Software Requirements Specification for Ekta Donation Website

Group 19

Group Mentor: Diti

Group Members:

202201315 Dani Jal Nileshbhai
202201281 Shah Sakshi Dharmeshkumar
202201354 Ram Kulkarni
202201292 Devamm Patel
202201275 Kunj Mahendra Bhuva
202201271 Rishabh Jain
202201362 Tanay Kewalramani
202201303 Pandya Ansh Ashutoshbhai
202201264 Yadav Alpeshkumar Banshbahadur
202201363 Mangukiya Harshkumar Ashwinbhai

Contents

Chapter 1

Introduction

1.1 Purpose of the Document

This document outlines the Software Requirements Specification (SRS) for the Ekta Group Donation Website. Its purpose is to provide an in-depth understanding of the system's functionality, objectives, and target audience, enabling clear communication between stakeholders, designers, and developers.

1.2 Overview of Ekta Group Donation System

The Ekta Group Donation Website aims to connect donors with individuals and organizations in need of donations. The platform allows donors to browse various donation categories, make secure donations, and track their donation history. Beneficiaries can list their needs, and admins can manage requests, ensuring donations are allocated effectively.

1.3 Scope

This project includes the creation of a web-based donation management system. The system will allow for user registration, donation tracking, beneficiary management, and provide analytics for admins. This document covers the functional and non-functional requirements of the system, laying the foundation for design, development, testing, and deployment.

Chapter 2

Requirements

2.1 Functional Requirements

1. **User Registration and Authentication**
    - Secure registration and login for donors, beneficiaries, and admins.
    - Role-based access control ensures users access only relevant features.
2. **Donation Management**
    - Donors can select categories and specify donation amounts or items.
    - Beneficiaries can request assistance, detailing their specific needs.
3. **Donation Tracking and History**
    - Donors can view their donation history, including amounts, categories, and recipient information.
    - Real-time updates on donation impact where applicable.
4. **Category and Cause Management**
    - Admins can create and manage donation categories (e.g., food, shelter, education) and specific causes within each category.
5. **Beneficiary Approval System**
    - Beneficiary requests require admin approval before appearing publicly.
    - Admins can view details, verify requests, and approve or deny them as needed.
6. **Notifications and Alerts**
    - Donors receive confirmation emails upon donation.
    - Beneficiaries are notified of request status updates (e.g., pending, approved, denied).
7. **Search and Filter Options**
    - Users can search for donation opportunities by location, category, or beneficiary need.

- Advanced filters help donors find causes that match their interests.

8. **Reporting and Analytics**
    - Admins can access analytics reports on donations (e.g., total donations, top donors, popular categories).
9. **Secure Payment Gateway**
    - A secure payment gateway allows donors to make payments safely.
    - The system records and manages transaction histories for transparency.

2.2 Non-Functional Requirements

- **Performance:** System responses should be efficient, especially during high-traffic events.
- **Reliability:** Ensure consistent uptime and data reliability.
- **Scalability:** Accommodate an increasing number of users and donation requests.
- **Usability:** Design a user-friendly interface that is easy to navigate for all ages.
- **Data Privacy:** Protect donor and beneficiary information per data privacy regulations.
- **Mobile Accessibility:** Make the platform mobile-friendly for ease of access.

Chapter 3

User Stories

3.1 Donor Stories

1. As a donor, I want to donate anonymously, so my personal information remains private.
    - **Acceptance:** Option to donate anonymously.
    - **Success:** Donation is processed without displaying the donor's name.
    - **Failure:** If the donation isn't processed anonymously, the donor's details are shown publicly.
2. As a donor, I want to save my payment details securely, so I can make donations faster in the future.
    - **Acceptance:** Donor is prompted to save payment details after donation.
    - **Success:** Payment details are saved securely for future donations.
    - **Failure:** If saving fails, donor receives an error, and details aren't stored.
3. As a donor, I want to filter beneficiaries by location, so I can support causes close to me.
    - **Acceptance:** Location filter is available on the beneficiary search.
    - **Success:** System displays beneficiaries within the selected location range.
    - **Failure:** If the filter fails, all beneficiaries are shown regardless of location.
4. As a donor, I want to receive periodic updates on my donations' impact, so I feel connected to the cause.
    - **Acceptance:** Donor can opt-in for updates during donation.
    - **Success:** Donor receives updates on how donations are used.
    - **Failure:** If updates fail, donor receives an error message and no updates.
5. As a donor, I want to add comments or messages with my donations, so I can share words of encouragement.
    - **Acceptance:** Comment box is available in the donation form.

- **Success:** Comment is successfully saved and sent with the donation.
- **Failure:** If saving fails, donor receives an error, and the comment isn't submitted.

6. As a donor, I want to receive tax receipts for my donations, so I can claim deductions where applicable.
    - **Acceptance:** Donors can download or receive tax receipts after donation.
    - **Success:** Donor receives a valid tax receipt.
    - **Failure:** If the receipt cannot be generated, an error appears, and the donor is prompted to retry.

3.2 Beneficiary Stories

1. As a beneficiary, I want to edit my request details, so I can update information if my needs change.
    - **Acceptance:** Beneficiaries can edit request details from their profile.
    - **Success:** Changes are saved and displayed to donors.
    - **Failure:** If editing fails, an error appears, and changes are not saved.
2. As a beneficiary, I want to add images or documents to my requests, so I can provide more context for donors.
    - **Acceptance:** Option to upload images and documents is available.
    - **Success:** Uploaded images/documents are displayed on the request page.
    - **Failure:** If upload fails, an error message is shown, and no media is added.
3. As a beneficiary, I want to withdraw my request if my needs are met, so resources are used efficiently.
    - **Acceptance:** Beneficiaries have a "Withdraw Request" option.
    - **Success:** Request is marked as withdrawn and no longer visible.
    - **Failure:** If withdrawal fails, an error appears, and the request remains active.
4. As a beneficiary, I want to receive notifications about new donations, so I know when help is on the way.
    - **Acceptance:** Beneficiary is notified when a new donation is made.
    - **Success:** Notifications are sent to the beneficiary's email.
    - **Failure:** If notifications fail, an error message appears, and the beneficiary receives no updates.

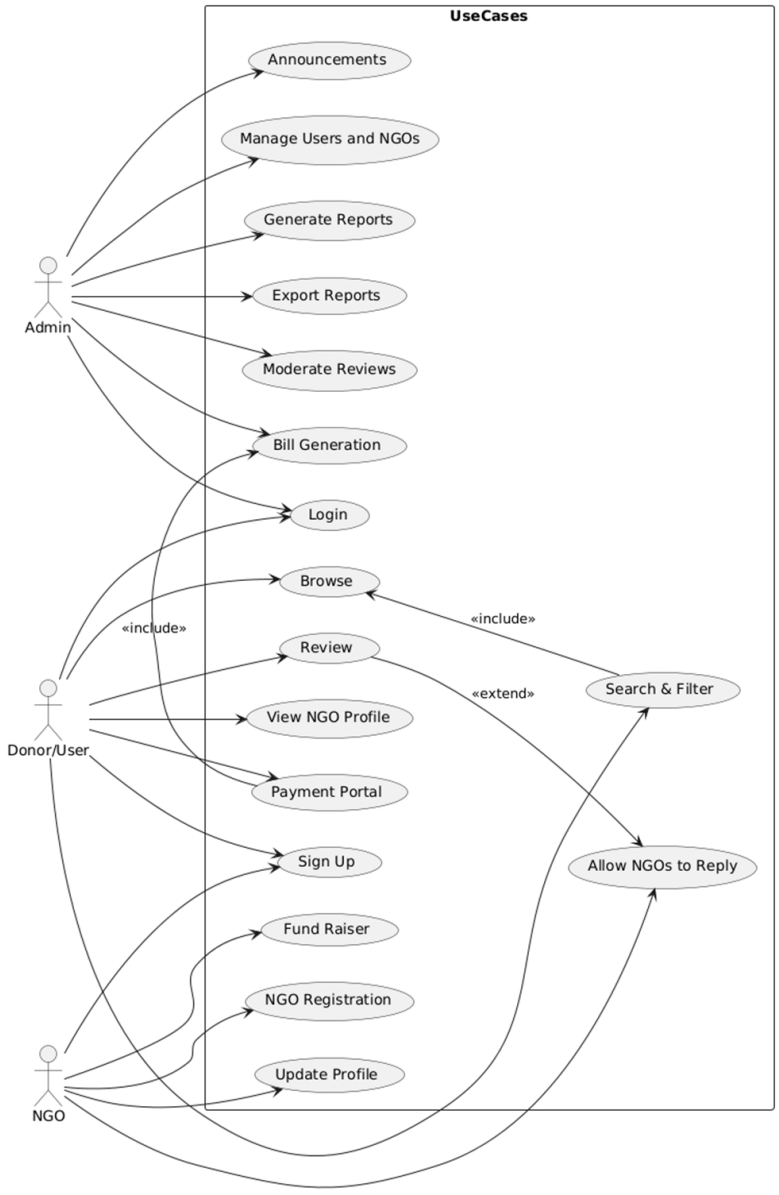
3.3 Admin Stories

1. As an admin, I want to monitor user activities, so I can detect and prevent any misuse of the platform.
    - **Acceptance:** Admins have access to an activity log or dashboard.
    - **Success:** Activities are logged and displayed on the admin dashboard.
    - **Failure:** If logging fails, an error appears, and logs are incomplete.
2. As an admin, I want to track recurring donations, so I can plan and allocate resources accordingly.
    - **Acceptance:** Recurring donations are marked on the admin dashboard.
    - **Success:** Recurring donations are tracked and displayed.
    - **Failure:** If tracking fails, recurring donations are not highlighted, and an error message appears.
3. As an admin, I want to generate custom reports on donation trends, so I can better understand user behavior.
    - **Acceptance:** Reporting tool is available on the admin dashboard.
    - **Success:** Report is generated with selected filters.
    - **Failure:** If report generation fails, an error message is shown, and no report is created.

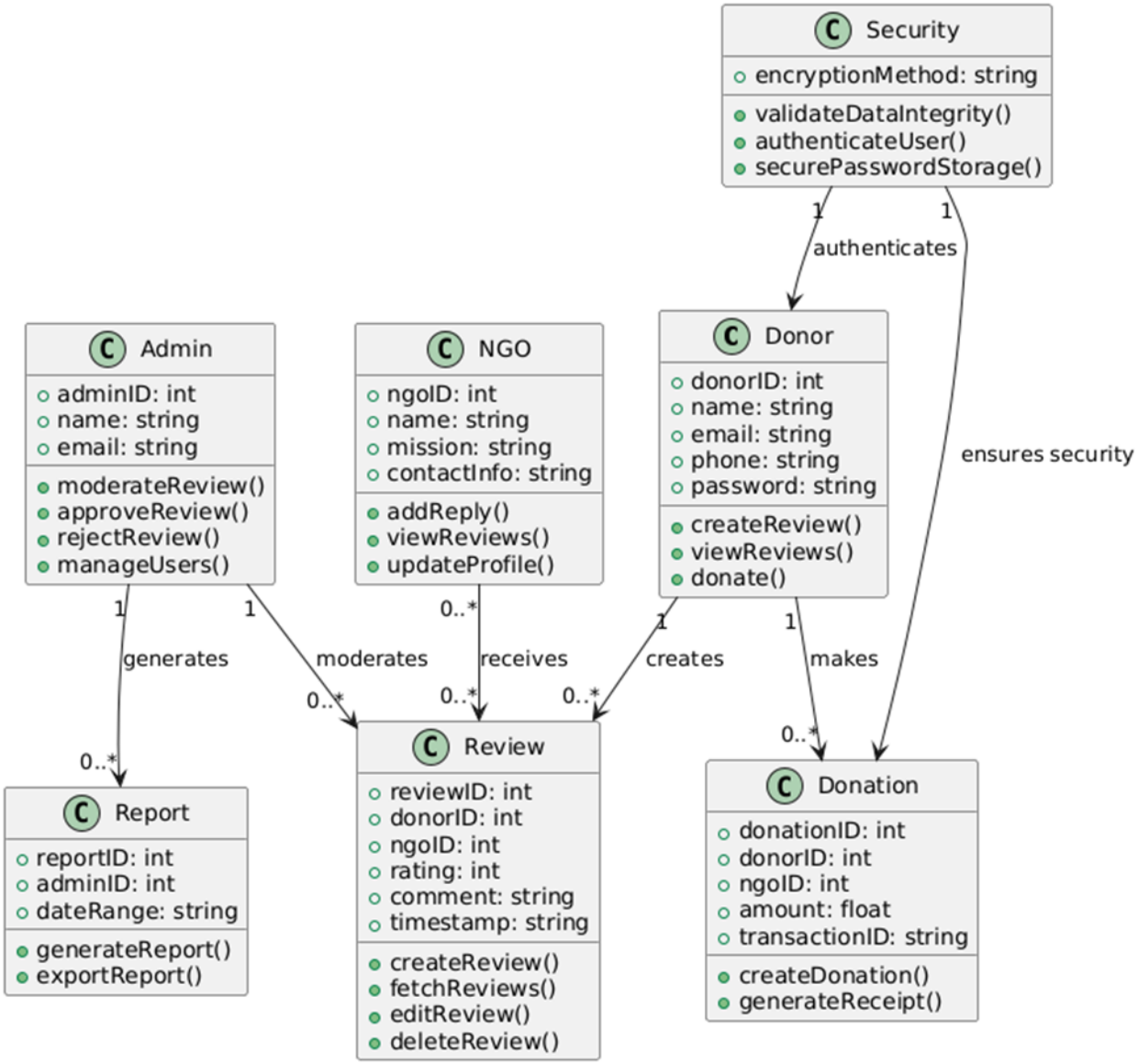
Chapter 4

System Behaviour

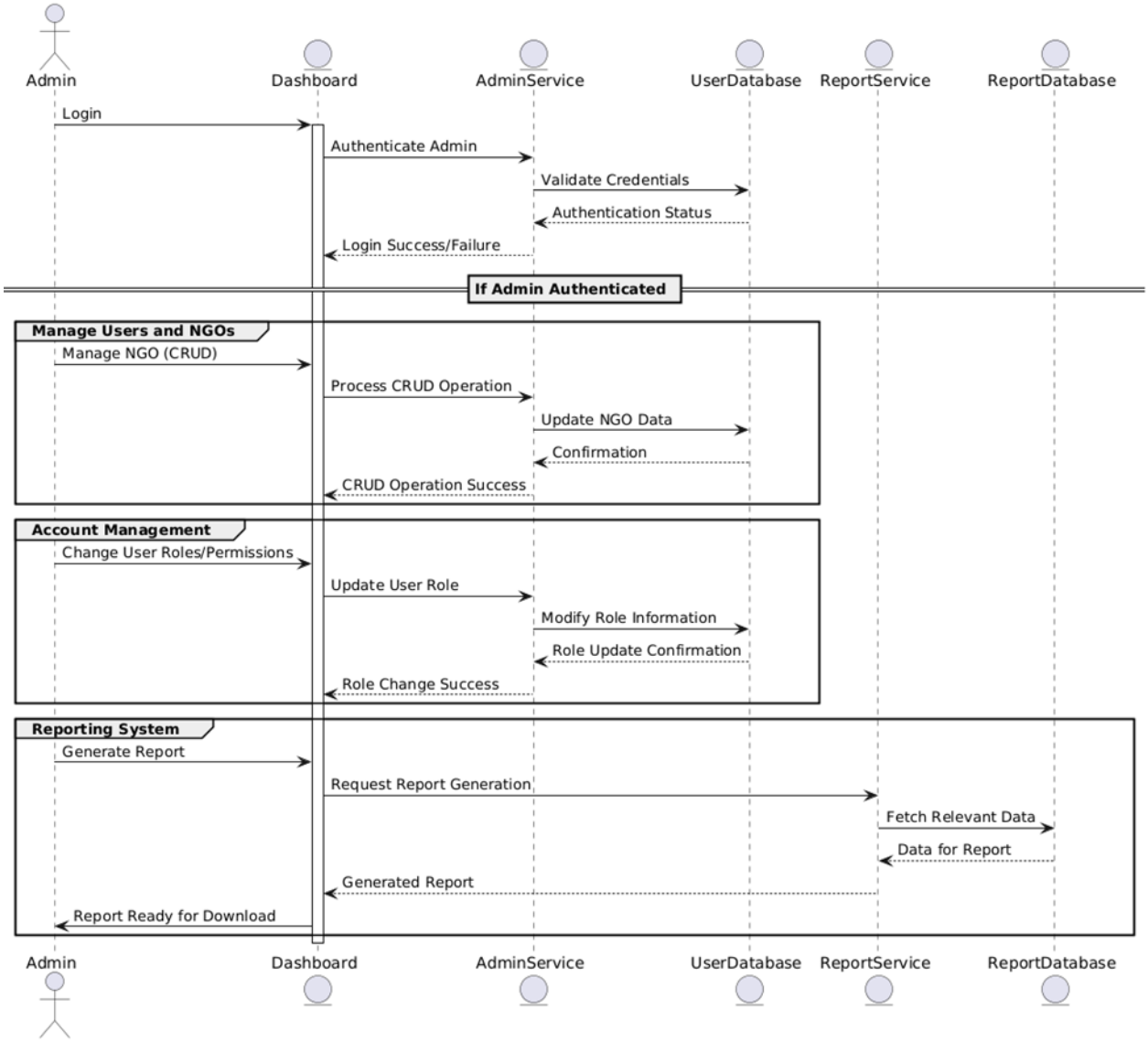
4.1 Use Case Diagram

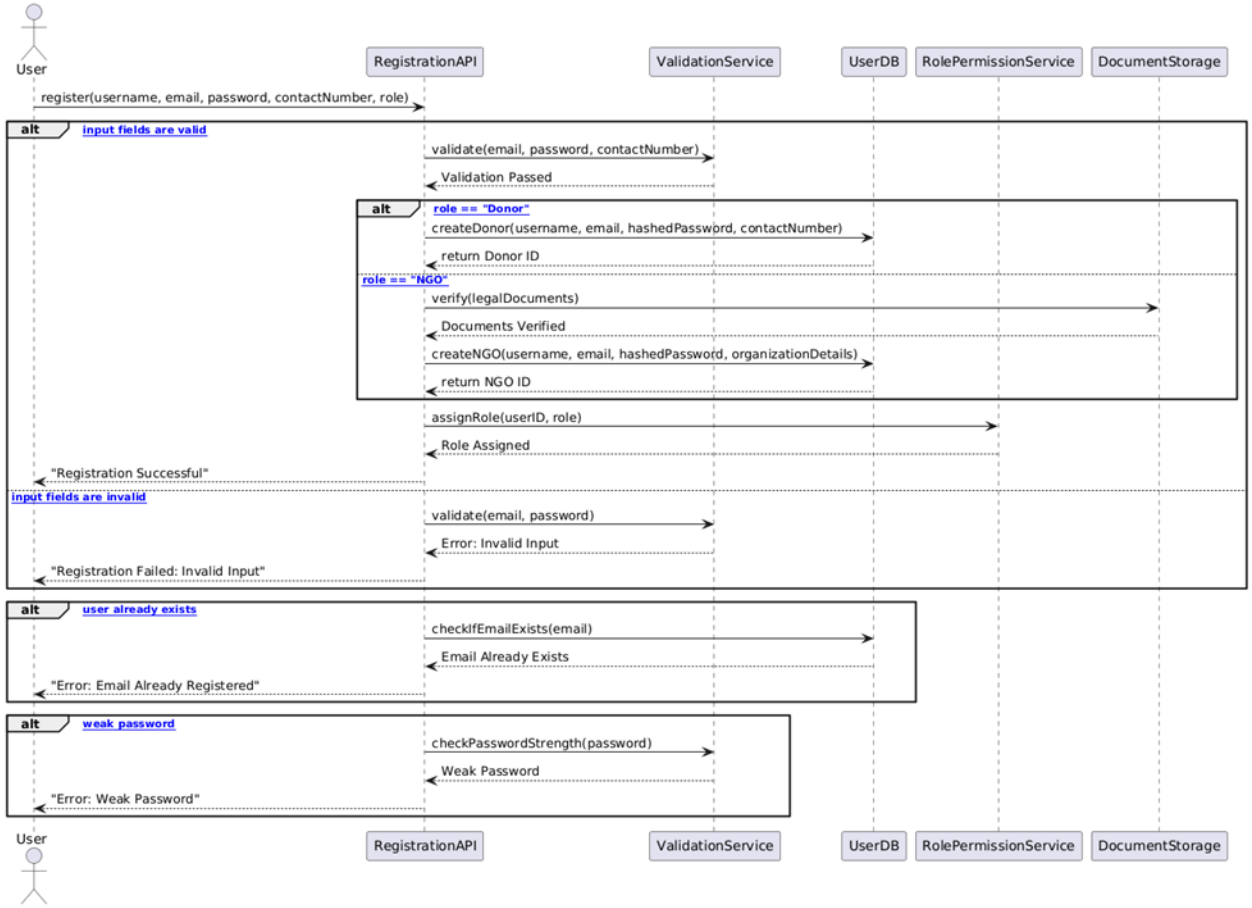


4.2 Class Diagram

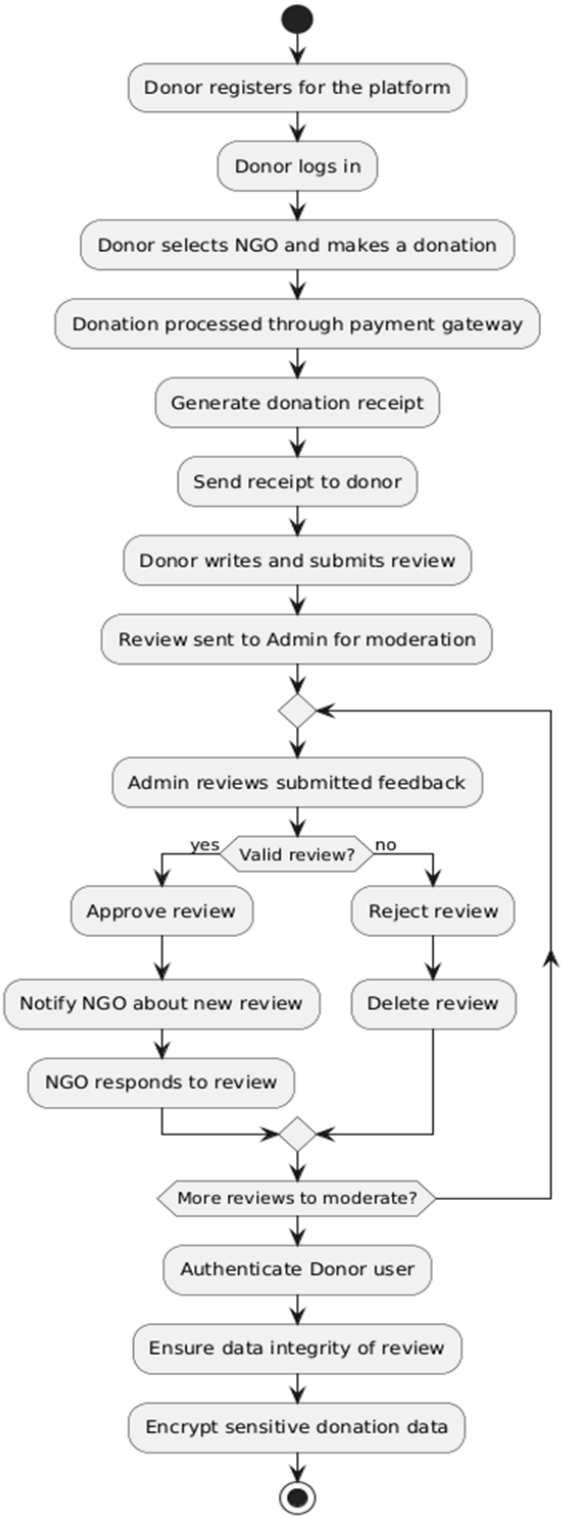


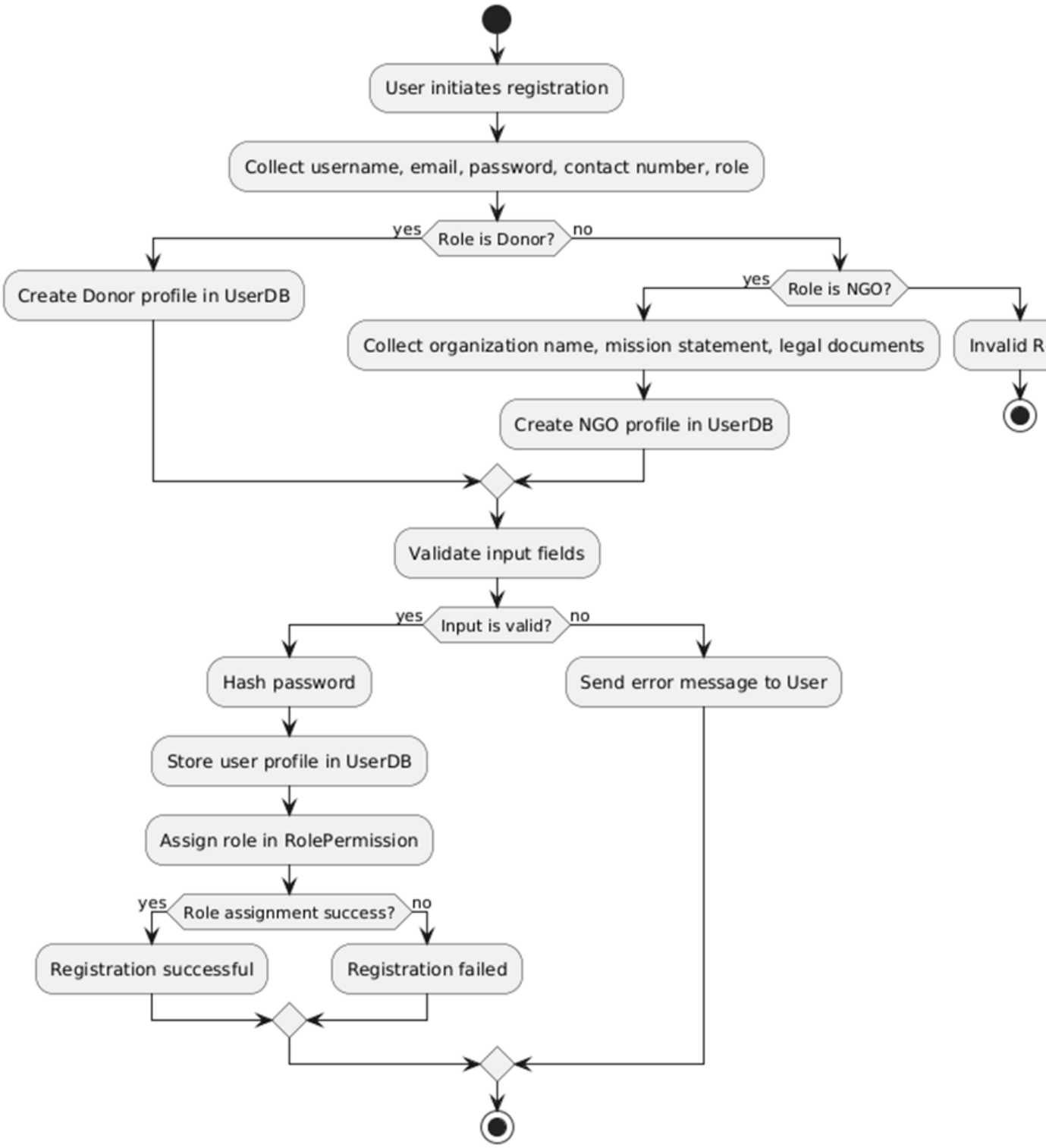
4.3 Sequence Diagram




User
RegistrationAPI
ValidationService
UserDB
RolePermissionService
DocumentStorage
register(username, email, password, contactNumber, role)
alt
input fields are valid
validate(email, password, contactNumber)
Validation Passed
alt
role == "Donor"
createDonor(username, email, hashedPassword, contactNumber)
return Donor ID
role == "NGO"
verify(legalDocuments)
Documents Verified
createNGO(username, email, hashedPassword, organizationDetails)
return NGO ID
assignRole(userID, role)
Role Assigned
"Registration Successful"
input fields are invalid
validate(email, password)
Error: Invalid Input
"Registration Failed: Invalid Input"
alt
user already exists
checkIfEmailExists(email)
Email Already Exists
"Error: Email Already Registered"
alt
weak password
checkPasswordStrength(password)
Weak Password
"Error: Weak Password"
User
RegistrationAPI
ValidationService
UserDB
RolePermissionService
DocumentStorage

4.4 Activity Diagram




User initiates registration
Collect username, email, password, contact number, role
Role is Donor?
yes
no
Create Donor profile in UserDB
Role is NGO?
yes
Collect organization name, mission statement, legal documents
Invalid R
Create NGO profile in UserDB
Validate input fields
Input is valid?
yes
no
Hash password
Send error message to User
Store user profile in UserDB
Assign role in RolePermission
Role assignment success?
yes
no
Registration successful
Registration failed

Chapter 5

Product Backlog

5.1 Sprint 1: User Registration and Authentication

Duration: 2 weeks
Stories:

- **Donor Registration**
  - Backend Story: User Registration API
  - Tasks:
    - Implement API for user registration.
    - Validate input fields.
    - Hash and securely store passwords.
    - Include optional fields like contact numbers.
    - Set up user roles and permissions.
- **NGO Registration**
  - Backend Story: NGO Registration API
  - Tasks:
    - Implement API for NGO registration.
    - Store organization details, mission statements, and legal documents.
    - Create a profile for each NGO.

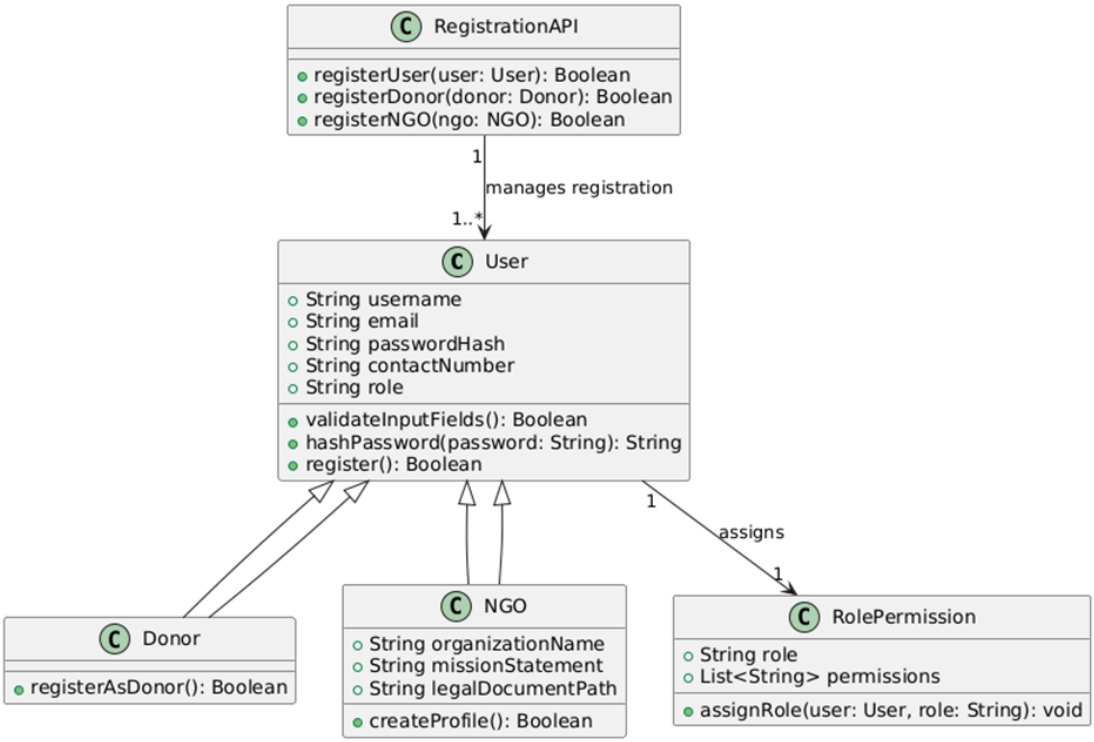
Estimated Effort: 42.9 $\sim$ 43 FP

Functionality and Effort Calculation

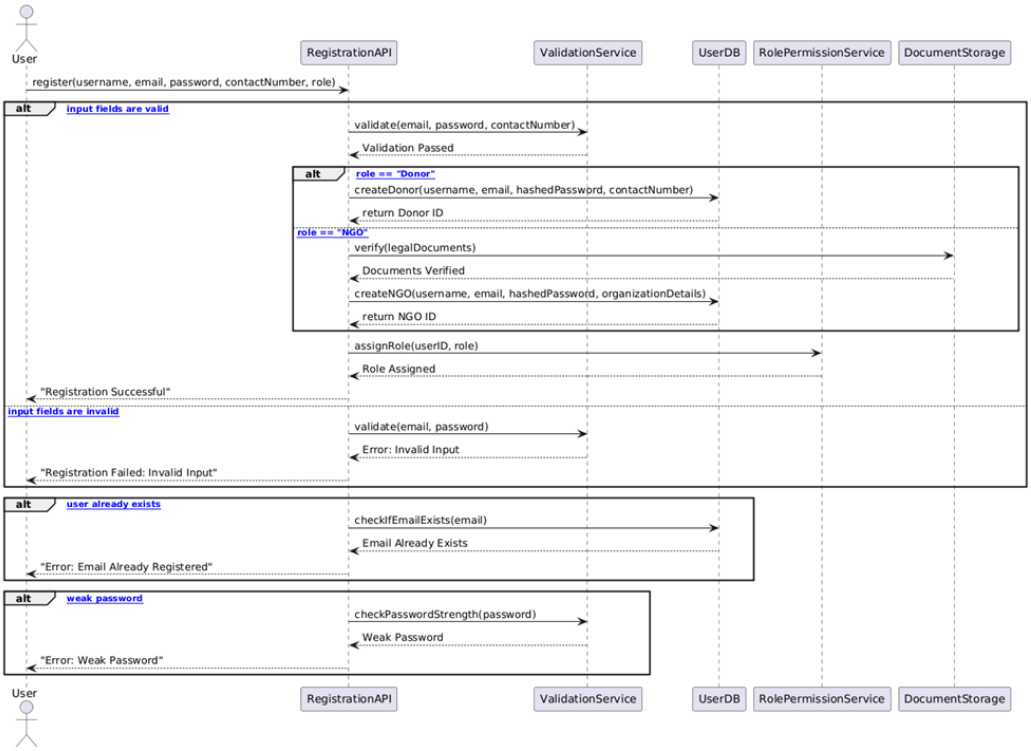
Functionality	UFP	AFP	Total FP
API for user registration	$(1\times4)+(1\times5)+(1\times4)+(1\times7)+(0\times5) = 20$	0.65	13
Validate input fields	$(1\times4)+(0\times5)+(0\times4)+(0\times7)+(0\times5) = 4$	0.65	2.6
Hash and securely store passwords	$(1\times4)+(0\times5)+(0\times4)+(1\times7)+(0\times5) = 11$	0.65	7.15
Include optional fields	$(1\times4)+(0\times5)+(0\times4)+(1\times7)+(0\times5) = 11$	0.65	7.15
Set up roles & permissions	$(1\times4)+(1\times5)+(1\times4)+(1\times7)+(0\times5) = 20$	0.65	13

Diagrams

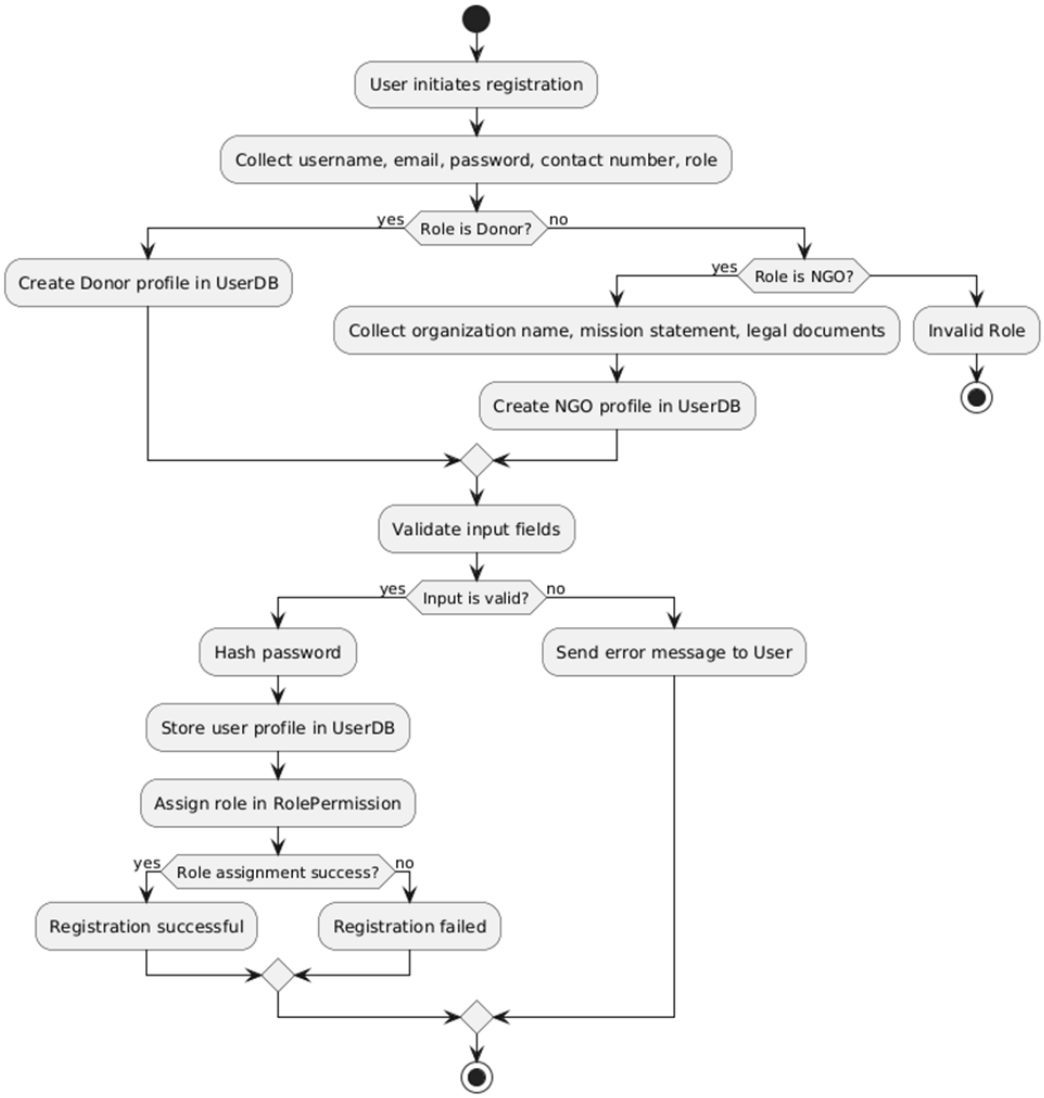
- Class Diagram



- Sequence Diagram



- Activity Diagram



5.2 Sprint 2: Admin Verification and NGO Listing

Duration: 2.5 weeks
Stories:

- **Admin Verification**
  - Backend Story: Verification API
  - Tasks:
    - Implement verification process for NGO documents.
    - Include status updates and notifications.
- **Search and Filter NGOs**
  - Backend Story: NGO Listing and Filtering API
  - Tasks:
    - Create API to retrieve NGO profiles with filtering options.
    - Optimize database queries for search and filter operations.

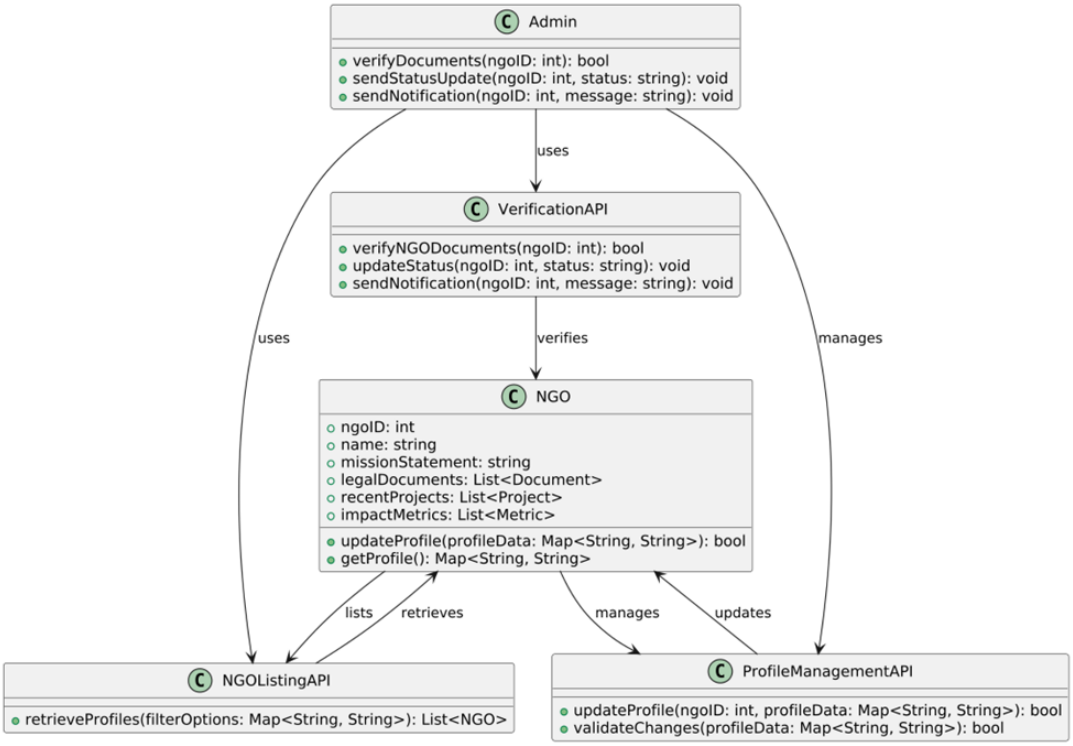
Estimated Effort: 49 FP

Functionality and Effort Calculation

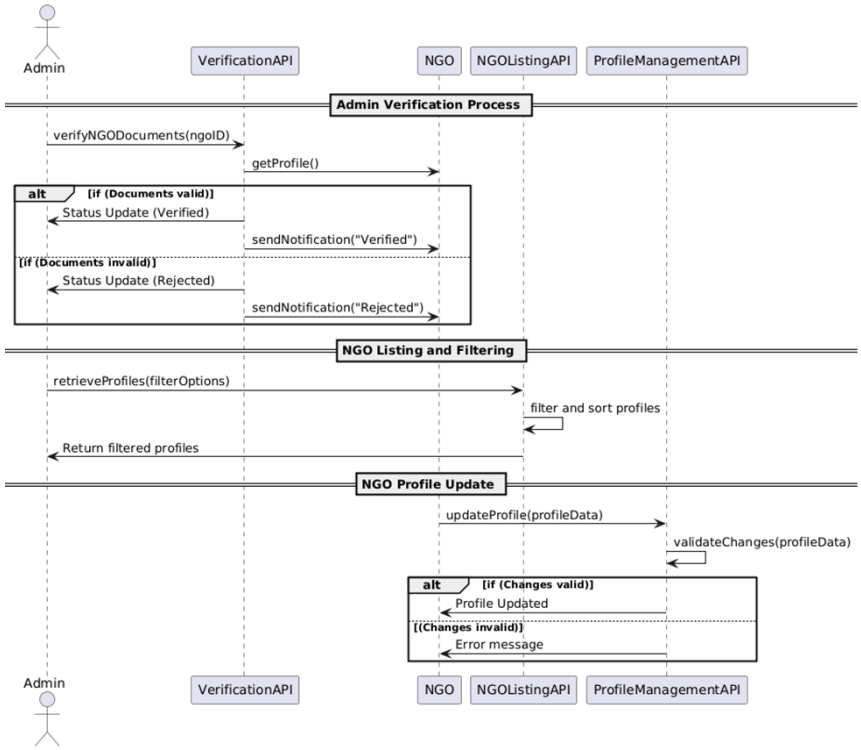
Functionality	UFP	AFP	Total FP
Verification NGO	(1×4)+(1×5)+(1×4)+(1×7)+(0×5) = 20	0.65	13
Status updates & notifications	(1×4)+(1×5)+(1×4)+(1×7)+(0×5) = 20	0.65	13
Retrieve NGO profile	(1×4)+(1×5)+(1×4)+(1×7)+(0×5) = 20	0.65	13
Validity for changes & integrity	(1×4)+(0×5)+(1×4)+(1×7)+(0×5) = 15	0.65	9.75
NGO to update projects	(1×4)+(1×5)+(0×4)+(1×7)+(0×5) = 16	0.65	10.4

Diagrams

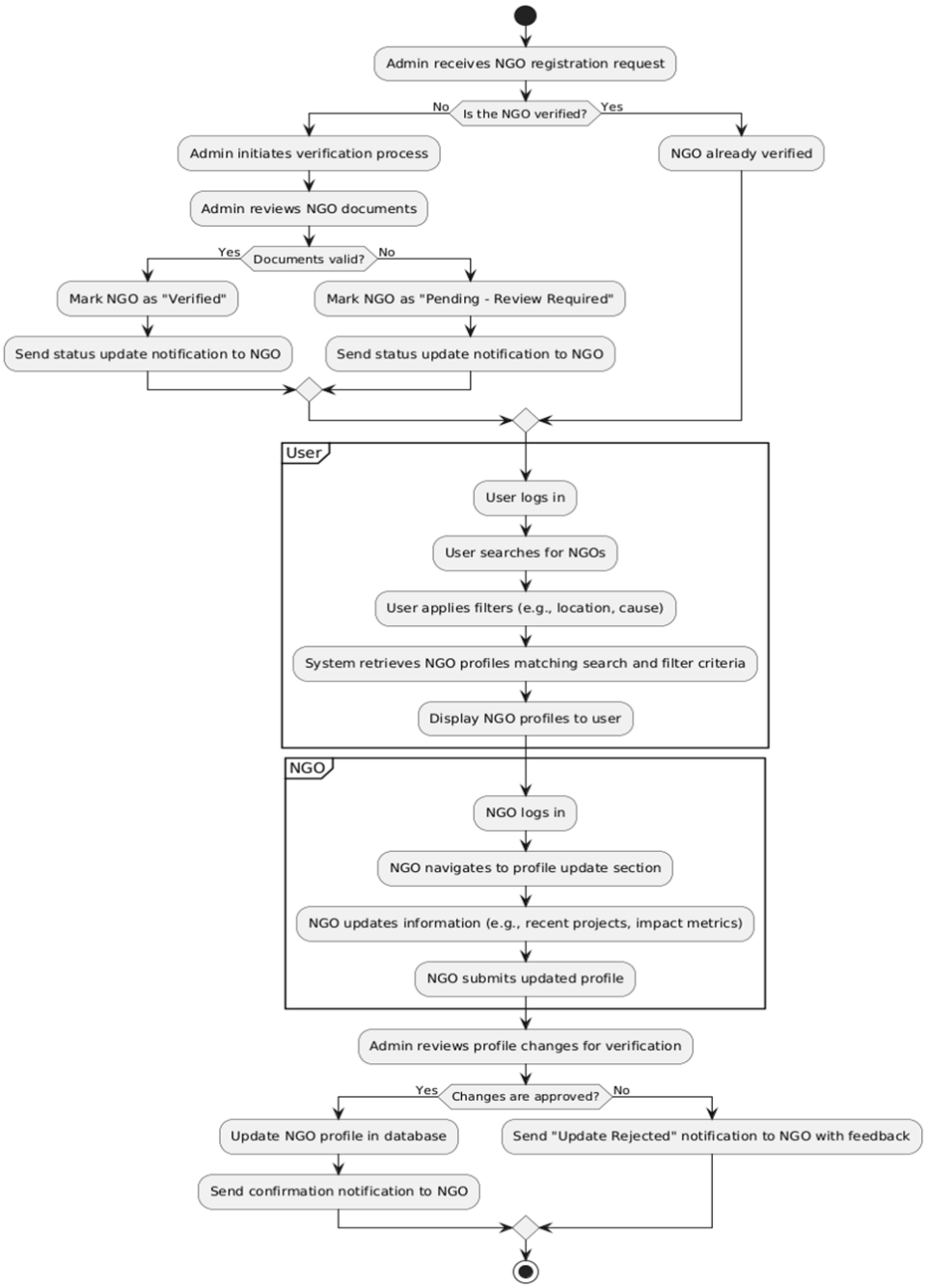
- Class Diagram



- Sequence Diagram



- Activity Diagram



5.3 Sprint 3: Donation Management

Duration: 3 weeks
Stories:

- **Donation Process**
  - Backend Story: Payment Processing API
  - Tasks:
    - Integrate payment gateways.
    - Ensure secure processing and storage of payment information.
    - Generate donation receipts and transaction IDs.
- **NGO Donation Management**
  - Backend Story: Bill Generation API
  - Tasks:
    - Create a billing system for receipts.
    - Store transaction records securely.
    - Provide downloadable donation reports.

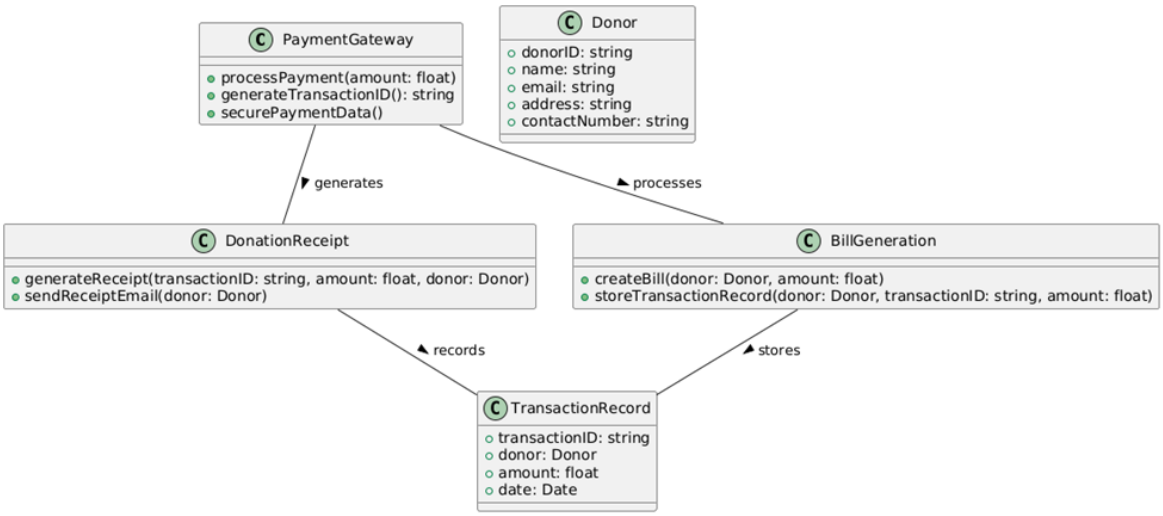
Estimated Effort: 61 FP

Functionality and Effort Calculation

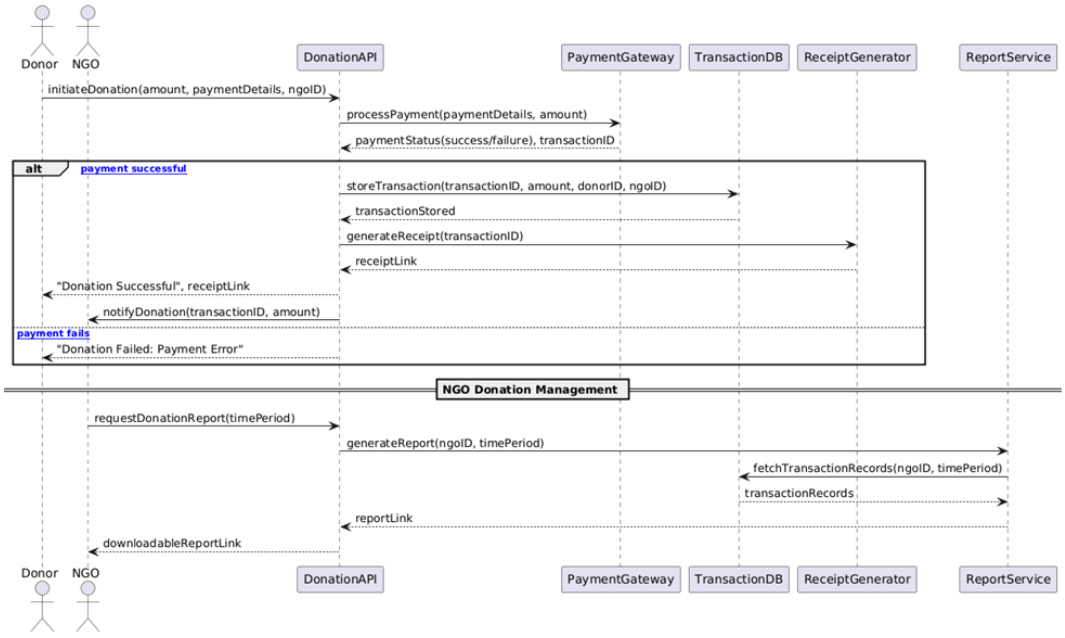
Functionality	UFP	AFP	Total FP
Integrated payment gateways	(1×4)+(1×5)+(1×4)+(0×7)+(1×5) = 18	0.65	11.7
Generate donation receipts	(1×4)+(1×5)+(1×4)+(1×7)+(0×5) = 17	0.65	11.05
Create a billing system for receipts	(2×4)+(2×5)+(1×3)+(1×7)+(0×5) = 23	0.65	15.6
Store transaction records securely	(0×3)+(1×4)+(0×3)+(1×7)+(0×5) = 11	0.65	7.15
Provide donation reports	(2×4)+(1×5)+(1×4)+(1×7)+(0×5) = 23	0.65	15.6

Diagrams

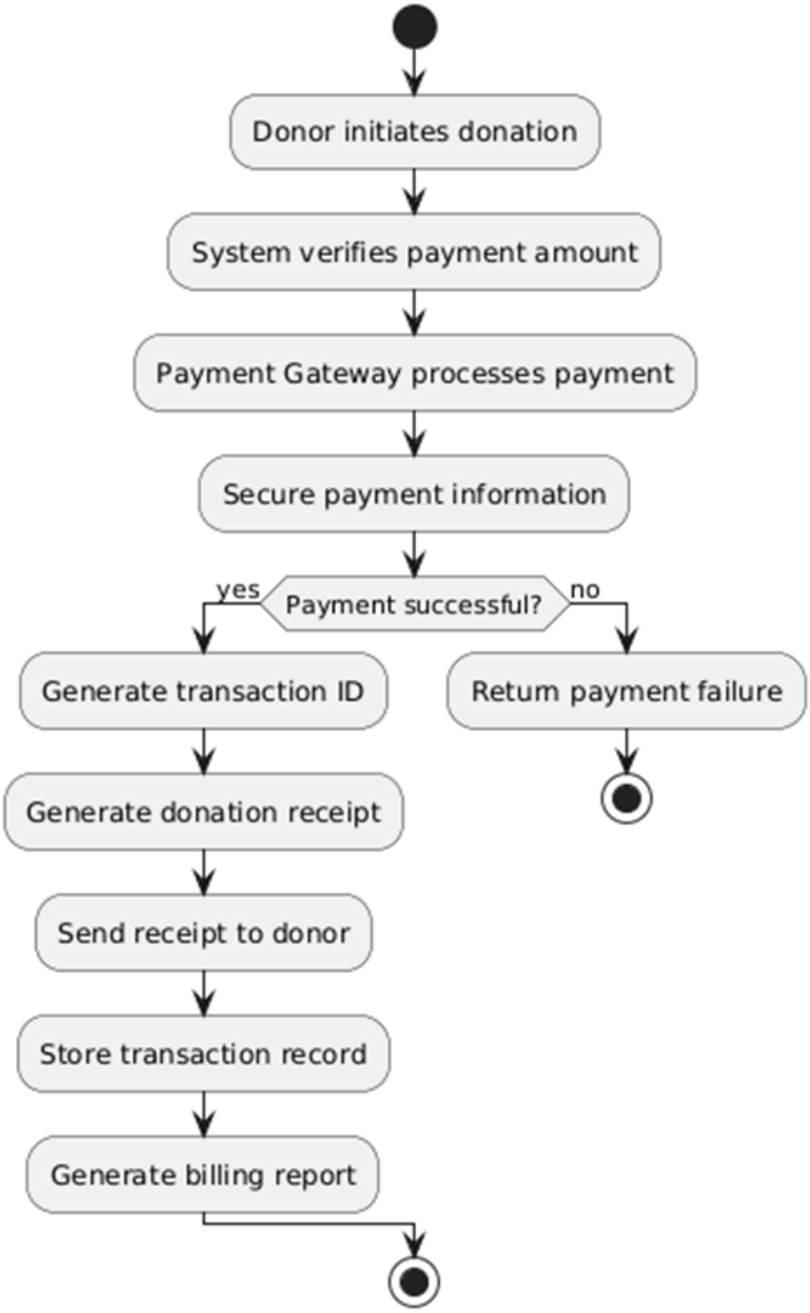
- Class Diagram



- Sequence Diagram



- Activity Diagram



5.4 Sprint 4: Admin Dashboard

Duration: 2.5 weeks
Stories:

- **Manage Users and NGOs**
  - Backend Story: Admin Management System
  - Tasks:
    * Implement CRUD operations for NGOs, donors, and admins.
    * Create tools for account management.
    * Provide auditing capabilities.
- **Reporting System**
  - Backend Story: Reporting System API
  - Tasks:
    * Create APIs to generate reports.
    * Allow exporting of reports in formats like PDF or CSV.

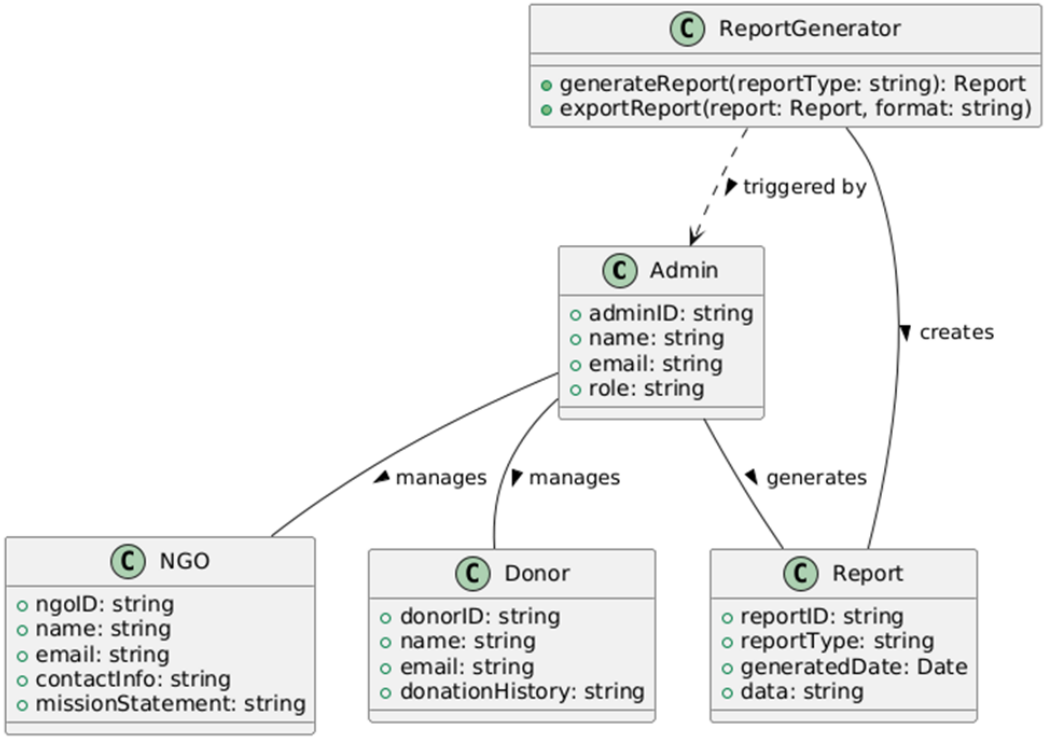
Estimated Effort: 53 FP

Functionality and Effort Calculation

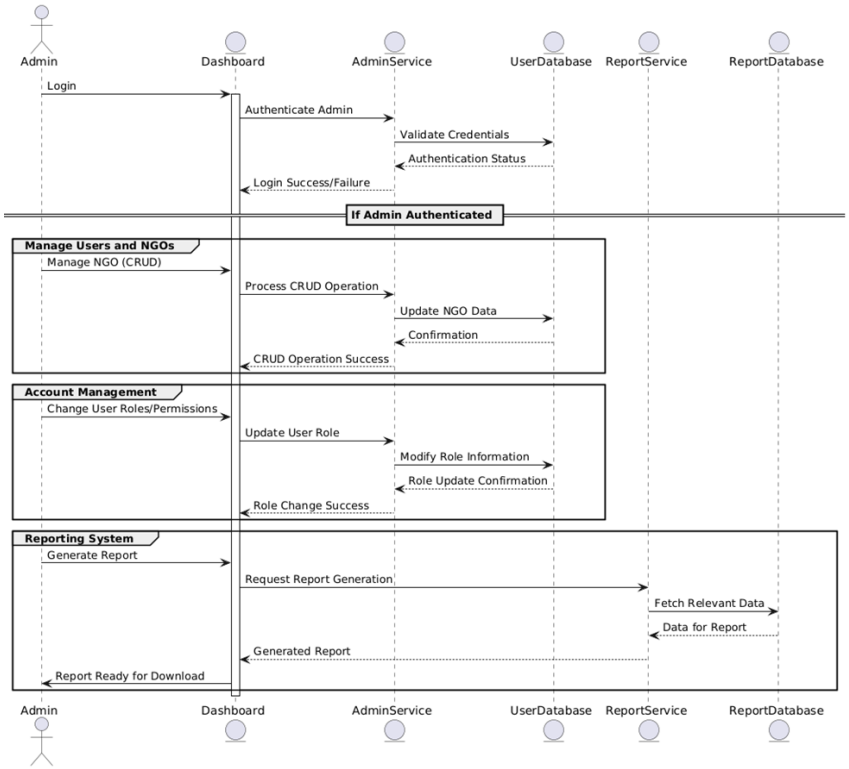
Functionality	UFP	AFP	Total FP
Implement CRUD operations	(1×4)+(1×5)+(1×4)+(1×7)+(0×5) = 20	0.65	13
Account management	(1×4)+(0×5)+(0×4)+(1×7)+(0×5) = 11	0.65	7.15
Provide auditing capabilities	(1×4)+(1×5)+(1×4)+(1×7)+(0×5) = 20	0.65	13
Generate reports	(1×4)+(1×5)+(1×4)+(1×7)+(0×5) = 20	0.65	13
Allow export data	(1×4)+(0×5)+(0×4)+(1×7)+(0×5) = 11	0.65	7.15

Diagrams

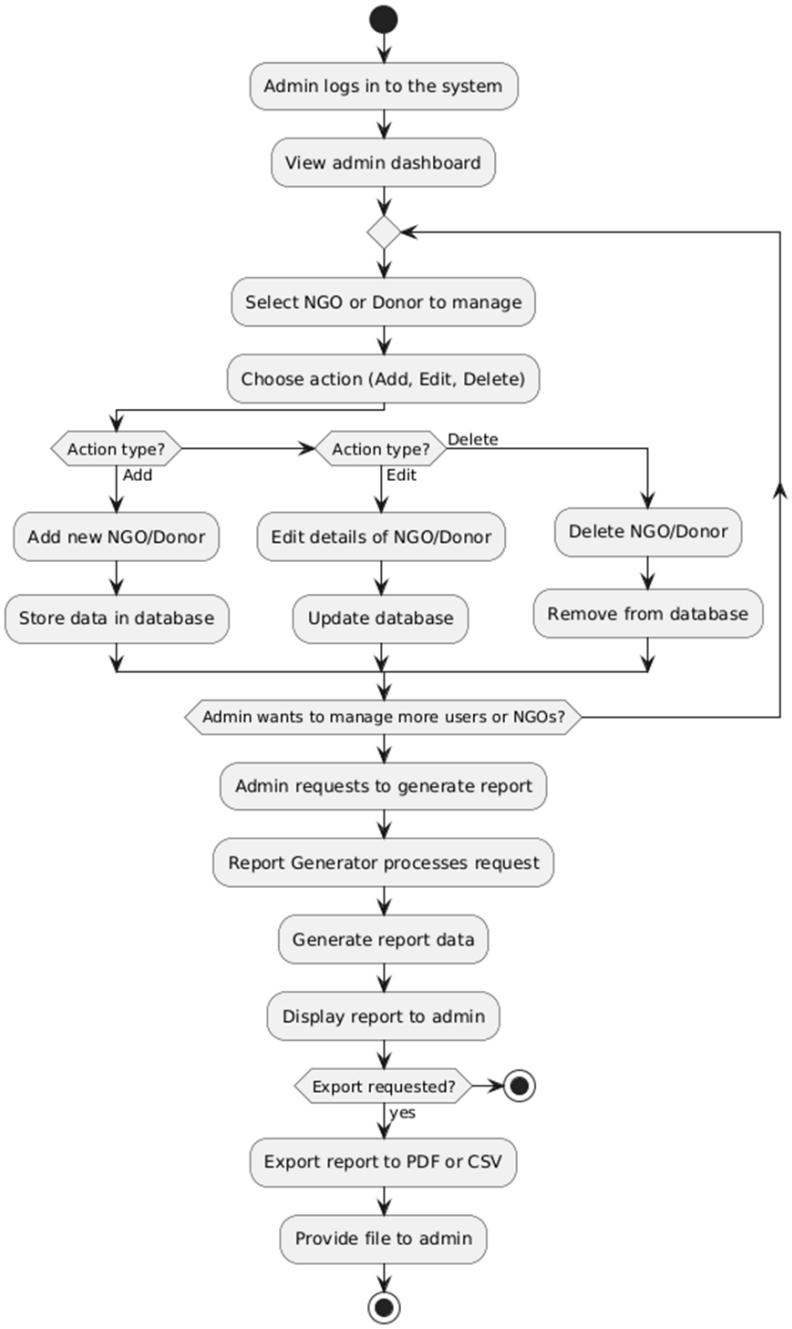
- Class Diagram



- Sequence Diagram



- Activity Diagram



5.5 Sprint 5: User Feedback and Support

Duration: 1.5 weeks
Stories:

- **Donor Feedback**
  - Backend Story: Review System API
  - Tasks:
    - Implement API for creating and fetching reviews.
    - Allow NGOs to reply to reviews.
    - Ensure reviews are moderated by admins.
- **NGO Response**
  - Backend Story: Review Response API
  - Tasks:
    - Enable NGOs to respond to feedback.

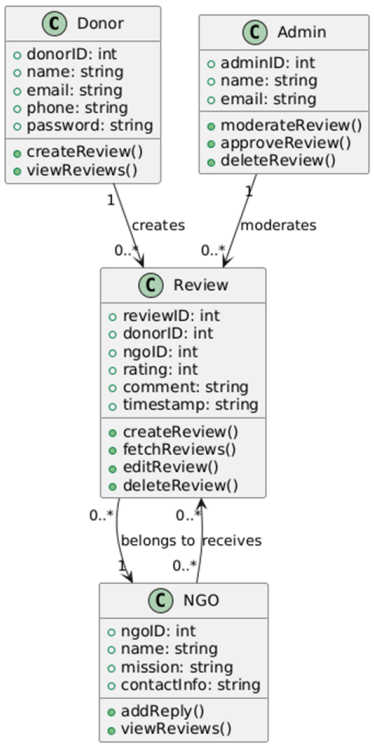
Estimated Effort: 33.15 FP

Functionality and Effort Calculation

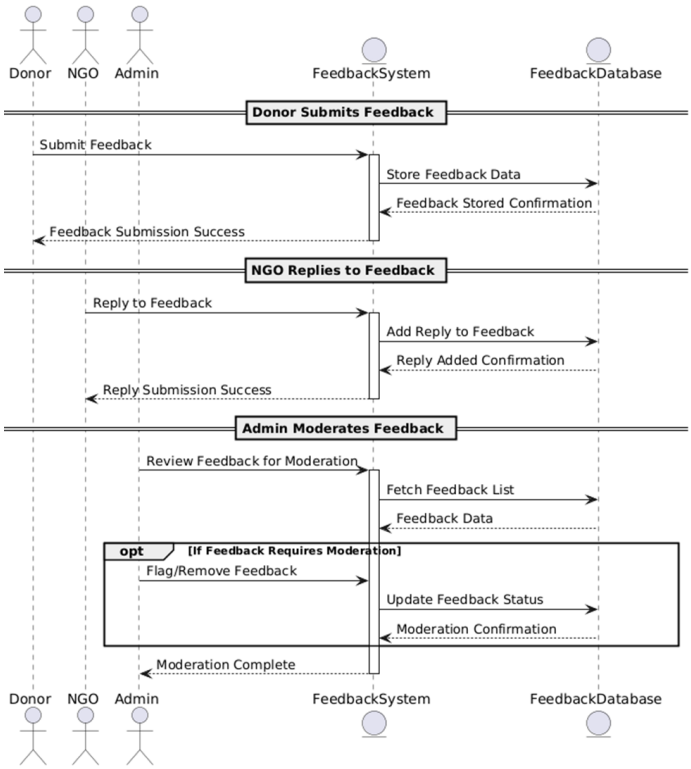
Functionality	UFP	AFP	Total FP
Review system	(1×4)+(1×5)+(1×4)+(1×7)+(0×5) = 20	0.65	13
Allow NGOs to reply	(1×4)+(0×5)+(0×4)+(1×7)+(0×5) = 11	0.65	7.15
Reviews moderation by admins	(1×4)+(1×5)+(1×4)+(1×7)+(0×5) = 20	0.65	13

Diagrams

- Class Diagram



- Sequence Diagram



5.6 Sprint 6: Advanced Reporting and Analytics

Duration: 3 weeks
Stories:

- **Customizable Reports**
  - Backend Story: Advanced Reporting API
  - Tasks:
    - Implement APIs to generate customizable reports.
    - Allow filtering by date range, NGO, and donor demographics.
    - Enable advanced visualizations like charts and graphs.
- **Analytics Dashboard**
  - Backend Story: Analytics API
  - Tasks:
    - Build APIs to fetch real-time analytics.
    - Integrate data visualization libraries for charts.
    - Include KPIs like donation trends, top donors, and NGO impact metrics.

Estimated Effort: 58 FP

Functionality and Effort Calculation

Functionality	UFP	AFP	Total FP
Generate customizable reports	(2×4)+(2×5)+(1×4)+(1×7)+(0×5) = 26	0.65	16.9
Advanced visualizations	(1×4)+(1×5)+(1×4)+(1×7)+(0×5) = 20	0.65	13
Real-time analytics API	(1×4)+(1×5)+(1×4)+(1×7)+(0×5) = 20	0.65	13
Data visualization KPIs	(1×4)+(1×5)+(0×4)+(1×7)+(0×5) = 16	0.65	10.4

Chapter 6

Proposed Model for Development

6.1 Model Chosen

The agile model is chosen for developing the NGO Donation App. This model allows for frequent feedback and iteration, ensuring that the system meets user needs. Each sprint's outcome is reviewed and improved based on stakeholder feedback, enabling the team to refine features like donation processing, user registration, and reporting capabilities.

6.2 Requirements Gathering

Objective: Collaborate with stakeholders (e.g., NGOs, donors, admins) to fully understand the needs and features required for the donation system.
Activities:

- Conduct user interviews to gather requirements and insights from NGOs and donors.
- Develop initial prototypes to visualize and validate core functionality, such as user registration and donation flow.
- Perform user testing on prototypes to refine requirements based on real user feedback.

Output: A comprehensive list of requirements reflecting core features, such as registration, authentication, donation processing, and reporting.

6.3 Sprint Planning

Objective: Organize the gathered requirements into manageable tasks, prioritize based on dependencies and business value, and set up a roadmap.
Activities:

- Divide the requirements into smaller tasks, grouped into stories (e.g., donor registration, payment processing, admin verification).

- Prioritize tasks for each sprint, ensuring a balanced workload and clear objectives per sprint.
- Set sprint goals for specific functional outcomes (e.g., completing user registration API or donation processing).

Output: A well-defined sprint backlog with clearly prioritized tasks and goals for each sprint cycle.

Development and Testing

Objective: Develop and test each feature, ensuring functionality, quality, and security.
Activities:

- Implement the designated features for each sprint (e.g., backend APIs for registration, authentication, and profile management).
- Conduct unit tests to ensure each feature functions as intended.
- Perform integration testing to validate that new features integrate well with existing components.
- Carry out regular code reviews to ensure high code quality and catch issues early.

Output: Incrementally developed, tested, and quality-assured features added to the app.

Sprint Review

Objective: Present sprint progress to stakeholders, gather feedback, and make iterative improvements.
Activities:

- Demonstrate completed features (e.g., NGO registration, donor feedback system, payment gateways) to stakeholders.
- Collect feedback from users and stakeholders to assess usability and functionality.
- Identify potential adjustments or refinements based on the review.

Output: Stakeholder feedback to guide adjustments and improvements for the next sprint.

Sprint Retrospective

Objective: Reflect on each sprint to improve team dynamics, processes, and tools for future development.
Activities:

- Review the sprint process, discussing areas where the team excelled and where improvements are needed.
- Identify action items, such as changes to communication channels or adjustments in task management.

Output: List of improvements and action items to enhance team productivity and sprint efficiency.

Repeat

Objective: Use insights and feedback to refine the development process and continue iteration until the full NGO Donation App meets the project goals.
Activities:

- Apply lessons learned and new insights into each subsequent sprint.
- Continue iterating and adding functionality, moving closer to delivering a complete app that meets user needs.

Output: Progressive enhancements, resulting in a fully developed NGO Donation App with high usability, security, and performance.

Chapter 7

Technology Used

The NGO Donation App is designed to facilitate and simplify donation processes for NGOs, donors, and administrators. Built with modern technologies, the frontend, backend, and database layers are developed with a focus on scalability, security, and performance.

7.1 Frontend (Presentation Layer)

Technologies: HTML, CSS, ReactJS
Features:

- Dynamic and responsive user interfaces.
- User registration and login forms.
- Donation forms.
- NGO profiles and listings.
- Admin dashboards.

7.2 Backend (Application Layer)

Technologies: node.js, express.js framework
Features:

- API development for CRUD operations.
- Payment processing and integration.
- Feedback and review management.
- Security measures like role-based access control (RBAC).

7.3 Database (Data Layer)

Technologies: mongoDB
Features:

- Data models for users, NGOs, donations, feedback, and reports.
- Secure storage for transaction records.
- Query optimization for search and filtering.

7.4 Development Tools

Technologies: Visual Studio Code (VSCode), GitHub
Features:

- VSCode for integrated development, debugging, and source control.
- GitHub for version control and seamless collaboration.

Summary: This technology stack ensures a robust, user-friendly, and secure platform for handling donations and supporting NGO operations. Each component has been selected to balance ease of development, performance, and scalability.