UniTrade

Project Report Submitted in Partial Fulfillment of the Requirements for the Degree of

Bachelor of Engineering
In
Computer Science & Engineering

Submitted by

Lakshita Dave :(Roll No. 22UCSE4023)

Muskaan Agarwal :(Roll No. 22UCSE4025)

Under the Supervision of

Anil Gupta

Professor

Department of Computer Science and Engineering

MBM University, Jodhpur

May, 2025

Department of Computer Science & Engineering

MBM University, Ratanada, Jodhpur, Rajasthan, India –342011

CERTIFICATE

This is to certify that the work contained in this report entitled **"UniTrade**" is submitted by the group members Ms. Lakshita Dave (Roll. No: 22UCSE4023) and Ms. Muskaan Agarwal (Roll. No: 22UCSE4025) to the Department of Computer Science & Engineering, MBM University, Jodhpur, for the partial fulfillment of the requirements for the degree of **Bachelor of Engineering** in **Computer Science.**

They have carried out their work under my supervision. This work has not been submitted elsewhere for the award of any other degree or diploma.

The project work in our opinion, has reached the standard fulfilling of the requirements for the degree of Bachelor of Engineering in accordance with the regulations of the University.

Anil Gupta
Professor
(Supervisor)

Dr. Shrwan Ram Balach
(Head of Department)

DECLARATION

We, **Lakshita Dave and Muskaan Agarwal,** IV year, hereby declare that this seminar/project titled **"UniTrade"** is a record of original work done by us under the supervision and guidance of **Anil Gupta, Professor, CSE**.

We further certify that this work has not formed the basis for the award of the Degree/Diploma/Associateship/Fellowship or similar recognition to any candidate of any university and no part of this report is reproduced as it is from any other source without appropriate reference and permission.

Lakshita Dave (22UCSE4023)

Muskaan Agarwal (22UCSE4025)

VIIIth Semester, Computer Science

ACKNOWLEDGEMENT

We would like to express our heartfelt appreciation to everyone who contributed to the completion of our college project on "**UniTrade**". We are grateful for the support and assistance provided by various individuals and institutions

We would like to express our deep sense of gratitude to our project supervisor **Anil Gupta** for his able guidance and useful suggestions, which helped us in completing the project work in time. His insights and knowledge in the field were invaluable to us.

We take immense pleasure in thanking Dr. Shrwan Ram, Head of Department of Computer Science department, and Dr. Jayshree Vajpayee, Dean of MBM University for having permitted us to carry out this project and providing us with the necessary resources and fostering a conducive environment for research and learning. The department's commitment to academic excellence played a significant role in enabling us to pursue this project

Lakshita Dave

Muskaan Agarwal

ABSTRACT

In today's academic environment, university students often seek cost-effective solutions for acquiring essential items such as textbooks, electronics, and other learning resources. At the same time, many students wish to sell or donate items they no longer need. However, the lack of a secure, trusted, and convenient platform tailored specifically for students often makes this exchange inefficient or inaccessible. UniTrade addresses this gap by providing a dedicated mobile application that creates a closed, university-based marketplace for the buying and selling of second-hand goods.

Developed using Flutter, UniTrade ensures cross-platform compatibility and a smooth, responsive user experience. It incorporates Firebase services for backend support, including real-time database management, user authentication, and messaging. A key feature of UniTrade is its university-specific access, enforced through email domain verification during signup. This ensures that only verified university students can use the platform, significantly enhancing trust and safety in all transactions.

The application supports item listing across multiple categories—books, electronics, furniture, stationery, and more. Buyers can browse through listings, filter by category, search based on keywords, and communicate directly with sellers using an in-app chat feature. Each user has a profile where their listings, ratings, and transaction history are maintained to improve transparency and accountability within the platform.

UniTrade promotes sustainability by encouraging reuse and reducing waste. It also improves convenience by enabling students to engage in local, on-the-spot exchanges within the university campus, eliminating the need for long-distance travel or dealing with unknown buyers or sellers from generic platforms. Additionally, the app encourages student interaction and builds a sense of community through trusted peer-to-peer exchanges.

Contents

List of Figures

Chapter 1

Introduction to Project

1.1 Project Description

UniTrade is a mobile-based marketplace application developed using Flutter, specifically designed to facilitate the buying and selling of second-hand items among university students. It provides a secure and trusted environment by restricting access to verified students through university email authentication. The platform allows students to list items such as books, electronics, stationery, furniture, and more, enabling direct peer-to-peer transactions within the campus.

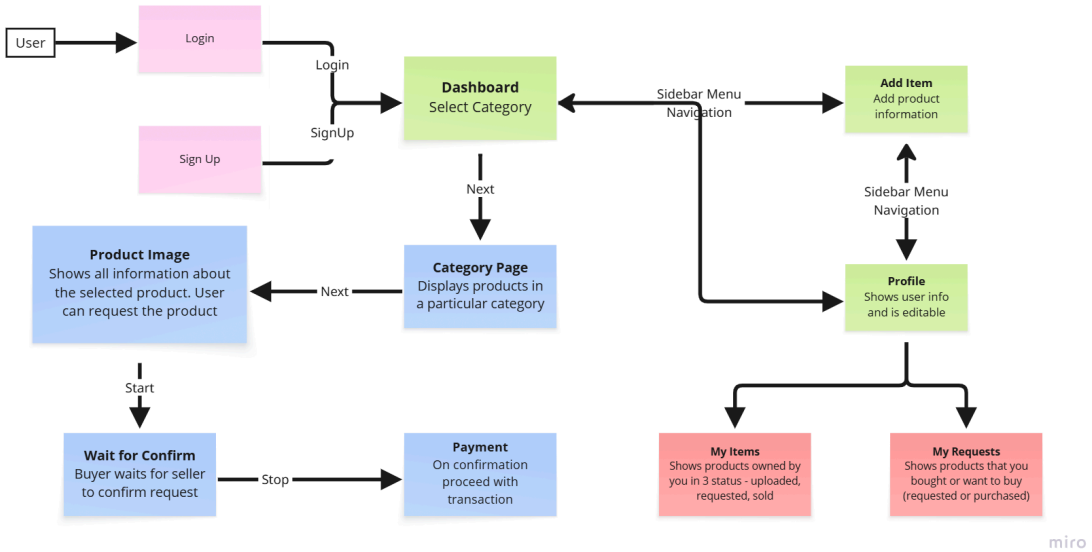


Fig 1.1 : Flow Diagram

The app includes essential features like category-based item listings, search and filter options, user profiles, an in-app chat system, and a clean, responsive UI. Backend functionalities are powered by Firebase, including Firestore for data storage, Firebase Auth for secure login, and Firebase Storage for image hosting. By focusing on a single university community, UniTrade enhances trust, ensures faster transactions, and fosters a culture of sustainability through reuse.

1.2 Motivation

The primary motivation behind UniTrade stems from the common yet unresolved problem faced by university students—how to conveniently and safely buy or sell second-hand items within the student community. Traditional platforms like OLX or Facebook Marketplace do not offer campus-specific filters, often leading to interactions with unknown users, concerns over safety, and logistical challenges.

During our time at the university, we noticed the frequent demand for used books, gadgets, and furniture, but students lacked a reliable platform to find or offer these items. This inspired us to create a mobile app tailored to university life, where students can connect securely, save money, and support one another.

Furthermore, UniTrade is not just a transactional tool—it contributes to sustainable practices by promoting the reuse of items, reducing waste, and minimizing the carbon footprint associated with new purchases and deliveries.

1.3 Need of the Project

There is a growing need for campus-specific digital tools that simplify student life, and UniTrade addresses a critical gap in this space. Key reasons why this project is needed include:

- Security and Trust: Generic platforms don’t ensure the safety or verification of users. UniTrade authenticates users with their university email, building a secure environment.
- Affordability: Many students are on a tight budget. Buying second-hand items is a practical solution, but without a proper platform, opportunities are lost.
- Convenience: With a localized app, students can find, communicate, and meet up on campus, making transactions quick and easy.
- Community Engagement: The platform fosters peer interaction and support, encouraging a culture of sharing and mutual benefit.
- Sustainability: Promoting reuse directly contributes to environmental sustainability, a value increasingly important to students and institutions.

In summary, UniTrade is not just a mobile app—it is a community-driven solution that addresses financial, social, and environmental challenges within the university ecosystem.

Chapter 2

Requirement Modeling

This chapter explains the detailed decomposition of the requirements for our UniTrade - A university marketplace application.

2.1 Functional Requirements

Functional requirements define the specific behavior and functions of the system. For the UniTrade marketplace application, these include:

2.1.1 User Management

- User Registration and Login: Users must register using their university email ID. Email verification ensures that only valid university students can join the platform. Login requires secure credentials and session handling.
- Profile Management: Users can update personal information such as name, contact number. The profile also stores the user's listed items, and transaction history.

2.1.2 Upload Item:

- Users can upload second-hand items under defined categories such as:
    - Books
    - Electronics

    - Lab Instruments
    - Uniform
    - Others
- While uploading, users must provide item title, description, images, price, and optional tags.
- Uploaded item is visible in the user profile “My Items” menu

2.1.3 Purchase Items:

- Users request for a listed item.
- Owner can see sold products in “My Items” whereas buyer can see products in “My Requests”.
- The owner then accepts the request by contacting the user.
- Further the product is sold.

2.1.4 Administrative Functions:

Admins can:

- Manage global settings such as category updates.
- Ban or restrict users violating guidelines.
- View user reports and resolve disputes.

2.2 Non-Functional Requirements

Non-functional requirements specify the quality attributes and constraints of the system.They describe how the system should behave.

- Performance: The app must load quickly and handle multiple simultaneous users without significant delays.
- Usability: The user interface should be intuitive and easy to navigate.
- Security: User data must be protected through encryption and secure authentication mechanisms.
- Scalability: The system should be able to scale to accommodate increasing numbers of users and bicycles.
- Maintainability: The codebase should be well-documented and structured to facilitate maintenance and updates.

2.3 Deployment Environment

The deployment environment outlines the technical specifications and conditions under which the application will operate. The application will be deployed on both iOS and Android devices, leveraging Flutter for cross-platform compatibility.Backend, developed in Python, will run on web servers capable of handling Python scripts and interfacing with a MySQL database.The MySQL database will be hosted on a reliable server environment, ensuring data integrity and availability.Integration with external APIs, such as Firebase API, for user authentication. The application will require a stable internet connection for real-time data synchronization and communication with the backend services.

Frontend:

- Developed using Flutter (Dart language)
- Deployed to Android (Google Play Store) and optionally to iOS (App Store)

Backend:

- Firebase for authentication, Firestore database, storage, and cloud functions
- Hosting and serverless functions managed by Firebase

Development Tools:

- Android Studio / VS Code
- Firebase Console
- Postman (for testing APIs if using custom backend)

Chapter 3

System Design

The system design chapter elaborates on how the UniTrade marketplace application will be structured and organized. This includes the overall system architecture, the decomposition of the application into modules, and the design of the database that will store and manage data.

3.1 System Architecture

The system architecture of the UniTrade marketplace application is designed to ensure seamless interaction between users and the system. It follows a client-server model, comprising the following components:

- Client Side: The mobile application developed using Flutter, providing cross-platform compatibility for both iOS and Android devices. The app includes user interfaces for registration, login, product browsing, uploading items, and assisting transactions.
- Server Side: The backend server, implemented in Python, handles business logic, processes user requests, and communicates with the database.
- Database: MySQL is used to manage user data, product inventory, and other relevant information. The database server ensures data integrity and supports efficient data retrieval.
- External APIs: Integration with Firebase API for user authentication.

3.2 Module Decomposition

The application is divided into several key modules, each responsible for specific functionalities:

4.1 User Module

- User registration & login via university email
- Profile creation and editing
- View own listings, and requests

4.2 Listing Module

- Upload new items (category-based)
- Browse items by category or search term
- Filter by price, date, or condition
- View item details

4.3 Request & Notification Module

- Request for items
- Contact the owner
- Money transaction takes place

3.3 Database Design

The database design is designed for efficient data management and retrieval. The following are the main tables in the MySQL database:

Users Table

- User_ID: Unique identifier for each user.
- Username: User's login name.
- email_id: User's email address.
- Phone_no: Unique phone number
- password: Encrypted user password.

```
mysql> desc users;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| user_id  | int         | NO   | PRI | NULL    |       |
| username | varchar(40) | NO   |     | NULL    |       |
| mail_id  | varchar(40) | YES  | UNI | NULL    |       |
| phone_no | int         | YES  | UNI | NULL    |       |
| password | varchar(40) | NO   |     | NULL    |       |
| logged   | tinyint(1)  | YES  |     | 0       |       |
+----------+-------------+------+-----+---------+-------+
6 rows in set (0.01 sec)
```

- Logged: A variable to check on user

Categories Table

- cat_id: Unique identifier for each category.
- cat_name: Category's name

```
mysql> desc categories;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| cat_id   | int         | NO   | PRI | NULL    |       |
| cat_name | varchar(30) | NO   |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

Products Table

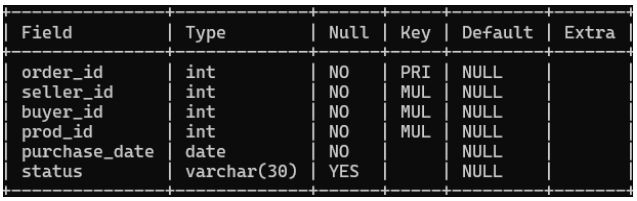
- prod_id : Unique identifier for each product
- prod_name : Product's name
- Cat_id: Product's category's identifier
- owner_id : Product's owner's identifier
- Prod_img: Product's image
- Price: Price of item
- Company: Brand/Company name of product
- Mfg_date: Manufacturing date of product
- Exp_date: Optional Expiry date of product
- Specification: Condition and some features of product
- Prod_status: The current status of product, is it uploaded, requested by user (contact is pending) or is it sold.

```
mysql> desc products;
+---------------+---------------+------+-----+---------+-------+
| Field         | Type          | Null | Key | Default | Extra |
+---------------+---------------+------+-----+---------+-------+
| prod_id       | int           | NO   | PRI | NULL    |       |
| prod_name     | varchar(40)   | NO   |     | NULL    |       |
| cat_id        | int           | NO   | MUL | NULL    |       |
| owner_id      | int           | NO   | MUL | NULL    |       |
| prod_img      | blob          | NO   |     | NULL    |       |
| price         | int           | NO   |     | NULL    |       |
| company       | varchar(40)   | NO   |     | NULL    |       |
| mfg_date      | date          | NO   |     | NULL    |       |
| exp_date      | date          | YES  |     | NULL    |       |
| specification | varchar(8000) | YES  |     | NULL    |       |
| prod_status   | varchar(30)   | YES  |     | NULL    |       |
+---------------+---------------+------+-----+---------+-------+
11 rows in set (0.01 sec)
```

Orders Table

- Order_ID: Unique identifier for each transaction.
- Seller_ID: Identifier of the user selling the product.

- Buyer_ID: Identifier of the user requesting or purchasing the product.
- Prod_ID: Identifier of product to be sold
- Purchase_date: Date of transaction
- Status: Product sold or not



Field	Type	Null	Key	Default	Extra
order_id	int	NO	PRI	NULL	
seller_id	int	NO	MUL	NULL	
buyer_id	int	NO	MUL	NULL	
prod_id	int	NO	MUL	NULL	
purchase_date	date	NO		NULL	
status	varchar(30)	YES		NULL	

Chapter 4

Project Implementation

4.1 Technology Stack

1. Frontend (Client-Side – Mobile App)
    - Framework: Flutter
    - Language: Dart
    - Target Platforms: Android & iOS
    - UI Toolkit: Material Design (built-in with Flutter)
2. Backend (Server-Side)
    - Language: Python
    - Framework: Flask (Lightweight, REST APIs)
    - API Type: RESTful APIs
3. Database (Data Layer)
    - Database System: MySQL
    - ORM: SQLAlchemy (for Flask)
    - Tables:
        - ➢ Users
        - ➢ Categories(Books, Electronics, etc)
        - ➢ Products
        - ➢ Orders

4.2 Code Implementation

The code implementation for the UniTrade marketplace application involves several screens and integration with backend services.

4.2.1 User registration and Login

```
Form(
key: _formKey,
child: Column(
children: [
CustomTextFormField(
    controller: nameController,
    keyboardType: TextInputType.text,
    hintText: 'First Name*',
    validator:nameValidator,
    autoValidateMode: AutovalidateMode.onUserInteraction,
    obscureText: false,
    suffix: false,
),CustomTextFormField(  // CustomTextFormField
    controller: lastNameController,
    keyboardType: TextInputType.text,
    hintText: 'Last Name',
    validator: lastNameValidator,
    autoValidateMode: AutovalidateMode.onUserInteraction,
    obscureText: false,
    suffix: false,
),CustomTextFormField(  // CustomTextFormField
    controller: emailController,
    keyboardType: TextInputType.emailAddress,
    hintText: 'Email id*',
    validator:emailSignUpValidator,
    autoValidateMode: AutovalidateMode.onUserInteraction,
    obscureText: false,
```

Fig: 4.1 - Login Code

LoginScreen: Allows users to log in to the mobile application, provides fields for entering phone number or emailID and password. It includes options for registering a new account if the user doesn't have one.

SignupScreen: Enables users to create a new account for the application and Requests necessary information such as phone number, password,etc .Using python, during login we check if credentials are correct by sending data to backend to be compared with database entries and during sign up post data to store in the database.

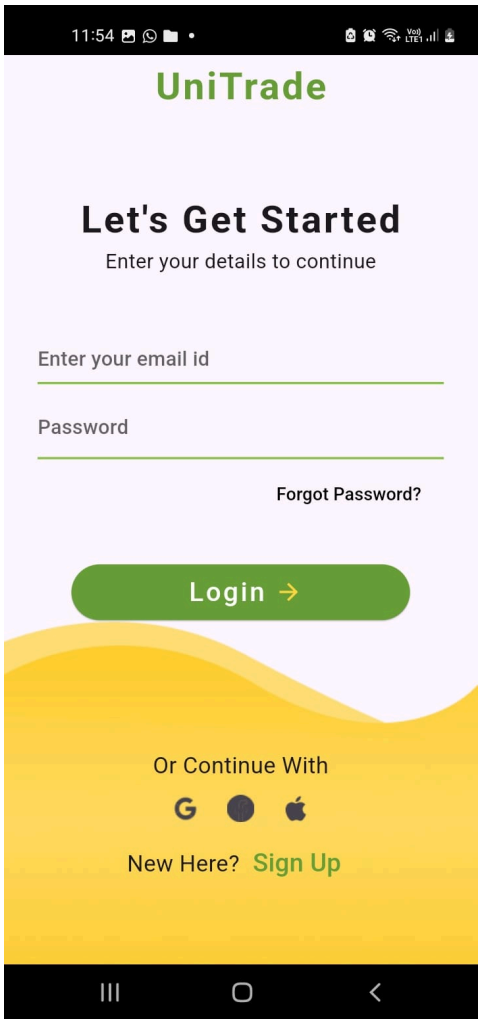


Fig 4.2 - Login Screen

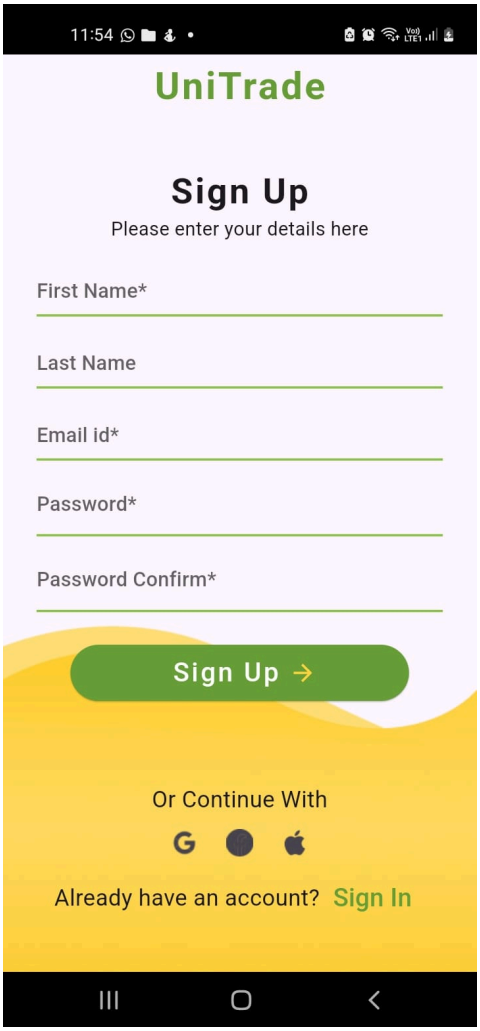


Fig 4.3 - Sign Up Screen

4.2.2 DashboardScreen

Displays the main interface of the application after successful login. It provides users with various available categories (Books, Electronics, Instruments, etc). The sidebar in it has 2 buttons: Profile and Add Item. By selecting any of these two we can navigate to other screens.

```dart
class CategoryCard extends StatelessWidget {
  final String imageUrl;
  final String title;

  const CategoryCard({
    required this.imageUrl,
    required this.title,
  });

  @override
  Widget build(BuildContext context) {
    return Container(
      width: 40.w,
      height: 25.h,
      decoration: BoxDecoration(
        color: AppColors.placeHolderColor.withOpacity(0.15),
        borderRadius: BorderRadius.circular(2.w),
      ),
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: <Widget>[
          CircleAvatar(
            backgroundImage: NetworkImage(imageUrl),
            radius: 14.w,
          ),
          SizedBox(height: 1.h),
          Padding(
            padding: EdgeInsets.symmetric(horizontal: 2.w),
```

Fig 4.4 - Dashboard Code

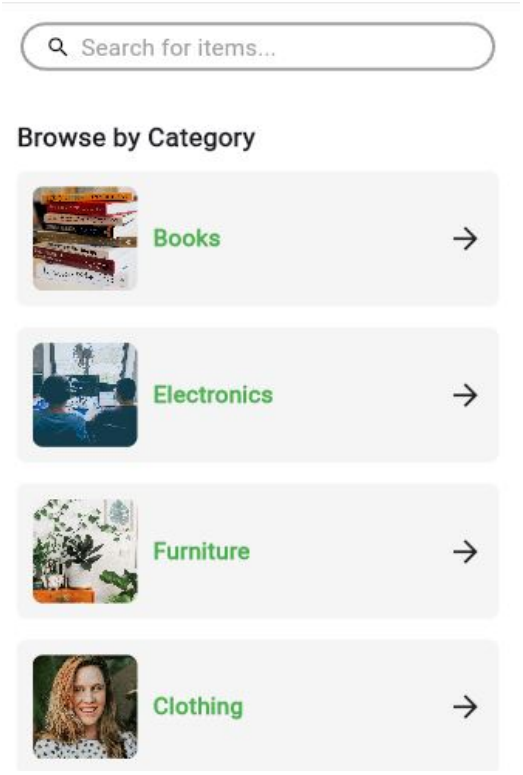
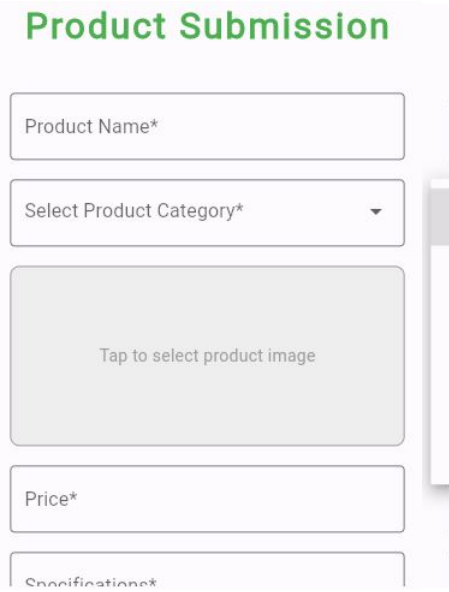
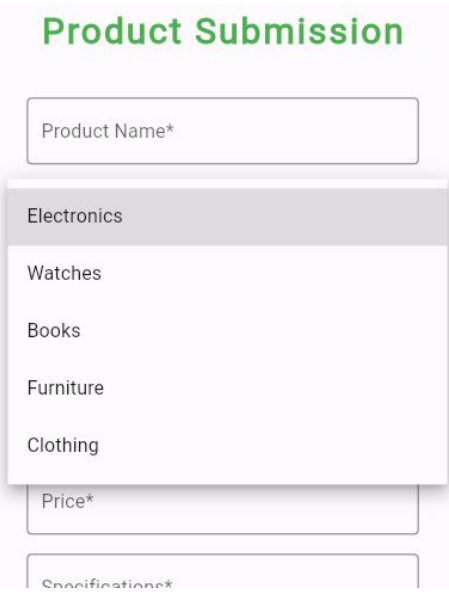


Fig 4.5 - Categories Screen

4.2.3 Add Item Screen - .

Displays a form to input details about the new product to be sold. It includes product name, category, company name, its image, price, manufacturing date and its specifications. For image upload we use the camera plugin in flutter.





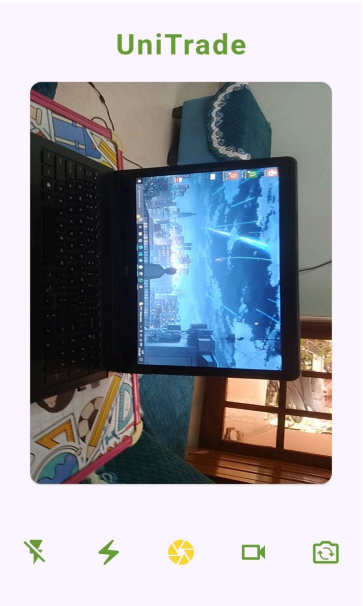


Fig 4.6 - Add Item Form **Fig 4.7 - Select Category** **Fig 4.8 - Upload Image**

4.2.4 Python - Backend

```
File  Edit  Format  Run  Options  Window  Help
from flask import Flask, request, jsonify
from flask_cors import CORS

app = Flask(__name__)
CORS(app)


def init_db():
    with sqlite3.connect("UniTrade.db") as conn:
        cursor = conn.cursor()
        cursor.execute('''create table if not exists users(
                        user_id int primary key,
                        username varchar(40) not null,
                        mail_id varchar(40) unique,
                        phone_no int unique,
                        password varchar(40) not null,
                        logged bool default False
                        );''')
        cursor.execute('''create table categories(
                        cat_id int primary key,
                        cat_name varchar(30) not null
                        )''')
        cursor.execute('''insert into categories
                        values
                        (101, "Books"),
                        (102, "Electronics"),
                        (103, "Lab Instruments"),
                        (104, "Uniform"),
                        (105, "Others")''')
        cursor.execute('''create table products(
                        prod_id int primary key,
                        prod_name varchar(40) not null,
                        cat_id int not null,
                        owner_id int not null,
                        prod_img blob not null,
                        price int not null,
                        company varchar(40) not null,
                        mfg_date date not null,
                        exp_date date,
                        specification varchar(8000),
                        prod_status varchar(30) check (prod_status in ("uploaded","requested","sold")),
                        foreign key(cat_id) references categories(cat_id),
                        foreign key(owner_id) references users(user_id)
                        );''')
```

```python
@app.route('/api/login', methods=['POST'])
def login():
    data = request.json
    email = data.get('email')
    password = data.get('password')
    query = "Select * from users where email = ? and password = ?"
    params = (email,password,)
    with sqlite3.connect("UniTrade.db") as conn:
        cursor = conn.cursor()
        cursor.execute(query, params)
        items = cursor.fetchall()
    return jsonify(items)

@app.route("/api/register", methods=["POST"])
def register():
    data = request.json
    with sqlite3.connect("UniTrade.db") as conn:
        cursor = conn.cursor()
        try:
            cursor.execute("INSERT INTO users (username, email_ID, passowrd) VALU
            conn.commit()
            return jsonify({"message": "User registered successfully"}), 201
        except sqlite3.IntegrityError:
            return jsonify({"error": "Email already exists"}), 400

@app.route("/api/categories", methods=["GET"])
def get_items():
    query = "SELECT * FROM categories"
    with sqlite3.connect("UniTrade.db") as conn:
        cursor = conn.cursor()
        cursor.execute(query)
        items = cursor.fetchall()
    return jsonify(items)
```

```python
@app.route("/api/products", methods=["GET"])
def get_items():
    category = request.args.get("category")
    query = "SELECT * FROM products"
    params = ()
    if category:
        query += " WHERE category = ?"
        params = (category,)
    with sqlite3.connect("UniTrade.db") as conn:
        cursor = conn.cursor()
        cursor.execute(query, params)
        items = cursor.fetchall()
    return jsonify(items)

@app.route("/api/addProduct", methods=["POST"])
def add_item():
    data = request.json
    with sqlite3.connect("UniTrade.db") as conn:
        cursor = conn.cursor()
        cursor.execute("Select cat_ID from categories where cat_name = ?",data['category'])
        cat_ID = cursor.fetchall()
    with sqlite3.connect("UniTrade.db") as conn:
        cursor = conn.cursor()
        cursor.execute("INSERT INTO products (prod_name, cat_ID, owner_ID, prod_img, price,
                       (data["prod_name"], cat_ID,  data["user_ID"], data["prod_img"], data
        conn.commit()
        return jsonify({"message": "Product added successfully"}), 201

if __name__ == '__main__':
    app.run(debug=True)
```

Fig 4.9 - Backend Logic Code

Routes are made for each API. There are 2 methods- “get data” and “post data”.

1. api/login: user authentication by verifying number/email and password against stored credentials in the database.
2. api/register: Stores the new user details securely in the database
3. api/categories: Retrieves all categories’ data from categories table.
4. api/products: Gets all products’ data of particular category selected.
5. api/addProduct: Stores the details of new products to sell in the database.

Chapter 5

Conclusion & Future Work

5.1 Conclusion

The UniTrade application successfully addresses a common yet often overlooked need within university campuses - the buying and selling of second-hand items such as books, electronics, and other essentials among students. By creating a dedicated platform exclusively for university students, the app promotes a secure, trusted, and localized environment for transactions.

Developed using Flutter for cross-platform support and a Python + MySQL backend for reliable server-side processing, UniTrade ensures smooth user interactions, quick listing uploads, real-time search, and efficient communication between buyers and sellers. The integration of role-based access and an admin panel also adds layers of control and safety, ensuring quality and appropriate content on the platform.

This project not only improves affordability and sustainability on campus but also fosters a sense of community by encouraging students to interact, trade, and help each other.

5.2 Future Work

While the current implementation of the UniTrade application provides a solid foundation, there are several areas for future enhancement and many features that need to be implemented in our project before it operates in real world:

- Real-Time Chat System - Implement socket-based or Firebase real-time chat for a seamless messaging experience.

- Recommendation Engine - Use machine learning to suggest items based on user preferences and search history.
- Payment Gateway Integration - Allow in-app payments through Razorpay, Stripe, or UPI for faster transactions.
- Delivery Coordination Tools - Add features like pickup scheduling, in-app maps for on-campus exchange points.
- Multi-University Support - Expand the platform to support multiple universities with campus-specific filters and logins.
- Advanced Moderation Tools - AI-driven moderation to detect inappropriate listings or spam.
- Dark Mode & Accessibility Features - Enhance UI for better user experience and inclusivity.
- Progressive Web App (PWA) - Extend functionality to web browsers for broader accessibility.

References

- Flutter(Frontend): https://reactnative.dev/
- Python(Backend) - https://www.w3schools.com/python/
- MySQL(database) - https://dev.mysql.com/doc/
- Stackoverflow : https://stackoverflow.com/ For errors
- Firebase google login: https://firebase.google.com/docs/auth/web/google-signin