RAJALAKSHMI ENGINEERINGCOLLEGE

RAJALAKSHMI NAGAR, THANDALAM – 602 105

RAJALAKSHMI
ENGINEERING COLLEGE

Laboratory Record Notebook

Name: . . . . . .

Year / Branch / Section: . . . . . .

University Register No. : . . . . . .

College Roll No. : . . . . . .

Semester: . . . . . .

RAJALAKSHMI ENGINEERING COLLEGE

RAJALAKSHMI NAGAR, THANDALAM – 602 105

BONAFIDE CERTIFICATE

Name: _____

Academic Year: _____ Semester: _____ Branch : _____

Register No: | |

Certified that this is the bonafide record of work done by the above student

in the CS19442 – *Software Engineering* *Laboratory during the year 2023-2024*

Signature of Faculty in-charge

Submitted for the Practical Examination held on .....

Internal Examiner

External Examiner

ONLINE INTRANET KNOWLEGDE MANAGEMENT SYSTEM FOR COLLEGE

Table of Contents

OVERVIEW OF THE PROJECT

This project is aimed at developing an online intranet knowledge management system that is of importance to either an organization or a college. The system (KMS) is an Intranet based application that can be accessed throughout the organization or a specified group or Department. This system can be used as a knowledge or information management system for the college. Students or Staff logging should be able to upload any kind of technical information. Students or Staffs logging in may also access or search any information put up by others. Knowledge Management System should facilitate knowledge sharing from the grass root level like project teams to the entire college or organization. The Knowledge Management System is a web-based system, which provides a platform for students and the staff of an organization or institution to upload any kind of technical or course related data. Students or Staffs logging in may also access or search any information put up by others. Knowledge Management System facilitates knowledge sharing from the grass root level like project teams to the entire college or organization.

.

SOFTWARE REQUIREMENTS SPECIFICATION(SRS)

EX_NO:1 **DATE:**

TABLE OF CONTENTS:

1. INTRODUCTION

1.1 About project:

- It consists of three modules Admin, Staff, and Student, Where Admin can Add/Modify/Delete Departments, Subjects and Deletes the Documents or Staff or Student can share their technical knowledge. Student or Staff can upload the technical documents. Students can post the queries directly to the respective staff's inbox. Staff can answer the queries posted in their inbox. Thus, the software can be used as Knowledge management system for colleges. Students or Staff logging in can access or search the information uploaded by others. This comprehensive approach transforms the educational experience by providing a robust framework for knowledge management in the college setting.

1.2 Scope:

- The scope of the Online Knowledge Management System (KMS) for a college includes creating a centralized platform to store and manage academic content like course materials and research papers, and administrative documents such as policies and student records. It incorporates collaboration tools like virtual classrooms and discussion forums to enhance communication and group work among students and faculty. The system features advanced search and retrieval functions, ensuring users can quickly find relevant information.

1.3 Features:

- Document Management: Centralized storage and management of documents, manuals, procedures and other files.
- Faster Information Retrieval: powerful search capabilities enable users to quickly find the information they need, reducing time wasted on searching for information.

2. PROJECT REQUIREMENTS

2.1 General Requirements:

1. User Interface (UI):

-Easy input and management of knowledge articles, documents, Resources. Intuitive navigation and search functionality

2. Content Management:

-Document and content management system for storing and organizing educational resources, research papers, course materials and administrative documents.

3. Security Features:

- Security measures to protect sensitive educational data. Access control and permissions management.

4. Notification System:

-Email or in-system notifications for updates, new contents and system announcements.

5. Feedback Mechanisms:

- Tools for users to provide feedback on content, suggest improvements or ask questions.

3. FUNCTIONAL REQUIREMENTS

3.1 User Authentication and Authorization

- Users shall be able to register and log in to the system.
- User roles shall include students, faculty, and staff with different levels of access permissions.
- Administrators shall have the authority to manage user accounts and roles

3.2 Document Management System

- Users shall be able to upload, categorize, and search documents.
- Documents shall be categorized by type (e.g., academic, administrative, extracurricular).
- Users shall be able to view, download, and comment on documents.

3.3 Announcement Board

- Administrators shall be able to post announcements and updates.
- Announcements shall be displayed prominently on the system dashboard.
- Users shall be able to view and comment on announcements

3.4 Search Functionality

- The system shall provide a search feature for users to find documents, discussions, events, and announcements.
- Search results shall be displayed based on relevance.

4. Non-functional Requirements

4.1 Performance

- The system shall be responsive and load quickly, even during peak usage times.
- Response times for user interactions shall be within acceptable limits.

4.2 Security

- User authentication and data transmission shall be encrypted using secure protocols.
- Access control mechanisms shall prevent unauthorized access to sensitive information.
- Regular security audits and updates shall be performed to mitigate security risks

4.3 Usability

- The user interface shall be intuitive and user-friendly.
- Help documentation and tooltips shall be provided to assist users in navigating the systems.
- The system shall support accessibility standards to accommodate users with disabilities.

4.4 Interoperability:

- Integration: The system should be capable of integrating with other systems and data sources, using standard APIs and data formats (e.g., RESTful services, XML, JSON).
- Compatibility: The system should be compatible with various operating systems, databases, and browsers.

AGILE – SCRUM METHODOLOGY

EX_NO:2 **DATE:**

1.**Product Backlog**

The product backlog defines the different features and functionalities. the website intends to achieve. It outlines the value that the website would add to the users.

1.1. User Authentication and Access
1.2. Set up Knowledge Management
1.3. Admin Dashboard and Localization
1.4. Report Generation
2.Scrum Backlog

Sprint 1 Backlog :

1) User Authentication
User Story: As a user, I want to securely register and log in to the e-parking system so that I can access my account.
Tasks:

- Implement user registration form with email and password.
- Develop email verification feature.
- Implement login functionality.
- Create password recovery feature.

Basic UI Setup
Tasks:
• Set up the basic layout and navigation.
• Design login, registration, and dashboard screens.

Sprint 2:

Sprint Goal: Set up Knowledge Management Requirements.
Project Management

- Task : Create database schema for storing details.
- Task : Implement knowledge creation functionality
- Task : Develop UI for adding and removing knowledge/data

Sprint 3

Admin Dashboard

Tasks:

• Develop an admin dashboard.
• Display user statistics and system logs.
• Implement user management features.

Localization

Tasks:

- Implement multi-language support.
- Allow users to select their preferred language.
- Translate key UI elements and notifications.

Sprint 4:

Sprint Goal: Implementation of Report Generation

Report Generation

Task: Develop report generation functionality for project managers.

Sprint Goal: Implementation Knowledge Management.

Milestone Management

Task: Develop UI for inserting, deleting, searching, editing data.

Task: Implement backend logic for tracking knowledge management.

Evaluation of Activities with data

Task: Integrate task status updates with knowledge management logic.

Task: Implement notifications for activities done by the user.

Task: Implement machine learning model to evaluate success or failure data management.

○ .

USER STORIES

EX_NO:3 **DATE:**

AS A STUDENT:

USER STORY - 1: Access Course Materials

- user Story: As a student, I want to access course materials so that I can review lecture notes, readings, and assignments.

Acceptance Criteria: I can search for and download course-specific documents, lecture notes, and readings**.**

USER STORY - 2: Find Campus Resources

User Story: As a student, I want to find information about campus resources like the library, counseling services, and extracurricular activities.

Acceptance Criteria: I can easily navigate to sections with comprehensive information about various campus resources.

USER STORY - 3: Ask Questions

User Story : As a student, I want to ask questions about my coursework so that I can get help from my professors or peers.

Acceptance Criteria: I can post questions in course-specific forums and receive notifications when someone responds.

USER STORY -4: Tracking Course Progress

User Story: As a student, I want to track my progress in each course so that I can stay on top of my studies and know what assignments or exams are upcoming.

Acceptance Criteria:

The system should provide a dashboard showing course progress, including completed and upcoming tasks.

Students should receive reminders for upcoming deadlines.

AS A FACULTY:

USER STORY -5: Upload Course Content

User Story: As a faculty member, I want to upload and organize course content so that my students can access it conveniently.

Acceptance Criteria: I can upload documents, create folders, and set permissions for student access.

USER STORY -6: Manage Student Information

User Story: As an administrative staff member, I want to manage student information so that I can keep records up-to-date and accessible.

Acceptance Criteria: I can update student profiles, track academic progress, and manage enrollment details**.**

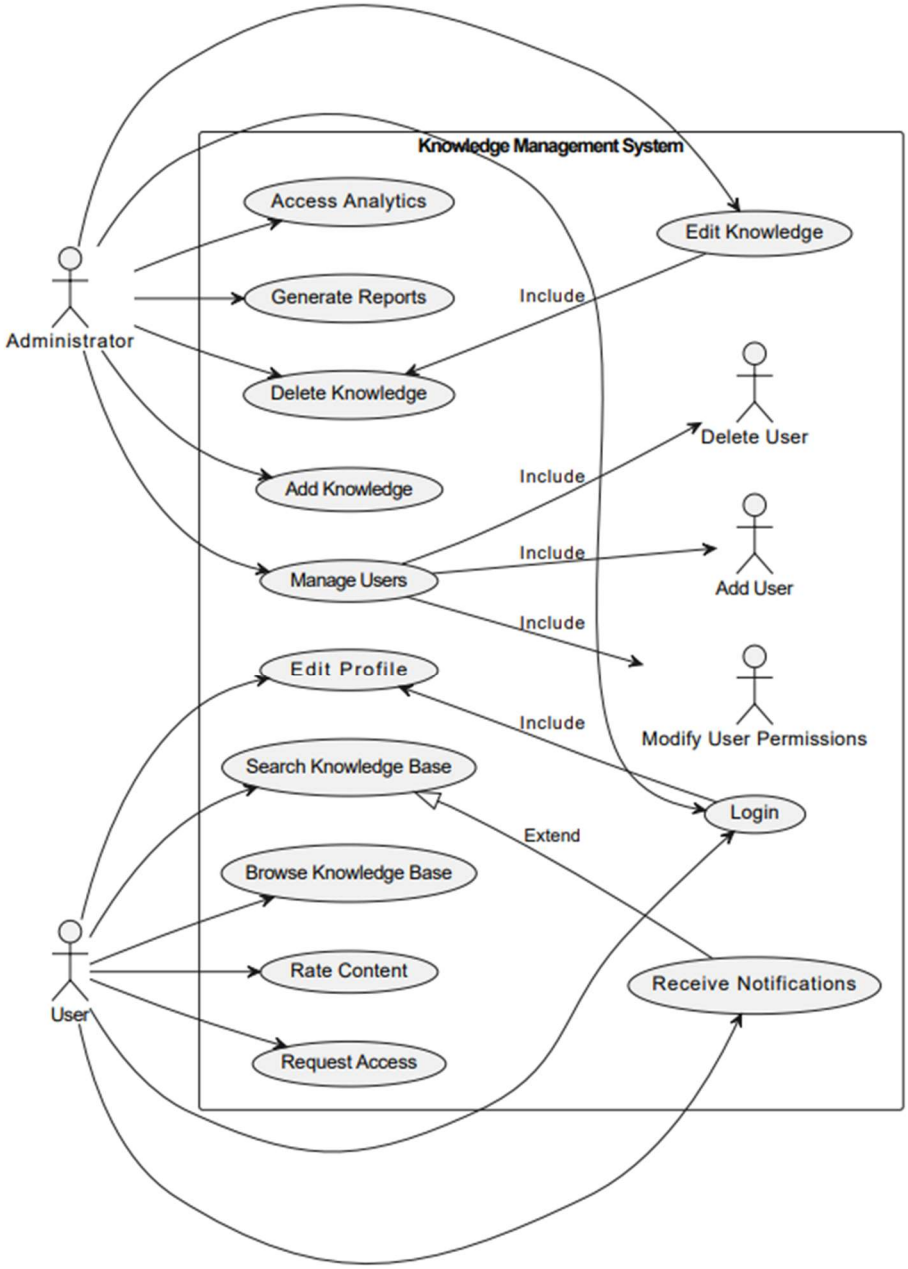
USER STORY – 7: Post Announcements

User Story: As an **administrative staff member**, I want to post announcements about important events and deadlines so that the college community stays informed.

Acceptance Criteria: I can create announcements, set visibility to specific groups (e.g., students, faculty), and schedule notifications**.**

USECASE DIAGRAM

EX_NO:4 **DATE:**



Administrator

Access Analytics:

View and analyze data about how the system is used.

Generate Reports:

Create reports based on system data.

Delete Knowledge:

Remove content from the system.

Add Knowledge:

Add new content to the system.

Manage Users:

Delete Users:

Remove users from the system.

Add Users:

Add new users to the system.

Modify User Permissions:

Change what users can do.

Edit Knowledge:

Update or change existing content**.**

Login:

Access the system by logging in.

Student

Edit Profile:

Update personal information.

Search Knowledge Base:

Look for specific content in the system.

Browse Knowledge Base:

Explore content in the system.

Rate Content:

Give feedback on content.

Request Access:

Ask for permission to access certain content or features.

Receive Notifications:

Get updates about important information or changes

NON-FUNCTIONAL REQUIREMENTS (NFR)

EX NO:5 **DATE:**

1. **Performance and Scalability:**

Response Time:

The system must load and display search results within 2 seconds of a query being submitted.

Scalability:

The system should handle up to 5,000 concurrent users without performance degradation, accommodating peak times during exams and future growth.

2. Security:

Data Protection:

All personal and academic data must be encrypted using industry-standard encryption techniques both at rest and in transit.

Access Control:

The system must implement role-based access control (RBAC) to ensure only authorized users can access sensitive information and perform administrative functions.

3. Usability:

User Interface:

The system interface must be intuitive and user-friendly, allowing users to navigate and perform tasks with minimal training.

Accessibility:

The system should comply with accessibility standards such as WCAG 2.1, ensuring it is usable by people with disabilities.

4. Maintainability:

Documentation:

Comprehensive documentation should be provided for both users and administrators to facilitate troubleshooting and training.

5. Compatibility:

Browser Support:

The system must be compatible with major web browsers, including Chrome, Firefox, Safari, and Edge.

Device Support:

The system should be fully functional on desktops, laptops, tablets, and smartphones.

6. Auditability:

Logging:

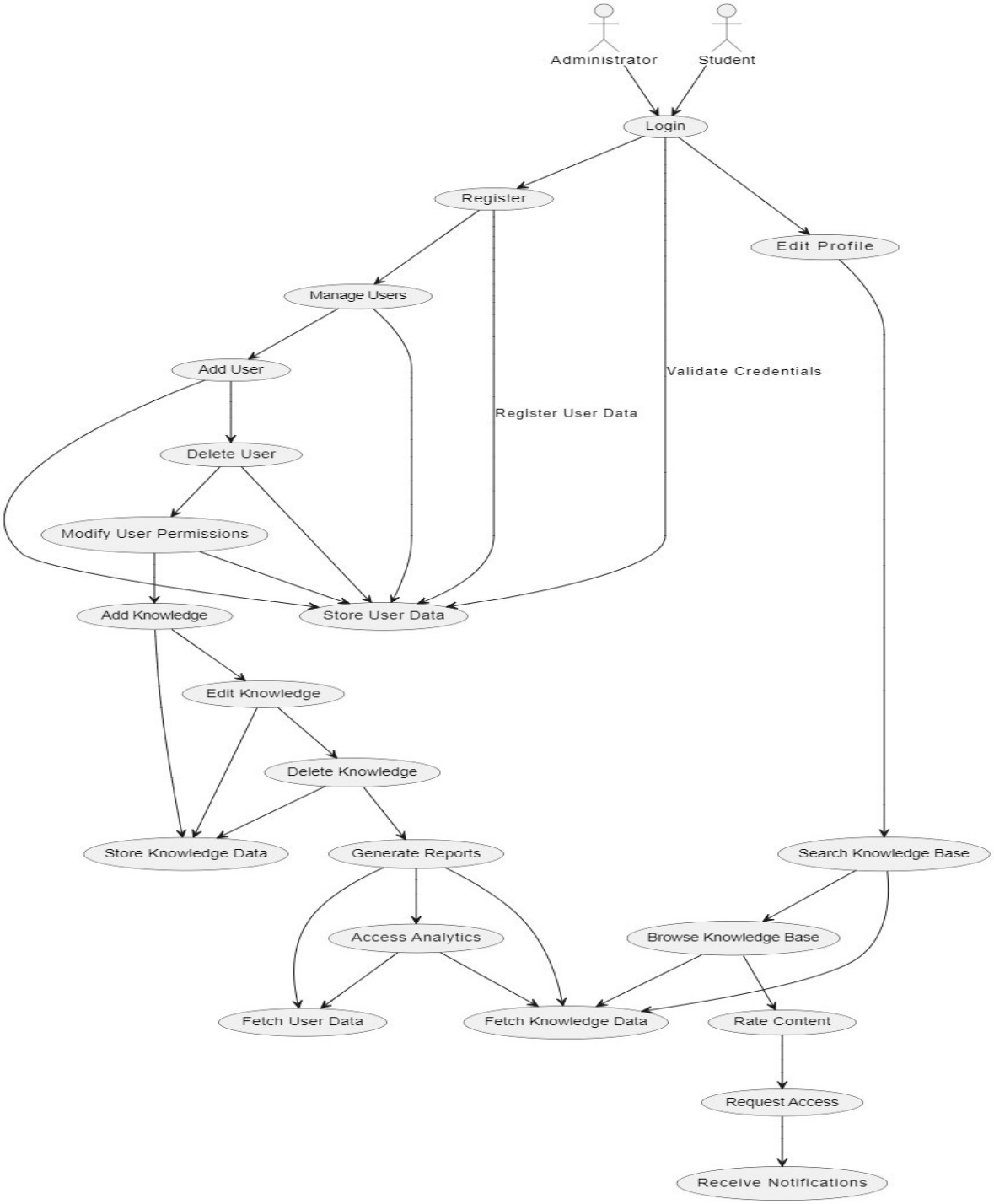
The system should maintain detailed logs of user activities, including access to sensitive data and administrative actions.

Audit Trails:

Audit trails should be available to track changes to knowledge base content and user permissions, ensuring accountability and transparency.

OVERALL PROJECT ARCHITECTURE

EX NO:6 **DATE:**



Actor Attributes:

Both the Administrator and the Student have their own unique attributes. For example, the Administrator may have attributes like "username," "password," and "role," while the Student may have attributes like "username," "password," and "enrollment status."

These attributes are crucial for identifying and authenticating users within the system.

Use Case Attributes:

Each use case, whether for the Administrator or the Student, may have specific attributes associated with it.

For instance, the "Login" use case for both actors would require attributes such as "username" and "password" to authenticate and grant access to the system.

Other use cases may have additional attributes. For example, the "Add Knowledge" use case for the Administrator may have attributes like "title," "description," and "category" for adding new knowledge entries.

Database Interactions:

Attributes are also essential in database interactions, where data related to users and knowledge entries are stored and retrieved.

The attributes associated with storing user data may include "user ID," "username," "password hash," etc.

Similarly, attributes related to knowledge data may include "knowledge ID," "title," "description," "author," etc.

Relationship between Attributes:

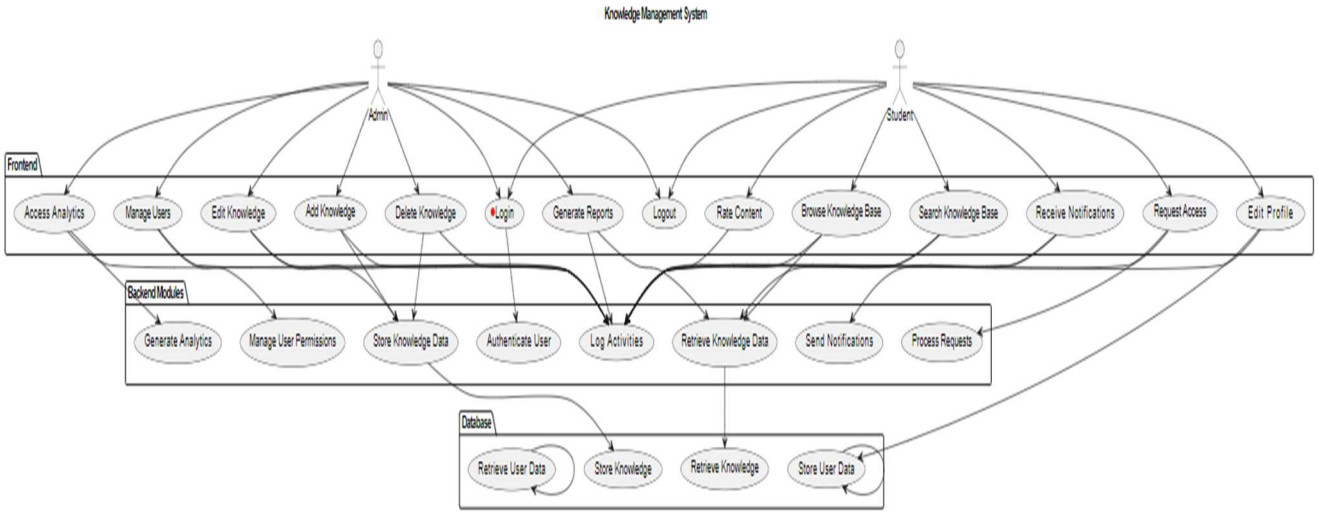
Attributes are interconnected with each other and with the actions performed by actors and use cases.

For example, the "Login" use case requires the attributes "username" and "password" to validate user credentials stored in the database.

Likewise, when the Administrator adds new knowledge to the system, attributes like "title," "description," and "category" are associated with the knowledge entry and stored in the database.

BUSINESS ARCHITECTURE DIAGRAM

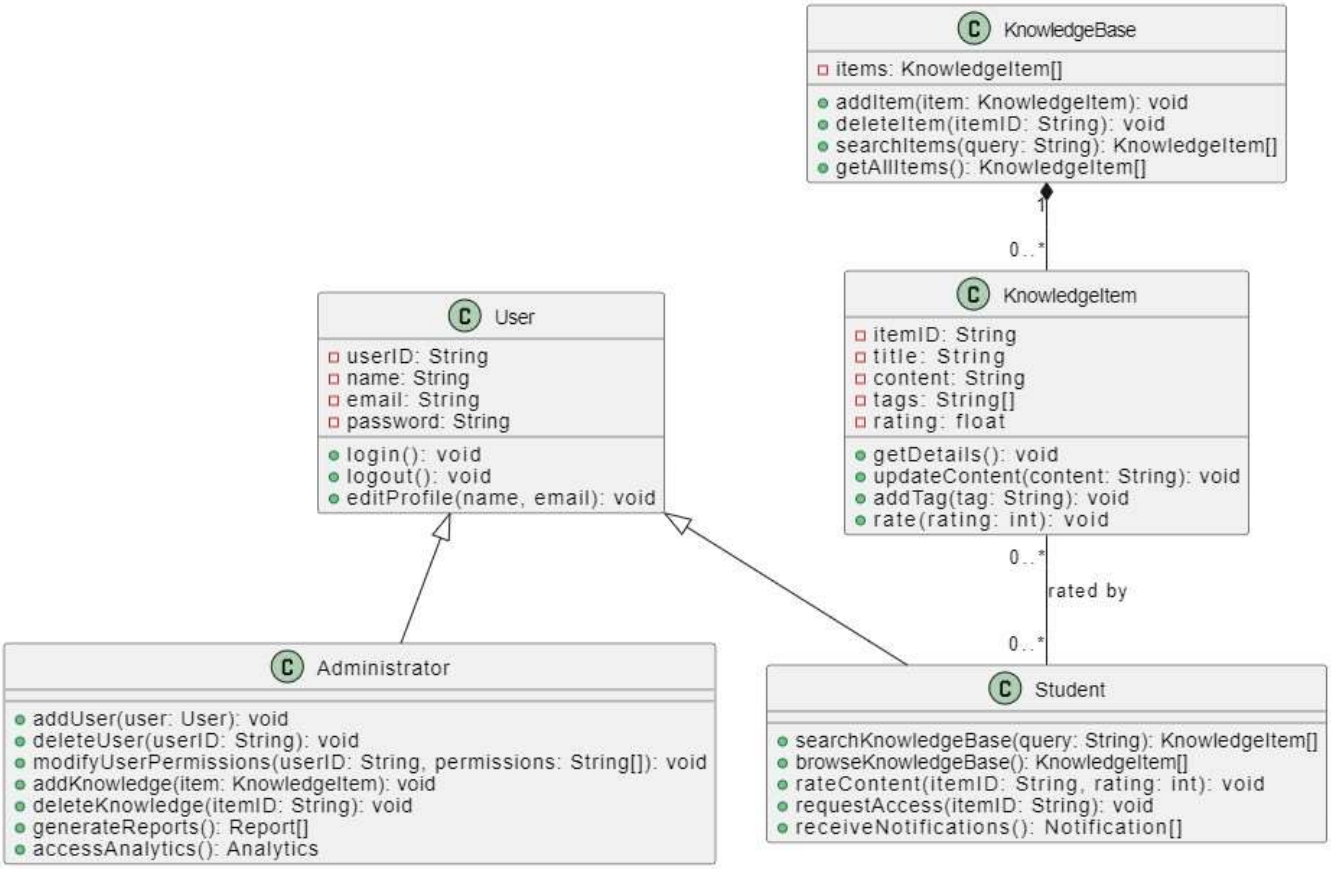
EX_NO:7 **DATE:**



CLASS DIAGRAM

EX_NO:8 **DATE:**

A class diagram is a type of UML(Unified Modeling Language) diagram that represents the structure and behavior of a system or application by illustrating the classes, attributes, methods, and relationships between objects. Class Diagrams are widely used in software development to visualize the static design of a system.



The above class diagram's breakdown into the classes and their relationships is given below,

User: Represents a user with attributes like user, name, email, and password. Methods include login(), logout(), and editProfile(name, email).

Administrator: Inherits from User. Additional methods for managing users (addUser(), deleteUser(), modifyUserPermissions()) and knowledge items (addKnowledge(), deleteKnowledge()), and generating reports and accessing analytics.

Student: Also inherits from User. Methods include searching and browsing knowledge items (searchKnowledgeBase(), browseKnowledgeBase()), rating content (rateContent()), requesting access to items, and receiving notifications.

KnowledgeBase: Manages KnowledgeItem instances. Methods include adding, deleting, and searching for items (addItem(), deleteItem(), searchItems(), getAllItems()).

KnowledgeItem: Represents items in the knowledge base, such as articles or documents. Attributes include itemID, title, content, tags, and rating. Methods include retrieving details (getDetails()), updating content (updateContent()), adding tags, and rating items.

Login: Represents the login credentials associated with a User. Attributes include mailID and otp, with a method validate() to authenticate the user.

The relationships depicted in the diagram are:

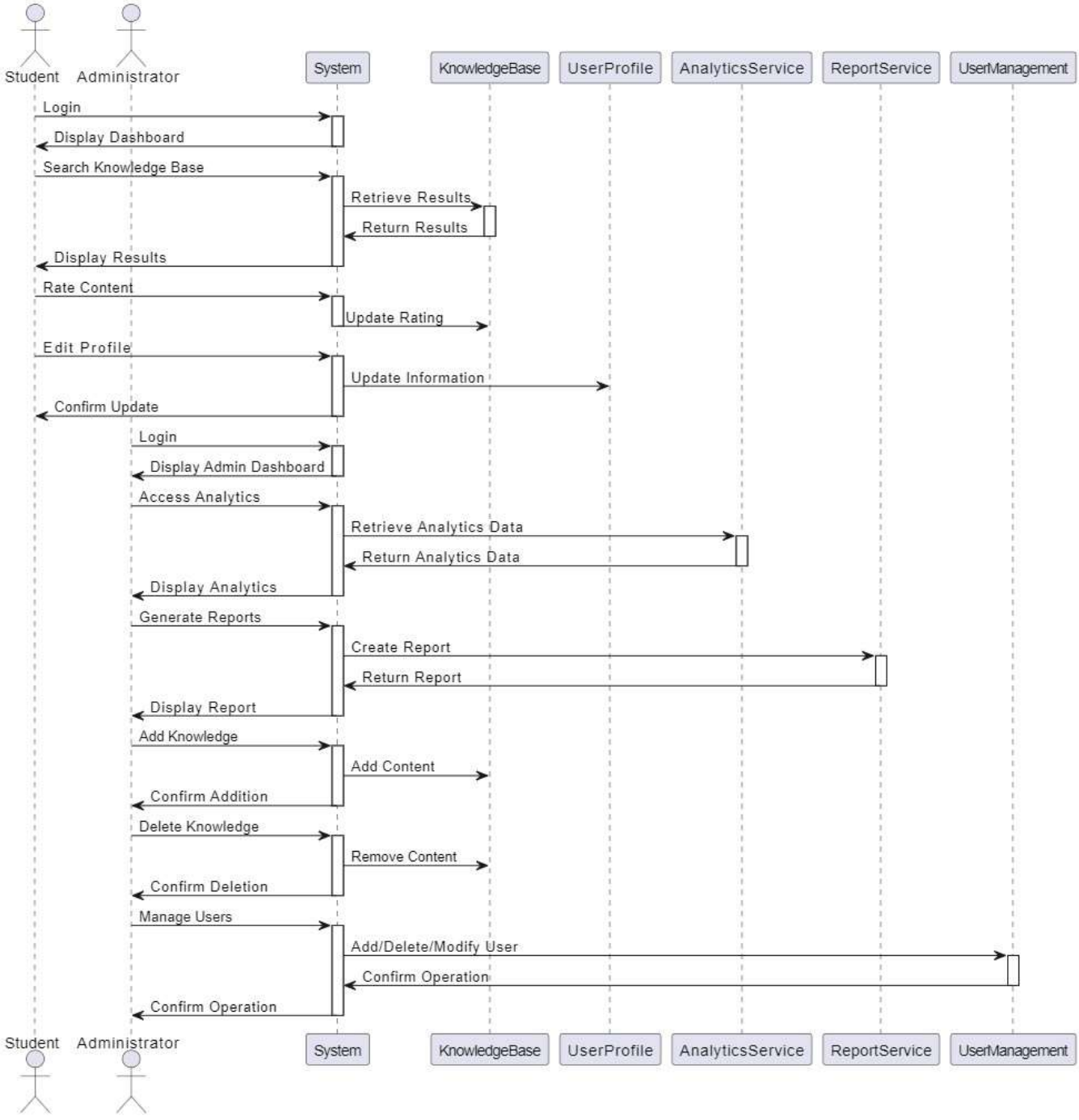
User has a one-to-one relationship with Login (each User has one Login).
KnowledgeBase has a one-to-many relationship with KnowledgeItem (each KnowledgeBase manages multiple KnowledgeItems).
KnowledgeItem has a many-to-many relationship with Student (items can be rated by multiple students).

KnowledgeBase is managed by Administrator and accessed by Student, indicating their roles in managing and using the knowledge base.
Administrator issues KnowledgeItems, showing their authority over the knowledge base content.
This diagram illustrates the structure and interactions within a system involving users, administrators, students, and a knowledge base with items and login credentials.

SEQUENCE DIAGRAM

EX NO:9 **DATE:**

.



Student Actions:
Login:

Actor: Student
Action: The student initiates a login request to the system.
System Response: The system processes the login and displays the student's dashboard.
Search Knowledge Base:

Actor: Student
Action: The student searches for specific information in the knowledge base.
System Response:
The system retrieves relevant results from the Knowledge Base.
The system displays the search results to the student.
Rate Content:

Actor: Student
Action: The student rates a piece of content in the knowledge base.
System Response:
The system updates the rating for that content in the Knowledge Base.
Edit Profile:

Actor: Student
Action: The student edits their profile information.
System Response:
The system updates the student's information in the UserProfile.
The system confirms the update to the student.
Administrator Actions:
Login:

Actor: Administrator
Action: The administrator initiates a login request to the system.
System Response: The system processes the login and displays the administrator's dashboard.
Access Analytics:

Actor: Administrator
Action: The administrator accesses system analytics.
System Response:
The system retrieves analytics data from the AnalyticsService.
The system displays the analytics data to the administrator.
Generate Reports:

Actor: Administrator
Action: The administrator generates a report.
System Response:
The system creates the report using the Report Service.The system displays the generated report to the administrator.
Add Knowledge:

Actor: Administrator
Action: The administrator adds new content to the knowledge base.
System Response:
The system adds the content to the Knowledge Base.
The system confirms the addition to the administrator.
Delete Knowledge:

Actor: Administrator
Action: The administrator deletes content from the knowledge base.
System Response:
The system removes the content from the Knowledge Base.
The system confirms the deletion to the administrator.
Manage Users:

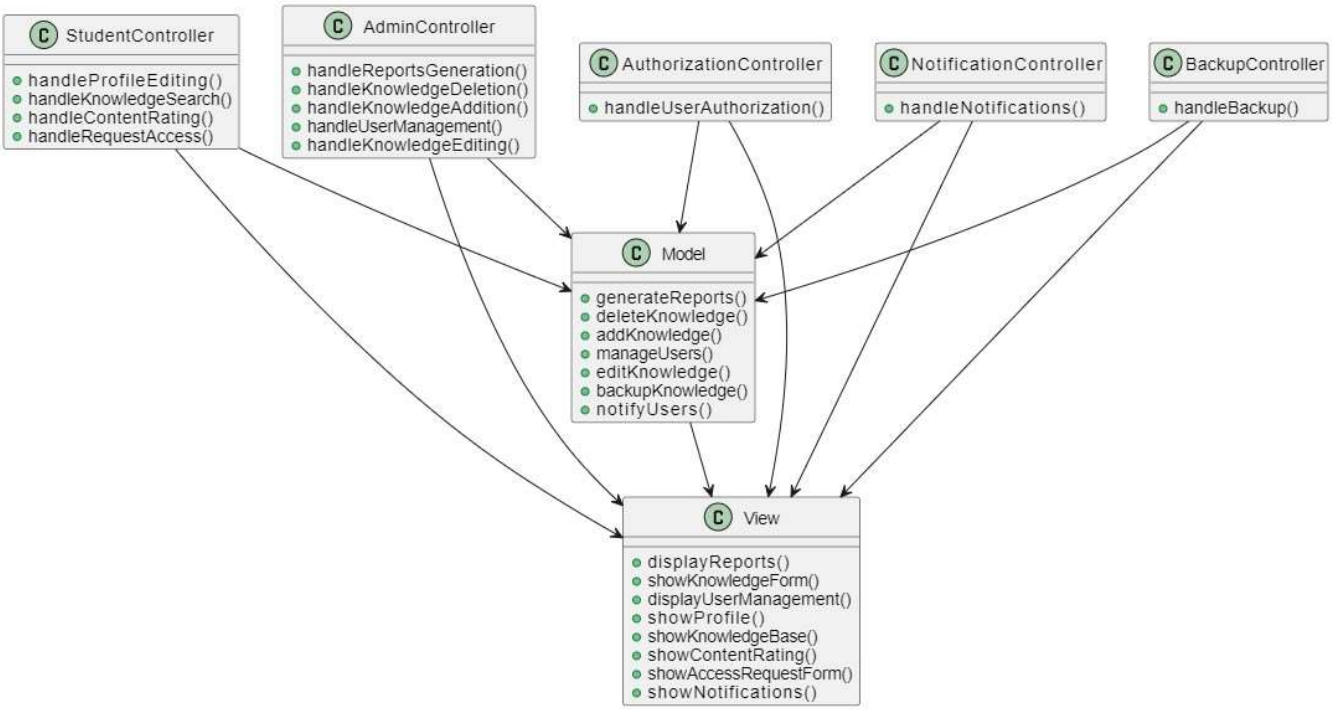
Actor: Administrator
Action: The administrator manages user accounts, which includes adding, deleting, or modifying user permissions.
System Response:
The system performs the requested user management action using the User Management service.
The system confirms the operation to the administrator.
Summary
Student Interactions:

The student logs in, searches the knowledge base, rates content, and edits their profile.
These actions involve the system interacting with the Knowledge Base, User Profile, and related services to retrieve, update, and display information.
Administrator Interactions:

The administrator logs in, accesses analytics, generates reports, and manages knowledge and user accounts.
These actions involve the system interacting with Analytics Service, Report Service, Knowledge Base, and User Management to perform administrative tasks and display the results.
Each interaction is represented by an arrow indicating the request and response between the actors (students and administrators) and the system components, ensuring a clear understanding of how the system processes each action.

ARCHITECTURAL PATTERN (MVC)

EX NO:10 **DATE:**



Model:

Represents the data and business logic of the application.

Contains methods for generating reports, managing knowledge (addition, deletion, editing), user management, backup knowledge, and notification.

View:

Handles the presentation layer of the application.

Provides interfaces for displaying reports, forms for knowledge management, user management, profiles, knowledge base, content rating, access request, and notifications.

StudentController:

Handles actions related to student users.

Includes methods for editing profiles, searching the knowledge base, rating content, and requesting access to knowledge.

AdminController:

Handles actions related to administrator users.

Includes methods for generating reports, managing knowledge (addition, deletion, editing), and user management.

AuthorizationController:

Handles user authorization/authentication.

Contains methods for managing user permissions and access control.

NotificationController:

Handles notifications for users.

Contains methods for sending notifications to users about various events or updates.

BackupController:

Handles backup-related functionalities.

Includes methods for backing up knowledge data.

Relationships:

Controllers to Model:

All controllers interact with the Model to perform data-related operations such as generating reports, managing knowledge, user authorization, notifications, and backups.

Controllers to View:

Each controller interacts with the View to present the results of the operations to the users. For example, the StudentController may interact with the View to display a profile editing form or search results from the knowledge base.

Model to View:

The Model communicates with the View to update the user interface based on changes in data or state. For instance, when new knowledge is added or deleted, the View is updated accordingly to reflect these changes.

This architecture follows the MVC (Model-View-Controller) pattern where the Model represents data and logic, the View handles presentation, and the Controller manages user interactions and application flow. The additional controllers address specific aspects such as user authorization, notifications, and backups, enhancing the modularity and scalability of the system.