
Software Requirements and Design Document

for

COMMUNE

Prepared by: Aamna Saeed

Meesam E Tammar

Tashfeen Hassan

Fast NUCES

26/11/24

Table of Contents

Table of Contents	ii
1. Introduction.....	1
1.1 Purpose.....	1
1.2 Product Scope.....	1
1.3 Title.....	1
1.4 Objectives	1
1.5 Problem Statement	2
2. Overall Description.....	2
2.1 Product Perspective.....	2
2.2 Product Functions	3
2.3 List of Use Cases:	3
2.4 Extended Use Cases:	4
2.5 Use Case Diagram:	20
3. Other Nonfunctional Requirements.....	21
3.1 Performance Requirements	21
3.2 Safety Requirements.....	21
3.3 Security Requirements.....	21
3.4 Software Quality Attributes.....	22
3.5 Business Rules	22
3.6 Operating Environment	23
3.7 User Interfaces	23
4. Domain Model	29
5. System Sequence Diagram	30
6. Sequence Diagram.....	31
7. Class Diagram.....	40
8. Component Diagram	40
9. Package Diagram.....	41
10. Deployment Diagram.....	42

1. Introduction

1.1 Purpose

Commune is a full-stack Java desktop application designed for local community interaction. This software represents the initial release of the application, focusing on providing a localized platform for neighborhood engagement, commerce, security, and information sharing. This SRS covers all the primary functionalities of the application, outlining its purpose, scope, and objectives in addressing the needs of local communities.

1.2 Product Scope

This project's scope is mainly focused on neighborhoods and localities. Binding people that share the same geographic location together and providing them opportunities and conveniences like buying and selling items, organizing and planning community events, maintaining neighborhood watch initiatives, and sharing local updates. This is a highly localized platform that caters specifically to community needs, unlike general social media networks.

1.3 Title

Commune: A Full-Stack Java Desktop Application for Local Community Interaction

1.4 Objectives

The primary objectives of Commune are:

1. **Fostering Community Engagement:** Create a virtual space for neighbors to interact, plan, and collaborate on events or activities.
2. **Facilitating Local Commerce:** Enable users to buy and sell goods or services within their locality, making it convenient and more trusted.
3. **Enhancing Security:** Provide a platform for organizing neighborhood watch programs, sharing safety updates, and reporting suspicious activities.
4. **Efficient Information Sharing:** Allow residents to post updates on important matters like local services, emergencies, and general neighborhood news.

1.5 Problem Statement

Modern neighborhoods often experience a disconnect compared to former times due to busy schedules and the lack of a localized platform to keep neighbors informed and engaged. Broad social media platforms, while widespread, fail to address community-level needs effectively. This results in:

- Limited avenues for neighbors to communicate or collaborate on common goals.
- A lack of trust and convenience in buying and selling items within the locality.
- Inadequate systems for sharing critical neighborhood safety information.

Commune aims to bridge this gap by creating a tailored digital platform. It will provide features that streamline local commerce, enable event organization, strengthen community ties, and enhance neighborhood security through coordinated watch programs. The application is designed to foster a sense of belonging and collaboration among neighbors, addressing the disconnect prevalent in modern society.

2. Overall Description

2.1 Product Perspective

Commune is a new, self-contained product developed to address the specific needs of local communities. It is not part of an existing product family but represents a unique solution tailored to modern neighborhood challenges. The application integrates several key functionalities into a cohesive platform, including:

- Localized e-commerce features.
- Event planning and organization tools.
- Security-focused modules for neighborhood watch programs.
- Efficient mechanisms for sharing critical information.

The system operates as a standalone desktop application but could integrate with other systems or services (e.g., email notifications or local directories) to enhance functionality.

2.2 Product Functions

The major functions of Commune include:

- **User Management:** Allow users to create accounts, manage profiles, and define roles (e.g., buyer, seller, and organizer).
- **Local Marketplace:** Facilitate buying and selling of goods or services within the community.
- **Event Planning:** Enable users to create, manage, and participate in community events.
- **Neighborhood Watch:** Provide tools for organizing and participating in security initiatives, including reporting suspicious activities.
- **Information Sharing:** Allow users to post updates, news, and alerts relevant to the community.
- **Search and Filter:** Include features for efficiently finding relevant items, events, or posts.
- **Notifications:** Deliver updates to users regarding new posts, events, or marketplace activities.

2.3 List of Use Cases:

- Register New User
- Buy Items
- Sell Items
- Contact People
- Organize Event
- Browsing Lost and Found Item
- Babysitting and Childcare
- Create Polls And Surveys
- Share Resources
- Organize Carpooling
- Join Carpooling
- Posting Alerts

2.4 Extended Use Cases:

2.4.1 Use Case 1: Register New User

Scope: Commune

Level: User goal

Primary Actor: Resident (New User)

Stakeholders and Interests:

New Users: Want a quick, simple, and secure registration process.

Admin: Ensures that only legitimate users join the platform and maintain system security.

Preconditions:

User has access to the platform (via web or app).

User has a verified email address and/or phone number.

Post conditions:

User is successfully registered and can use the platform's features, including posting, joining groups, and making requests.

Main Success Scenario:

1. User visits the registration page.
2. User enters required information (name, email, phone number, address).
3. System verifies the provided email address and address information.
4. System sends a confirmation email to the user.
5. User clicks the confirmation link in the email.
6. The system completes the registration process and grants the user access to the platform.

Extensions:

- 3a. Address is invalid.

3a1. System notifies the user of an incorrect Address.

3a2. User is prompted to correct and re-enter the address.

3b. Email is invalid.

3b1. System notifies the user of an incorrect Email.

3b2. User is prompted to re-enter the correct email.

5a. User does not confirm the email.

5a1. Registration remains incomplete until email confirmation.

5a2. System sends a reminder to the user after a specific time period (e.g. 24 hours).

5b. Email confirmation link expires.

5b1. System notifies the user and prompts them to request a new confirmation link.

2.4.2 Use Case 2: Buy Items

Use Case Name: Buy Items

Scope: Commune

Level: User Goal

Primary Actor: Resident (User)

Stakeholders:

Seller: wants to sell items quickly and safely.

Buyer: wants to buy items in good condition at a fair price.

Admin: Ensures secure transaction and user satisfaction.

Preconditions:

Buyer must be registered and logged in.

Items must not be marked as sold/unavailable.

Post conditions:

Item is successfully sold and marked sold/unavailable.

Main Success Scenario:

1. Buyer logs into the platform.
2. Buyer navigates to the "Buy Items" section.
3. Buyer browses available items.
4. Buyer selects an item to purchase.
5. Buyer contacts the seller to inquire about the item.
6. Buyer and seller agree on terms of sale (price, payment, delivery).
7. Buyer completes the purchase (payment is made).

Extensions:

4a. Item is no longer available.

4a1. Buyer receives a notification
that the item is sold.

5a. Buyer cannot reach seller.

5a1. Buyer is unable to contact

seller, the system notifies the buyer to try again later.

2.4.3 Use Case 3: Sell Items

Use Case Name: Sell Items

Scope: Commune

Level: User Goal

Primary Actor: Resident (User)

Stakeholders:

Seller: wants to sell items quickly and safely.

Buyer: wants to buy items in good condition at a fair price.

Admin: Ensures secure transaction and user satisfaction.

Preconditions:

Seller must be registered and logged in.

Seller has an item to post for sale.

Postconditions:

Item is successfully listed, purchased, and marked as sold or removed

Main Success Scenario:

1. Seller logs into the platform.
2. Seller navigates to the "Sell Items" section.
3. Seller creates a new listing by providing item details (title, description, price, condition, photos).
4. System verifies the listing details.
5. Seller submits the listing for buyers to view.
6. Seller negotiates with buyer (if necessary) and agrees to terms of sale.
7. Seller marks the item as unavailable once the sale is completed.

Extensions:

- 4a. Incomplete listing information.

- 5a. Item violates platform policies.
 - 4a1. System prompts the seller to complete the missing details.
 - 5a1. The system flags the listing and notifies the seller of the violation.
 - 7a1. Seller can relist the item or negotiate with a new buyer.
- 7a. Buyer backs out.

2.4.4 Use Case 4: Contact People

Scope: Commune

Level: User Goal

Primary Actor: Resident

Stakeholders:

Resident: Wants to contact neighbors or other users easily.

Admin: Ensures that communication features are working securely.

Preconditions:

User must be registered and logged in.

Postconditions:

Message is successfully sent or received by the intended recipient.

Main Success Scenario:

1. User logs into the platform.
2. User navigates to the "Contact People" section.
3. User selects or searches for the recipient.
4. User writes a message and sends it.
5. The recipient receives the message

and responds if necessary.

Extensions:

3a. Recipient not found.

3a1. System notifies the user
that the recipient is not
found.

5a. Message fails to send.

5a1. System prompts the
user to resend or
correct any errors.

2.4.5 Use Case 5: Organize Event

Scope: Commune

Level: User Goal

Primary Actor: Resident

Stakeholders:

Resident: Wants to organize or participate in neighborhood events.

Admin: Ensures events are organized smoothly and responsibly.

Preconditions:

Resident must be logged in and authorized to create events.

Postconditions:

Event is successfully created and published for other residents to view or attend.

Main Success Scenario:

1. Resident logs into the platform.
2. Resident navigates to the "Organize Event"

section.

3. Resident creates a new event by filling in
event details (name, date, location, description).

4. System verifies and
publishes the event for
other residents to view.

5. Other residents can RSVP to the event.

Extensions:

4a. Incomplete event details.

4a1. System prompts the
organizer to fill in
missing details.

5a. Event fails to publish.

5a1. System prompts the
organizer to retry.

2.4.6 Use Case 6: Browsing Lost and Found Item

Scope: Commune

Level: User Goal

Primary Actor: Resident

Stakeholders:

Resident: Wants to post or find lost items in the neighborhood.

Admin: Ensures that items are posted and tracked responsibly.

Preconditions:

Resident must be logged in.

Postconditions:

Item is successfully found or posted as lost.

Main Success Scenario:

1. Resident logs into the platform.
2. Resident navigates to the "Lost and Found" section.
3. Resident creates a new lost item post or searches for a found item.
4. System verifies and publishes the post.
5. Other residents view and respond to the post.
6. If found, return item to owner and post is taken down.

Extensions:

- 4a. Lost item not found.
 - 4a1. System notifies the user to keep searching or update the post.
- 5a. Post violates policies.
 - 5a1. System flags the post and notifies the poster.

2.4.7 Use Case 7: Babysitting and Childcare

Scope: Commune

Level: User Goal

Primary Actor: Resident

Stakeholders:

Parents (Resident): Want to find trusted babysitters/childcare providers.

Babysitters (Resident): Want to offer their services safely.

Admin: Ensures that childcare arrangements are done securely.

Preconditions:

Resident must be registered and logged in.

Postconditions:

Babysitting or childcare arrangement is successfully completed.

Main Success Scenario:

1. Parent or babysitter logs into the platform.
2. User navigates to the "Babysitting and Childcare" section.
- 3a. If Parent, then searches for babysitters.
- 3b. If babysitter, then posts their services.
4. System verifies the details.
5. Parent and babysitter agree on terms (time, payment).
6. Babysitting service is provided.

Extensions:

4a. Service provider not available.

4a1. System notifies the user that no
babysitters are available.

2.4.8 Use Case 8: Create Polls and Surveys

Scope: Commune

Level: User Goal

Primary Actor: Resident

Stakeholders:

Resident: Wants to create or participate in surveys.

Admin: Ensures survey quality and user participation.

Preconditions:

Resident must be registered and logged in.

Postconditions:

Survey is successfully created or completed by the user.

Main Success Scenario:

1. Resident logs into the platform.
2. Resident navigates to the "Polls/Survey" section.
3. Resident chooses to either create a new poll/survey or fill out an existing poll/survey.
- 4a. If creating, Resident enters poll/survey details (title, questions, options).
- 4b. If filling, Resident selects answers and submits the survey.
5. Survey is saved and published and if new, published for others to participate in.

Extensions:

4a. Incomplete survey creation.

4a1. System prompts the user to
complete missing survey
details.

5a. Survey submission fails.

5a1. System prompts the user to
resubmit the survey.

2.4.9 Use Case 9: Lend Tools/Share Resources

Scope: Commune

Level: User Goal

Primary Actor: Resident

Stakeholders:

Resident: Wants to borrow or lend tools/resources within the neighborhood.

Admin: Ensures the safe exchange and return of resources.

Preconditions:

Resident must be registered and logged in.

Lender must have items listed for lending.

Postconditions:

Resource/tool is successfully borrowed or lent, and returned after use.

Main Success Scenario:

1. Resident logs into the platform.
2. Resident navigates to the "Tool Lending"
section.
3. Resident selects a tool/resource to borrow
or lists a tool to lend.

4. System verifies and facilitates communication between lender and borrower.

5. Transaction is completed, and the item is exchanged.

6. The tool is returned, and the transaction is closed.

Extensions:

4a. Tool not available.

4a1. System notifies the borrower that the tool is unavailable.

6a. Tool is not returned.

6a1. System flags the issue and notifies the lender to take action.

2.4.10 Use Case 10: Organize Carpooling

Scope: Commune

Level: User Goal

Primary Actor: Resident

Stakeholders:

Resident: Wants to organize or join carpools for local transportation.

Admin: Ensures the safety and coordination of carpooling.

Preconditions:

Resident must be registered and logged in.

Post conditions:

Carpool is successfully organized or joined by residents.

Main Success Scenario:

1. Resident navigates to the "Organize Carpool" section.
2. Resident provides carpool details
(pickup points, timing, number of passengers).
3. System verifies and
publishes the carpool.
4. Other residents can join the carpool.

Extensions:

- 4a. Incomplete carpool details.
 - 4a1. System prompts the
user to complete
missing details.
- 5a. Carpool is full.
 - 5a1. System notifies the
resident that the
carpool is full and offers
alternatives.

2.4.11 Use Case 11: Join Carpool

Scope: Commune

Level: User Goal

Primary Actor: Resident

Stakeholders:

Resident: Wants to join a carpool for convenient transportation.

Carpool Organizer: Wants to fill available seats with neighborhood residents.

Admin: Ensures the safety and organization of carpools.

Preconditions:

Resident must be registered and logged in.

A carpool must be available for the resident to join.

Post conditions:

Resident is successfully added to the carpool and can participate in the carpool arrangement.

Main Success Scenario:

1. Resident logs into the platform.
2. Resident navigates to the "Carpool" section.
3. Resident browses available carpools or searches for a specific carpool.
4. Resident selects a carpool to join based on schedule and location.
5. System verifies the availability of seats in the carpool.
6. Resident joins the carpool and coordinates pickup and drop-off details with the organizer.

Extensions:

5a. Carpool is full.

5a1. System notifies the resident that the

carpool has no available seats.

5a2. System prompts resident to search for
alternative carpools.

6a. Carpool is canceled.

6a1. System notifies the resident that the
carpool is no longer available.

2.4.12 Use Case 12: Posting Alerts

Scope: Commune

Level: User Goal

Primary Actor: Neighborhood Watch Member

Stakeholders:

Watch Member: Wants to post neighborhood alerts for security purposes.

Residents: Want to stay informed about potential threats or important notifications.

Admin: Ensures alerts are posted responsibly.

Preconditions:

Watch member must be logged in.

Postconditions:

Alert is successfully posted, and residents are notified.

Main Success Scenario:

1. Watch member logs into the platform.
2. Watch member navigates to the "Watch Alerts"
section.
3. Watch member creates a new alert

with details (description, location, urgency).

4. System verifies and posts
the alert.

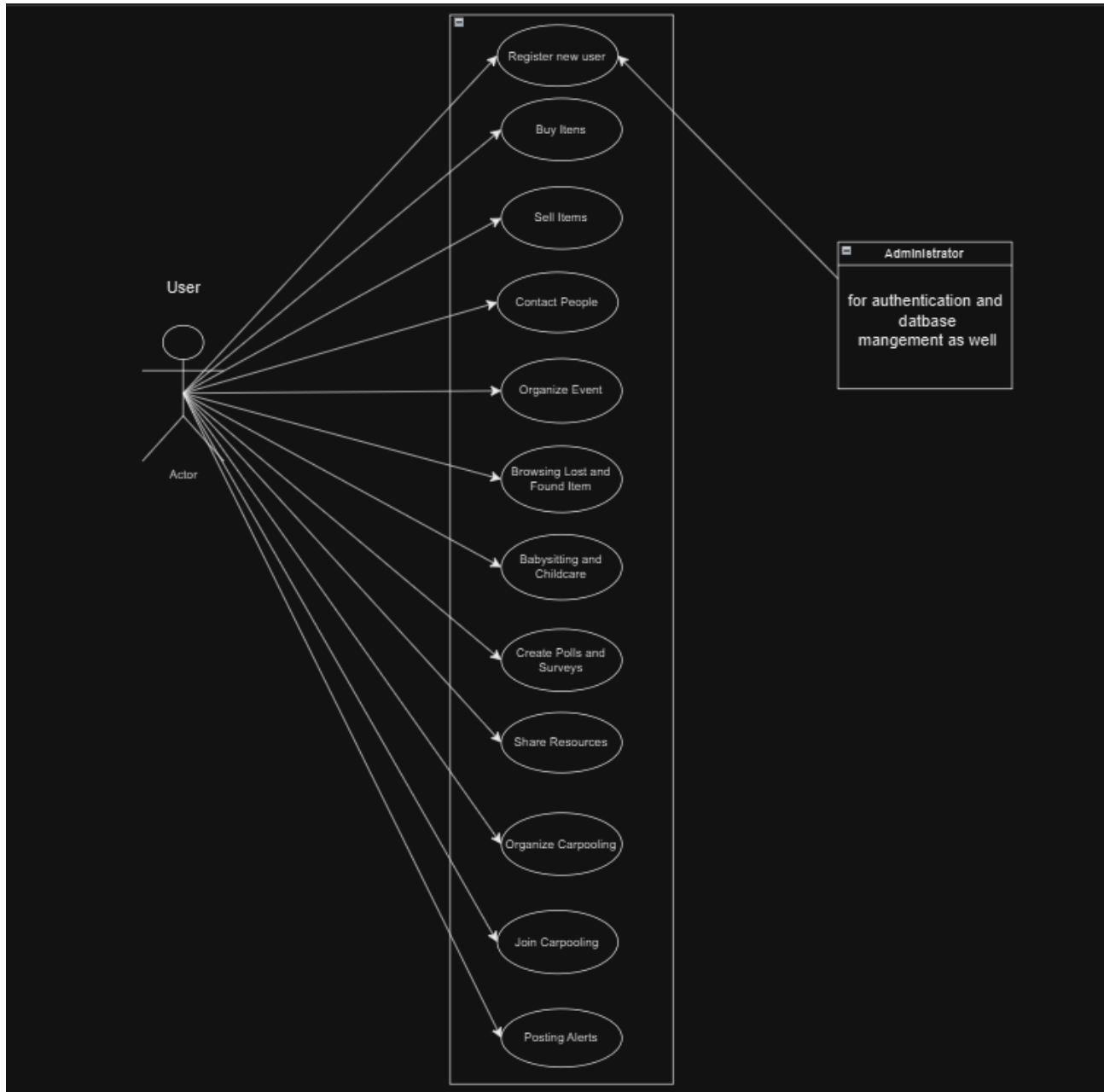
5. System sends a
notification to all
users about the alert.

Extensions:

4a. Alert fails to post.

4a1. System notifies the
watch member
to retry posting.

2.5 Use Case Diagram:



3. Other Nonfunctional Requirements

3.1 Performance Requirements

- The application must support at least 100 concurrent users without a noticeable degradation in performance.
- All actions, including searches, posting updates, and creating events, should complete within 2 seconds under normal usage.
- The system should efficiently handle data queries related to use cases like searching lost and found items or filtering marketplace posts.

3.2 Safety Requirements

- The application must prevent the loss of critical data, such as user profiles and community posts, by implementing regular automated backups.
- There should be safeguards to prevent accidental deletion of essential community content, such as double-confirmation dialogs for sensitive actions.
- The application should comply with local data protection regulations to avoid any legal risks related to misuse or mishandling of user information.

3.3 Security Requirements

- User authentication must use a secure method such as hashed and salted passwords.
- Sensitive data, including user information and community updates, should be encrypted during transmission and storage.
- The system should include role-based access control to restrict features (e.g., only event organizers can create events).
- Logs of all administrative actions must be maintained for auditing purposes.
- The application must comply with privacy regulations, such as GDPR or local equivalents, ensuring that user data is used responsibly and only with consent.

3.4 Software Quality Attributes

1. **Flexibility:** The software should be modular and easily configurable to adapt to different community requirements or extend functionalities in the future. New features, such as additional marketplace categories or event types, should be easy to integrate without significant rework.
2. **Availability:** The system should be highly reliable and available with minimal downtime. It must operate effectively during high-traffic periods, with a targeted uptime of 99.9% to ensure users can rely on its services at all times.
3. **Maintainability:** The codebase should follow clean coding practices, including adherence to design patterns and proper documentation. This ensures that future developers can easily understand, debug, and enhance the system. Regular updates and maintenance schedules should also be implemented.
4. **Adaptability:** The application should be adaptable to changes in user needs or technology trends. It must support updates to external dependencies, such as third-party libraries or APIs, without disrupting functionality.
5. **Reusability:** Core components, such as the user management system and notification module, should be designed for reusability. These modules could be repurposed for other projects or extended to support additional features within the Commune platform.

3.5 Business Rules

1. Administrator Role:

- Administrators have the authority to delete user accounts if necessary (e.g., due to violations of terms or inactivity).
- Administrators can manage and moderate content posted in the application, including removing inappropriate or irrelevant posts.
- Administrators can assign or revoke special permissions for users (e.g., event organizers or moderators).

2. User Role:

- Users have access to all primary functionalities, including buying and selling items, organizing events, participating in neighborhood watch programs, and sharing information.
- Users can create and manage their profiles, including setting preferences and updating contact information.

- Users must adhere to community guidelines and terms of service when interacting on the platform.

These rules ensure a structured and efficient system for managing user roles and responsibilities while maintaining a safe and productive community environment.

3.6 Operating Environment

The Commune application will operate in the following environment:

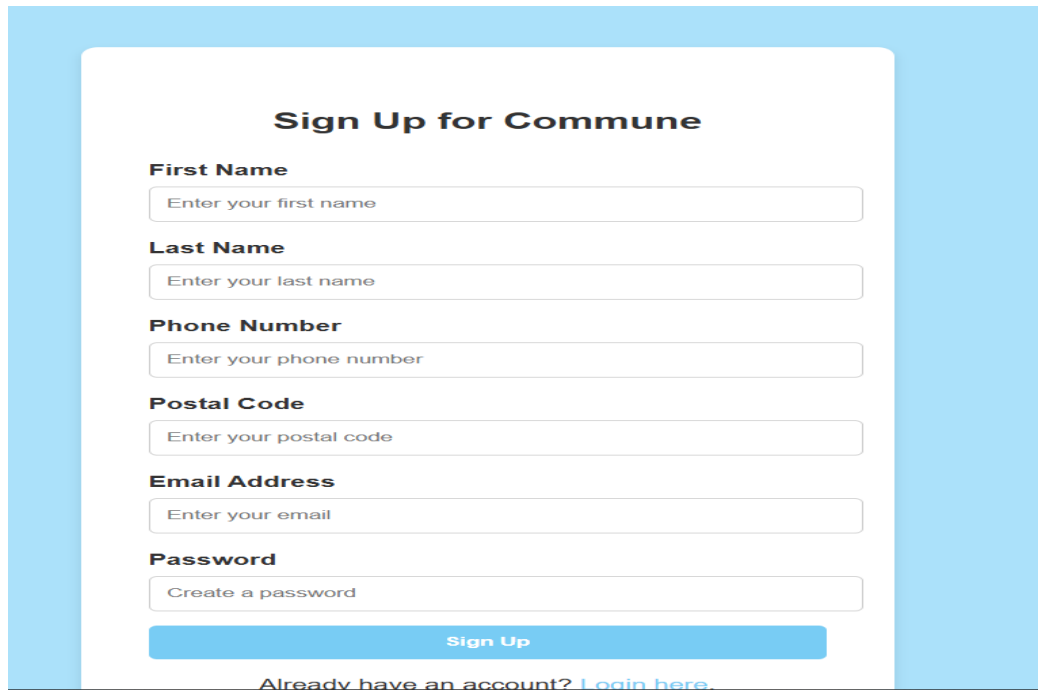
- **Hardware Platform:**
 - Minimum requirements: Dual-core processor, 4GB RAM, 500MB disk space.
 - Recommended: Quad-core processor, 8GB RAM, SSD storage.
- **Operating System:**
 - Supported on Windows 10 or later, macOS 11 or later, and major Linux distributions (e.g., Ubuntu 20.04 or later).
- **Software Dependencies:**
 - Java Runtime Environment (JRE) 11 or higher.
 - Database: MySQL 8.0 or equivalent relational database.
 - Web components: HTML, CSS, JavaScript (integrated using Thymeleaf templates).
- **Additional Tools:**
 - Requires an internet connection for some functionalities, such as fetching updates or sending notifications.

3.7 User Interfaces

The user interface of the Commune application is designed with a combination of **HTML**, **CSS**, and **JavaScript**, using the **Thymeleaf** template engine to render dynamic content. The interface is structured to be intuitive and user-friendly, adhering to modern design principles.

3.7.1.1 Characteristics of the User Interface:

Screen Layouts:



The image shows a 'Sign Up for Commune' form. It has a light blue header and a white body. The form fields are: First Name, Last Name, Phone Number, Postal Code, Email Address, and Password. Each field has a placeholder text. Below the fields is a blue 'Sign Up' button. At the bottom, there is a link: 'Already have an account? [Login here.](#)'

Sign Up for Commune

First Name
Enter your first name

Last Name
Enter your last name

Phone Number
Enter your phone number

Postal Code
Enter your postal code

Email Address
Enter your email

Password
Create a password

Sign Up

Already have an account? [Login here.](#)

3.7.2 Key Features of the Interface:

1. Input Fields:

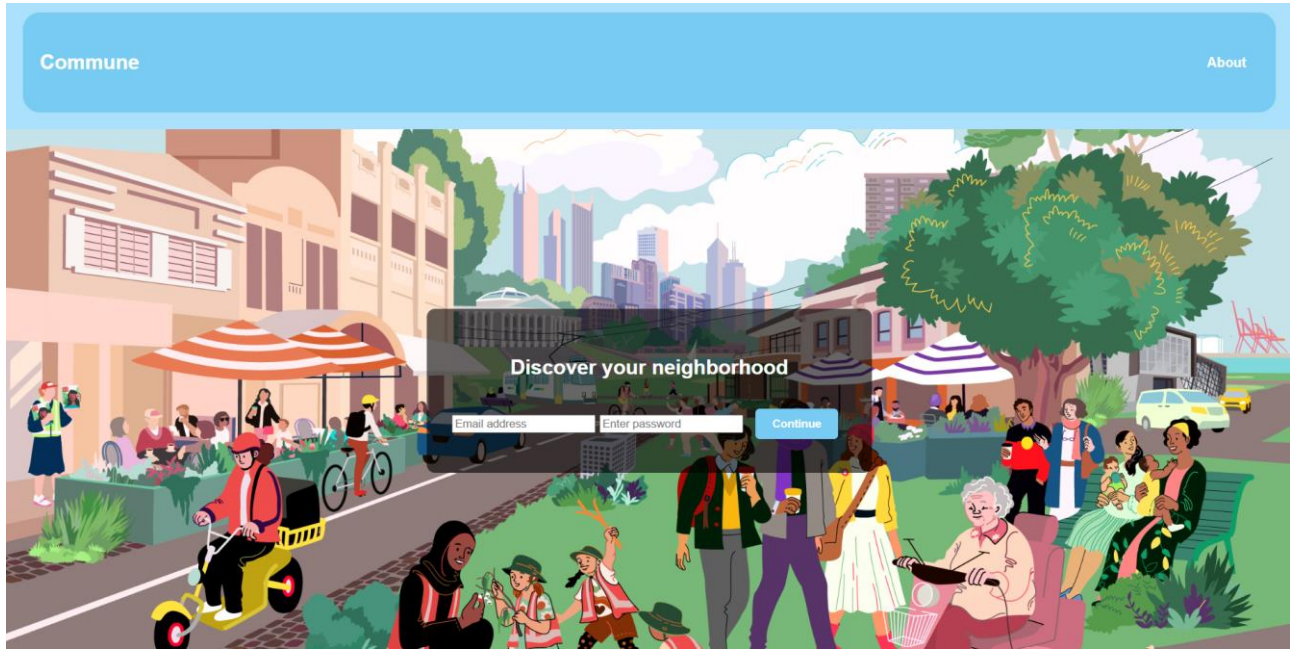
- **First Name and Last Name:** For the user's personal identification.
- **Phone Number:** Ensures contact information for communication or verification.
- **Postal Code:** Captures the user's locality for location-based services.
- **Email Address:** Used for login credentials and further communication.
- **Password:** Ensures security for the user's account.

2. Action Button:

- **Sign Up:** Submits the user's information to register them in the application.

3. Link for Existing Users:

- An option is provided to redirect existing users to the login page via the "**Already have an account? Login here**" link.



Explanation:

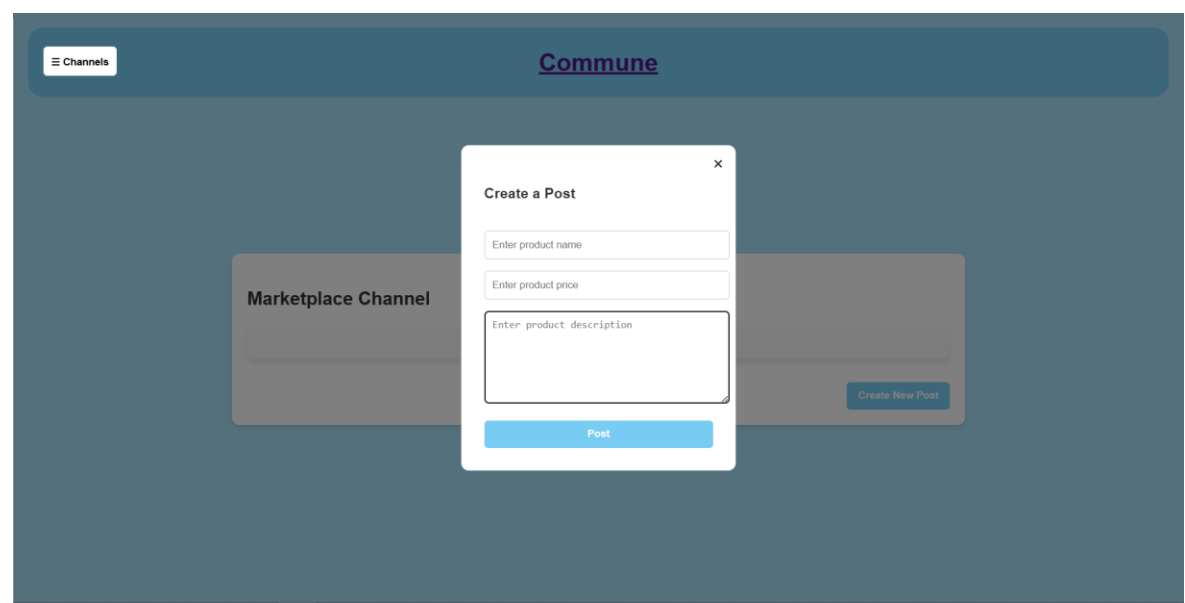
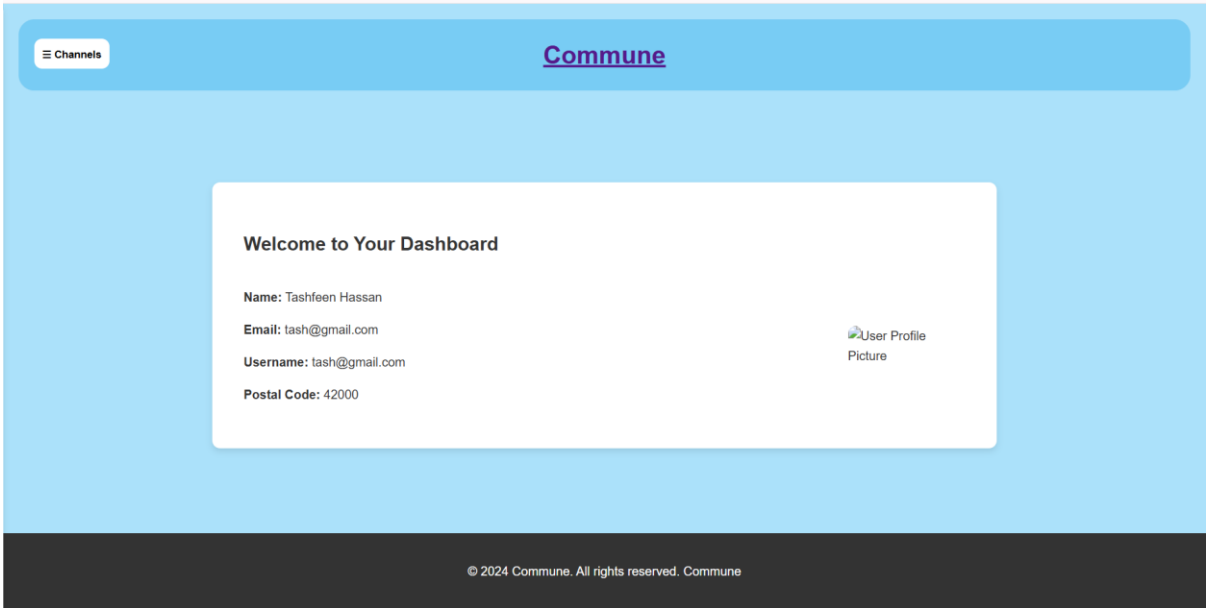
The phrase "**Discover your neighborhood**" invites users to explore and engage with their local community through the platform.

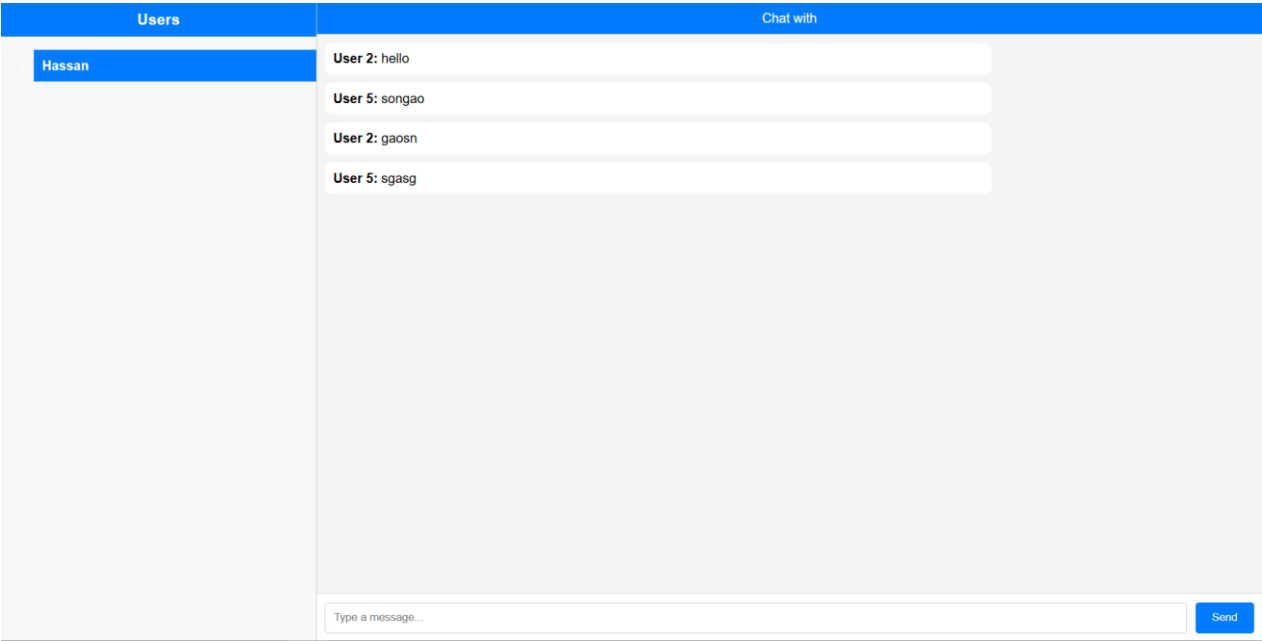
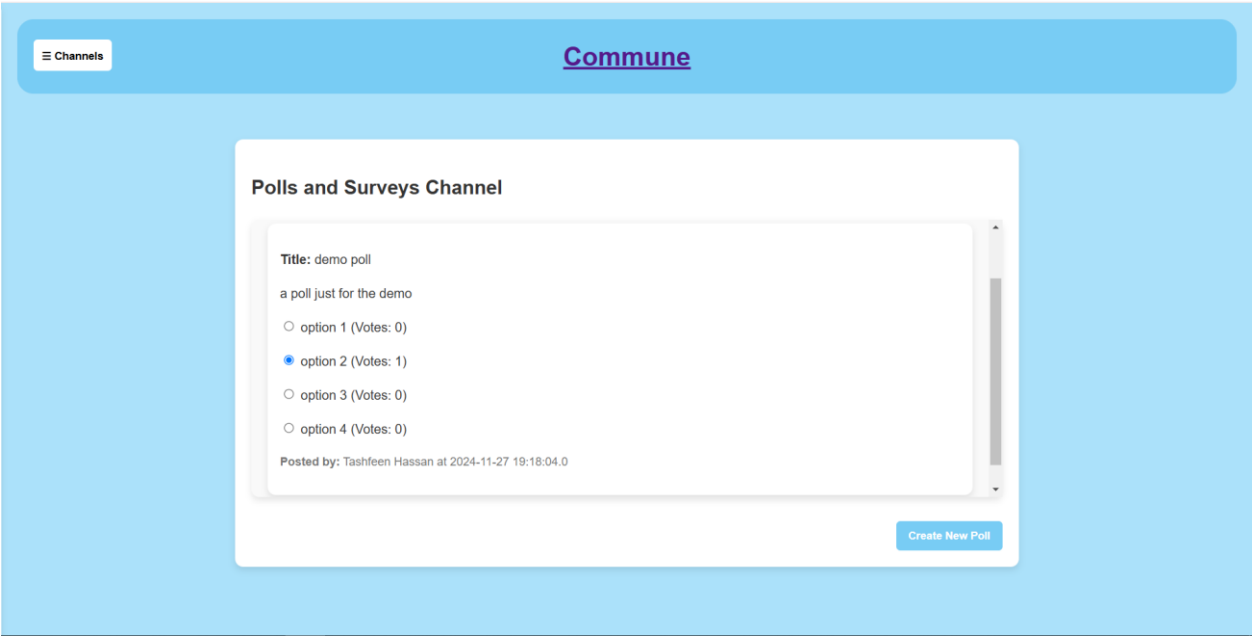
The screenshot displays a web interface for adding an administrator. At the top, a light blue header bar contains the word "Commune" on the left and "About" and "Login" links on the right. The main content area is a light blue gradient. Centered in this area is a white rectangular form titled "Add Admin". The form contains six input fields, each with a label above it and placeholder text inside: "First Name" (placeholder: "Enter your first name"), "Last Name" (placeholder: "Enter your last name"), "Phone Number" (placeholder: "Enter your phone number"), "Postal Code" (placeholder: "Enter your postal code"), "Email Address" (placeholder: "Enter your email"), and "Password" (placeholder: "Enter your password").

3.7.3 Key Features of the Interface:

1. Header Section:

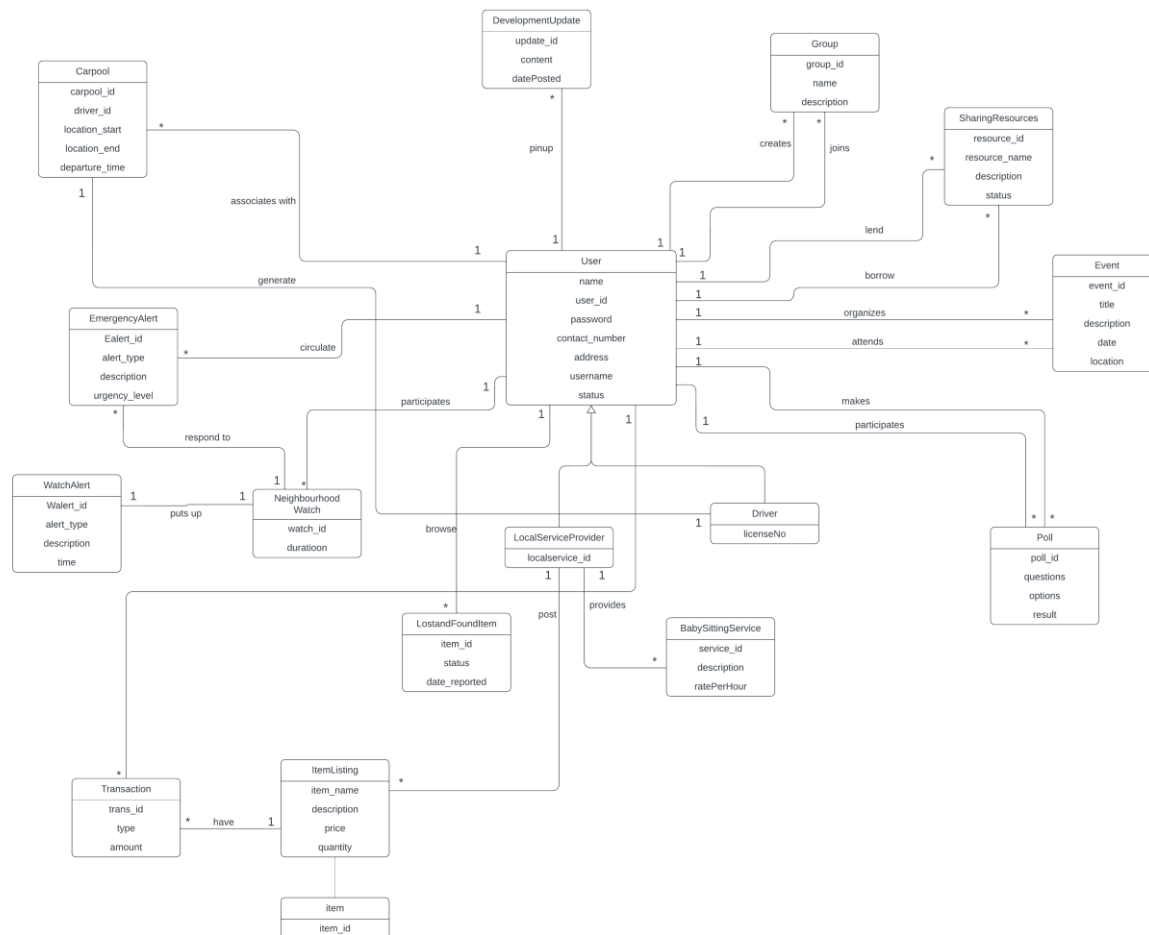
- The application name, "**Commune**", is displayed in the top-left corner, ensuring consistent branding.
- **Navigation Links:**
 - "**About**": Provides information about the platform.
 - "**Login**": Redirects to the login page for access to administrative or user functionalities.





4. Domain Model

DOMAIN MODEL

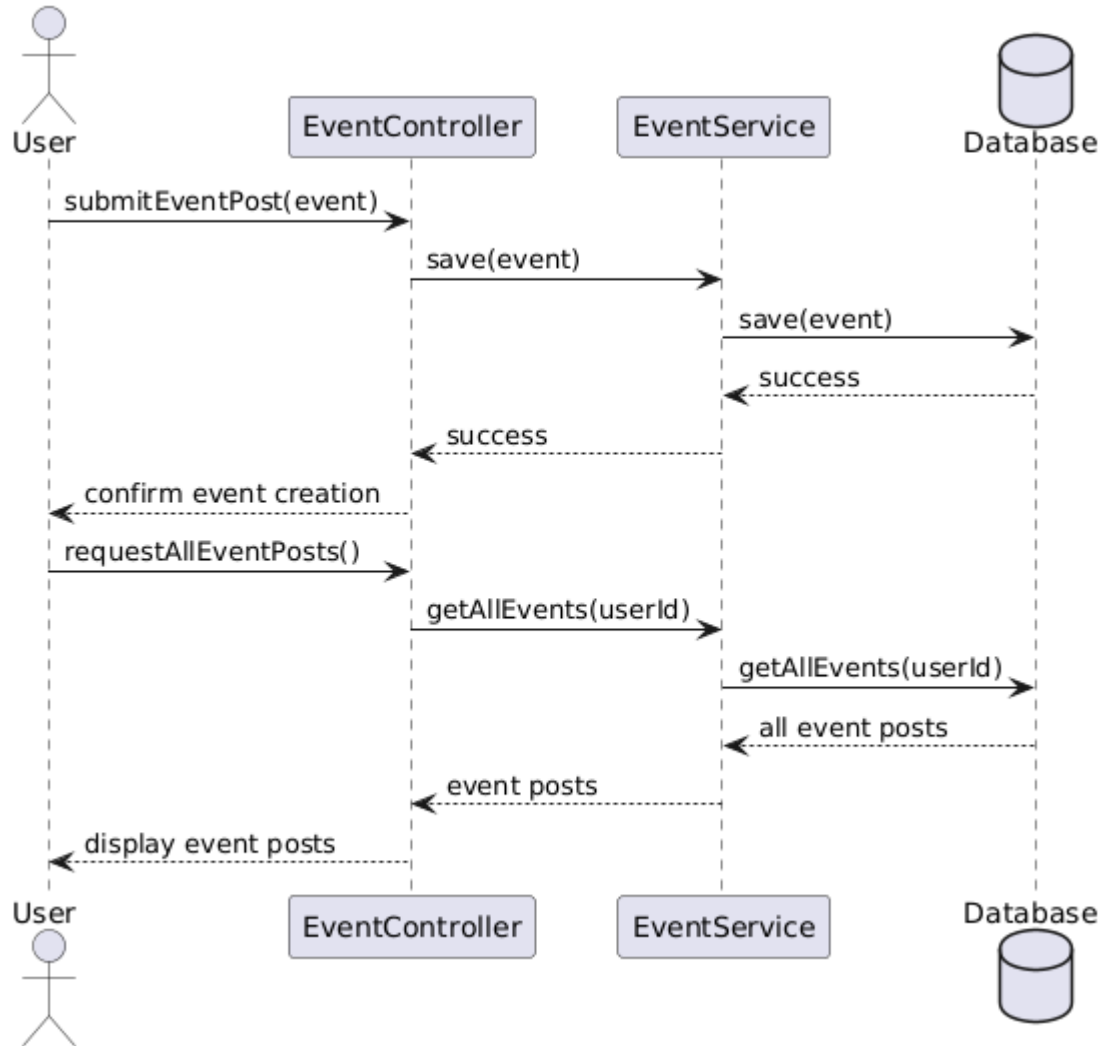


5. System Sequence Diagram

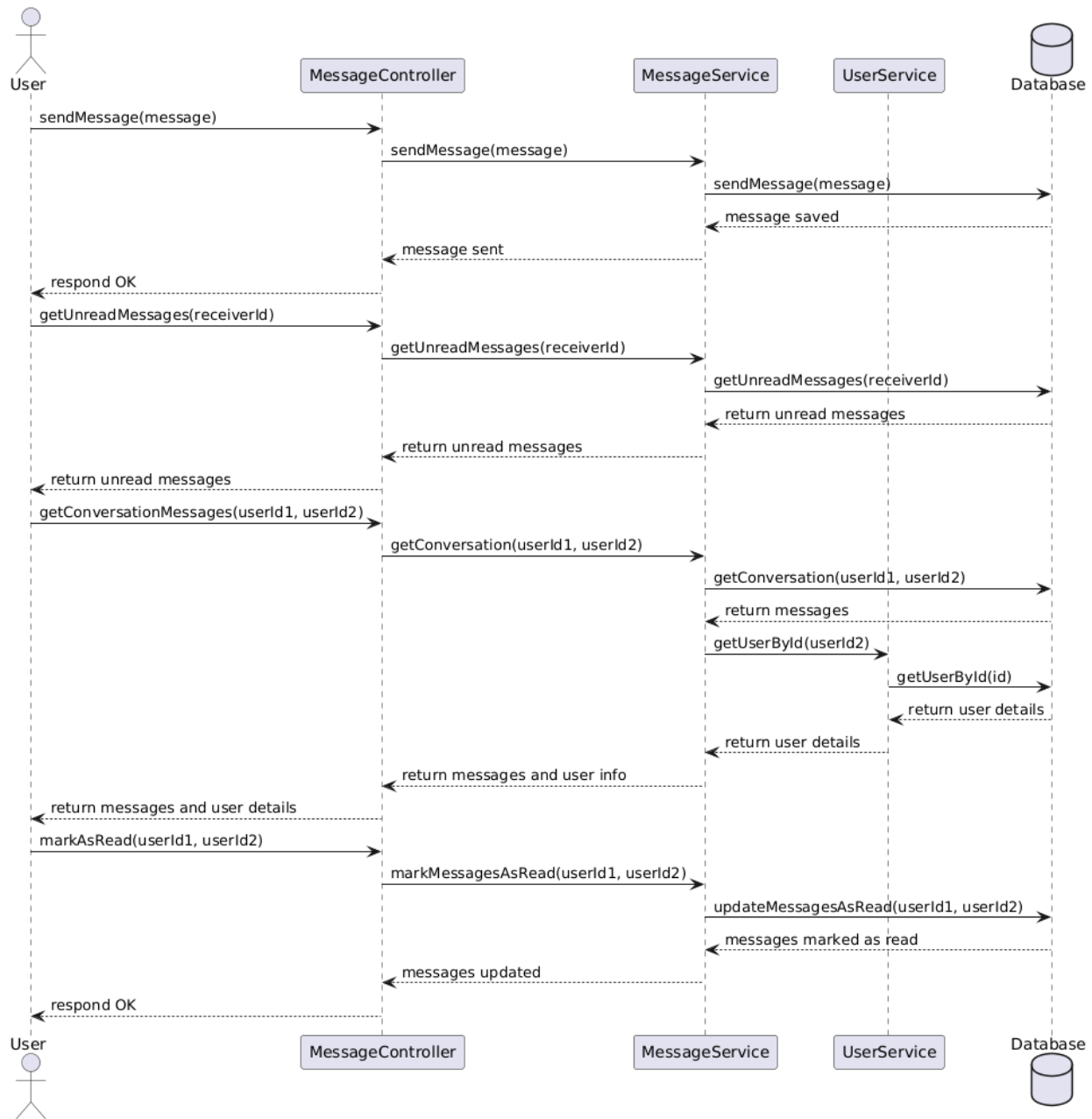


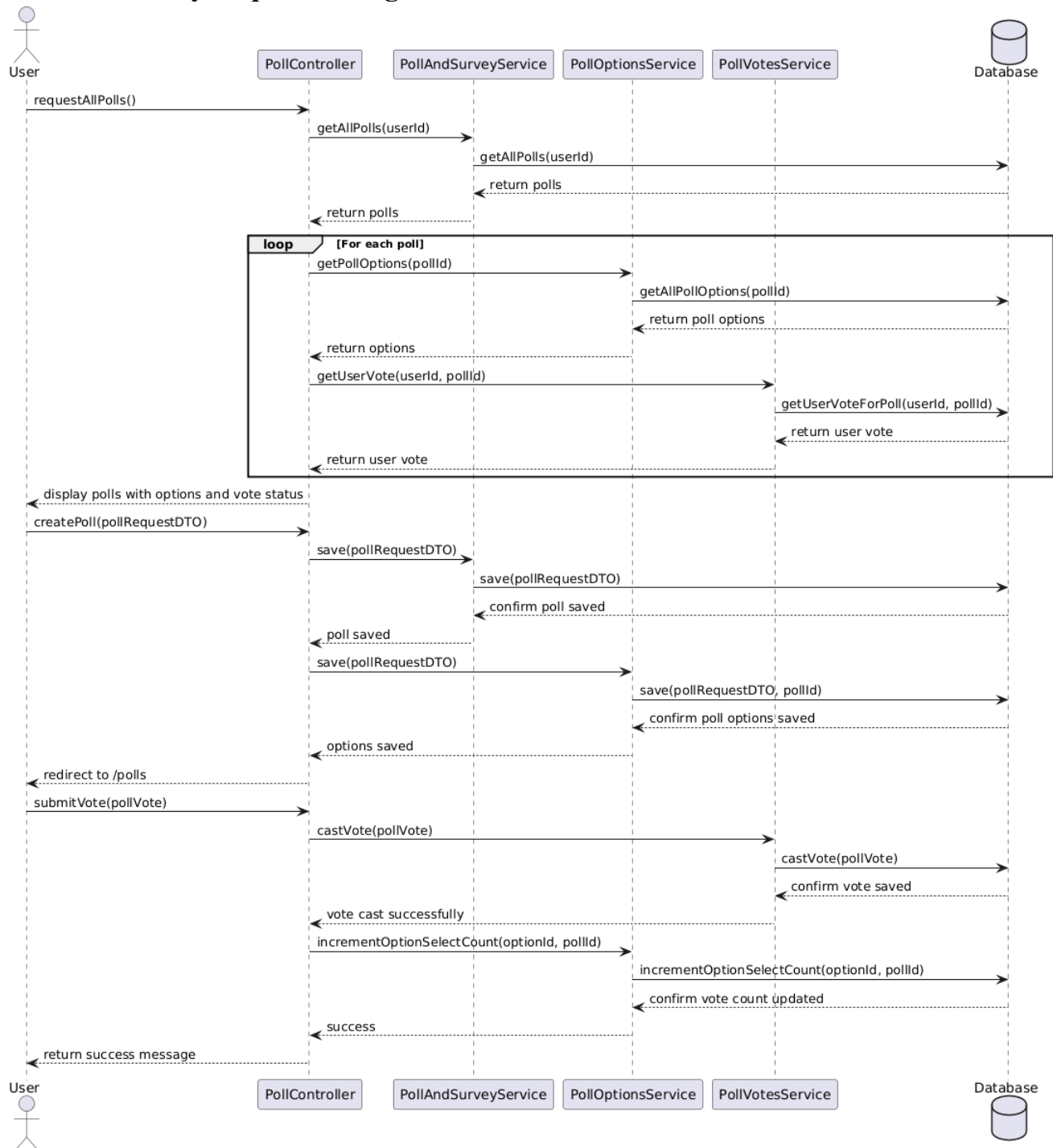
6. Sequence Diagram

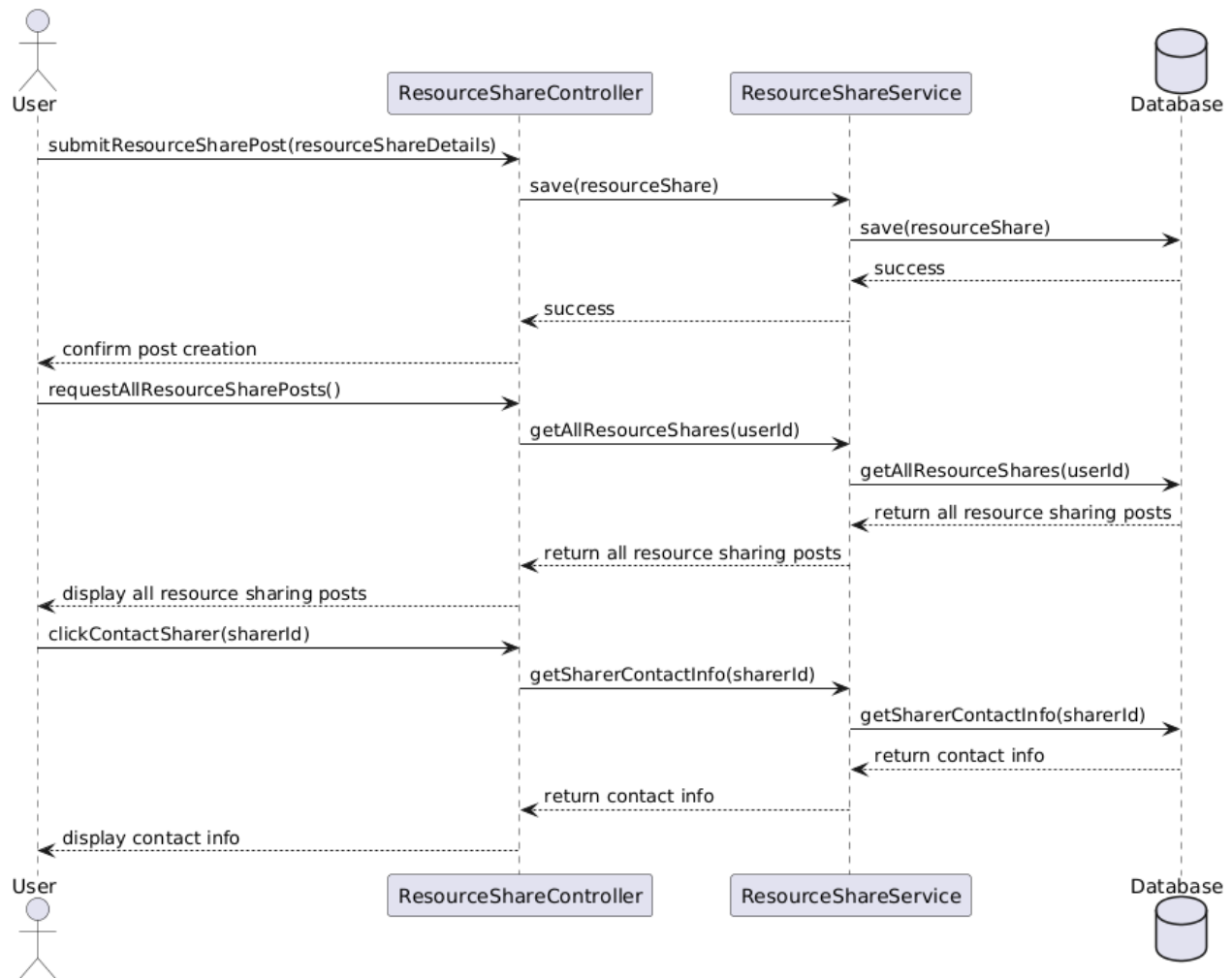
Events Sequence Diagram:

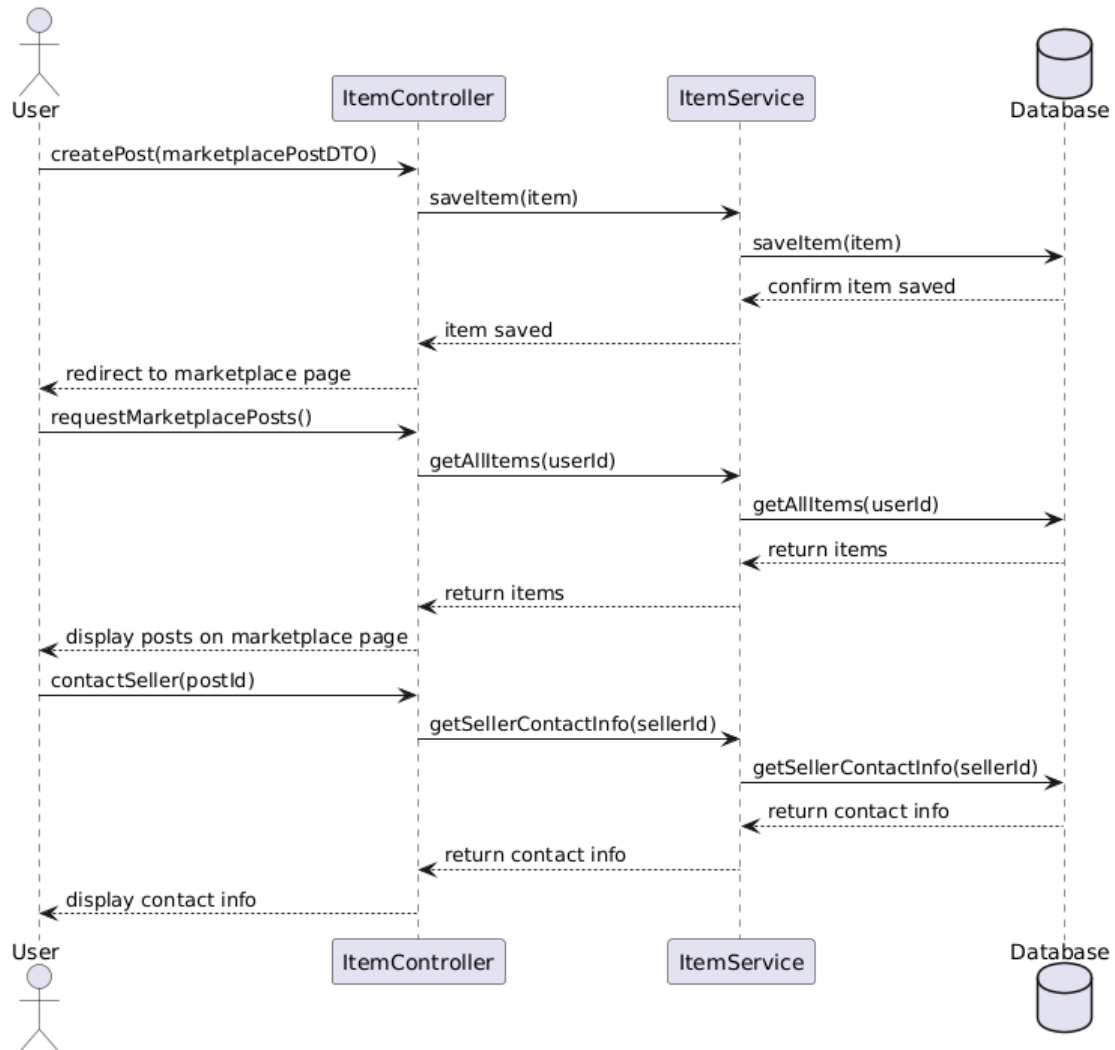


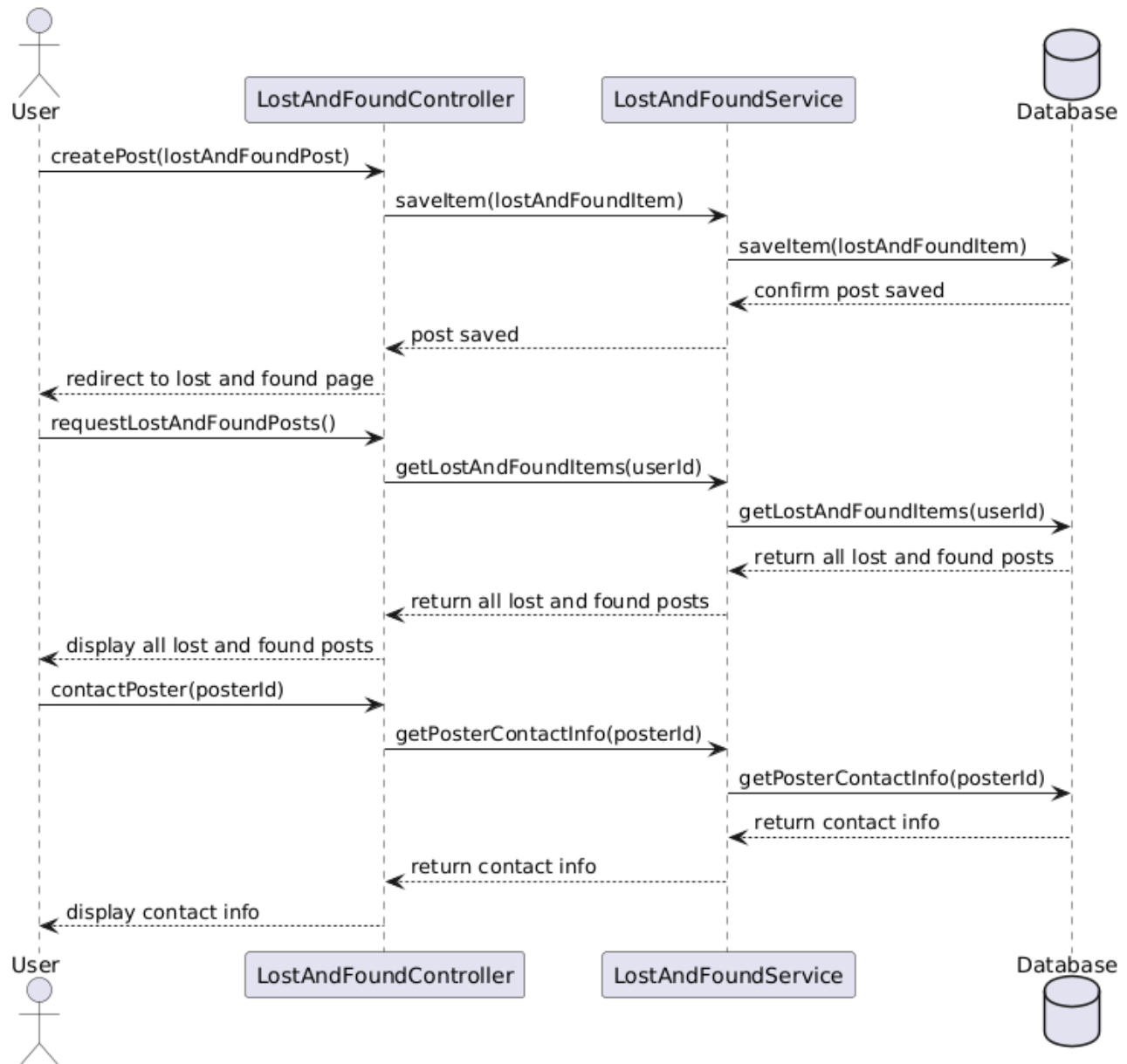
Messages Sequence Diagram:

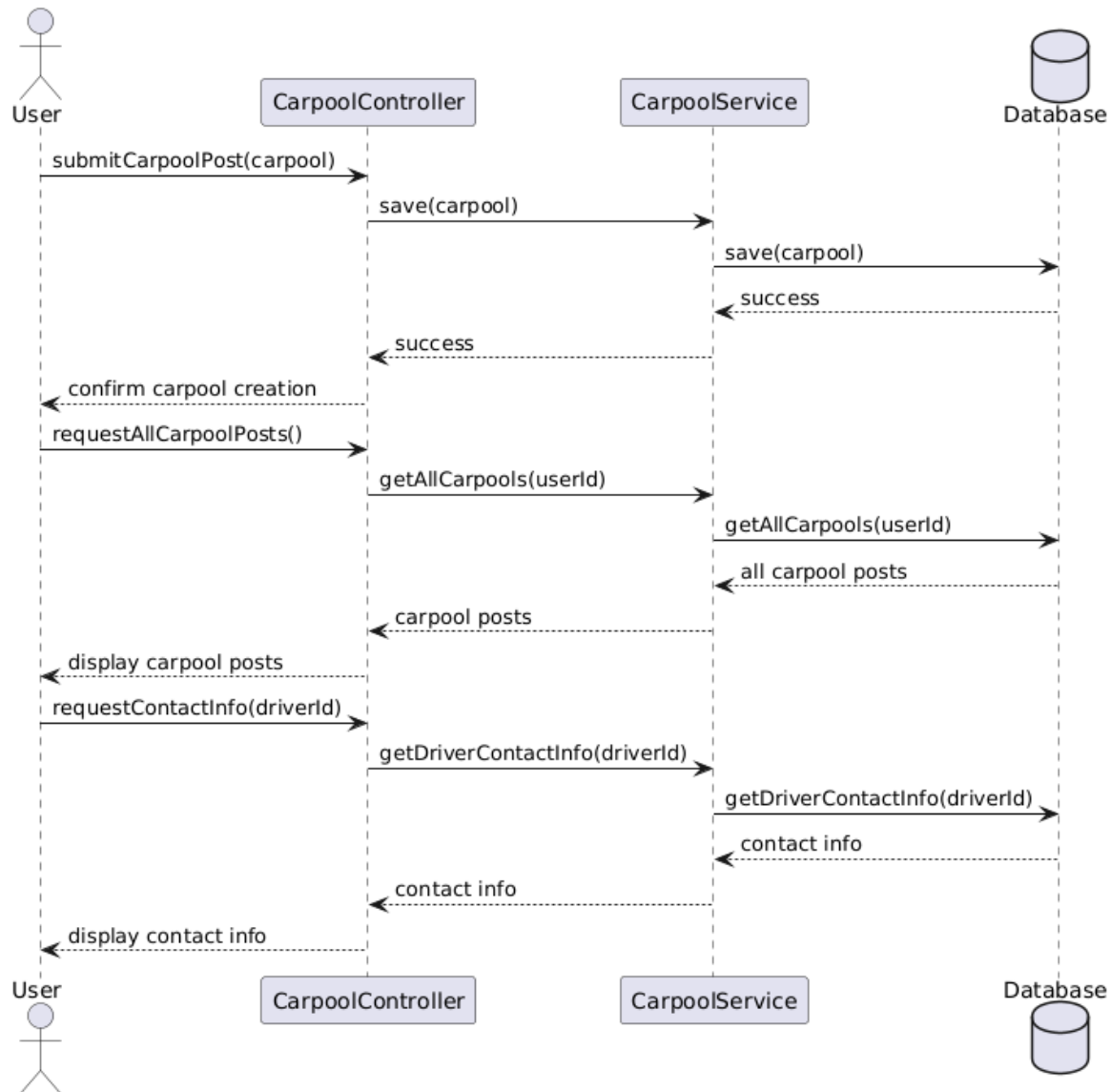


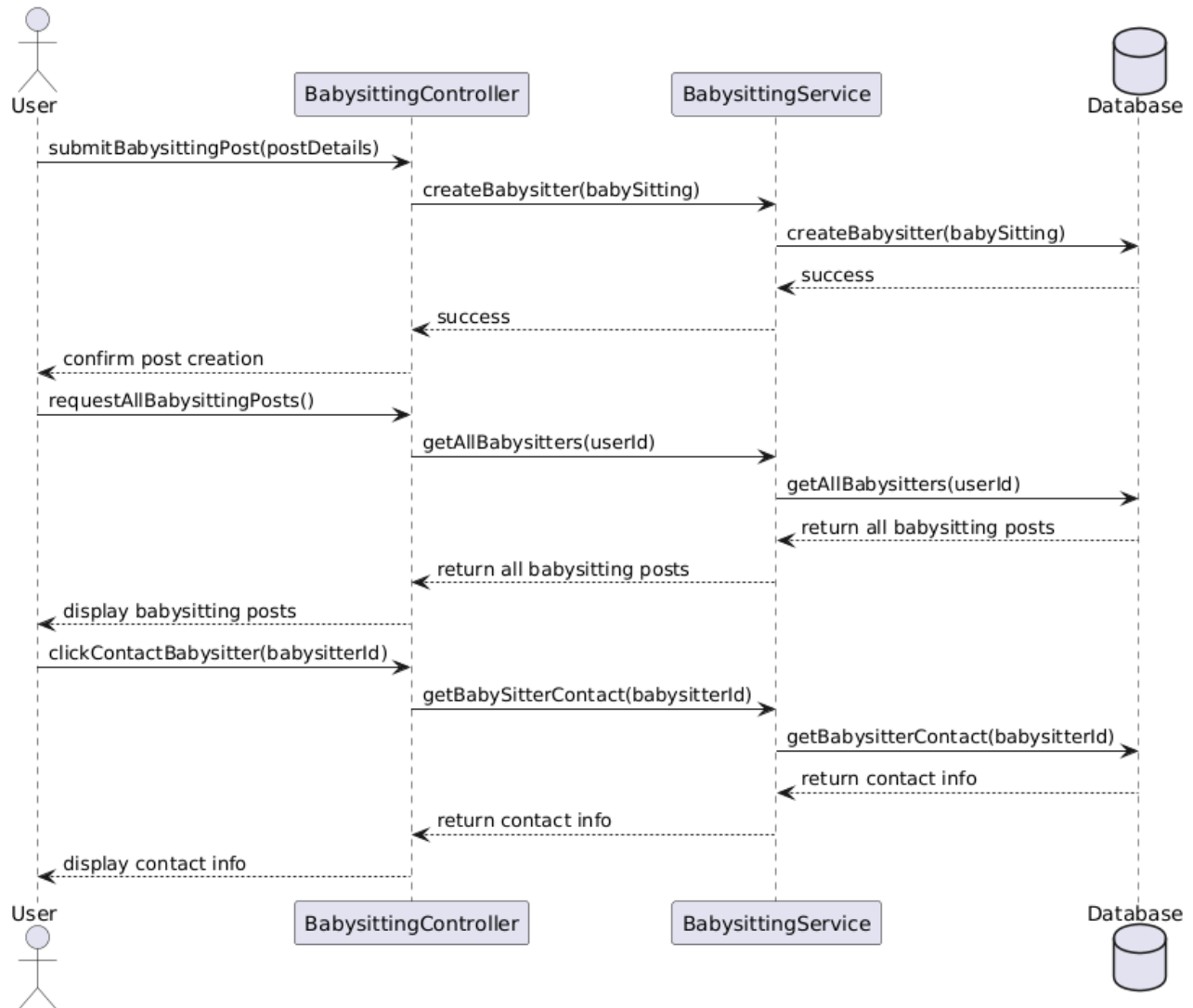
Poll and Survey Sequence Diagram:

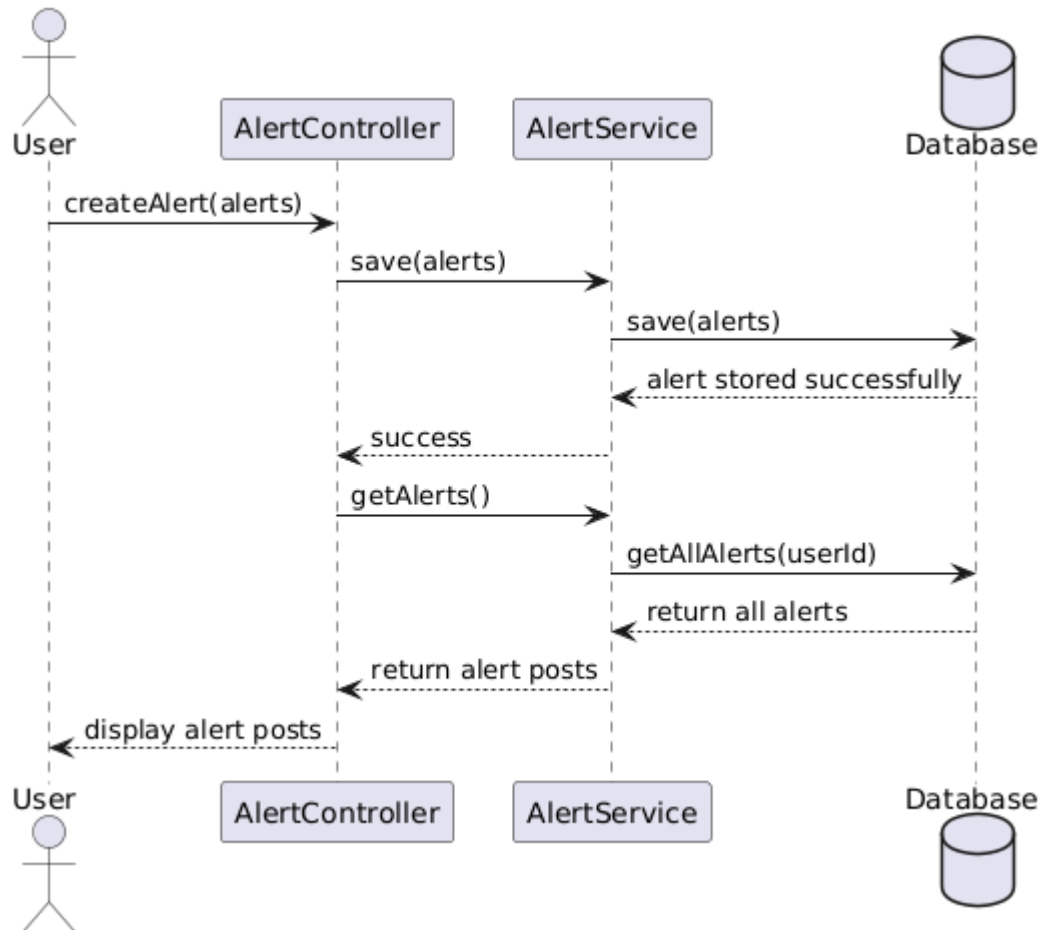
Resources Sharing Sequence Diagram:

Marketplace Sequence Diagram:

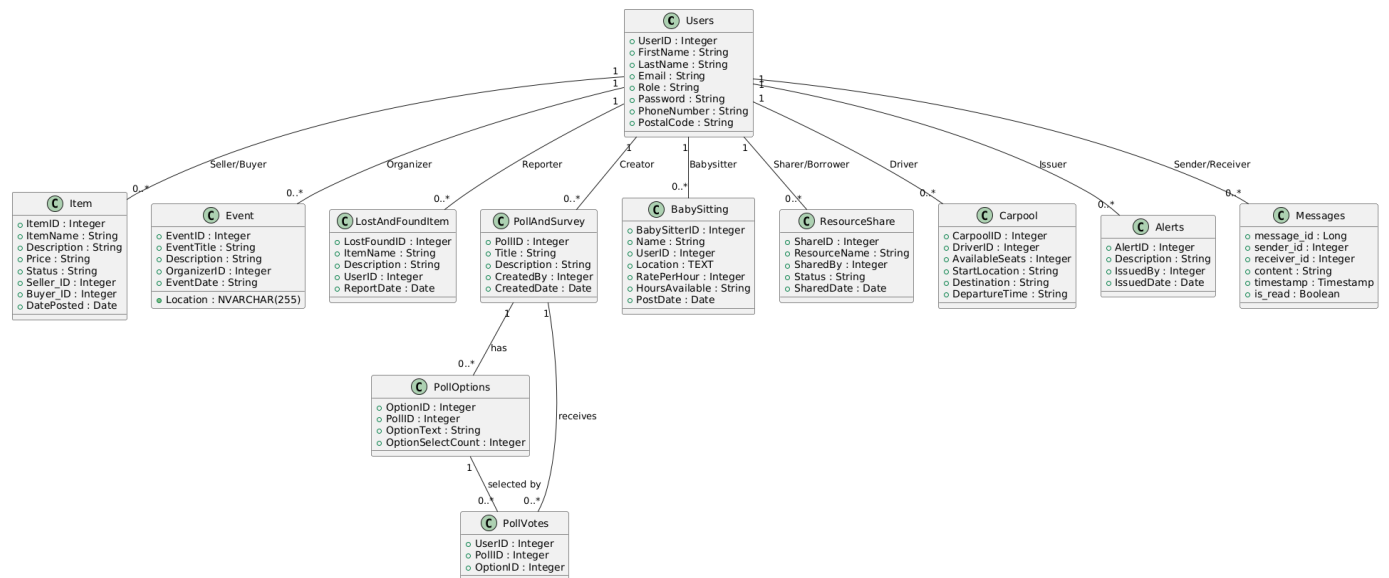
Lost and Found Sequence Diagram:

Carpool Sequence Diagram:

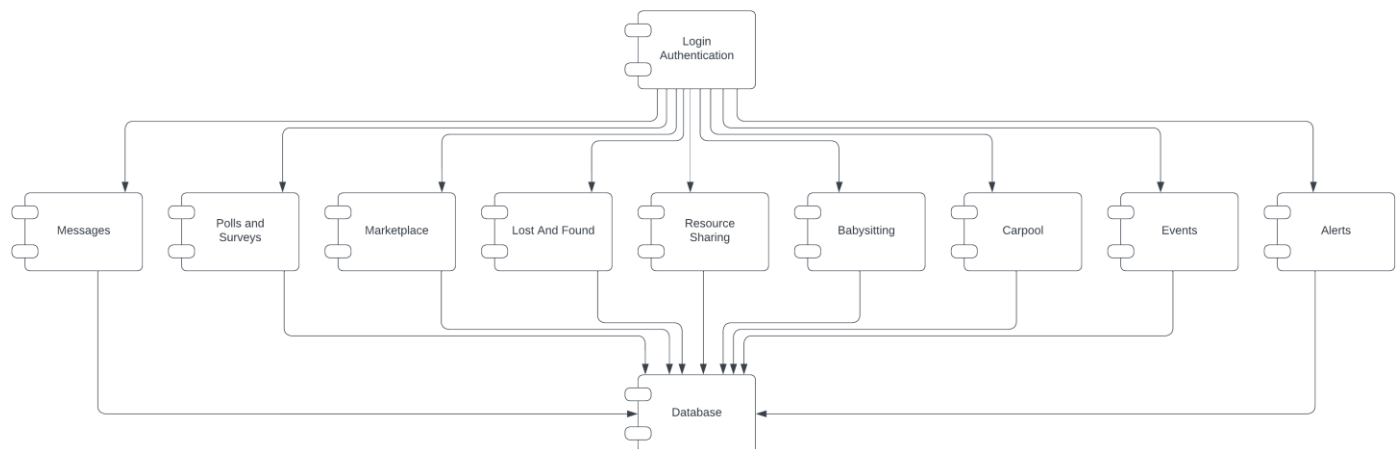
Babysitting sequence Diagram:

Alert Sequence Diagram:

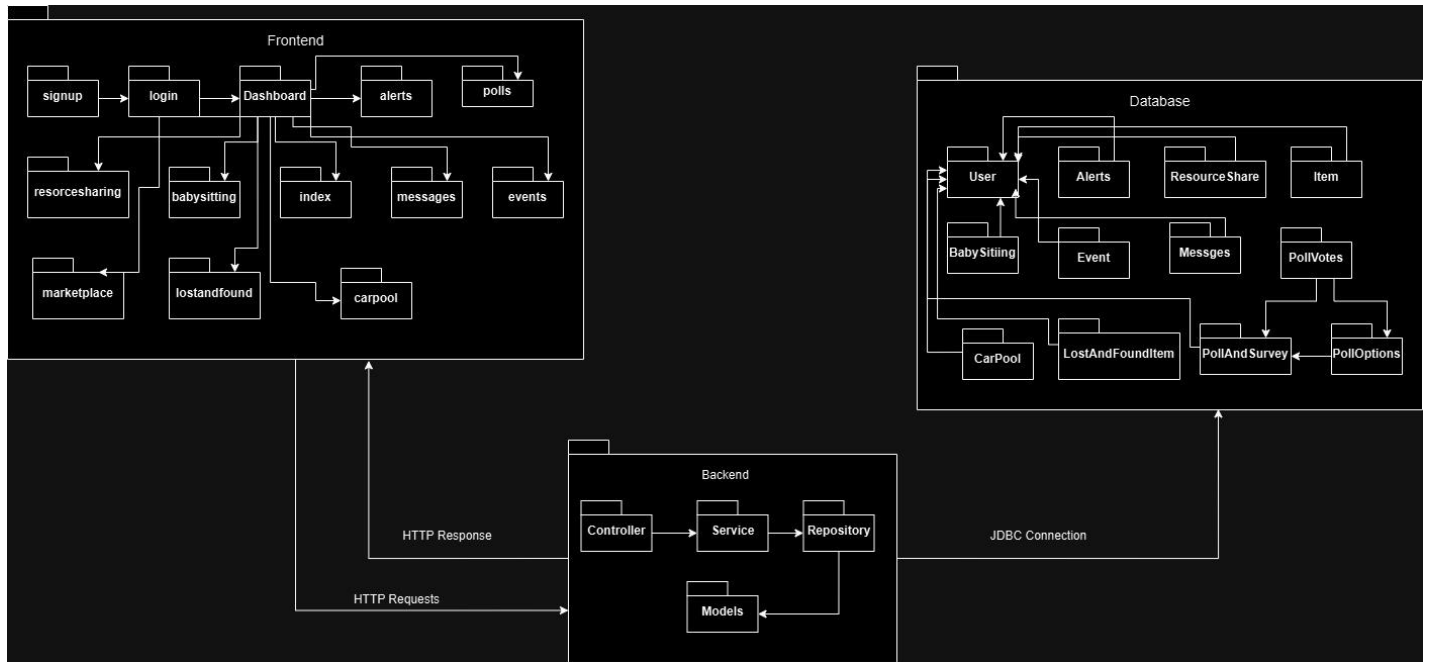
7. Class Diagram



8. Component Diagram



9. Package Diagram



10. Deployment Diagram

