Project Report on

Notes Management System

Submitted to

Department of Computer Science and Engineering

Nepal Engineering College

By

Asim Sonju Shrestha (020-309)

Dipson Thapa (020-313)

Date: 06/30/2024

Table of Contents

List of Figures

USE CASE ANALYSIS

Use case analysis is a technique used in software and systems engineering to identify, clarify, and organize system requirements by defining the interactions between a user (actor) and a system to achieve specific goals. This analysis helps ensure that the system meets user needs and provides a clear understanding of system functionality and user interactions.

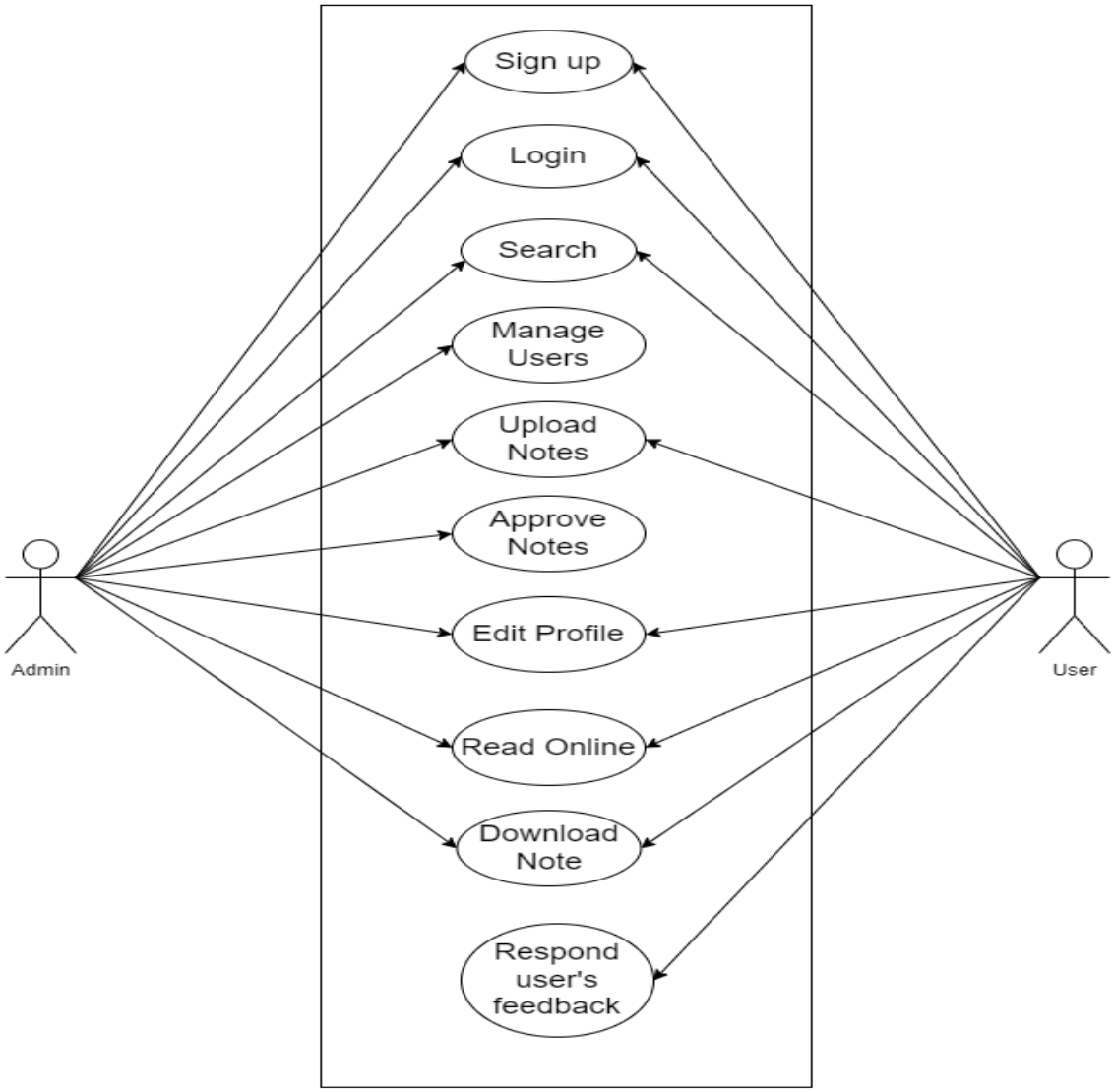

Figure 1: Use Case Diagram (User and admin)

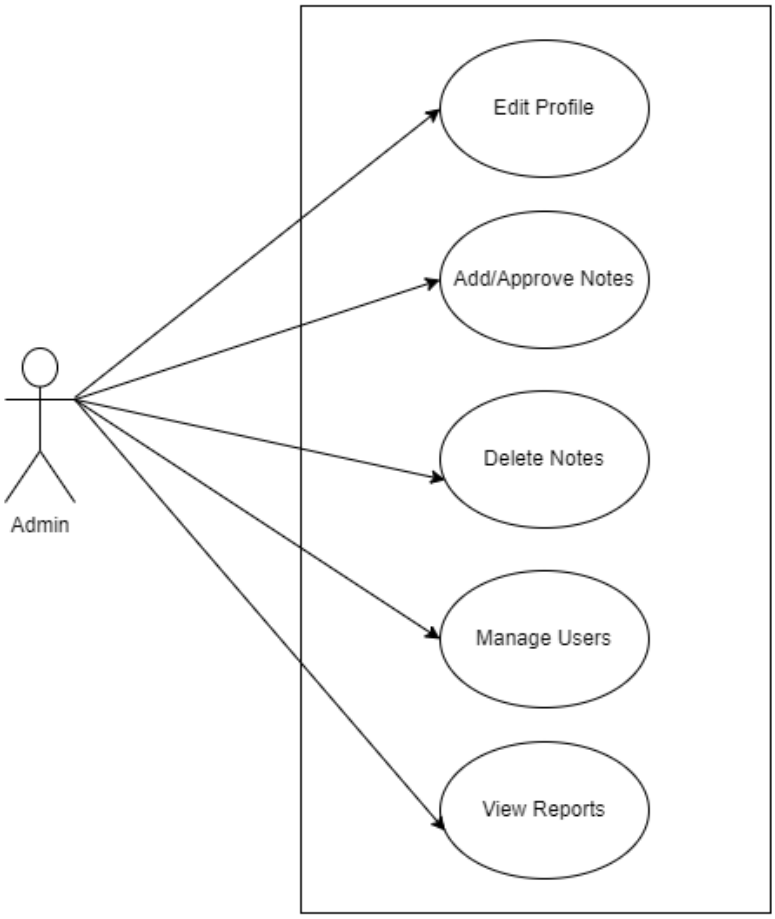

Figure 2: Use Case Diagram for Admin

Approve Note

Admin

This use case describes the process of an admin approving a note uploaded by a user.

- The admin is registered with the Notes Management System.
- The admin has logged into their account.
- The admin navigates to the pending approvals section.
- The admin reviews the list of notes pending approval.
- The admin selects a note to review.

- The system displays the note details (title, description, category, uploaded by, upload date, etc.).
- The admin reviews the note content for appropriateness and accuracy.
- If the note meets the system's guidelines, the admin marks the note as approved.
- The system updates the note's status to approved.
- The note becomes available for all users to view and download.
- If the note does not meet the guidelines, the admin marks the note as rejected.
- The system updates the note's status to rejected and sends a notification to the user with the reason for rejection.

Manage Users

Admin

This use case describes the process of an admin managing user accounts in the system.

- The admin is registered with the Notes Management System.
- The admin has logged into their account.
- The admin navigates to the user management section.
- The admin reviews the list of registered users.
- The admin selects a user to manage.
- The system displays the user’s details (name, email, registration date, status, etc.).
- The admin can update user details if necessary.
- The admin can change the user's status (activate, deactivate, or ban).
- The system updates the user's status and logs the action.
- If the user is banned, the system sends a notification to the user about the action.
- The admin can also reset the user's password.
- The system logs all actions taken by the admin.

View Reports

Admin

This use case describes the process of an admin viewing system reports.

- The admin is registered with the Notes Management System.
- The admin has logged into their account.
- The admin navigates to the reports section.
- The admin selects the type of report they want to view (upload activity, user activity, system performance, etc.).
- The system generates the selected report.
- The admin reviews the report data (e.g., number of uploads, user activity logs, system errors).
- The admin can download the report for further analysis.
- The system logs the admin's report viewing activity.

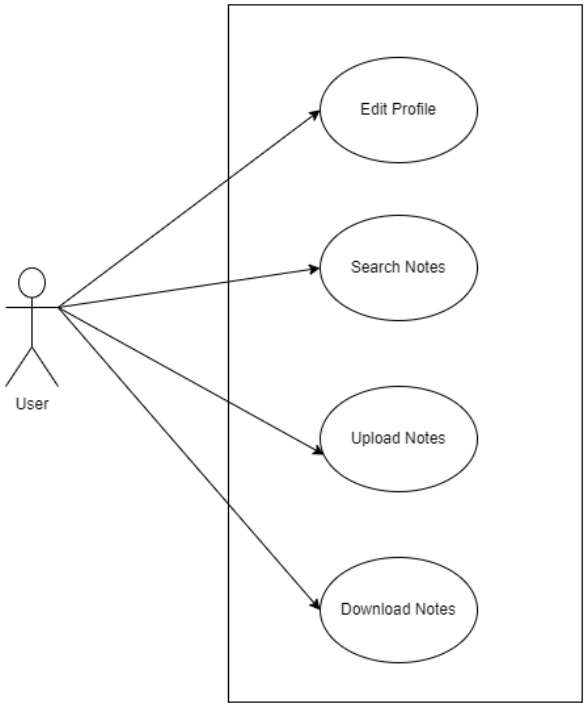

Figure 3: Use Case Diagram for User

Upload Note

User

This use case describes the process of a user uploading a note to the system.

- The user is registered with the Notes Management System.
- The user has logged into their account.
- The user has the note file ready for upload.
- The user navigates to the upload section of the system.
- The user selects the note file from their device.
- The user provides necessary details (title, description, category, etc.) about the note.
- The user agrees to the copyright policy of the system.
- The user clicks the upload button.
- The system verifies the file format and size.
- The system checks for any duplicate files.
- If the file is valid and not a duplicate, the system uploads the note and updates the database with the new entry.
- The system generates a confirmation message for the successful upload.
- The note is marked as pending approval by an admin.
- The user receives a notification about the pending approval status.
- The system logs the upload activity in the user's record.
- If the file is invalid or a duplicate, the system displays an error message and the process ends.

View Notes

User

This use case describes the process of a user viewing available notes in the system.

- The user is registered with the Notes Management System.
- The user has logged into their account.
- The user navigates to the note’s library section.
- The user uses search filters (title, category, upload date, etc.) to find specific notes.
- The user selects a note from the search results.
- The system displays the note details (title, description, upload date, uploaded by, etc.).
- The user can view the note content if it is approved.
- If the note is not approved, the user sees a message indicating its pending approval status.
- The user can download the approved note or save it to their favorites.
- The system logs the view activity in the user's record.

ACTIVITY DIAGRAM

An activity diagram is a type of UML (Unified Modeling Language) diagram that visually represents the flow of actions or processes within a system. It is used to model the dynamic behavior of a system, illustrating parallel activities, decisions, and conditions. Activity diagrams are commonly used in software engineering and business process modeling to provide a clear depiction of activity flows for system analysis and design.

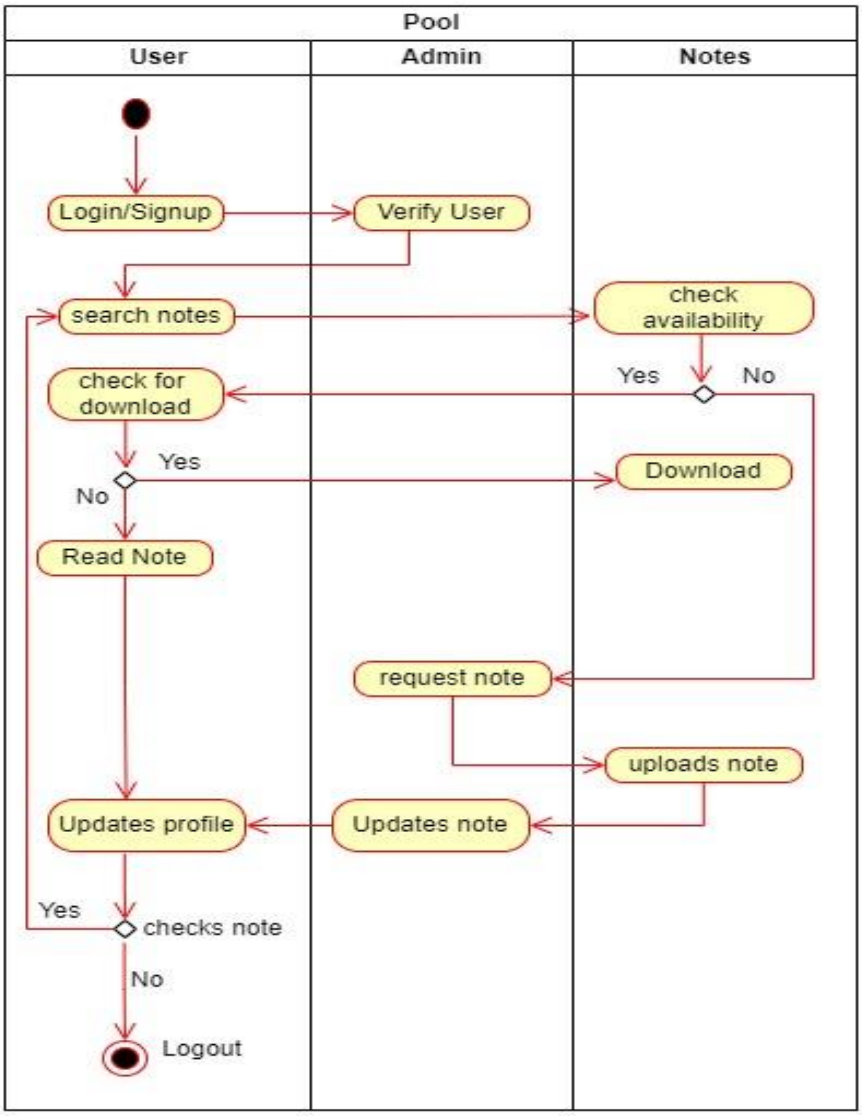

Figure 4: Activity Diagram/Swimlane Diagram

Swimlane/Activity Diagram

This swimlane/activity diagram provides a detailed view of the activities within the Notes Management System, organized by the roles involved. Here are the key points:

1. **Swimlanes (Roles):**
    - **User:** Represents the end-users of the system.
    - **Admin:** Represents the administrators of the system.
    - **Notes:** Represents the notes management subsystem.
2. **Activities:**
    - **User:**
        - **Login/Signup:** Users log in or sign up to access the system.
        - **Search notes:** Users search for notes within the system.
        - **Check for download:** Users check if the desired notes are available for download.
        - **Read Note:** Users read the notes.
        - **Updates profile:** Users update their profile information.
        - **Checks note:** Users verify the availability or condition of a note.
        - **Logout:** Users log out from the system.
    - **Admin:**
        - **Verify User:** Admin verifies the user's credentials or account.
        - **Request note:** Admin processes the user's request for a note.
        - **Uploads note:** Admin uploads new notes to the system.
        - **Updates note:** Admin updates existing notes in the system.
    - **Notes:**
        - **Check availability:** The system checks the availability of the requested notes.

- **Download:** The system facilitates the download of notes if available.

3. **Flow of Activities:**
    - Users log in or sign up and are verified by the admin.
    - Users search for notes and check if they are available for download.
    - If available, users download and read the notes; if not, they update their profile or check other notes.
    - Admins handle note requests, upload new notes, and update existing notes as needed.
    - Users can update their profiles and eventually log out of the system.
4. **Decisions:**
    - **Check for download:** Determines if the note is available for download.
    - **Check availability:** The system checks if the requested note is available.
5. **End Points:**
    - The activities culminate in the user logging out of the system after performing the necessary actions.

The activity diagram for the Notes Management System effectively illustrates the various activities and functionalities available to users after successful authentication and authorization. By using specific symbols and notations, the diagram clearly communicates the flow of activities, decision points, and the overall functionality of the system. Activity diagrams offer numerous benefits, including visualization, process modeling, documentation, communication, and validation, making them a valuable tool in the development and understanding of complex systems like the Notes Management System.

ENTITY RELATION DIAGRAM

Introduction

The Notes Management System is designed to facilitate the creation, management, and approval of notes within an academic or professional setting. The system has two primary user roles: Users and Admins. Users can upload and manage their notes, while Admins are responsible for approving or rejecting these notes and managing user. This document provides an in-depth explanation of the entities, relationships, attributes, and keys used in the system's Entity-Relationship (ER) diagram.

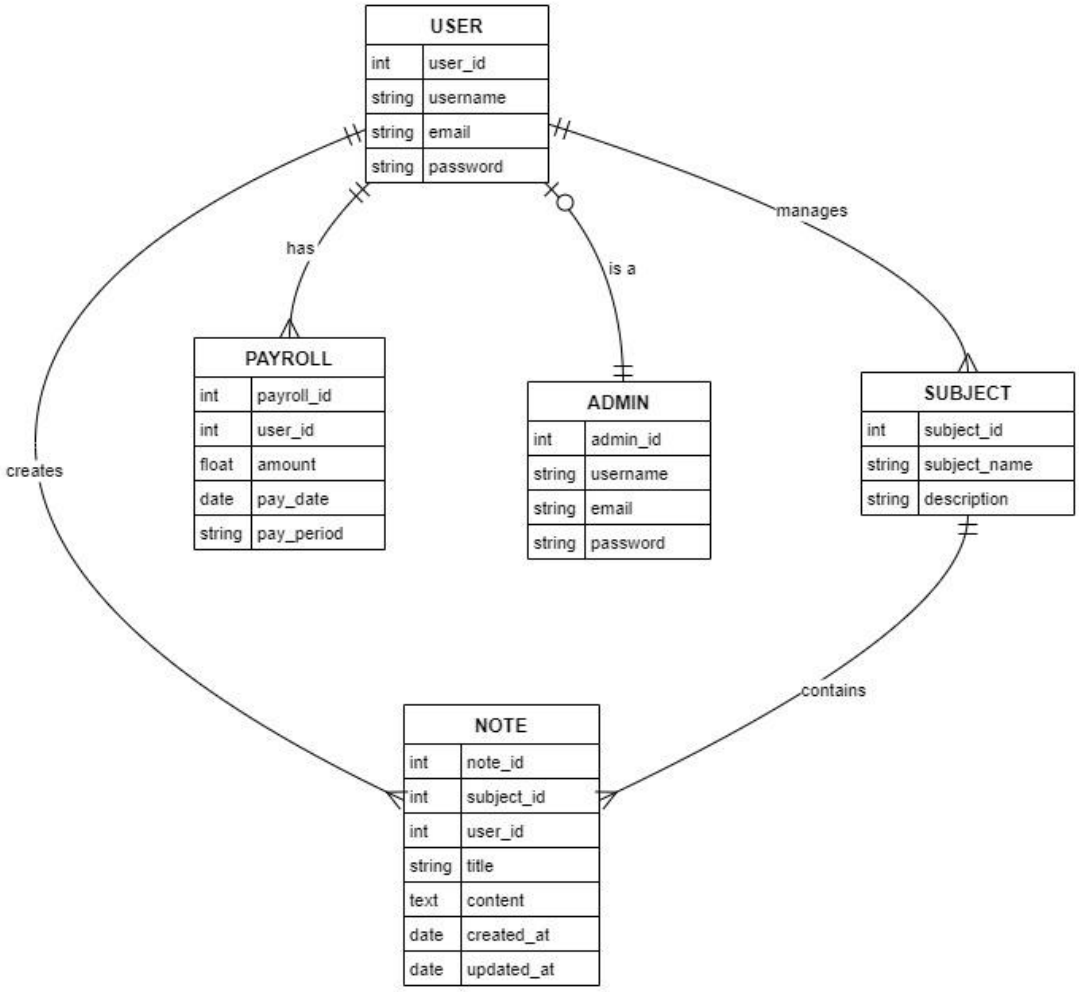

Figure 5: ER Diagram

This Entity-Relationship (ER) diagram represents the structure and relationships of a Notes Management System.

1. **Entities and Attributes:**
    - **USER:**
        - user_id: Integer, Primary Key
        - username: String
        - email: String
        - password: String
    - **ADMIN:**
        - admin_id: Integer, Primary Key
        - username: String
        - email: String
        - password: String
    - **SUBJECT:**
        - subject_id: Integer, Primary Key
        - subject_name: String
        - description: String
    - **NOTE:**
        - note_id: Integer, Primary Key
        - subject_id: Integer, Foreign Key
        - user_id: Integer, Foreign Key
        - title: String
        - content: Text
        - created_at: Date
        - updated_at: Date

- **PAYROLL:**
    - payroll_id: Integer, Primary Key
    - user_id: Integer, Foreign Key
    - amount: Float
    - pay_date: Date
    - pay_period: String

2. **Relationships:**
    - **USER and ADMIN:**
        - is a: A user can be an admin. This indicates an inheritance relationship where an admin is a special type of user.
    - **USER and PAYROLL:**
        - has: A user has payroll information. This indicates that each user can have multiple payroll records.
    - **ADMIN and SUBJECT:**
        - manages: An admin manages subjects. This relationship indicates that an admin can be responsible for multiple subjects.
    - **SUBJECT and NOTE:**
        - contains: A subject contains notes. This relationship indicates that each subject can have multiple notes associated with it.
    - **NOTE and USER:**
        - creates: A user creates notes. This relationship indicates that each note is created by a user.
3. **Primary and Foreign Keys:**
    - **Primary Keys:** Unique identifiers for each entity (e.g., user_id for USER, admin_id for ADMIN, etc.).

- **Foreign Keys:** Attributes that link entities together (e.g., user_id in NOTE references USER, subject_id in NOTE references SUBJECT, etc.).

4. **Associative Details:**
    - The diagram captures the hierarchical and relational structure of users, admins, subjects, notes, and payroll within the system, showing how they interact and relate to each other.

The ER Diagram for the Notes Management System provides a comprehensive view of the data structure and relationships within the system. By clearly defining the entities, attributes, keys, and relationships, the diagram serves as a valuable tool for database design and implementation. Understanding this diagram is crucial for developers, database administrators, and stakeholders to ensure the efficient and effective operation of the Notes Management System.

CLASS DIAGRAM

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

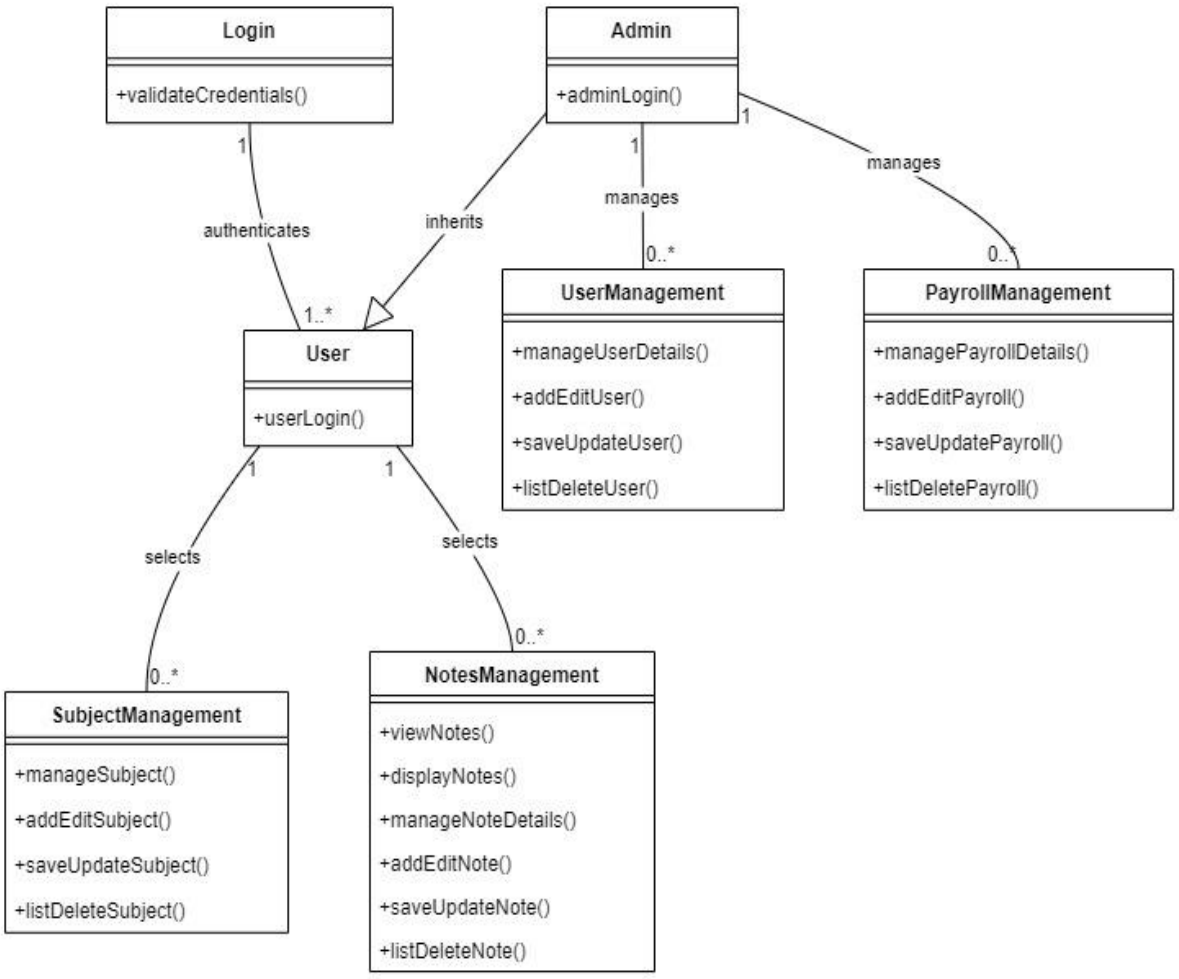

Figure 6: Class Diagram

This class diagram represents a system with several classes and their relationships, specifically focused on user authentication, administration, and management of various entities such as users, payroll, subjects, and notes. Here is a breakdown of the diagram:

1. **Login Class**
    - **Attributes/Methods:**
        - +validateCredentials()
    - **Associations:**
        - Authenticates one or more Users (1..*).
2. **User Class**
    - **Attributes/Methods:**
        - +userLogin()
    - **Associations:**
        - Authenticated by Login (1..*).
        - Selects one or more instances of SubjectManagement (0..*).
        - Selects one or more instances of NotesManagement (0..*).
3. **Admin Class (inherits from User)**
    - **Attributes/Methods:**
        - +adminLogin()
    - **Associations:**
        - Manages zero or more instances of UserManagement (0..*).
        - Manages zero or more instances of PayrollManagement (0..*).
4. **UserManagement Class**
    - **Attributes/Methods:**
        - +manageUserDetails()
        - +addEditUser()
        - +saveUpdateUser()
        - +listDeleteUser()

- **Associations:**
    - Managed by Admin (0..*).

5. **PayrollManagement Class**
    - **Attributes/Methods:**
        - +managePayrollDetails()
        - +addEditPayroll()
        - +saveUpdatePayroll()
        - +listDeletePayroll()
    - **Associations:**
        - Managed by Admin (0..*).
6. **SubjectManagement Class**
    - **Attributes/Methods:**
        - +manageSubject()
        - +addEditSubject()
        - +saveUpdateSubject()
        - +listDeleteSubject()
    - **Associations:**
        - Selected by User (0..*).
7. **NotesManagement Class**
    - **Attributes/Methods:**
        - +viewNotes()
        - +displayNotes()
        - +manageNoteDetails()
        - +addEditNote()

        - +saveUpdateNote()
        - +listDeleteNote()
    - **Associations:**
        - Selected by User (0..*).

Summary of Relationships:

- **Inheritance:**
    - Admin inherits from User.
- **Associations:**
    - Login authenticates Users.
    - Admin manages UserManagement and PayrollManagement.
    - Users select SubjectManagement and NotesManagement entities for interaction.

This class diagram captures the high-level structure and interactions between different components in the system, providing a clear overview of how users and admins manage various aspects such as users, payroll, subjects, and notes.

SEQUENCE DIAGRAM

Sequence diagrams are a type of UML (Unified Modeling Language) interaction diagram that detail how operations are carried out. They capture the interaction between objects in the context of a collaboration, showing the order in which messages are sent between the different lifelines (object instances) over time. Key elements of sequence diagrams include:

- Actors: Represent external users or systems that interact with the system.
- Lifelines: Vertical bars representing the different objects involved in the interaction.
- Messages: Arrows between lifelines representing the flow of communication and the order of interactions.
- Activation bars: Rectangles on lifelines indicating the duration of an operation.
- Interaction fragments: Structured elements like alternatives, options, and loops that model complex interactions.

Sequence diagrams are used to:

- Visualize system behavior and interactions
- Document software design and architecture
- Facilitate communication and collaboration among stakeholders
- Clarify and validate system requirements
- Debug and troubleshoot system issues

When creating sequence diagrams, it's important to balance the level of detail, avoid clutter, and keep them up-to-date as the system evolves.

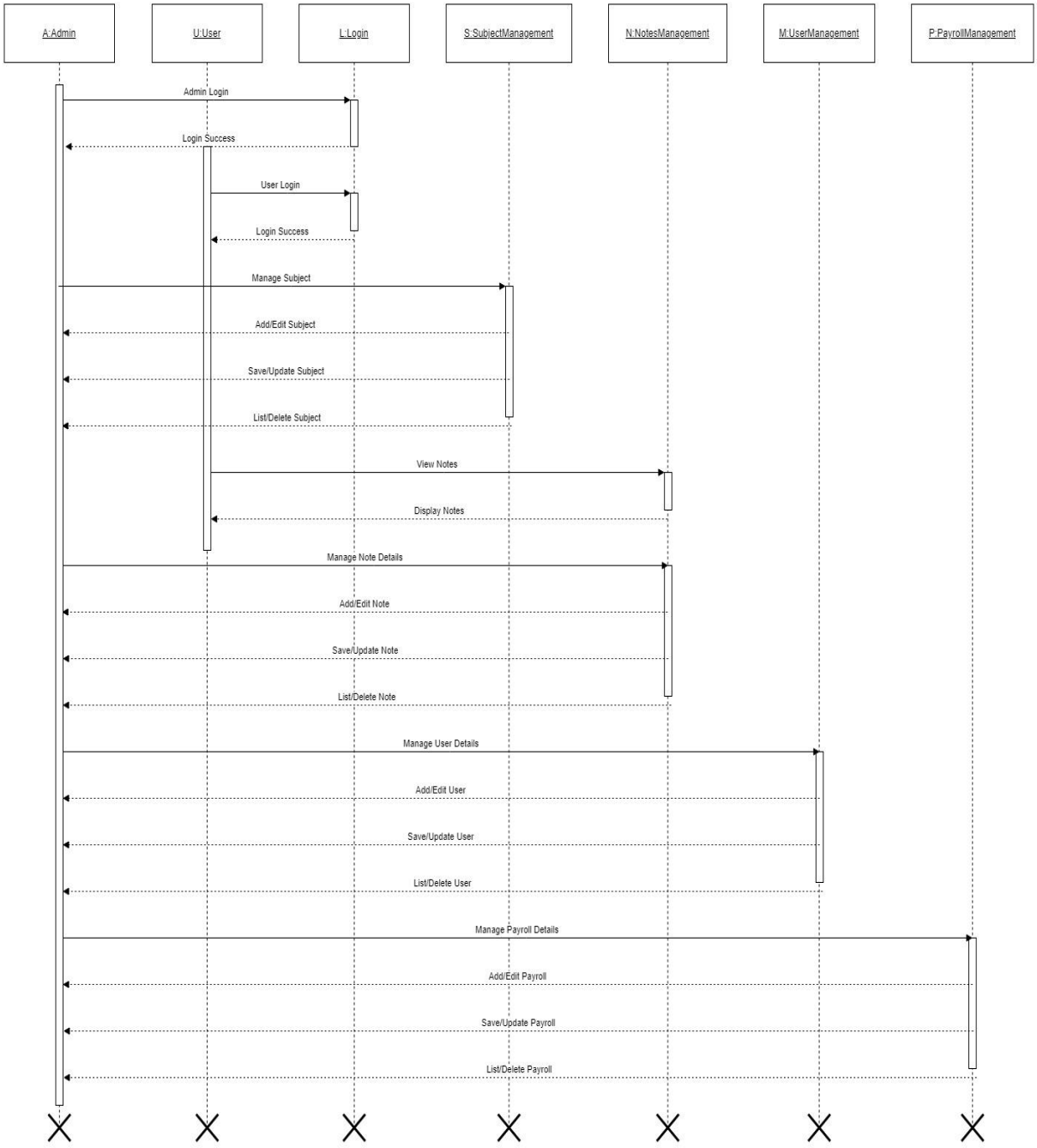

Figure 7: Sequence Diagram

This sequence diagram represents the interactions between different actors and the system for a notes management system. Here's a brief description of the key interactions:

1. **Actors and System Components**:
    - **A: Admin**: The system administrator.
    - **U: User**: A regular user of the system.
    - **L: Login**: The login process.
    - **S: SubjectManagement**: The module handling subject management.
    - **N: NoteManagement**: The module handling note management.
    - **M: UserManagement**: The module managing user details.
    - **P: PayrollManagement**: The module managing payroll details.
2. **Login Process**:
    - **Admin Login**: Admin initiates the login process.
    - **User Login**: User initiates the login process.
    - **Login Success**: Both admin and user receive confirmation of successful login.
3. **Subject Management**:
    - **Manage Subject**: Admin accesses the subject management module.
    - **Add/Edit Subject**: Admin can add or edit subject details.
    - **Save/Update Subject**: Changes to subjects are saved or updated.
    - **List/Delete Subject**: Admin can list or delete subjects.
4. **Note Management**:
    - **View Notes**: User views the notes.
    - **Display Notes**: The system displays the notes to the user.
    - **Manage Note Details**: Admin or user manages note details.
    - **Add/Edit Note**: Admin or user can add or edit notes.
    - **Save/Update Note**: Changes to notes are saved or updated.

- **List/Delete Note**: Admin or user can list or delete notes.

5. **User Management**:
    - **Manage User Details**: Admin manages user details.
    - **Add/Edit User**: Admin can add or edit user details.
    - **Save/Update User**: Changes to user details are saved or updated.
    - **List/Delete User**: Admin can list or delete users.
6. **Payroll Management**:
    - **Manage Payroll Details**: Admin manages payroll details.
    - **Add/Edit Payroll**: Admin can add or edit payroll details.
    - **Save/Update Payroll**: Changes to payroll details are saved or updated.
    - **List/Delete Payroll**: Admin can list or delete payroll records.

Each interaction is shown as a message or action between the actors and the system components, demonstrating the flow of operations within the notes management system.

COLLABORATION DIAGRAM

A collaboration diagram, also known as a communication diagram in UML 2.0, is a type of interaction diagram that shows the structural organization of objects in a system and the messages passed between these objects. It emphasizes the relationships between objects and the sequencing of messages exchanged between them to accomplish a specific task or operation.

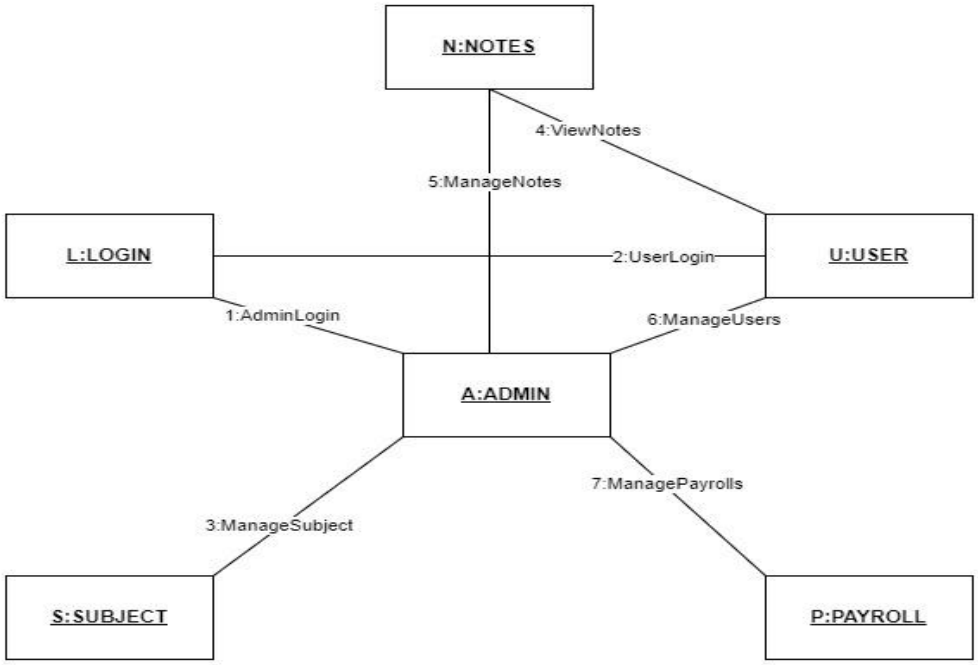

Figure 8: Collaboration Diagram

This collaboration diagram illustrates the interactions between different components of a notes management system. Here's a description of the key elements and their relationships:

1. N:NOTES: This is the central component for managing notes.
    - It interacts with U:USER through "4.ViewNotes" and "5.ManageNotes" operations.
2. U:USER: Represents the user of the system.
    - It can view and manage notes.

    - It interacts with L:LOGIN through "2.UserLogin" operation.
3. L:LOGIN: Handles the login functionality for both users and admins.
    - It's connected to U:USER for user logins.
    - It's also connected to A:ADMIN for "1.AdminLogin" operation.
4. A:ADMIN: Represents the administrator role in the system.
    - It has more privileges than a regular user.
    - It can manage users ("6.ManageUsers"), subjects ("3.ManageSubject"), and payrolls ("7.ManagePayrolls").
5. S:SUBJECT: Represents the subject or category component.
    - It's managed by the admin ("3.ManageSubject").
6. P:PAYROLL: Represents the payroll management component.
    - It's also managed by the admin ("7.ManagePayrolls").

The numbered operations indicate the sequence of interactions:

1. Admin Login
2. User Login
3. Manage Subject
4. View Notes
5. Manage Notes
6. Manage Users
7. Manage Payrolls

This diagram shows a hierarchical structure where the admin has overarching control over users, subjects, and payrolls, while regular users primarily interact with notes. The login component serves as the entry point for both user types. The collaboration diagram effectively illustrates the relationships and interactions between different parts of the notes management system.

DATA FLOW DIAGRAM(DFD)

Data Flow Diagrams (DFDs) are a traditional way to visualize the flow of data through a system. They are particularly useful for understanding how information moves within a system and between different entities. DFDs focus on the data and how it is processed, rather than on the control flow, making them an essential tool for system design, analysis, and documentation.

Key Components of a DFD

1. **Processes**:
    - Represented by circles or rounded rectangles.
    - Show how data is transformed or processed within the system.
    - Labeled with a verb-noun phrase (e.g., "Process Order").
2. **Data Stores**:
    - Represented by open-ended rectangles or parallel lines.
    - Indicate where data is stored within the system (e.g., databases, files).
    - Labeled with a noun (e.g., "Customer Database").
3. **External Entities**:
    - Represented by squares or rectangles.
    - Represent sources or destinations of data outside the system boundaries.
    - Labeled with a noun (e.g., "Customer", "Supplier").
4. **Data Flows**:
    - Represented by arrows.
    - Show the direction of data movement between processes, data stores, and external entities.
    - Labeled with the name of the data being moved (e.g., "Order Details").

Levels of DFDs

1. **Level 0 DFD (Context Diagram)**:
    - Provides a high-level overview of the system.
    - Shows the system as a single process and how it interacts with external entities.
    - Simplifies the view by focusing on the main functions and data flows.
2. **Level 1 DFD**:
    - Breaks down the Level 0 process into sub-processes.
    - Provides more detail about the system's internal processes and data flows.
    - Each process in the Level 0 DFD is expanded into a more detailed process in the Level 1 DFD.
3. **Level 2 (and beyond) DFD**:
    - Further decompose Level 1 processes into more detailed processes.
    - Continue to break down until the necessary level of detail is reached.
    - Each level offers a deeper understanding of the system's workings.

Creating a DFD

Steps to Create a DFD

1. **Identify the System Boundary**:
    - Define what is included in the system and what lies outside its boundaries.
2. **Determine the Major Processes**:
    - Identify the main functions that the system performs.
3. **Identify the Data Stores**:
    - Determine where data is stored within the system.

4. **Identify the External Entities**:
    - Determine who interacts with the system and what data they provide or receive.
5. **Draw the Data Flows**:
    - Connect processes, data stores, and external entities with arrows showing how data moves.
6. **Validate the Diagram**:
    - Ensure that all data flows make sense and accurately represent how the system functions.

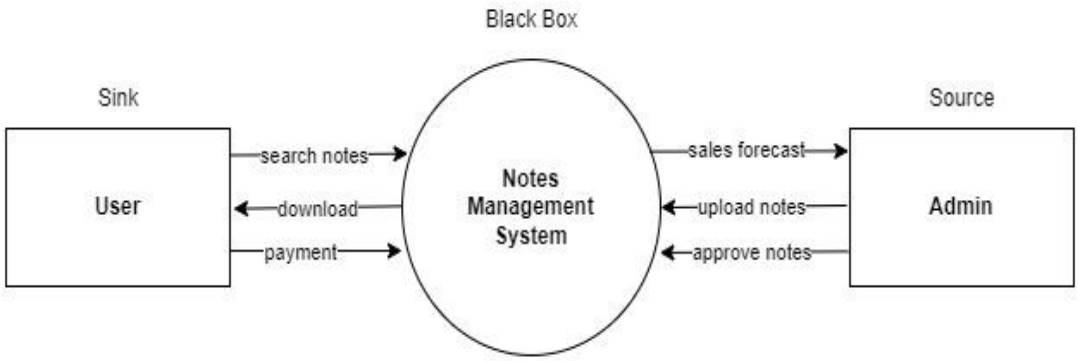

Figure 9: Context Level DFD

This context-level Data Flow Diagram (DFD) provides a high-level overview of a Notes Management System. Here are the key points:

1. **Actors:**
    - **User:** Represents the end-users interacting with the system.
    - **Admin:** Represents the administrators managing the system.
2. **Processes:**
    - **Notes Management System (Black Box):** The central system handling the interactions and operations related to note management.

3. **Data Flows:**

- From **User** to **System:**
  - **Search notes:** Users search for notes within the system.
  - **Download:** Users download notes from the system.
  - **Payment:** Users make payments for notes.
- From **System** to **User:**
  - **Search notes (response):** System provides search results to users.
  - **Download (response):** System facilitates the downloading of notes.
- From **Admin** to **System:**
  - **Upload notes:** Admins upload new notes to the system.
  - **Approve notes:** Admins approve notes before they are available to users.
  - **Sales forecast:** Admins may upload sales forecast data to the system.
- From **System** to **Admin:**
  - **Upload notes (response):** System acknowledges the uploaded notes.
  - **Approve notes (response):** System confirms the approval of notes.
  - **Sales forecast (response):** System provides sales forecast data back to the admin.

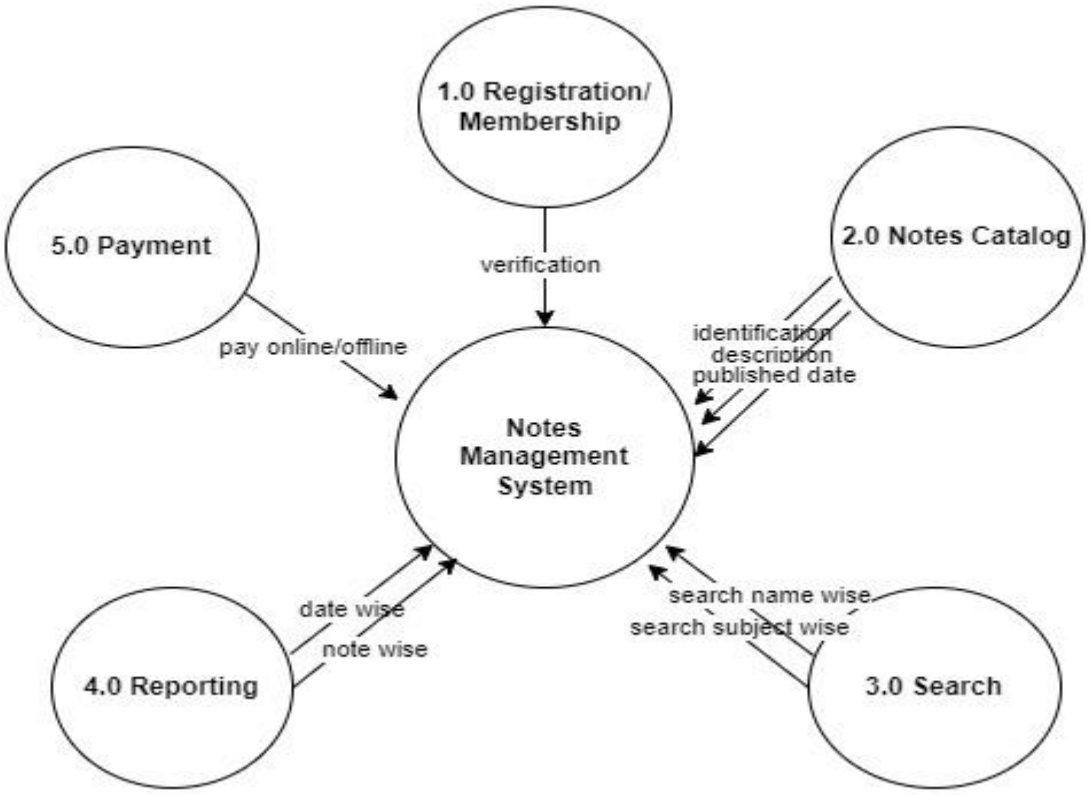

Figure 10: Level-0 DFD

The Level-0 Data Flow Diagram (DFD) for the Notes Management System provides a high-level overview of the system's major processes and data flows. Here are the key points:

- **Central Process (Notes Management System)**
    - The central process is the Notes Management System, which interacts with all other components.
- **1.0 Registration/Membership**
    - Users register or manage their membership.
    - The system verifies the user details.
- **2.0 Notes Catalog**
    - Users can add or manage notes.
    - The system handles the identification and description of notes.

- **3.0 Search**
    - Users can search for notes.
    - Searches can be conducted by name or subject.
- **4.0 Reporting**
    - The system provides reporting features.
    - Reports can be generated date-wise or note-wise.
- **5.0 Payment**
    - Users can make payments online or offline.
    - The system processes these payments.

Each of these components interacts directly with the central Notes Management System, with specific data flows as indicated in the diagram.

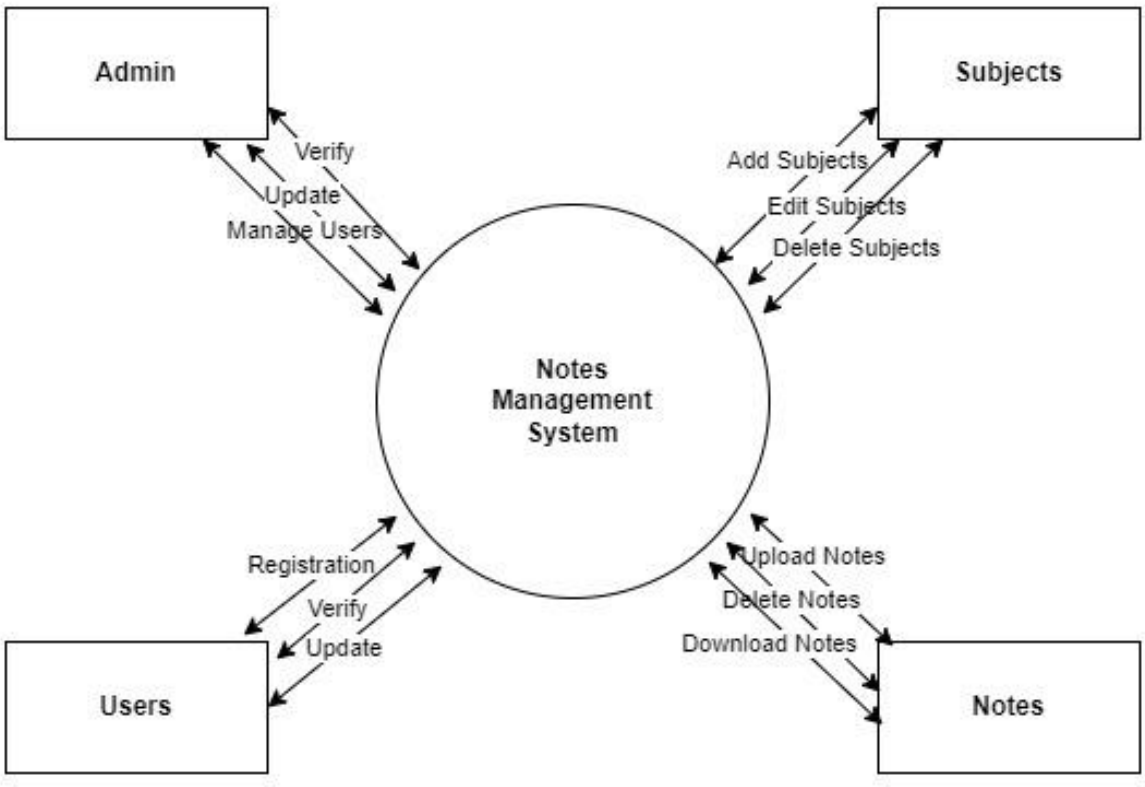

Figure 11: Level-1 DFD

The Level-1 Data Flow Diagram (DFD) for the Notes Management System provides a more detailed view of the system, breaking down the main processes and interactions with external entities. Here are the key points:

- **Central Process (Notes Management System)**
    - The central process remains the Notes Management System, which interacts with various external entities.
- **Admin**
    - The admin can verify users, update user information, and manage users.
    - Data flows: Verify, Update, Manage Users.
- **Users**
    - Users can register, verify their registration, and update their information.
    - Data flows: Registration, Verify, Update.
- **Subjects**
    - Subjects can be added, edited, or deleted by users or administrators.
    - Data flows: Add Subjects, Edit Subjects, Delete Subjects.
- **Notes**
    - Users can upload, delete, or download notes.
    - Data flows: Upload Notes, Delete Notes, Download Notes.

Each of these external entities interacts with the central Notes Management System, with specific data flows as indicated in the diagram.

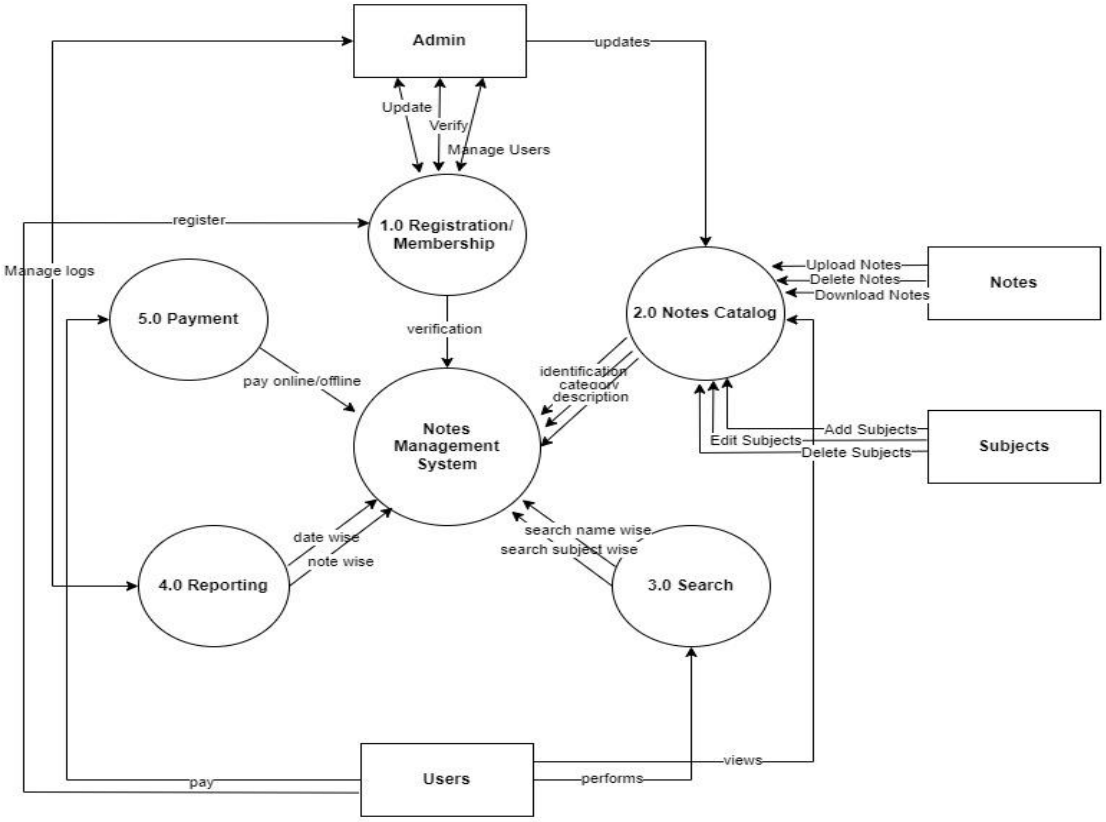

Figure 12: Level-2 DFD

The Level-2 Data Flow Diagram (DFD) for the Notes Management System provides a detailed breakdown of the system's processes, data flows, and interactions with external entities. Here are the key points:

- **Central Process (Notes Management System)**
    - The central process remains the Notes Management System, with detailed interactions for each sub-process.
- **Registration/Membership**
    - Users register, verify their registration, and update their information.
    - Admins manage users and verify updates.
    - Data flows: Register, Verify, Update.
- **2.0 Notes Catalog**
    - Users can upload, delete, and download notes.
    - Users or Admins can add, edit, or delete subjects.

    - Data flows: Upload Notes, Delete Notes, Download Notes, Add Subjects, Edit Subjects, Delete Subjects.
- **3.0 Search**
    - Users can search for notes by name or subject.
    - Data flows: Search Name Wise, Search Subject Wise.
- **4.0 Reporting**
    - The system provides date-wise and note-wise reports.
    - Admins manage logs.
    - Data flows: Date Wise, Note Wise.
- **5.0 Payment**
    - Users can make payments online or offline.
    - The system handles payment verification.
    - Data flows: Pay Online/Offline.
- **Admin**
    - Admins update the system, verify users, and manage user details.
    - Data flows: Update, Verify, Manage Users.

- **Users**
    - Users interact with the system for registration, searching notes, managing their catalog, and making payments.
    - Data flows: Register, Verify, Update, Perform, View, Pay.

- **Notes**
    - User's upload, delete, and download notes.
    - Data flows: Upload Notes, Delete Notes, Download Notes.
- **Subjects**
    - Users or Admins add, edit, or delete subjects.
    - Data flows: Add Subjects, Edit Subjects, Delete Subjects.

This Level-2 DFD illustrates the comprehensive interaction between users, admins, and the Notes Management System, detailing the various data flows and processes involved.

STATE CHART DIAGRAM

A state diagram (also known as a state machine or state chart diagram) is an illustration of all the possible behavioral states a software system component may exhibit and the various state changes it's predicted to undergo over the course of its operations.

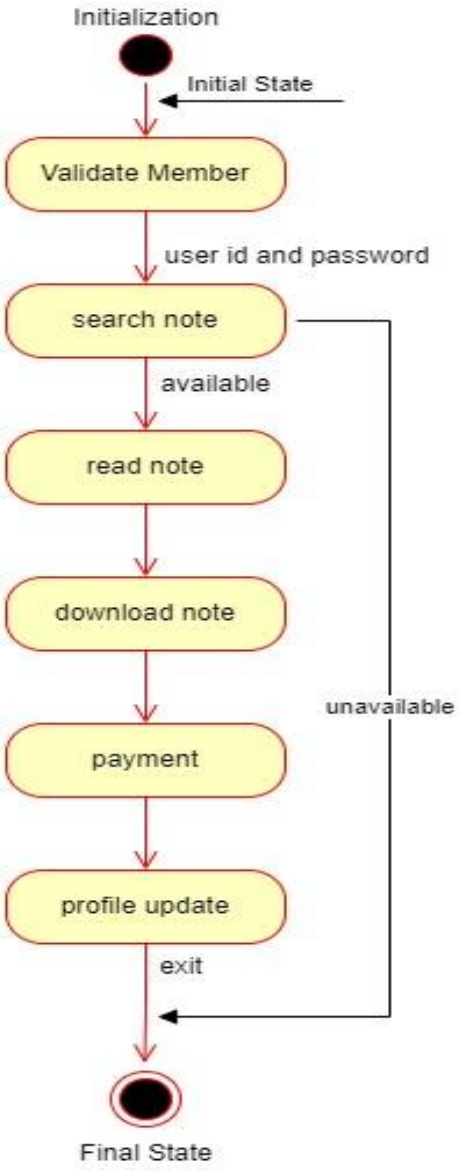

Figure 13: State Chart Diagram

This state chart diagram illustrates the flow of a note management system. Here's a description of the process:

1. **Initialization:** The process begins with an initial state.
2. **Validate Member**: The first action is to validate the user's membership, likely through a login process.
3. **Search Note:** After validation, the user can search for a note using their user ID and password.
4. **Read Note:** If a note is available, the user can read it.
5. **Download Note:** Following reading, there's an option to download the note.
6. **Payment:** After downloading, there's a payment state, suggesting some notes may require payment to access.
7. **Profile Update:** The user has the option to update their profile.
8. **Exit:** The process can end here, leading to the final state.

Key points:

- There's a conditional flow after "search note". If a note is unavailable, the process skips directly to the exit state.
- The diagram shows a linear progression through most states, with only one decision point.
- The system includes both free actions (search, read) and potentially paid features (download, requiring payment).
- User management is integrated, with validation at the start and profile updates near the end.

This diagram represents a simple, streamlined process for note management, covering authentication, search, access, download, payment, and user profile management.

COMPONENT DIAGRAM

An activity diagram is a type of UML (Unified Modeling Language) diagram that visually represents the flow of actions or processes within a system. It is used to model the dynamic behavior of a system, illustrating parallel activities, decisions, and conditions. Activity diagrams are commonly used in software engineering and business process modeling to provide a clear depiction of activity flows for system analysis and design.

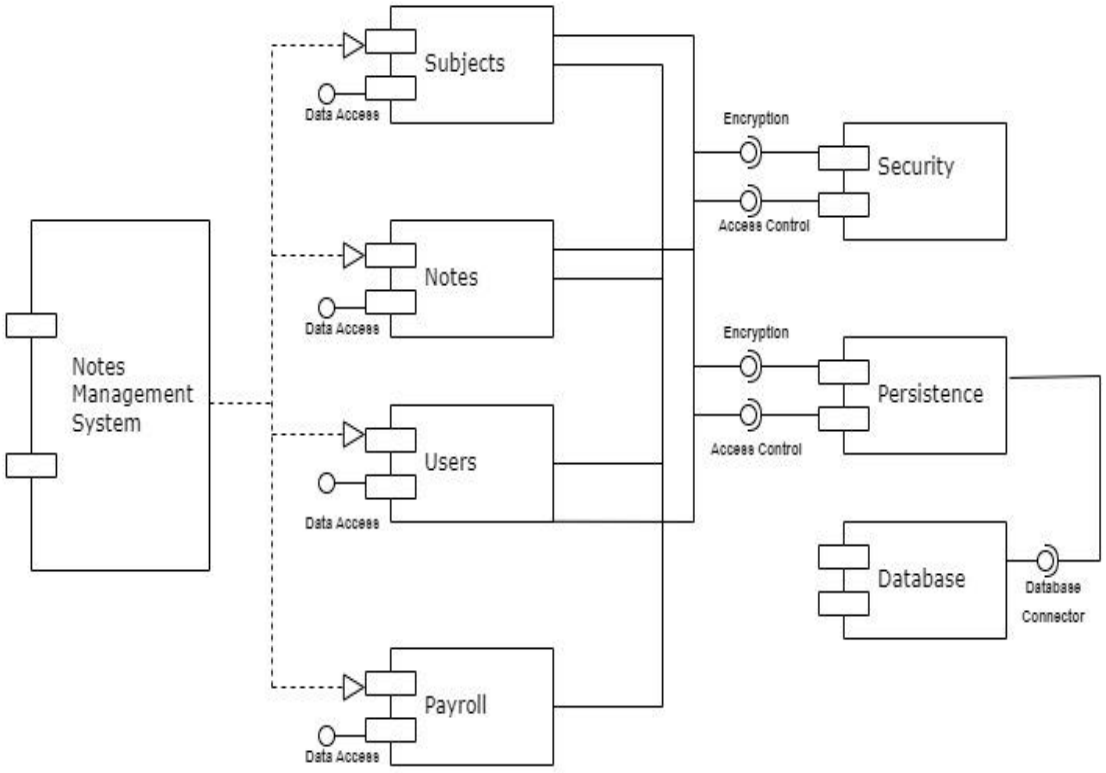

Figure 14: Component Diagram

Components

- Notes Management System: The central component that coordinates all other components and manages the overall functionality of the system.

- Subjects: Handles subject-related data, including creation, retrieval, and organization of subject information. It has a data access interface.
- Notes: Manages note content, organization, and metadata. This component also has a data access interface for creating, retrieving, and modifying notes.
- Users: Manages user accounts, profiles, and authentication. It includes a data access interface for user-related operations.
- Payroll: Processes and manages payroll information, potentially including salary calculations and payment records. It also has a data access interface.
- Security: Provides encryption and access control mechanisms to ensure data protection and user authorization. It interfaces with Subjects, Notes, and User's components.
- Persistence: Manages long-term data storage and retrieval, ensuring data integrity and availability. Like Security, it uses encryption and access control.
- Database: Central storage for all system data, likely a relational or NoSQL database depending on system requirements.
- Database Connector: Interfaces between the system components and the database, handling queries and data operations.
- Data Access: Present in Subjects, Notes, Users, and Payroll components, managing data retrieval and modification operations specific to each module.
- Encryption: Applied in both Security and Persistence components to protect sensitive data.
- Access Control: Implemented in Security and Persistence components to manage user permissions and data access rights. Authorization. By using specific symbols and notations, the diagram clearly communicates the flow of activities, decision points, and the overall functionality of the system. Activity diagrams offer numerous benefits, including visualization, process modeling, documentation, communication, and validation, making them a valuable tool in the development and understanding of complex systems like the Notes Management System.

DEPLOYMENT DIAGRAM

Deployment diagram is a diagram that shows the configuration of run time processing nodes and the components that live on them. Deployment diagrams is a kind of structure diagram used in modeling the physical aspects of an object-oriented system. They are often be used to model the static deployment view of a system (topology of the hardware).

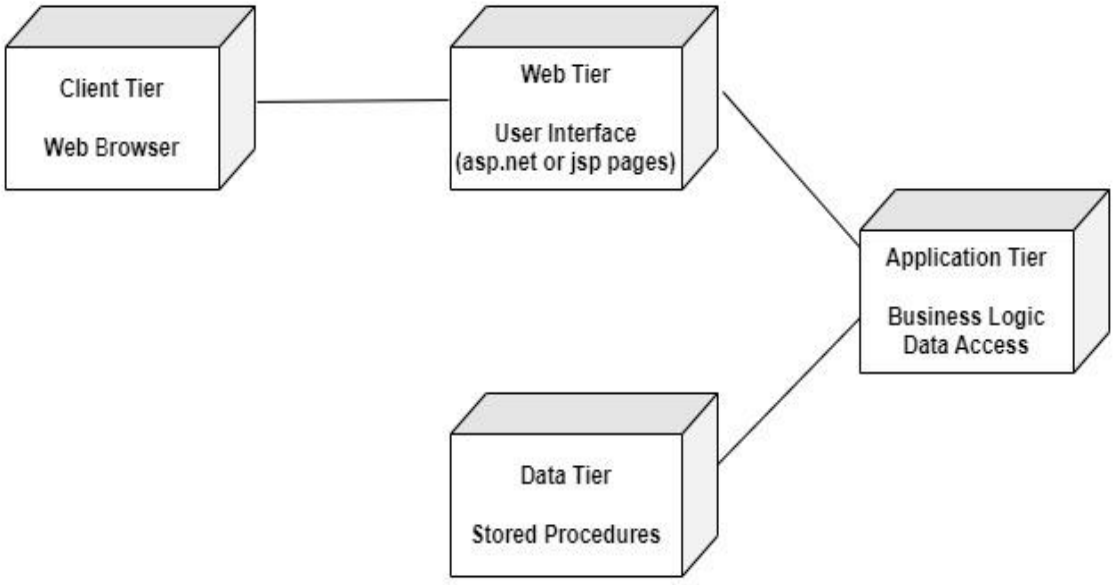

Figure 15: Deployment Diagram

This diagram depicts a multi-tier architecture for a software application deployment:

1. Client Tier:
    - This is the user-facing layer, typically a web browser.
    - It handles the presentation of the application to the end-user.
    - No business logic or data processing occurs here; it's primarily for user interaction and data display.

2. Web Tier:
    - Also known as the presentation layer.
    - Contains user interface components like JSP (Java Server Pages), HTML, CSS, and potentially JavaScript.
    - Responsible for collecting user input and presenting data in a user-friendly format.
    - Acts as a bridge between the client and the application tier.
3. Application Tier:
    - This is the core of the application, handling business logic and data access.
    - Business Logic: Implements the main functionality and rules of the application.
    - Data Access: Manages how the application interacts with the database, often using technologies like JDBC, ORM frameworks, or APIs.
    - This tier can be further divided into multiple layers for complex applications.
4. Data Tier:
    - Focuses on data storage and retrieval.
    - Utilizes stored procedures, which are precompiled SQL statements stored in the database.
    - Provides data integrity, security, and efficient data manipulation.

Benefits of this architecture:

- Scalability: Each tier can be scaled independently based on demand.
- Maintainability: Changes in one tier have minimal impact on others.
- Security: Data access is controlled through the application tier, not directly from the client.
- Performance: Tasks are distributed across tiers, potentially improving overall system performance.

Communication flow:

- Client communicates with the Web Tier.
- Web Tier interacts with the Application Tier.
- Application Tier accesses data through the Data Tier.

This type of architecture is common in enterprise-level applications, especially those requiring high scalability and clear separation of concerns. It allows for easier updates, better resource allocation, and improved security measures.

CONCLUSION

In conclusion, this report has provided a comprehensive exploration of Unified Modeling Language (UML) diagrams. We have thoroughly examined structural diagrams such as class, object, and component diagrams, which illustrate the static architecture of a system. Additionally, we've discussed behavioral diagrams including use case, sequence, and activity diagrams, which depict the dynamic aspects and interactions within a system.

Our in-depth analysis covered the semantics and best practices for creating each type of UML diagram. We explored how these diagrams serve as powerful tools for software developers, system analysts, and project managers in visualizing, specifying, constructing, and documenting the artifacts of software systems.