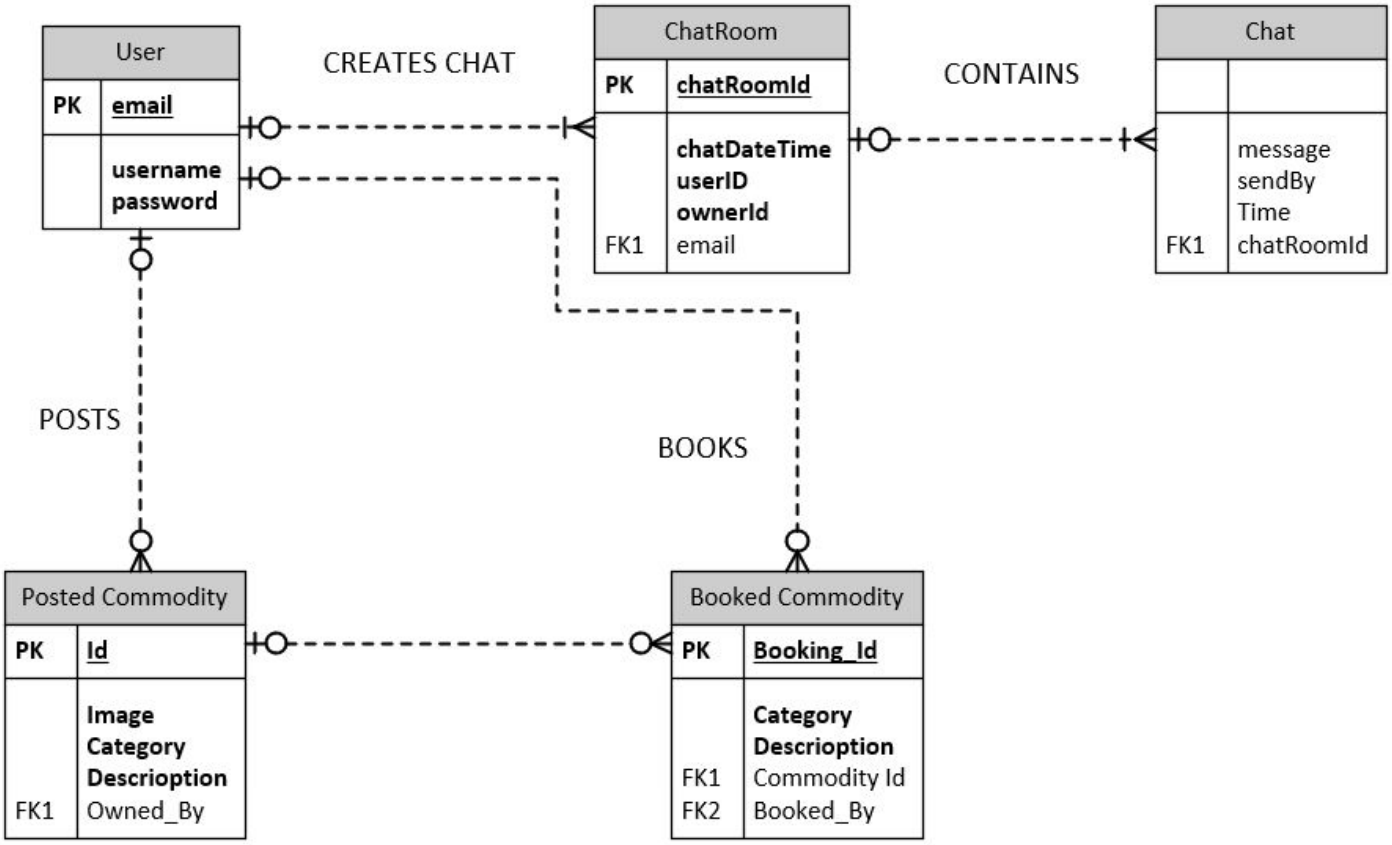


Figure 2.13: Database Diagram

The above diagram represents the entity-relationship model of the database structure that we will be using for our application. It can be seen that our application will have 5 tables with the functionalities as listed below:

UserTable: The details of the users are stored in this table. Whenever a user registers in the application the details of the registration, as well as login details, are stored into this table. The User table has EmailID which uniquely identifies every user.

ChatRoom: Once the user initiates a new chat with the owner of the commodity, a new chatroom Id gets created and it is saved in this table. This chat Room Id has further specific chats associated with its specific id.

Chat: Chat table is used to store all the chats of the user and the owner corresponding to its chat Room Id.

Posted Commodity: This table contains the details of all the commodities i.e the goods and services posted by the user. Every time a commodity is posted by a user an item id gets automatically generated by the table.

Booked Commodity: This table contains details of all the commodities being booked by the user

2.4 Imagga Image recognition algorithm

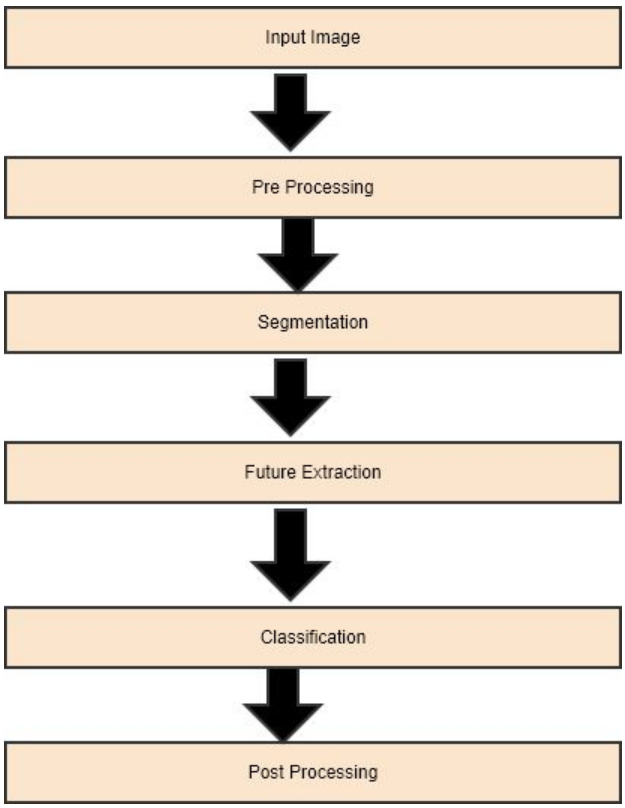


Figure 2.14: FlowChart of Image Classification processing algorithm

Following steps are followed while classification of images

1) Start
2) Scan the image.

3) Convert colour image into a grey image and then binary image.

4) Do preprocess like noise removal, skew correction etc.

5) Load the Image.

6) Do segmentation by separating lines from the textual image.

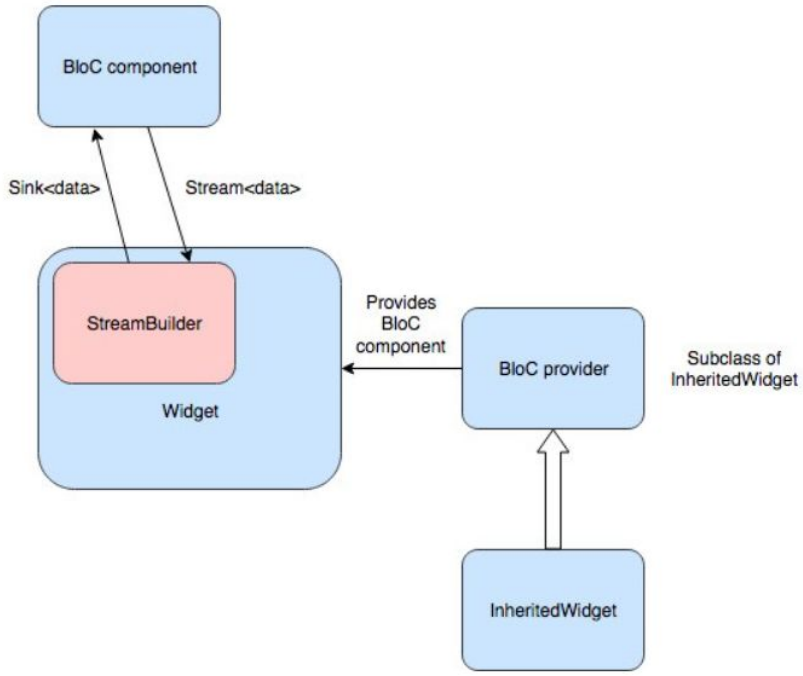
2.5 Selected architectural styles and patterns

2.5.1 Bloc architecture used by Flutter.

We choose Block architecture for our application because it allows easy evolution of an API design. BloC (Business Logic Component) is an architecture pattern introduced by Google at their IO conference this year. The idea behind it is to have separated components (BloC components) containing only the business logic that is meant to be easily shared between different Dart apps.

Building Blocks of BloC

In this section, I'm going to explain the main building components of BloC in Flutter. Let's start with a diagram to show how components are connected:



BloC component

At the core of BloC architecture, is the BloC component. It's the place where the business logic is happening

BloC provider

To inject the BloC component into the Screen / Widget, we use a class called Inherited Widget. This is a cheap and elegant way for dependency injection in Flutter. Every child of *InheritedWidget* will have an access to the components that *InheritedWidget* provides.

2.5.2 Three Tier Client-Server

A multilayered architecture is a client-server architecture in which presentation, application and data access functions are physically separated.

We choose a multitier architecture because it allows decoupling the complexity of the system, making it more flexible and reusable. Indeed, the developers acquire the power of modifying or adding specific layers without disrupting the entire application. Moreover, this kind of architecture allows to separate completely the access to data from the layer where the logic for presentation and the interaction with the customers is located: this is very important to make the system safer since the treated information are sensitive data.

More precisely, we adopted a three-tier architecture, composed of a Presentation (P) tier, an application (A) tier and a data access (D) tier: the mentioned separation between the clients and the data is possible thanks to the A tier.

Chapter 3

User interface design and Mobile Mock Screens

Here we present UX diagrams to show the flow which the Customer will follow to navigate inside the application.

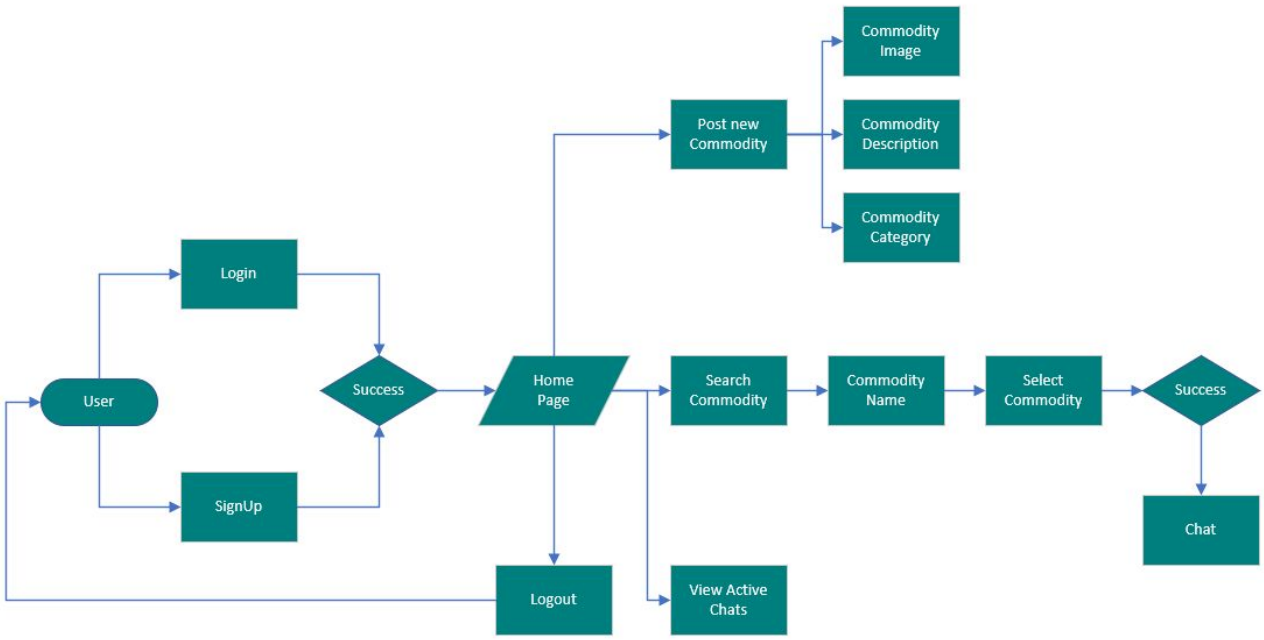


Figure 3.1: UX diagram -- User

As explained in the UX case diagram above, the users can register themselves by signing up in the application. Upon successful signup the user is being redirected to its home page which contains different options such as Posting new commodity, searching a commodity, viewing active chats as well as an option to logout from the application.

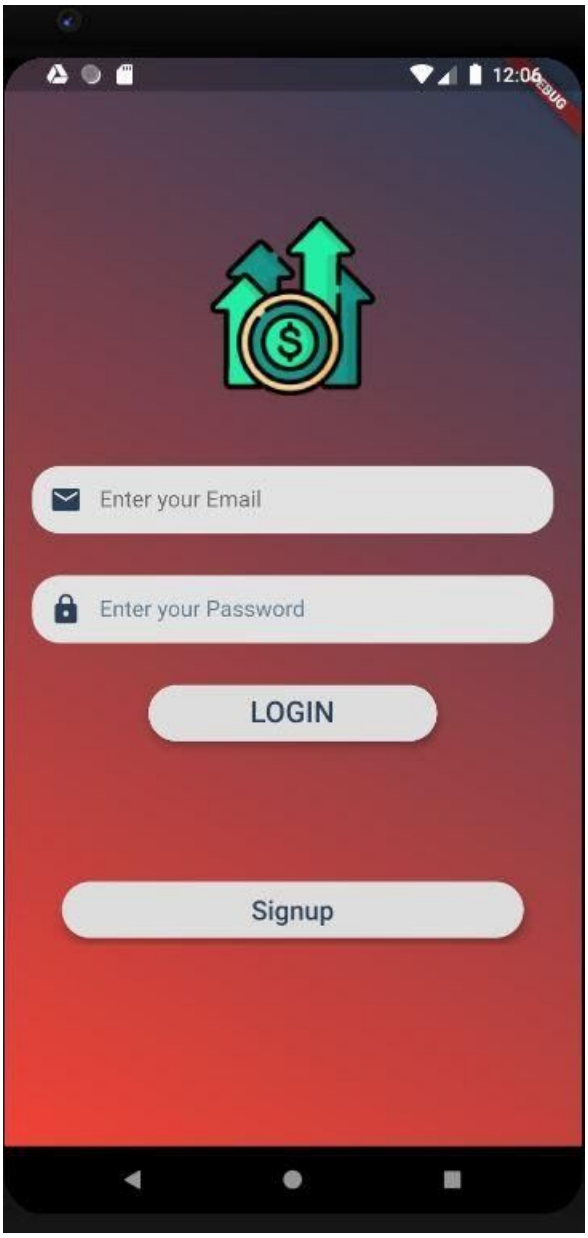
If the user chooses the option to post a new commodity they further have to provide to the system the details of the commodity such as the image, description as well as the commodity category.

If the user chooses to select search commodity they can search for a commodity by commodity name, upon which they will be shown with the different options of commodity matching the search query. The user further can select the and chat with the owner of the commodity from the item details screen.

Apart from this the home page also has the functionality where user can navigate to view which contains the history of all the conversation made by him and the owner of the commodity.

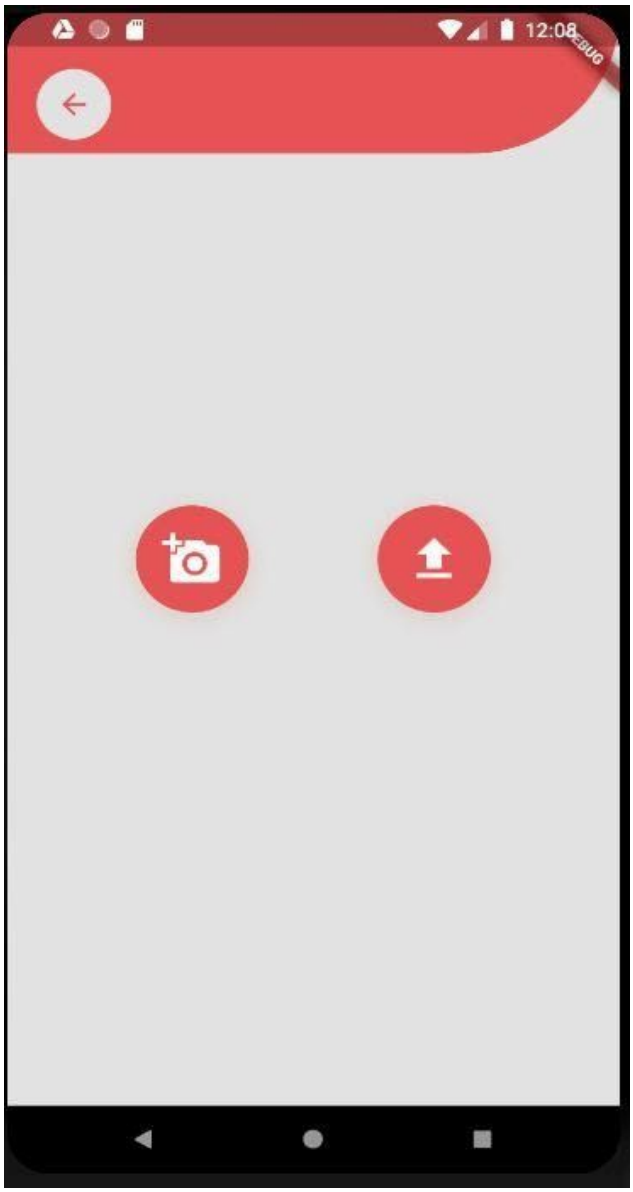
Application Mock Screens

1. Login Screen

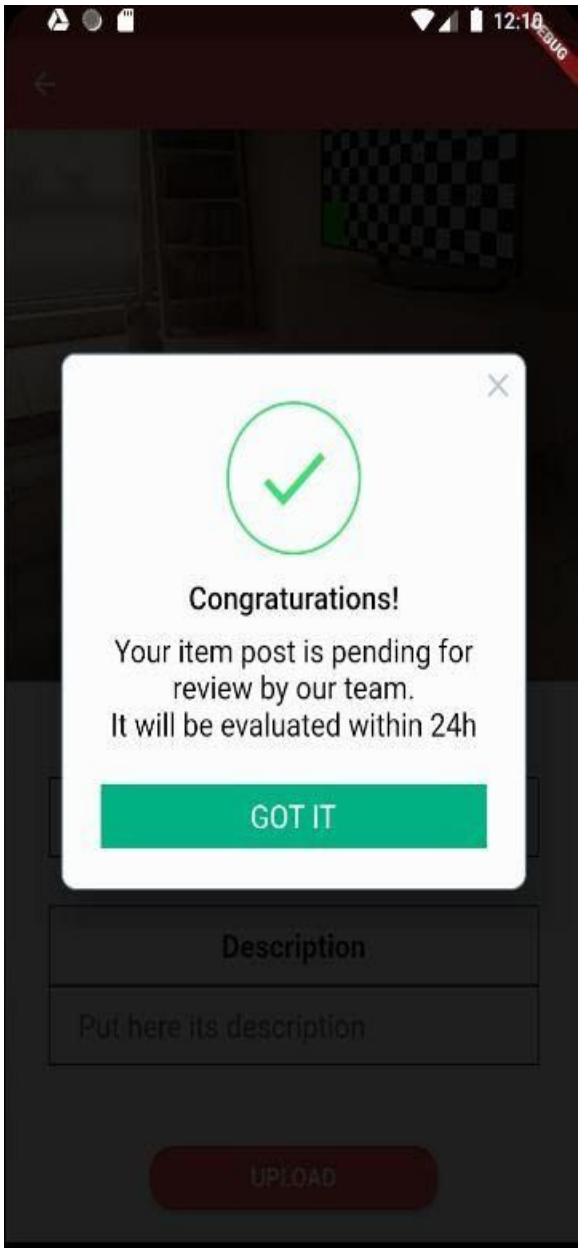


2. Home Screen

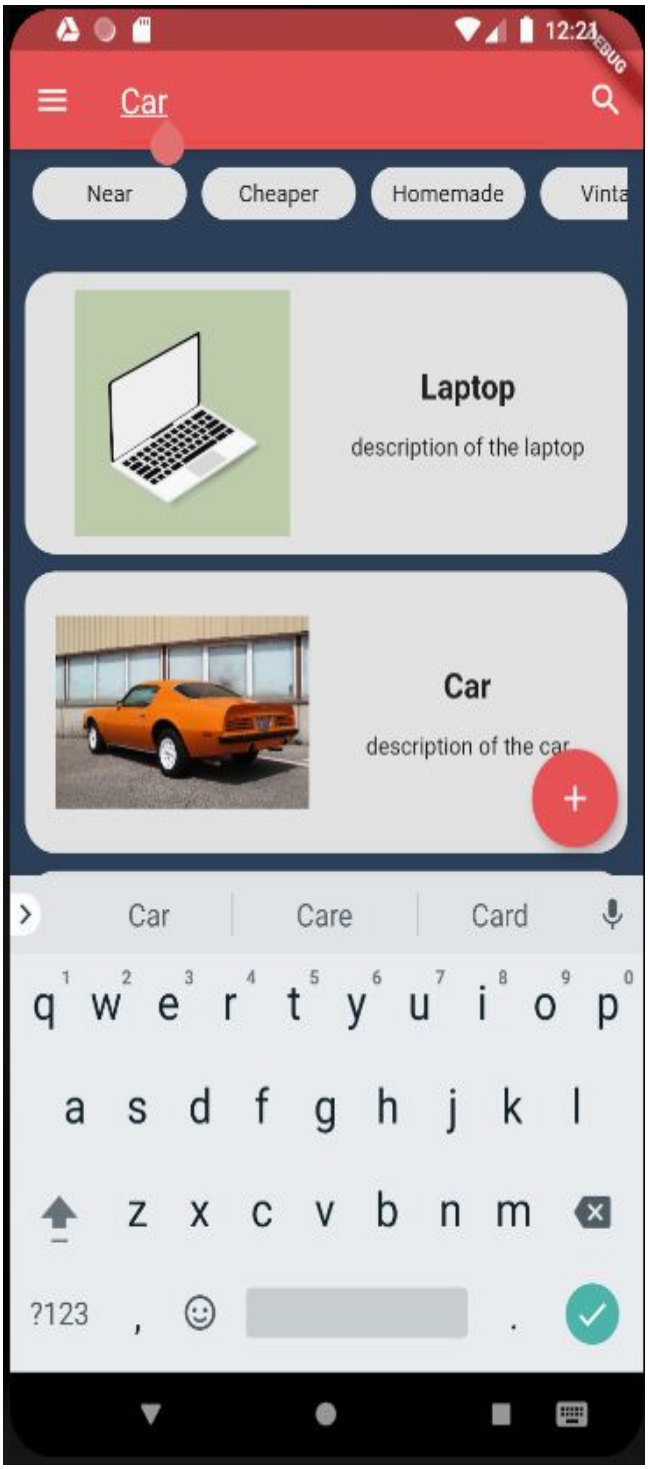
3. Upload Commodity Screen

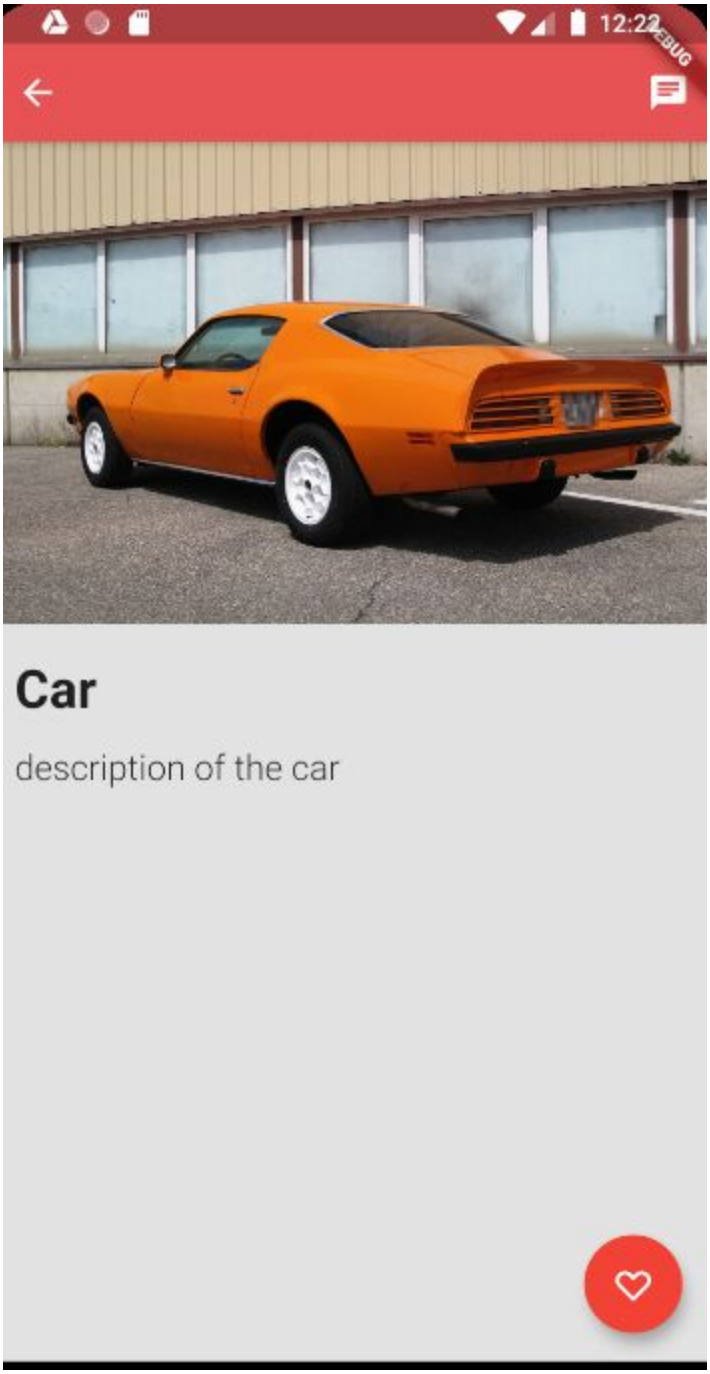



Title
panel, interior, room

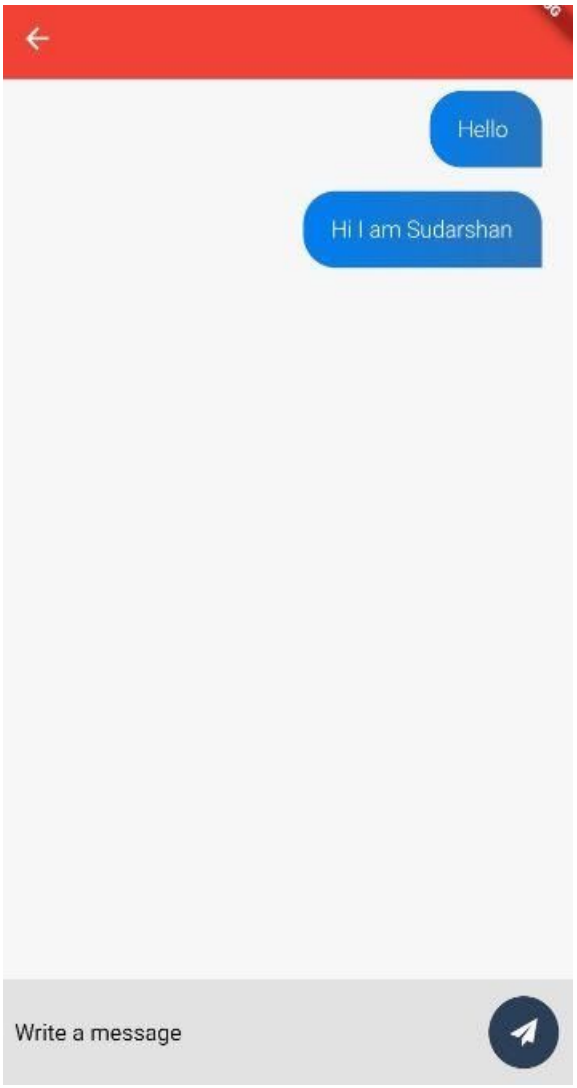

12:10
Congraturations!
Your item post is pending for review by our team.
It will be evaluated within 24h
GOT IT
Description
Put here its description
UPLOAD

4. Commodity Search Screen




12:22
DEBUG
Car
description of the car

5. User-Owner Chat Screen



Chapter 4

Implementation, integration and test plan

The system is divided into various subsystems:

- UserMobileApp
- External systems: Cloud Firestore, Imagga Image recognition API

These subsystems will be implemented, but also tested and integrated exploiting a bottom--up strategy, but considering also the importance of the various functionalities and trying to provide at each step of the plan a visible application feature. The components building the same subsystems will be implemented, integrated and tested for each subsystem and the integration and testing of the different subsystems will take place at a later stage (here we will focus only on the ApplicationServer subsystem). It is worth to notice that the external systems' components need not be implemented and tested just because they are external and they can be considered reliable.

The below table lists the main features available for Baratto users, their importance and implementation difficulty is shown below to better understand the decisions about implementation, testing and integration that will be taken in the rest of this section.

Feature	**Importance for the User**	**Difficulty of implementation**
Sign up and login	*High*	*Low*
Visualize home page data	*High*	*Medium*
Upload and post new Commodity	*High*	*High*

Automatic Image Classification & Moderation	*High*	*High*
Chat with the owner of Commodity	*High*	*High*
View Active Chats/Chat History	*High*	*High*
Selection of a commodity for trade	*High*	*High*

Table 5.1: Feature table User

It's important to note that the external APIs available for integration with the application i.e Imagga image recognition API have to be tested before the development of the application because functionalities such getting classification of Image as well as the moderation level of the requires these APIs to be in working condition.

For this reason and looking at the table that maps the features on their relevance for the users and their overall complexity, the components have to be implemented and tested :

Sign up and log in: sign up and log in features are an entry condition for the right functioning of the system, but they are not core features and they are not very complex

Visualize Home page Data: For this feature, the users must log in to the application get to see the commodities that the other registered users are ready to trade around him. It also should allow the users to navigate to functionalities such as viewing Active chats, Favorite Items, and Logout functionalities.

Upload and post new commodity: This is the most important feature of the application where the user can upload the image of the goods and service captured by the camera as well as additional description of the image before posting it for advertisement.

Automatic Image classification and moderation: this is another core function of the system where the image captured for posting the commodity is sent to the Imagga image recognition api for image classification and moderation.

Chat with the owner of the commodity: This is one of the most important functions of our application which enables the user registered in the application to chat with the

owner of the services being selected by them.

View active chats/chat history: This feature enables the user to refer to the history of conversation made between him and the owner of the item.

Selection of the commodity for trade: This feature enables users to select between different options of goods and services available for bartering that are posted by other registered users.

The verification and validation phases must start as soon as the development of the system begins in order to find errors as quickly as possible. As mentioned, the program testing to find bugs has to proceed in parallel with the implementation: unit testing has to be performed on the individual components and, as soon as the first (partial) versions of two components that have to be integrated are implemented, the integration is performed and tested: we incrementally perform integration to facilitate bug tracking. Moreover, scaffolding techniques must be used when needed.

Since the Baratto application is a relatively small system, the chosen integration technique is a structural one: the bottom-up approach is followed, but considering that the application has to be implemented and tested before the other services for the reasons explained before (the same argument holds for the most important features of each offered service).

Once the system is completely integrated, it must be tested as a whole to verify that functional and nonfunctional requirements hold.

Chapter 6

Hardware and Software Requirements

6.1 Mobile Application Requirements

Component Name	Minimum Requirement
Memory	2GB
Processor Type	ARMv7/ARM Cortex A8
Processor Speed	200 MHz
Hard Disk Space	32 MB
OS	Android 4.4 / iOS 6

Table 6.1: Requirements for Mobile Application

It is mandatory for all the smart devices that will be hosting our application to have internet connectivity as well as a working camera to post new commodity.

6.2 Software Requirements for development

Below there is the list of software that is required for the development

- Android Studio for developing Android as well as iOS application
- Cloud FiresStore database