NEIGHBOURHOOD WATCH APPLICATION

Software Design CSI473 Project Phase I

Adam Musakabantu Muyobo 202104060
Basetsana Prudence Hunyepa 20202798
Elizabeth Tankiso Otumile 202202896
Anita Amantle Kgosidiile 202000781
Musa Gae Monnahela 202002752

Requirements Analysis

This section highlights all the functional and nonfunctional requirements of the project.

Functional Requirements

User Management & Access

- Administrators
    - Approve/reject self-registered accounts (security officers and members).
    - Add, remove, suspend, block users (admins, officers, members).
    - Add/remove/suspend houses for monitoring.
    - Configure and maintain system settings.
    - Reset passwords (via email link).
    - Must always log in with two-factor authentication (OTP via email).
- Security Officers
    - Self-register in the mobile app; accounts must be approved by an administrator.
    - Log patrols by scanning QR codes at gates.
    - Add contextual comments like “gate open.”
    - View their complete patrol history.
    - Cannot edit or delete past patrol logs.
- Neighbourhood Watch Members
    - Self-register in the mobile app account approval required by the administrator.
    - Pay a subscription fee (X Pula/month) via secure online payment methods.
    - View patrol statistics for their area.
    - Interact with the community via posts and discussions.
    - Raise emergency alerts through the app.
    - Receive payment reminders, alerts, and notifications.
    - Accounts automatically suspended after 2 months of missed payments.

Patrol & Monitoring

- Each gate/hose is assigned a unique QR code.
- Patrol scan must capture officer ID, checkpoint ID, date/time, optional comments, and location data.
- System tracks patrol compliance:
    - If an officer scans fewer than X% of assigned checkpoints, they are provisionally suspended.
    - Alerts (SMS + email) sent to administrator and members.
    - Administrators can revoke or finalize suspension.
    - Officers reinstated only after 3 months suspension or admin approval.
    - Permanently deleted officers cannot rejoin.

Alerts & Community Features

- Emergency Alerts
    - Raised via mobile app button.
    - Visible to all members and triggers immediate notification to security officers (high priority, near real-time).
- Community Interaction
    - Members can post comments, engage in discussions, and receive community updates.

Payments & Subscriptions

- Members pay online via supported Botswana gateways (mobile money, card, Stripe, etc.).
- Automatic suspension after 2 months of missed payments.
- Voluntary subscription cancelation allowed.
- Payment records stored for reporting.

Reporting

- Generate weekly, monthly, and yearly reports.
- Reports include patrol compliance, anomalies, subscriptions, payments, officer activity.
- Access rules:
    - Admins: all reports.
    - Officers: personal patrol history.
    - Members: statistics for their neighbourhood.

Offline Functionality

- Mobile app must work offline for officers and members:
    - Scan QR codes and log patrols.
    - Store data locally until the internet is restored.
- Automatic synchronization once connectivity returns.

Non-Functional Requirements

Security

- Secure authentication (2FA for admins, OTP for sign-ups).
- Sensitive data (personal info, patrol logs, payment records) must be encrypted.
- Payment processing via PCI-DSS compliant gateways with tokenization.
- Audit logs for all sensitive actions (user changes, suspensions, payments).

Scalability

- The system should scale to support large neighbourhoods with many officers and members.

- Cloud-based infrastructure with load balancers and scalable databases.
- Use caching (e.g., Redis) for high-frequency data like patrol stats and notifications.

Performance

- API response time ≤ 300ms for standard requests.
- Emergency alert notifications dispatched within ≤ 5s.
- The system must manage high loads (hundreds of concurrent patrol logs + community posts + alerts).
- Efficient offline data sync with conflict resolution.

Availability & Reliability

- High availability architecture (99.9% uptime target).
- Multi-region or multi-AZ deployment.
- Automated backups with Recovery Point Objective (RPO) ≤ 15 min and Recovery Time Objective (RTO) ≤ 60 min.

Usability

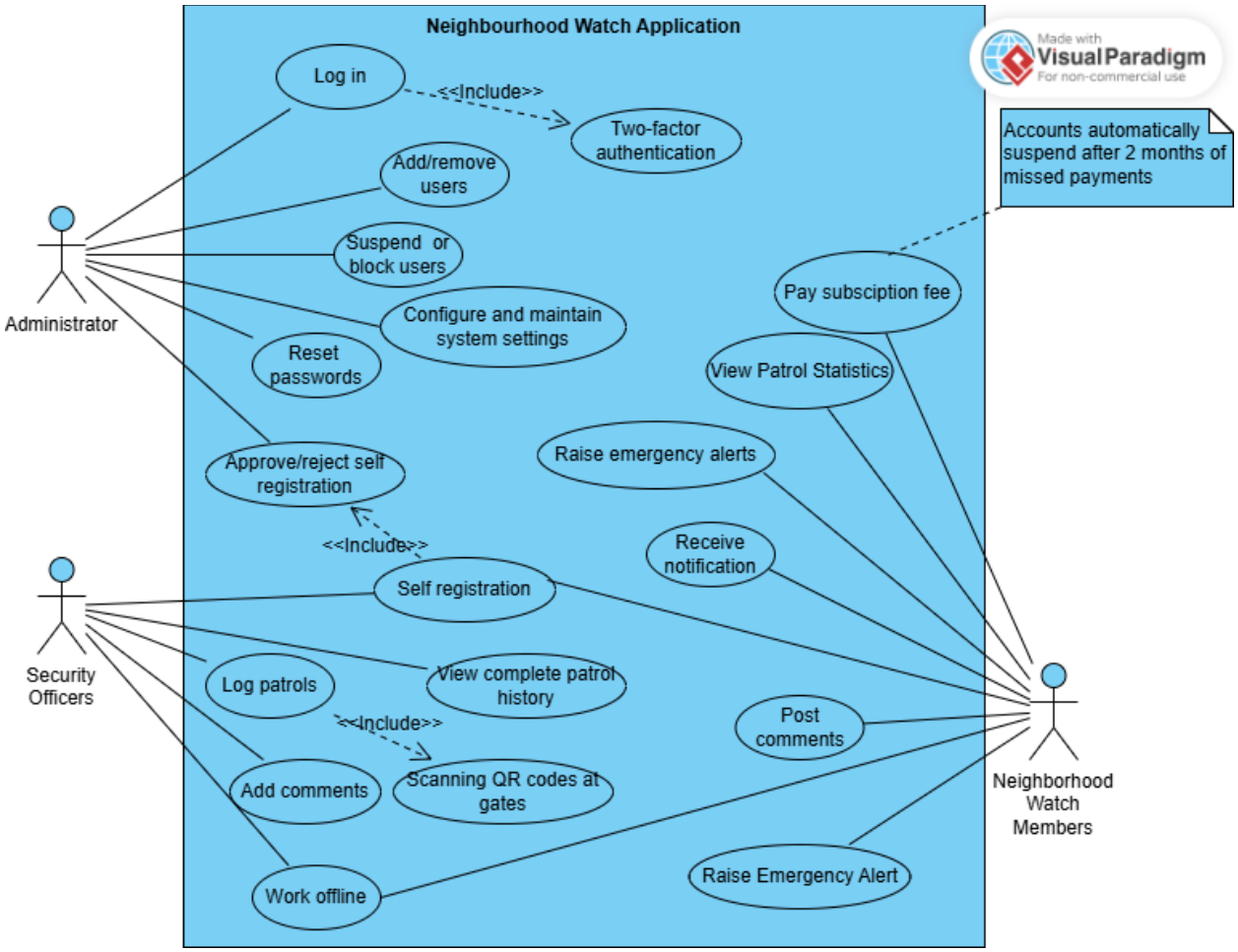
- Intuitive mobile app interface for officers and members.
- Accessible web interface for administrators.
- Local language support (English, Setswana).

Maintainability & Extensibility

- Modular service-oriented or microservices architecture.
- Support for future integration with wearable devices, USSD alerts, or third-party services

Use Case Model

Use Case Diagram



DETAILED SPECIFICATIONS OF USE CASES

Log In

Name of Use Case	Log In
Actors	Administrator, Security Officers, Neighbourhood Watch Members.
Entry condition	This use case starts when a user attempts to access the Neighbourhood Watch Application.
Flow of Events	1. The user opens the Neighbourhood Watch Application.

	2. The system presents the login screen, prompting for a username and password. 3. The user enters their credentials and clicks "Log In." 4. The system verifies the credentials. 5. The system determines the user's role (Administrator, Security Officer, or Neighbourhood Watch Member). 6. The system presents the appropriate dashboard or home screen for the user's role.
Exit condition	This use case terminates when the user is successfully logged in and sees their respective dashboard.
Exceptions	If the username or password is incorrect, the system displays an error message ("Invalid username or password.") and prompts the user to try again.
Special Requirements	For Administrators, two-factor authentication must be implemented after successful password verification to provide an extra layer of security.

Add/Remove Users

Name of Use Case	Add/Remove Users
Actors	Administrator
Entry condition	This use case starts when an Administrator has successfully logged in and navigates to the user management section of the application.
Flow of Events	1. The Administrator logs into the system and navigates to the User Management section. 2. The system displays a list of all existing users. 3. The Administrator initiates one of two actions: a. Add a User: i. Fills out a form with the new user's details.

	ii. Submit the form. iii. The system validates the information, creates the new account, and sends a notification to the new user. b. Remove a User: i. Selects an existing user. ii. Confirms the deletion. iii. The system permanently removes the user and all their associated data.
Exit condition	This use case terminates when the Administrator has either successfully added a new user or removed an existing user, and the system confirms the action.
Exceptions	If the Administrator provides invalid or incomplete information when adding a user or attempts to modify a user that does not exist, the system will display a specific error message and guide them to correct the issue.
Special Requirements	The system must maintain an audit log of all user creation and deletion actions, including the date, time, and the Administrator who performed the action

Configure and Maintain System Settings

Name of Use Case	Configure and Maintain System Settings
Actors	Administrator
Entry condition	This use case starts when an Administrator has successfully logged in and navigates to the "System Settings" or "Configuration" section of the application
Flow of Events	1. The Administrator logs into the system. 2. The Administrator selects the "System Settings" option from the administrative menu. 3. The system presents a dashboard with various configurable settings. 4. The Administrator selects a specific

	setting to view or modify. 5. The system displays the current value of the setting. 6. The Administrator makes the desired changes. 7. The Administrator saves the changes. 8. The system validates the new settings to ensure they are valid and conform to a proper format. 9. The system applies the new settings and displays a confirmation message to the Administrator.
Exit condition	This use case terminates when the Administrator has successfully updated the system settings, and the changes have been saved and applied.
Exceptions	An invalid value is entered, or a system error occurs while saving.
Special Requirements	If, for any reason, the Administrator's permissions are insufficient to modify a specific setting, the system displays an error message "Access denied."

Approve/Reject Self-Registration

Name of Use Case	Approve/Reject Self-Registration
Actors	Administrator
Entry condition	This use case starts when an Administrator successfully logs into the system and navigates to the applications dashboard.
Flow of Events	1. Administrator navigates to a list of pending registrations. 2. Administrator selects a recent registration to review. 3. Administrator reviews the user's details. 4. Administrator chooses to either Approve or Reject the registration. 5. The system updates the user's account status accordingly and sends a notification to the user.
Exit condition	This use case terminates when the Administrator

	has successfully approved or rejected a self-registration request, and the system has processed the action and sent the appropriate notifications.
Exceptions	If the system is unable to write the new user record to the database upon approval, the system rolls back the transaction and notifies the Administrator.
Special Requirements	Only Administrators should have access to this functionality. The system must verify the Administrator's permissions before allowing any actions.

Self-Registration

Name of Use Case	Self-Registration
Actors	Neighbourhood watch member and security officer
Entry condition	This use case starts when a new user accesses the application and selects the "login" or "Self-Register" option.
Flow of Events	1. The user enters their details (e.g., name, email, phone number, address). 2. The user submits the form. 3. The system validates the data and creates a pending user account. 4. The system sends a notification to the Administrator to review the application.
Exit condition	The user's registration details are successfully submitted and are awaiting approval.
Exceptions	Missing information, email already in use.
Special Requirements	The system must provide a clear and easy-to-use registration form.

Log Patrols

Name of Use Case	Log Patrols
Actors	Security officer

Entry condition	This use case starts when the Security Officer is authenticated and selects the "Log Patrols" function.
Flow of Events	1. The Security Officer opens the patrol logging interface. 2. The system displays a form with fields for patrol details such as date, time, duration, and route. 3. The Security Officer enters the required information. 4. The Security Officer can optionally scan QR codes located at neighbourhood gates to automatically log a checkpoint visit. 5. The Security Officer submits the log. 6. The system validates the data and saves the patrol record.
Exit condition	This use case terminates when the patrol log has been successfully saved to the system's database.
Exceptions	1. If required fields are left blank, the system will prompt the Security Officer to complete the form. 2. If the system is offline, the application will store the patrol log locally and inform the officer that it will synchronise when a connection is re-established.
Special Requirements	1. The application must support offline functionality for this use case. 2. Data entry must be optimized for speed and ease of use on mobile devices. 3. The system must accurately capture timestamps for patrol events.

View Complete Patrol History

Name of Use Case	View Complete Patrol History
Actors	Security officer

Entry condition	This use case starts when the Security Officer is authenticated and selects the "View Complete Patrol History" option.
Flow of Events	1. The Security Officer navigates to the patrol history screen. 2. The system retrieves all patrol records associated with the Security Officer's account. 3. The system displays the records in a list, showing key details for each entry. 4. The Security Officer can select a specific record to view more detailed information.
Exit condition	This use case terminates when the Security Officer has finished reviewing the patrol history and exits the screen.
Exceptions	1. If there are no patrol records, the system will display a message stating that the history is empty. 2. If the system fails to retrieve the data due to a server error, it will display a connection error message.
Special Requirements	1. The data must be presented in a clear, easy-to-read format. 2. The system must load the history quickly to provide a responsive user experience.

Add Comments

Name of Use Case	Add Comments
Actors	Security officer
Entry condition	This use case starts when an actor selects a specific patrol or incident report and chooses to add a comment.
Flow of Events	1. The actor selects the patrol or incident to comment on.

	2. The system presents a text field for the comment. 3. The actor types their comment. 4. The actor submits the comment. 5. The system saves the comment and associates it with the chosen record.
Exit condition	This use case terminates when the comment has been successfully posted to the system.
Exceptions	1. If the comment text is empty, the system will not allow the submission. 2. If the system is offline, the comment will be saved locally and synchronised later.
Special Requirements	1. The system must ensure that comments are associated with the correct patrol or incident record. 2. The system must be able to manage comments of varying lengths.

Work Offline

Name of Use Case	Work Offline
Actors	Security officer and Neighbourhood watch members
Entry condition	This use case starts when the Neighbourhood watch members and The Security Officer's device loses network connectivity.
Flow of Events	1. The device detects the loss of a network connection. 2. The application automatically switches to offline mode without interrupting the user's current activity. 3. The Security Officer continues to perform actions like logging patrols or adding comments. 4. The application stores all new data locally. 5. When the connection is restored, the application automatically synchronises the locally stored data with the main server.

Exit condition	This use case terminates when all locally stored data has been successfully synchronised with the server.
Exceptions	If the local storage is full, the system will warn the user that no more data can be stored until a connection is re-established.
Special Requirements	1. The synchronization process must be seamless and not require manual intervention. 2. The system must manage potential data conflicts that may arise from a lengthy period of being offline

Pay Subscription Fee

Name of Use Case	Pay Subscription Fee
Actors	Neighbourhood Watch member
Entry condition	This use case starts when a Neighbourhood Watch Member is logged in and navigates to the payment section.
Flow of Events	1. The member navigates to the subscription payment screen. 2. The system displays the current subscription status and outstanding fees. 3. The member selects a payment option and enters their payment details. 4. The system securely processes the payment. 5. The system confirms a successful payment and updates the member's subscription status.
Exit condition	This use case terminates when the payment has been successfully processed, and the member's account is up to date.
Exceptions	If the payment fails, the system will display an error message explaining the reason for the failure (e.g., "Insufficient funds," "Card declined").

Special Requirements	1. The payment process must be secure and compliant with financial regulations. 2. The system must provide a clear visual indicator of the member's subscription status.

Raise Emergency Alert

Name of Use Case	Raise Emergency Alert
Actors	Neighbourhood Watch member
Entry condition	This use case starts when a Neighbourhood Watch Member is logged in and an emergency arises.
Flow of Events	● The member selects the "Raise Emergency Alert" function. ● The system prompts the member to confirm the action. ● Upon confirmation, the system sends an immediate push notification to all relevant Security Officers and nearby members. ● The system logs the alert with the member's location and timestamp.
Exit condition	This use case terminates when the emergency alert has been successfully broadcast to all recipients.
Exceptions	If the member is not logged in or authorized, the system will not allow the alert to be sent.
Special Requirements	● The system must provide a highly visible and easy-to-access button for this use case. ● The alert must be delivered with minimal delay. ● The system must log the time, date, and location of the alert.

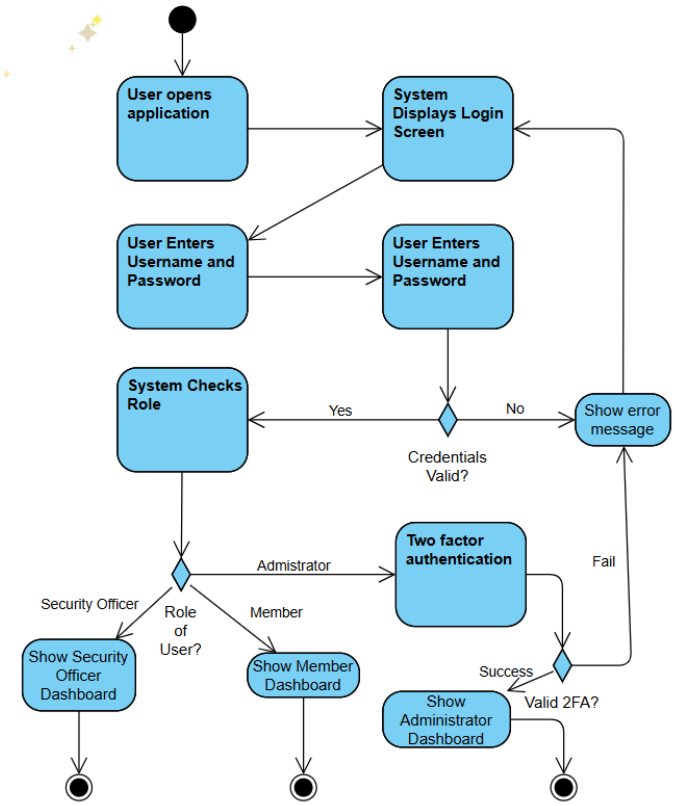
Receive Notification

Name of Use Case	Receive Notification

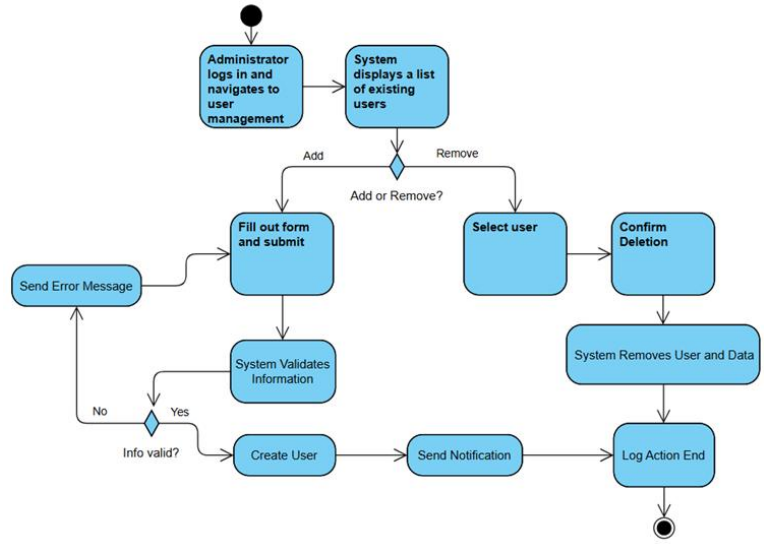
Actors	Neighbourhood Watch member
Entry condition	This use case starts when an event occurs that triggers a notification for the member.
Flow of Events	1. An event occurs in the system (e.g., an emergency alert is raised, a new comment is posted, a patrol is logged in their area). 2. The system's notification service sends a push notification to the member's device. 3. The member's device displays the notification. 4. The member opens the notification to be taken to the relevant section of the application.
Exit condition	This use case terminates when the member has received and viewed the notification.
Exceptions	If the user has disabled notifications on their device or in the application, the notification will not be delivered.
Special Requirements	1. The notification delivery must be dependable and fast. 2. The notifications must be distinguishable and informative (e.g., a special sound for an emergency alert).

Activity Diagrams

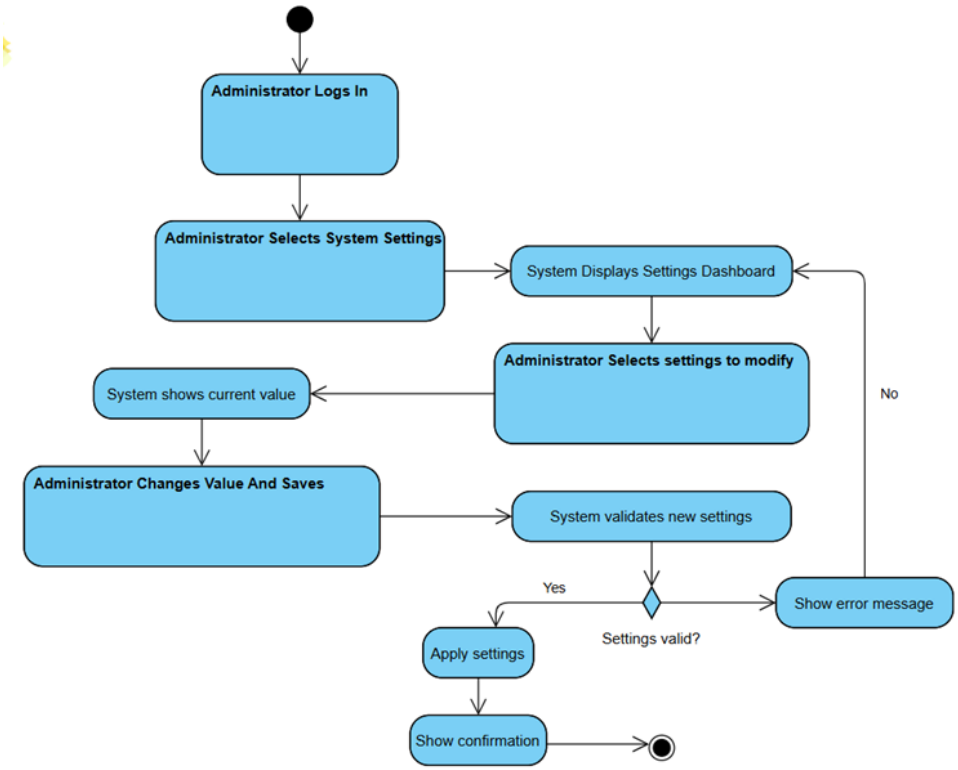
Log In



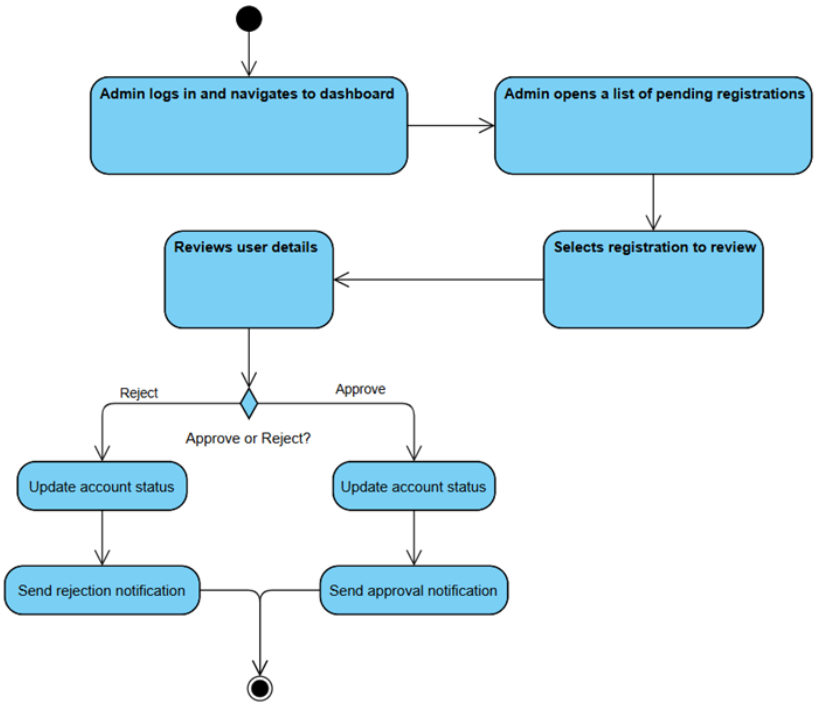
Add/Remove Users



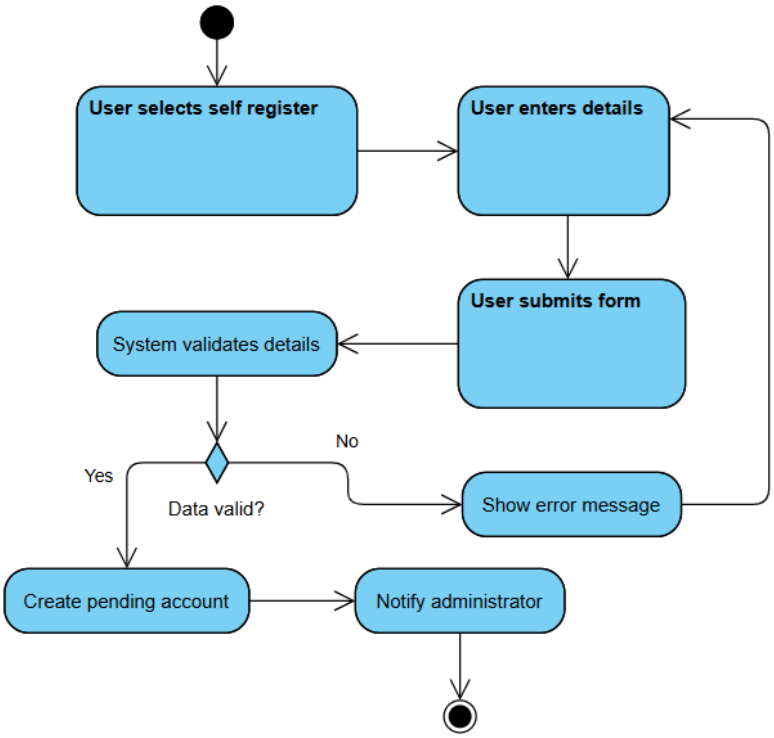
Configure and Maintain System Settings



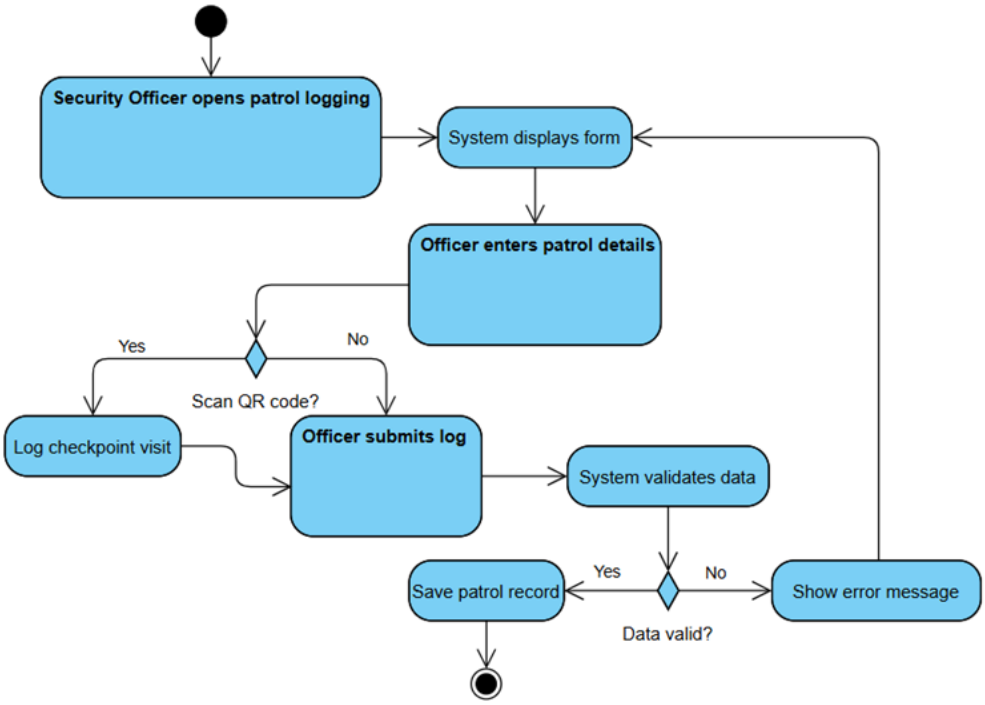
Approve/Reject Self Registration



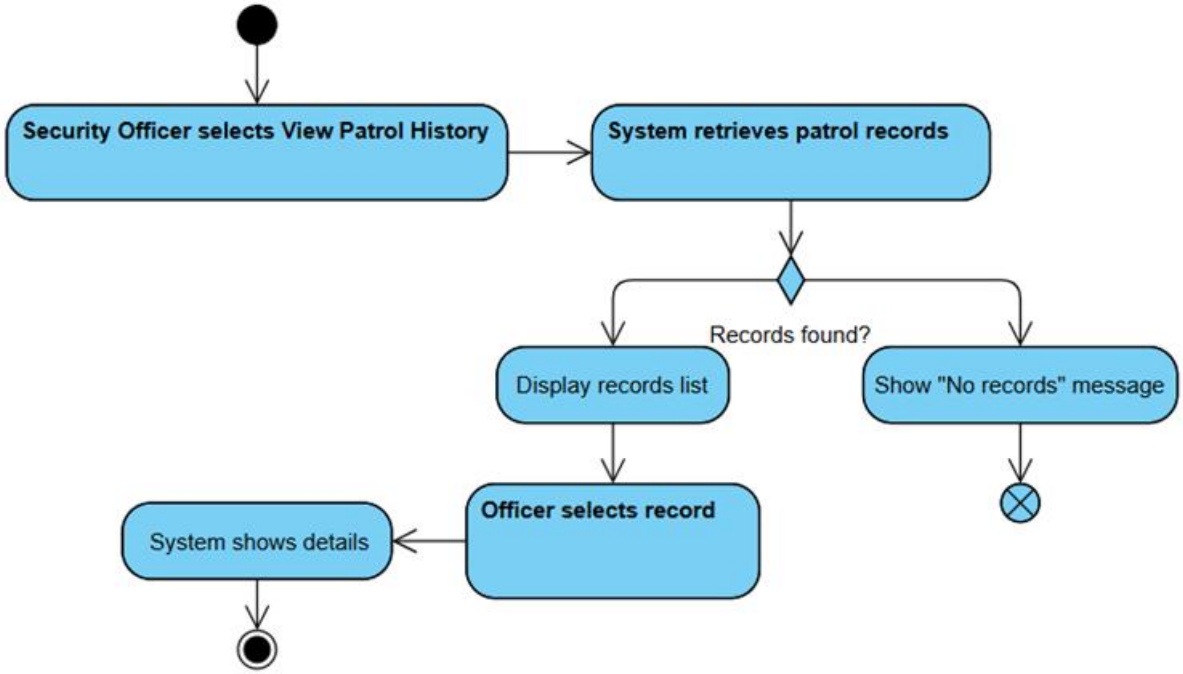
Self-Registration



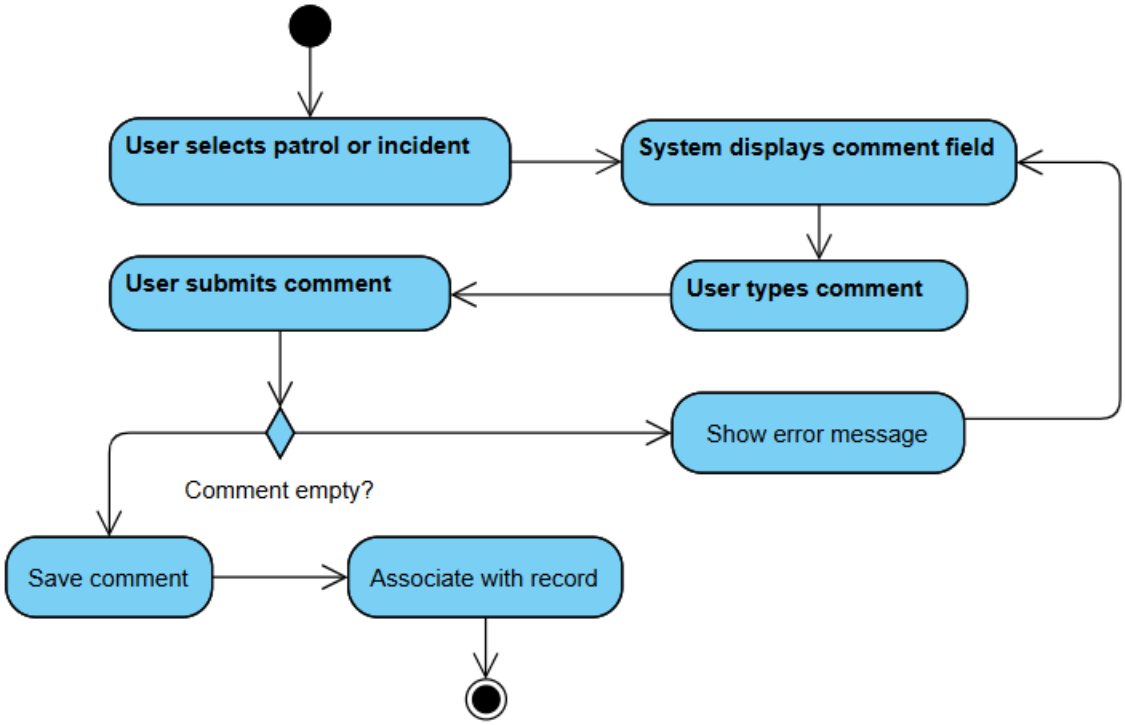
Log Patrols



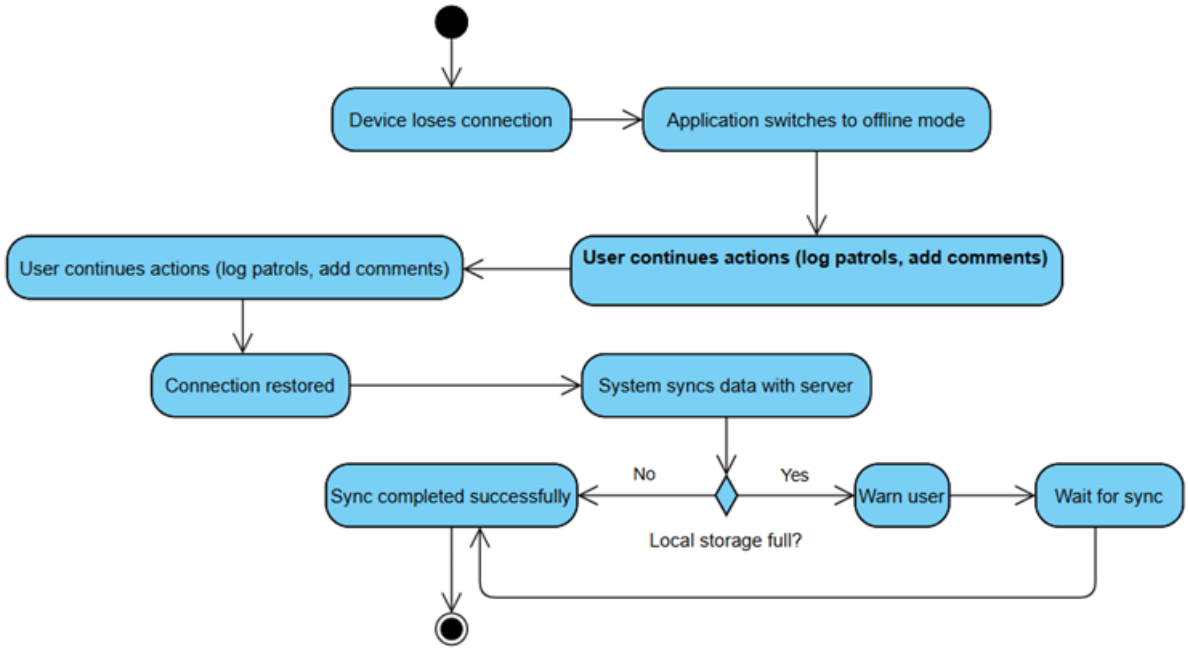
View Complete Patrol History



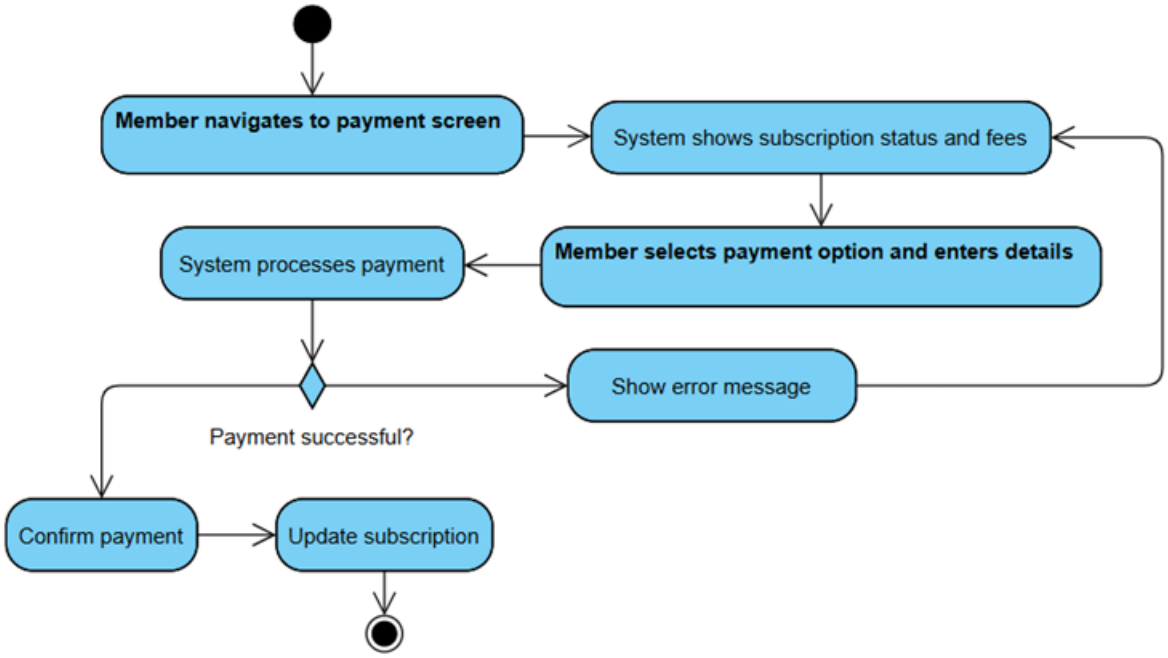
Add Comments



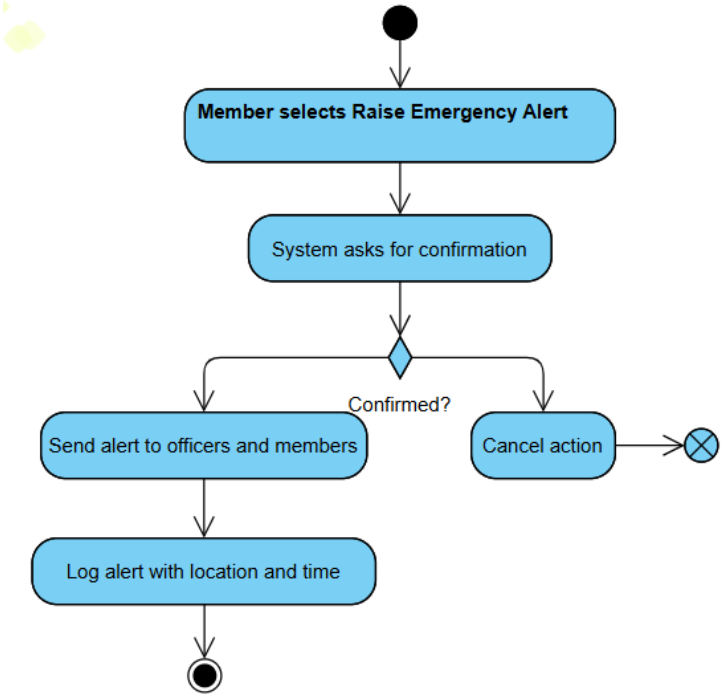
Work Offline



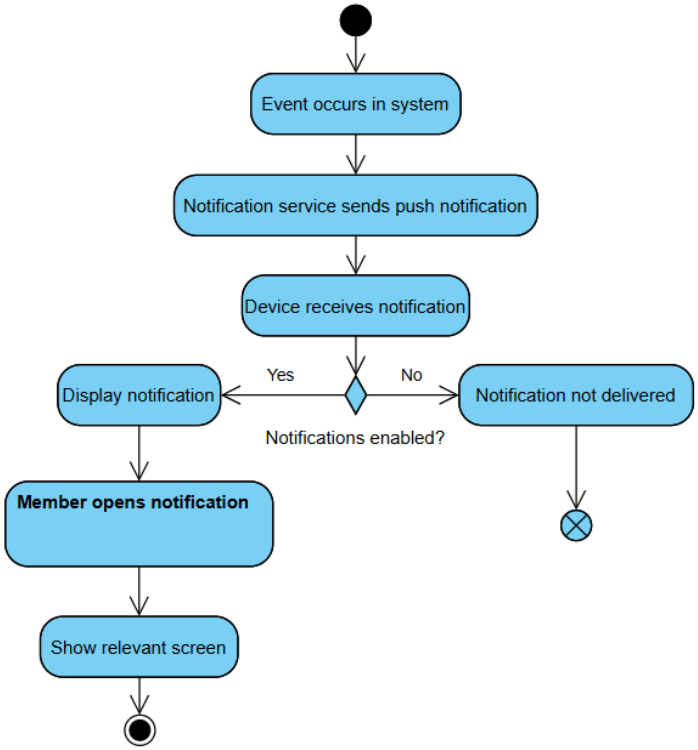
Pay Subscription Fee



Raise Emergency Alert



Receive Notification



System Design

Subsystem Identification

User Management Subsystem

- Administrator Management
- Officer Registration Approval
- Member Registration and Approval
- Authentication and Authorization

Patrol Monitoring Subsystem

- QR Code Scanning
- Patrol Logging
- Compliance Tracking
- Officer Suspension and Reinstatement

Membership and Subscription Subsystem

- Payment Processing
- Subscription Management
- Account Suspension/Un-subscription.

Community Interaction Subsystem

- Post and Comment Management
- Member Notifications
- Discussion Threads

Emergency and Alert Subsystem(notifications)

- Emergency Alert Trigger
- SMS/Email Notification
- Anomaly Reporting

Reporting and Analytics Subsystem

- Patrol Reports (Weekly/Monthly/Yearly)
- Compliance Reports
- Statistics Dashboard

Payment and Billing Subsystem

- Transaction Handling
- Payment Gateway Integration
- Audit Logs

Synchronization and Offline Subsystem

- Local Storage.
- Offline Mode Operations
- Data Synchronization

Security and Access Control Subsystem

- Data Encryption
- Role-Based Access Control
- Two-Factor Authentication

System Maintenance and Configuration Subsystem

- System Configuration
- Performance Monitoring
- Backup and Disaster Recovery
- Scalability Management

Backend Services Subsystem

- APIs, microservices, message queues, notification services.
- External service connectors (payment gateways, SMS/email APIs).

UI Subsystem

- Responsive web design for admins.
- mobile app UI for officers/members
- offline-ready mobile components, accessibility features.

Database Subsystem

- Data storage for users
- patrol logs, payments, subscriptions.
- alerts, backup & restore.
- indexing for performance.

Subsystem Relationships

Top Level Layer

- UI Subsystem
    1. Acts as the entry point for all users (admins via web, officers, and members via mobile).
    2. Directly connects users with the backend services and other subsystems.

Core Functional Layer

- User Management Subsystem
    1. Works with the UI Subsystem for registration, login, and authentication.
    2. Provides identity and access info to Security and Access Control.
    3. Linked with Membership and Subscription (approvals, account suspension).
- Patrol Monitoring Subsystem
    1. Collects patrol logs via UI (Mobile App).
    2. Syncs data with Sync and Offline Functionality for offline mode.
    3. Reports compliance issues to Emergency and Alerts.
- Membership and Subscription Subsystem
    1. Connected to Payment and Billing for subscription validation.
    2. Shares status with User Management for account activation/suspension.
- Community Interaction Subsystem
    1. Uses UI for comments and discussions.
    2. Relies on Notification and Alerts for updates.
- Emergency and Alert Subsystem
    1. Triggers notifications to officers/members via UI and external SMS/email.
    2. Informs Admins (via Web UI) when patrol anomalies occur.

Support Layer

- Reporting and Analytics Subsystem
    1. Pulls data from Patrol Monitoring, Membership and Subscription, and Payment and Billing.
    2. Displays insights through the UI.
- Payment and Billing Subsystem
    1. Works with Membership and Subscription for access rights.
    2. Interfaces with external payment gateways.
    3. Sends records to Reports and Analytics.
- Sync and Offline Functionality Subsystem
    1. Mediates between Mobile UI and Backend Services/Database.

2. Ensures data consistency when devices go online.

Infrastructure Layer

- Security and Access Control Subsystem
    1. Protects all data exchanges across subsystems.
    2. Works with User Management and Payment and Billing for authentication and encryption.
- System Maintenance and Configuration Subsystem
    1. Provides admins with system health monitoring, backups, and performance scaling.
    2. Supports all subsystems indirectly.
- Backend Services
    1. Central hub managing business logic across subsystems.
    2. Connects UI to the Database Subsystem.
- Database Subsystem
    1. Stores all persistent data: users, patrol logs, payments, alerts, reports.
    2. Supports Reporting and Analytics, Patrol Monitoring, Membership and Subscription, and Community Interaction.

Four-Tier Architecture of the Neighbourhood Watch Application

Presentation Tier (UI Layer)

Purpose: It is the interface between users and the system. It manages input/output, data visualization, patrol dashboards, alerts, and reports.

Includes: UI Subsystem (Web for Admins, Mobile for Officers and Members)

Application/Logic Tier (Business Logic)

Purpose: Implements system functionality and rules.

Includes: User Management, Patrol Monitoring, Membership and Subscription, Community Interaction, Emergency and Alerts, Payment and Billing, Reporting and Analytics, System Maintenance and Configuration.

Integration/Service Tier (Middleware and Support Services)

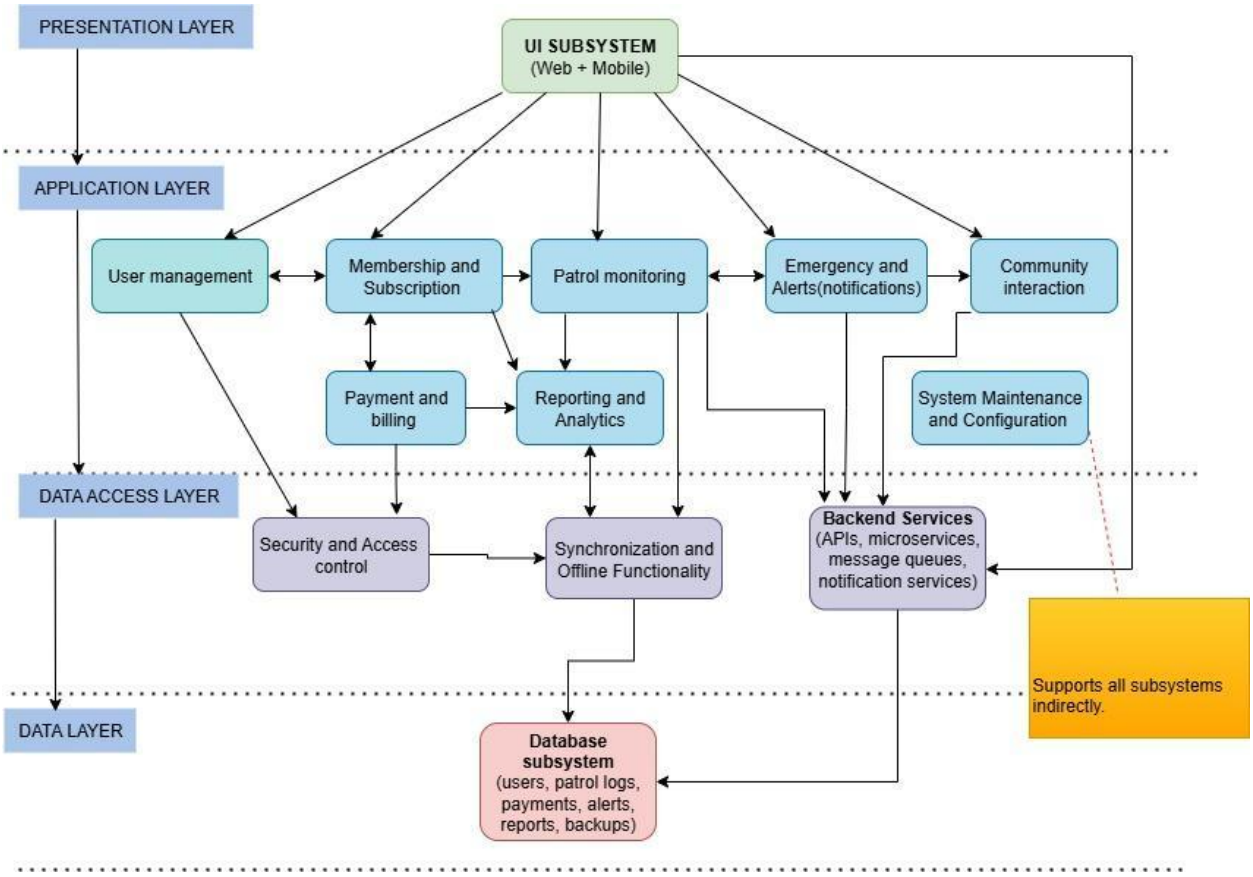
Purpose: Acts as a bridge between application logic and data services.

Includes: Synchronization and Offline Functionality, Security and Access Control, Backend Services.

Data Tier (Database & Storage)

Purpose: Centralized secure data management.

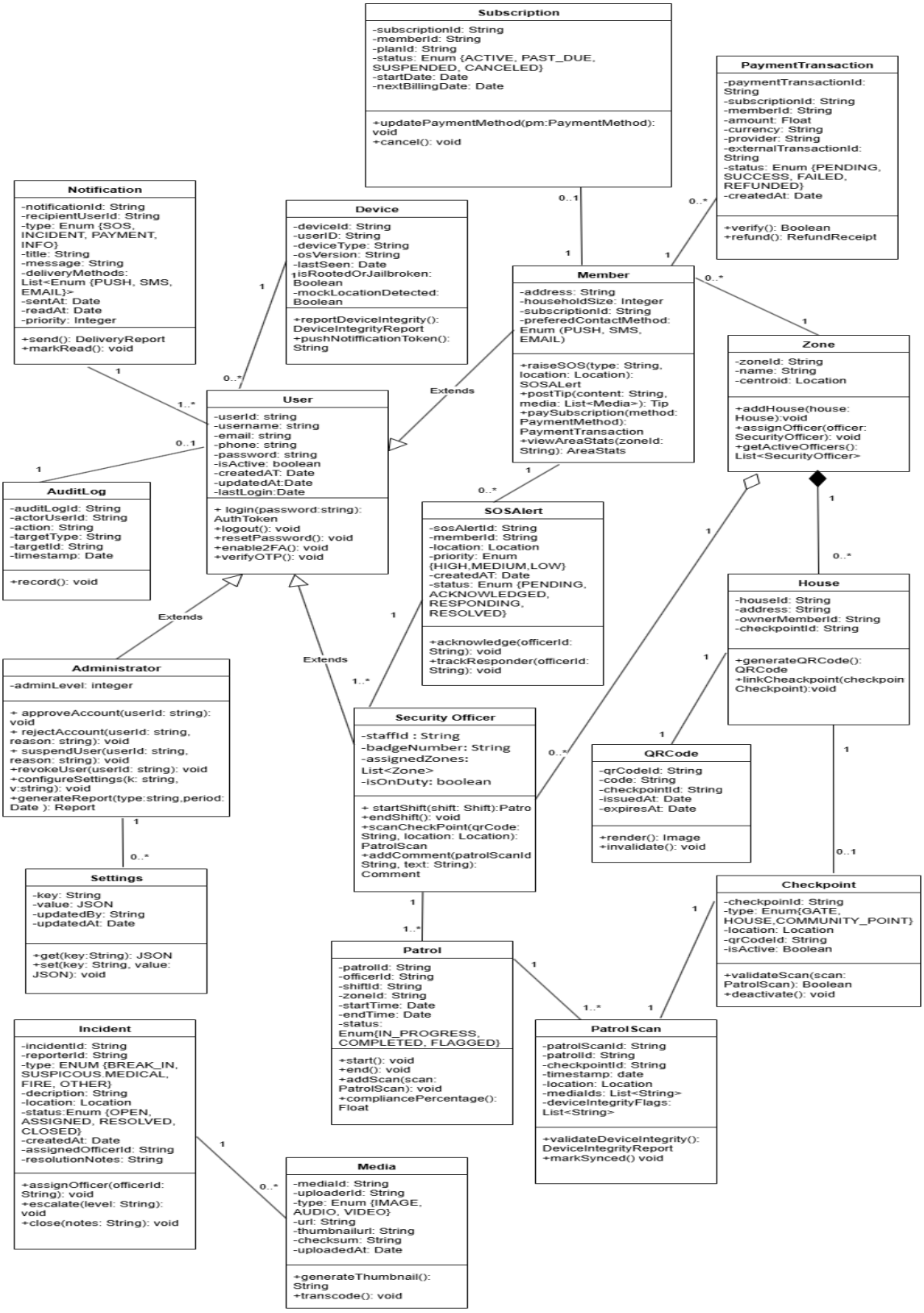
Includes: Database Subsystem, Data warehouses for analytics, backup systems



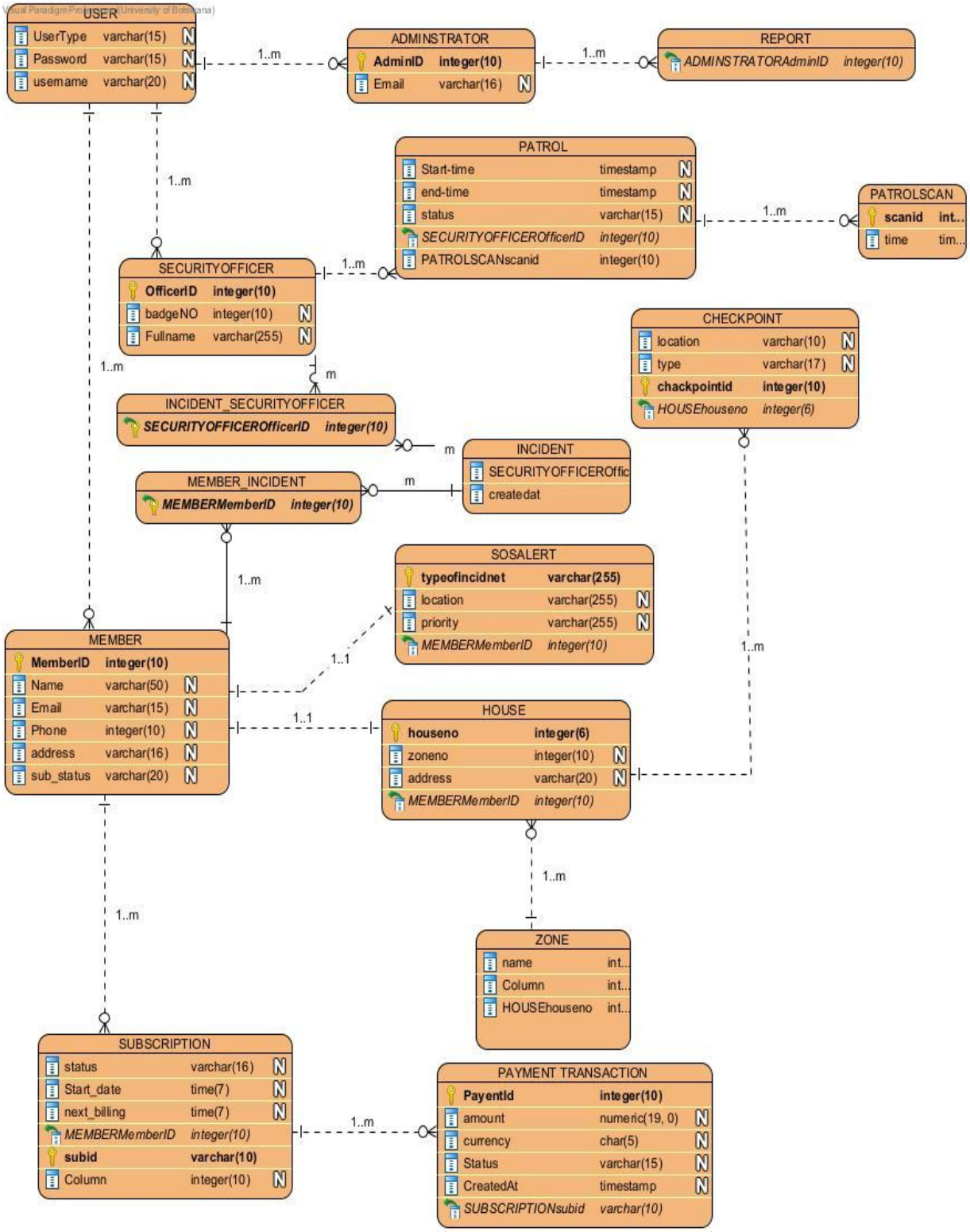
This setup ensures:

- Security (via dedicated Security & Access Control subsystem).
- Scalability (middleware for load balancing and APIs).
- Performance (Application layer separated from UI).
- Disaster Recovery (managed in Database layer and Maintenance subsystem).

Detailed Class Diagram



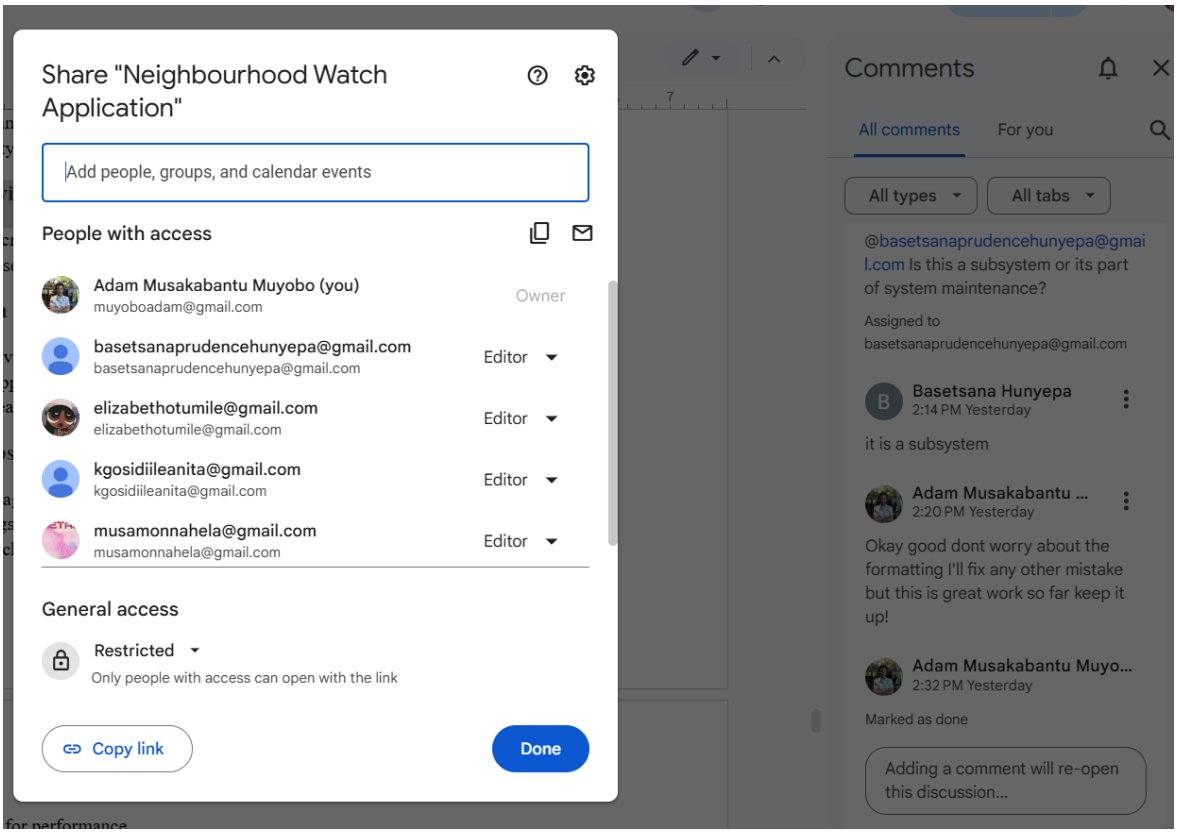
Relational Model



Collaboration Tools

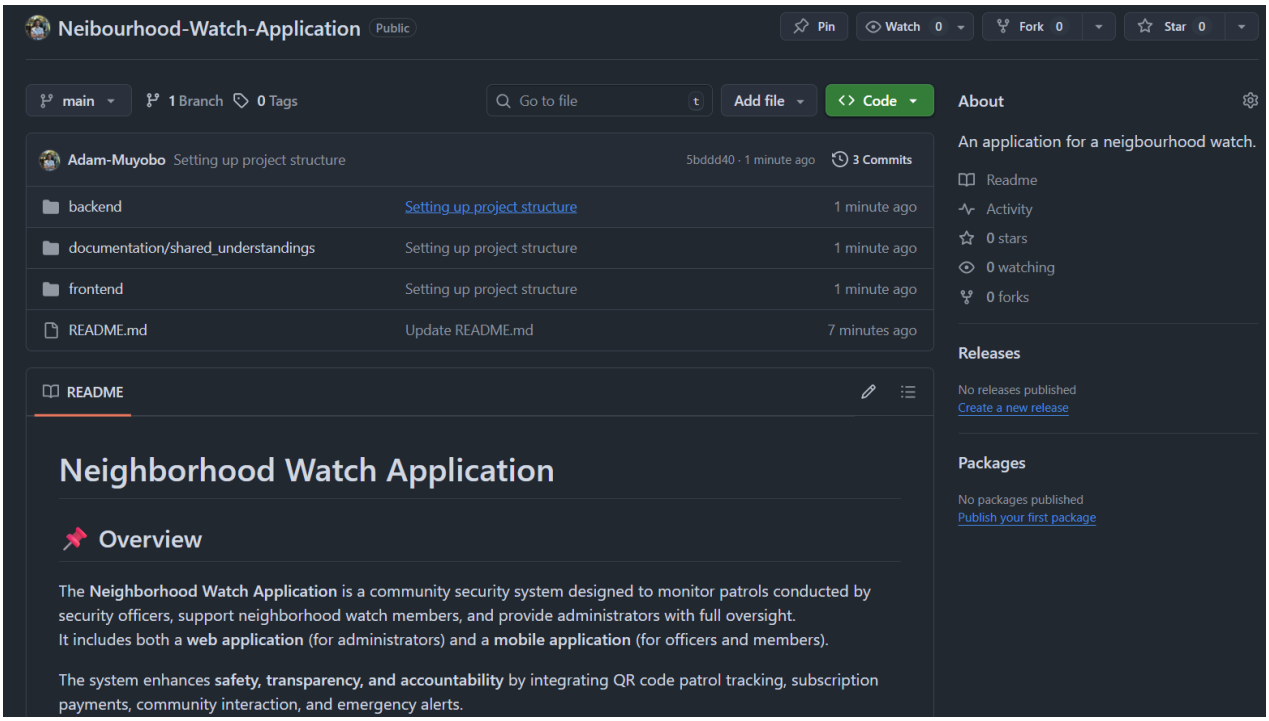
The collaboration tools used for this part of the project were Google Documents for a central document repository and simultaneous editing and GitHub for storing group documents and future code repository.

Google Documents



As you can see all members participating and using side comments to flag issues and resolve problems.

GitHub



Just a display of where our work will be stored and a central code repository also a collaboration space.