

# Campus Marketplace

CSIS – 4495 – 050

Applied Research Project

Professor: Bambang Sarif

Kenneth Rofuli – 300392382

## Introduction

Online marketplaces are part of everyday life, with platforms like Facebook Marketplace, Craigslist, and Kijiji widely used to buy, sell, or trade almost anything. However, these platforms are broad and not tailored for student communities. This often creates challenges such as interacting with strangers outside the campus, facing security concerns, or dealing with unreliable listings.

The goal of this project is to create a student-only mobile marketplace that focuses on the unique needs of college and university students. A mobile-first design ensures accessibility and convenience, as students primarily rely on smartphones for communication and transactions. The app provides a secure, campus-verified environment where students can buy and sell items such as textbooks, electronics, furniture, or dorm essentials.

### Main Research Question:

*How can we build a secure, student-focused mobile marketplace that combines affordability, convenience, and safety?*

This project assumes that by implementing secure login, campus-based user verification, real-time chat, and push notifications, peer-to-peer transactions can be made safer and more efficient. Beyond financial savings, the system aims to strengthen community trust and foster a more connected student body.

## Proposed Research Project

### Research Design and Objectives

The research follows a Design Science Research (DSR) methodology, focusing on building and evaluating a working prototype that addresses real-world challenges in student commerce. The mobile application will undergo iterative development, testing, and refinement based on user feedback.

#### Objectives:

1. Design a secure and user-friendly mobile app for student-to-student transactions within a single campus network.
  2. Implement a real-time chat system that enables communication between buyers and sellers.
  3. Integrate cloud-based image storage for posting and retrieving product photos efficiently.
  4. Conduct usability testing to measure trust, ease of use, and satisfaction among students.
  5. Produce a functional prototype demonstrating how mobile-first solutions enhance safety and accessibility in campus commerce.
- 

### Detailed Functionality Description

1. User Authentication and Profiles
  - Students create accounts using verified campus emails.
  - Profiles include name, email, campus, avatar, and optional bio.
  - Secure authentication handled via JWT tokens.
  - Only verified users can access the marketplace.

## 2. Listings Management

- Users can create, edit, and delete listings.
- Listings include title, description, price, category, and image.
- All images are uploaded and stored securely in the cloud (Firebase/Cloudinary).
- A built-in search and filter system allows users to browse by category or price range.

## 3. Real-Time Chat System

- Uses Socket.IO for instant communication between buyers and sellers.
- Displays unread message counts and provides notification badges for new messages.
- Enables students to negotiate and discuss item details within the app safely.

## 4. Push Notifications

- Implemented through Firebase Cloud Messaging (FCM).
- Alerts users about new messages, interest in their listings, or completed transactions.

## 5. Safety and Trust Features

- All users are authenticated through campus email domains.
- Reporting and moderation system (future enhancement) to flag inappropriate listings or users.
- No external buyers/sellers allowed, ensuring a closed community environment.

## 6. Admin Dashboard (Basic)

- A minimal web-based dashboard for viewing users and listings.
- Helps in managing platform activity and monitoring user reports.

## Methodology

### 1. System Development

- Prototype built with React Native for cross-platform mobile deployment (Android & iOS).
- Backend implemented using Node.js and Express.js.
- Data stored securely in MongoDB Atlas with proper indexing and schema validation.
- Real-time chat and notifications integrated using Socket.IO and Firebase Cloud Messaging.

### 2. Data Collection

- Participants: 15–20 student volunteers from the same campus.
- Process: Students will register, post listings, and complete at least one simulated transaction.
- Data Collected:
  - Quantitative: number of interactions, response times, and transaction success rate.
  - Qualitative: user satisfaction, safety perception, and ease of navigation.

### 3. Data Analysis

- Quantitative data analyzed through descriptive statistics (means, completion rates).
- Qualitative data analyzed using thematic coding from survey responses and interviews.

### 4. Justification

This mixed-method approach ensures that both the technical functionality and the user experience are evaluated. It focuses not only on building a working app but also understanding how students perceive its usefulness and trustworthiness.

## Technologies

Component	Technology Used	Purpose
Frontend	React Native	Cross-platform mobile UI
Backend	Node.js + Express.js	REST API & app logic
Database	MongoDB Atlas	Cloud-based data storage
Real-Time	Socket.IO	Chat and live notifications
Authentication	JWT	Secure login sessions
Cloud Storage	Cloudinary or Firebase	Image uploads

## Expected Results

By the end of the project, a functional mobile prototype will demonstrate a proof of concept for a secure and community-based campus marketplace. Students will be able to:

1. Create verified accounts and edit their profiles.
2. Post, update, or delete listings with images.
3. Search and filter items easily.
4. Engage in one-on-one real-time chat with peers.
5. Receive push notifications for key activities.

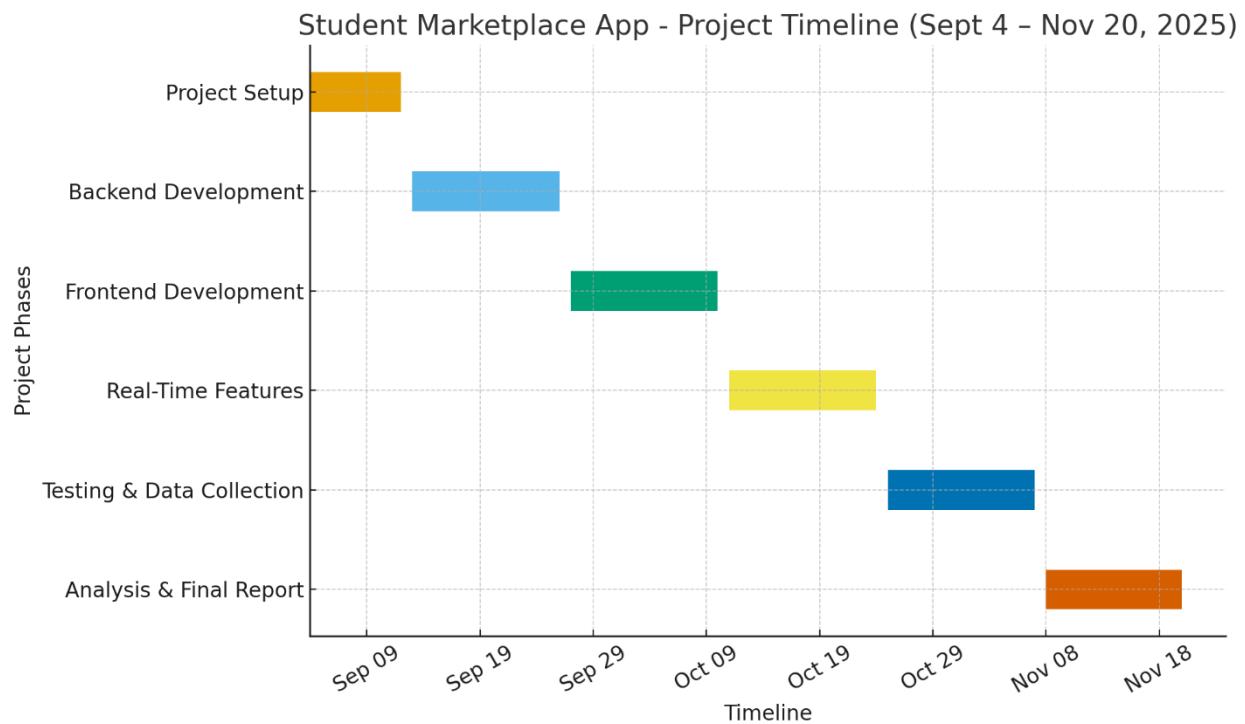
## Anticipated Contributions:

- Academic: Understanding how trust and usability affect adoption of mobile-first student marketplaces.
- Practical: A deployable prototype that can serve as a model for future campus-oriented e-commerce systems.

## Project Plan and Detailed Deliverables

Phase	Dates	Key Tasks	Deliverables
1. Setup	Sept 4–12	Finalize project scope, create GitHub repo, configure React Native + backend tools.	Working environment ready, GitHub repo initialized, project documentation started.
2. Backend Development	Sept 13–26	Build REST APIs for authentication, profiles, and listings.	Functional backend with database integration (users, listings, messages).
3. Mobile Frontend	Sept 27–Oct 10	Implement login, registration, profile management, and listing pages.	Usable front-end connected to backend APIs; users can register, log in, and view/post listings.
4. Real-Time Features	Oct 11–24	Integrate Socket.IO chat and FCM notifications.	Fully functional chat and notification system with unread count and alerts.
5. Testing	Oct 25– Nov 7	Conduct usability tests and surveys with student volunteers.	Collected feedback and usability data; bug fixes applied.
6. Analysis & Final Report	Nov 8–20	Analyze data, compile findings, finalize report and presentation.	Completed final paper, presentation slides, and live demo prototype.

Figure 1. 1 Project Timeline



## Implemented Features

### Introduction

The Campus Marketplace App was developed to provide students with a secure and user-friendly platform to buy, sell, and manage items within their campus community. The system combines a modern mobile interface with a robust backend infrastructure, ensuring both usability and reliability. Below is an overview of the key features implemented, categorized by functionality.

### Authentication System

The application features a secure and modern authentication system. Users can register by providing their name, email, password, and campus selection. Passwords are encrypted using **bcryptjs** for enhanced protection. Login and registration utilize **JWT (JSON Web Tokens)** for secure and verifiable user sessions. Tokens are stored safely on the device using **Expo Secure Store**, allowing users to remain logged in between app launches. The system also implements automatic logout upon token expiration to prevent unauthorized access and maintain data integrity.

```
▶   _id: ObjectId('68e621999edc4341000920e3')
    name : "Test1"
    email : "test1@test.com"
    password : "$2b$10$Ylk.tIpNzhvVwh4rMFV4nu9IH7vbnVTPRMZn6AOPRVhMZFwaoAhaG"
    campus : "Douglas"
    createdAt : 2025-10-08T08:32:25.784+00:00
    updatedAt : 2025-10-08T08:32:25.784+00:00
    __v : 0

    _id: ObjectId('68eb37d843672945ac2cc8e7')
    name : "Ken Test"
    email : "test@test.com"
    password : "$2b$10$bl7tSoVNLz5eq2DkuRiz6.nw4L4ahwM6owCVpmJ.edaBk6aGlQoZW"
    campus : "Douglas"
    favorites: Array (1)
    createdAt : 2025-10-12T05:08:40.322+00:00
    updatedAt : 2025-10-12T05:09:04.234+00:00
    __v : 1
```

Figure 2 – Database Entry for Users with encrypted passwords

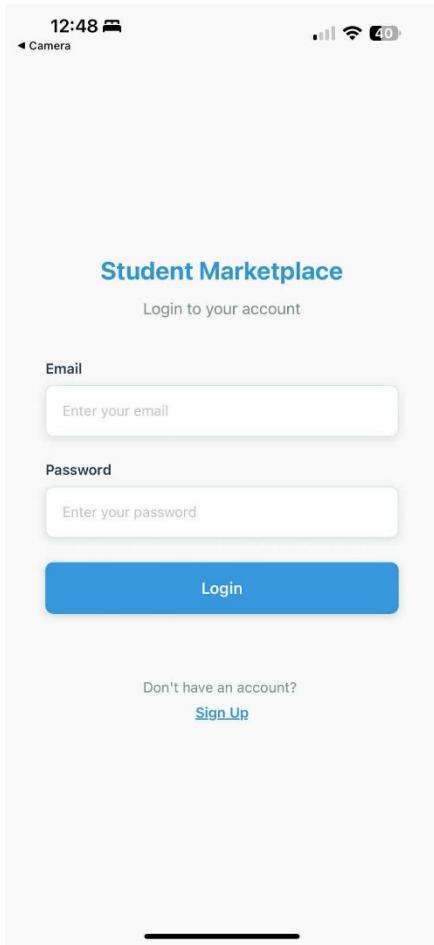


Figure 3 – Login Screen

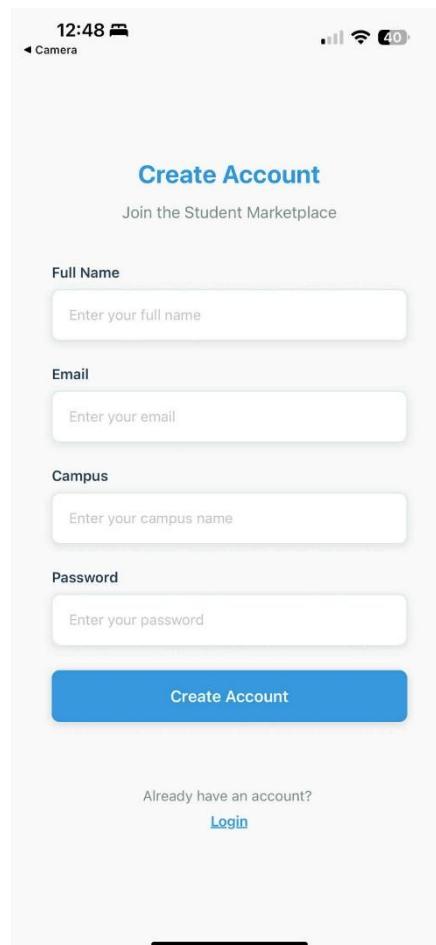


Figure 4 – Registration Screen

```
LOG 🔒 Attempting login to: https://studentmartketplace-backend.onrender.com/api/auth/login
LOG 🌐 Email: test@test.com
LOG ✅ Login successful: {"token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjY4ZWIZN2Q4NDM2NzI5NDVhYzJjYzhlnNyIsImhdCI6MTc2MTIwNjg3NSwiZXhwIjoxNzYxMjkzMjc1fQ.Ya8_1njvQgqTDXitdnKph7tQtZsVs4L52TXaBNu3w", "user": {"_v": 1, "_id": "68eb37d843672945ac2cc8e7", "campus": "Douglas", "createdAt": "2025-10-12T05:08:40.322Z", "email": "test@test.com", "favorites": ["68eb379443672945ac2cc8e4"], "name": "Ken Test", "password": "$2b$10$b17tSoVNLz5eq2DkuRiz6.nw4L4ahwM6owCVpmJ.edaBk6aGlQoZW", "updatedAt": "2025-10-12T05:09:04.234Z"}}

LOG 🔑 Token stored successfully
LOG ✅ User logged in: {"_v": 1, "_id": "68eb37d843672945ac2cc8e7", "campus": "Douglas", "createdAt": "2025-10-12T05:08:40.322Z", "email": "test@test.com", "favorites": ["68eb379443672945ac2cc8e4"], "name": "Ken Test", "password": "$2b$10$b17tSoVNLz5eq2DkuRiz6.nw4L4ahwM6owCVpmJ.edaBk6aGlQoZW", "updatedAt": "2025-10-12T05:09:04.234Z"}
```

Figure 5 – Console logs that tells us that login is successful

## **Listing Management**

Users can create, edit, and delete listings with essential details such as title, description, price, category, and images. A dedicated **My Listings** screen enables users to manage their own posts efficiently. Images are uploaded and stored via **Cloudinary**, ensuring fast and reliable cloud storage. The app integrates both **camera** and **gallery** options, allowing users to capture or select images directly. Confirmation prompts are included to avoid accidental deletions or modifications, ensuring a smooth and safe listing management experience.

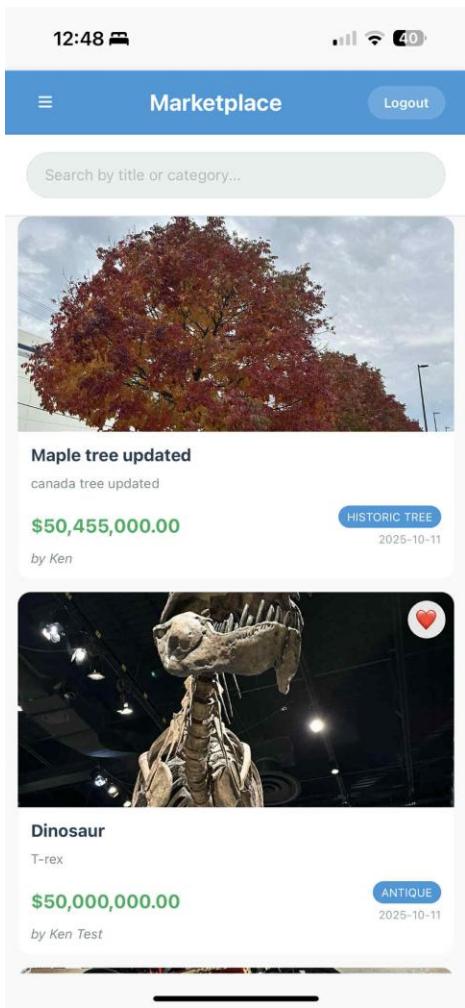


Figure 6 – Main Dashboard

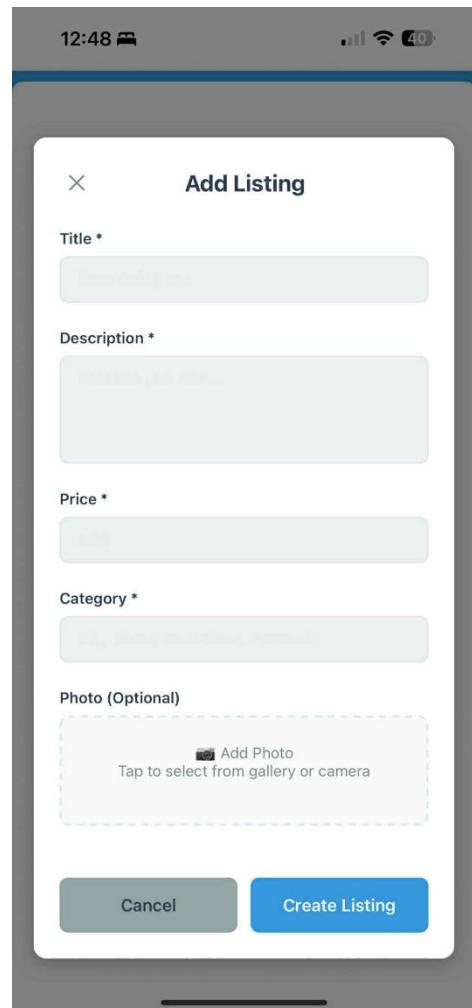


Figure 7 – Add Listing Modal

```
LOG 📸 Uploading image to Cloudinary...
LOG 📸 Starting Cloudinary upload...
LOG 📸 Cloudinary response: {"asset_folder": "", "asset_id": "e7f60e63aac1c2b364f9524ec8dbd654", "bytes": 39100, "created_at": "2025-10-23T08:10:16Z", "display_name": "listing", "etag": "4a549e0c6b724e2c897d9dd9f29581b3", "format": "jpg", "height": 965, "original_filename": "listing", "placeholder": false, "public_id": "e1t0glbjсхz6tehs7i0", "resource_type": "image", "secure_url": "https://res.cloudinary.com/dyxssxxhc/image/upload/v1761207016/e1t0glbjсхz6tehs7i0.jpg", "signature": "f603875c18e4b734a3af2dc733e38ab823d88cc9", "tags": [], "type": "upload", "url": "http://res.cloudinary.com/dyxssxxhc/image/upload/v1761207016/e1t0glbjсхz6tehs7i0.jpg", "version": 1761207016, "version_id": "6fc99a3d8683c4ddd0a03b383776b22", "width": 1287}
LOG ✅ Cloudinary upload successful: https://res.cloudinary.com/dyxssxxhc/image/upload/v1761207016/e1t0glbjсхz6tehs7i0.jpg
LOG ✅ Image uploaded successfully
LOG 📸 Creating listing: {"category": "Heavenly bodies", "cloudinaryId": "e1t0glbjсхz6tehs7i0", "description": "Constellation", "imageUrl": "Image URL attached", "price": 500000, "title": "Stars"}
```

Figure 8 - Console logs that tells us that creating a listing is successful

## Favorites System

The favorites feature allows users to save listings they are interested in by tapping a heart icon. Favorited listings are stored persistently and can be accessed through a dedicated **Favorites Screen**. The system provides real-time updates, instantly reflecting changes to favorites. Users are also restricted from favoriting their own listings, maintaining authenticity and fair user engagement across the marketplace.

## User Interface

The app's interface follows a clean, consistent, and responsive design that adapts to all device sizes. Listings are presented in professional card layouts, providing essential information at a glance. Each card supports interactive options such as “Message Seller” or “Report Listing” through a modal interface. The design emphasizes usability, smooth transitions, and appropriate touch feedback, resulting in an intuitive and visually cohesive user experience.

## Performance

The app has been optimized for high performance and reliability. Using **FlatList** ensures efficient rendering of large datasets with smooth scrolling and minimal memory usage. **Image optimization** and **API caching** significantly reduce load times and redundant requests. These measures contribute to fast navigation, low latency, and a stable experience even when handling numerous listings.

## Technical Infrastructure

The backend architecture is built using a **RESTful API** developed with **Express.js**, with data stored securely in a **MongoDB** database. The backend is deployed on **Render**, providing cloud-based scalability and reliability. Proper **CORS configuration** enables safe cross-origin communication between the mobile client and the server. Additional technical safeguards include comprehensive error handling and **Axios interceptors**, which automatically attach JWT tokens to API requests, ensuring secure and efficient data exchange.

## Screen Navigation

The app provides an organized navigation structure that allows users to move easily between key sections: **Dashboard** (browse all listings), **Favorites** (view saved items), **My Listings** (manage user posts), and **Login/Register** (authentication). Listing creation and editing are handled through modals, allowing users to perform these actions without leaving their current screen. This structure promotes a seamless and user-friendly workflow.

## Security Features

Security was prioritized throughout the entire development process. Passwords are hashed using **bcryptjs**, and all user actions requiring authentication are protected by **JWT tokens**. Sensitive data such as authentication tokens are securely stored using **encrypted storage** mechanisms on the device. Additionally, **server-side validation** is enforced to filter invalid or malicious inputs, ensuring data accuracy and preventing potential security threats. Together, these measures guarantee a high level of data protection and system integrity.

## Conclusion

Overall, the Campus Marketplace App demonstrates a strong integration of security, usability, and performance. From authentication and data management to interface design and system efficiency, each component has been developed with a clear focus on reliability and user experience. The combination of modern development practices, cloud-based infrastructure, and secure architecture ensures that the platform operates efficiently and safely for all users within the campus community.

## Moving Forward

Looking ahead, the next phase of development will focus on enhancing interactivity and community trust within the platform. Planned improvements include implementing a basic reporting system for listings, allowing users to flag inappropriate or misleading content. Additionally, a “Message Seller” functionality will be introduced to facilitate direct communication between buyers and sellers, creating a more dynamic and user-engaged marketplace experience.

## Planned Features

Further development goals include:

- Real-time messaging between users to support instant communication and improve transaction efficiency.
- Advanced search and filter options to help users find listings more accurately based on category, price range, or campus.
- User profiles and ratings to promote accountability and transparency within the community.
- Push notifications to alert users of new messages, updates, or listing activities in real time.

These planned features aim to elevate the platform’s interactivity, scalability, and overall user satisfaction, positioning the Campus Marketplace App as a comprehensive and trustworthy solution for campus-based buying and selling.

## Project Contract

Project Title: Campus Marketplace

Developer: Kenneth Rofuli

Course: CSIS 4495 – 050

Instructor: Bambang A.B. Sarif

Date: September 15, 2025

### Scope:

This contract outlines the development of a cross-platform mobile application designed to facilitate student-to-student buying and selling within a verified campus environment. The project will include a secure login system, listings management, real-time chat, and notification features.

### Responsibilities:

- The developer (Kenneth Rofuli) is solely responsible for all design, coding, testing, and documentation.
- The instructor will provide project oversight, feedback, and grading based on milestones.

### Deliverables:

1. Functional prototype demonstrating all key features.
2. Final written report and slide presentation.
3. Work log with dated progress entries.

### Signatures:

Developer: Kenneth Rofuli

Date: Sept. 11, 2025

## Work Log Table

Student Name: Kenneth (Solo Project)

Date	Hours	Description of Work Done
Sept 4, 2025	2	Initial research on student marketplaces; drafted introduction.
Sept 5, 2025	1.5	Set up GitHub repo with README and .gitignore; installed Node.js & MongoDB.
Sept 7, 2025	2	Designed ER diagram (Users, Listings, Messages); created schema draft.
Sept 9, 2025	2.5	Implemented Express.js server and tested MongoDB connection.
Sept 11, 2025	2	Added JWT authentication routes (register, login).
Sept 14, 2025	2.5	Created User profile model (name, email, campus, avatar).
Sept 16, 2025	3	Built CRUD APIs for listings (title, description, price, category, photo).
Sept 19, 2025	2	Tested backend APIs with Postman; fixed validation errors.
Sept 22, 2025	3	Started React frontend setup with Vite + Tailwind; created login/register forms.
Sept 25, 2025	2	Built profile & listing components; connected frontend to backend APIs.
Oct 1, 2025	2.5	Developed search and filter functionality for listings.
Oct 5, 2025	3	Added Socket.IO server setup and tested basic chat messages.
Oct 9, 2025	2	Implemented unread message count on chat UI.
Oct 13, 2025	2	Added notification bell and connected it to new message events.
Oct 17, 2025	3	Conducted initial system test (login → listing → chat); documented bugs/fixes.
Oct 22, 2025	2	Recruited test users; prepared survey form for feedback.
Oct 27, 2025	2.5	Collected first group feedback; noted UI flow issues.

Date	Hours	Description of Work Done
Nov 1, 2025	2	Continued user testing and interviews.
Nov 5, 2025	2	Performed data analysis (quantitative + thematic).
Nov 10, 2025	3	Drafted findings and results section.
Nov 14, 2025	2.5	Completed final research paper draft; designed slides.
Nov 18, 2025	2	Final review of prototype and testing.
Nov 20, 2025	1.5	Submitted final report, app, and presentation.

---

## References

- React Native. (n.d.). *Documentation*. Meta Open Source. Retrieved September 2025, from <https://reactnative.dev/>
- MongoDB. (n.d.). *MERN Stack Documentation*. MongoDB Inc. Retrieved September 2025, from <https://www.mongodb.com/mern-stack>
- Express.js. (n.d.). *Express — Node.js web application framework*. OpenJS Foundation. Retrieved September 2025, from <https://expressjs.com/>
- Socket.IO. (n.d.). *Documentation*. Retrieved September 2025, from <https://socket.io/docs/>
- Cloudinary. (n.d.). *Cloudinary Image and Video API Documentation*. Cloudinary Ltd. Retrieved September 2025, from <https://cloudinary.com/documentation>
- Expo Secure Store. (n.d.). *Securely store key-value pairs locally*. Expo. Retrieved September 2025, from <https://docs.expo.dev/versions/latest/sdk/securestore/>
- bcryptjs. (n.d.). *Password hashing library for Node.js*. npm. Retrieved September 2025, from <https://www.npmjs.com/package/bcryptjs>

- JSON Web Tokens (JWT). (n.d.). *Introduction to JSON Web Tokens*. Retrieved September 2025, from <https://jwt.io/introduction>
- Axios. (n.d.). *Promise-based HTTP client for the browser and Node.js*. npm. Retrieved September 2025, from <https://www.npmjs.com/package/axios>
- CORS. (n.d.). *CORS — Cross-Origin Resource Sharing middleware for Express.js*. npm. Retrieved September 2025, from <https://www.npmjs.com/package/cors>
- Render. (n.d.). *Cloud Application Hosting for Developers*. Render. Retrieved September 2025, from <https://render.com/docs>
- Firebase. (n.d.). *Cloud Messaging Overview*. Google. Retrieved September 2025, from <https://firebase.google.com/docs/cloud-messaging>