UNIVERSITY OF NUEVA CACERES

College of Computer Studies

System Requirement Specifications and Software Design

of

Class Management

for

Accountancy Review Management System

Prepared by:

Zeus Asico Tenerife

SYSTEM REQUIREMENT

FUNCTIONAL REQUIREMENTS

- ➔ The system shall allow the Dean and Program head to add Faculties to the system and should create an account for their faculty role.
- ➔ The system shall allow the Program head to assign the faculty to the subject.
- ➔ The system shall allow the faculty to create and edit a learning plan or syllabus.
- ➔ The system shall allow the faculty to submit their learning plan to the Dean or Program head.
- ➔ The system shall allow the Dean to approve, disapprove and give feedback to the learning plan from the faculty.
- ➔ The system shall allow the Dean or Program head to create and edit a class.
- ➔ The system shall allow the Program head to assign the faculties to the class
- ➔ The system shall allow the Program head to enroll the students to the class.
- ➔ The system shall allow the Program head to assign the subjects to the class.
- ➔ The system shall allow the Program head to assign the learning plan of every subject to the class.
- ➔ The system shall allow the students to view the class along with their subjects

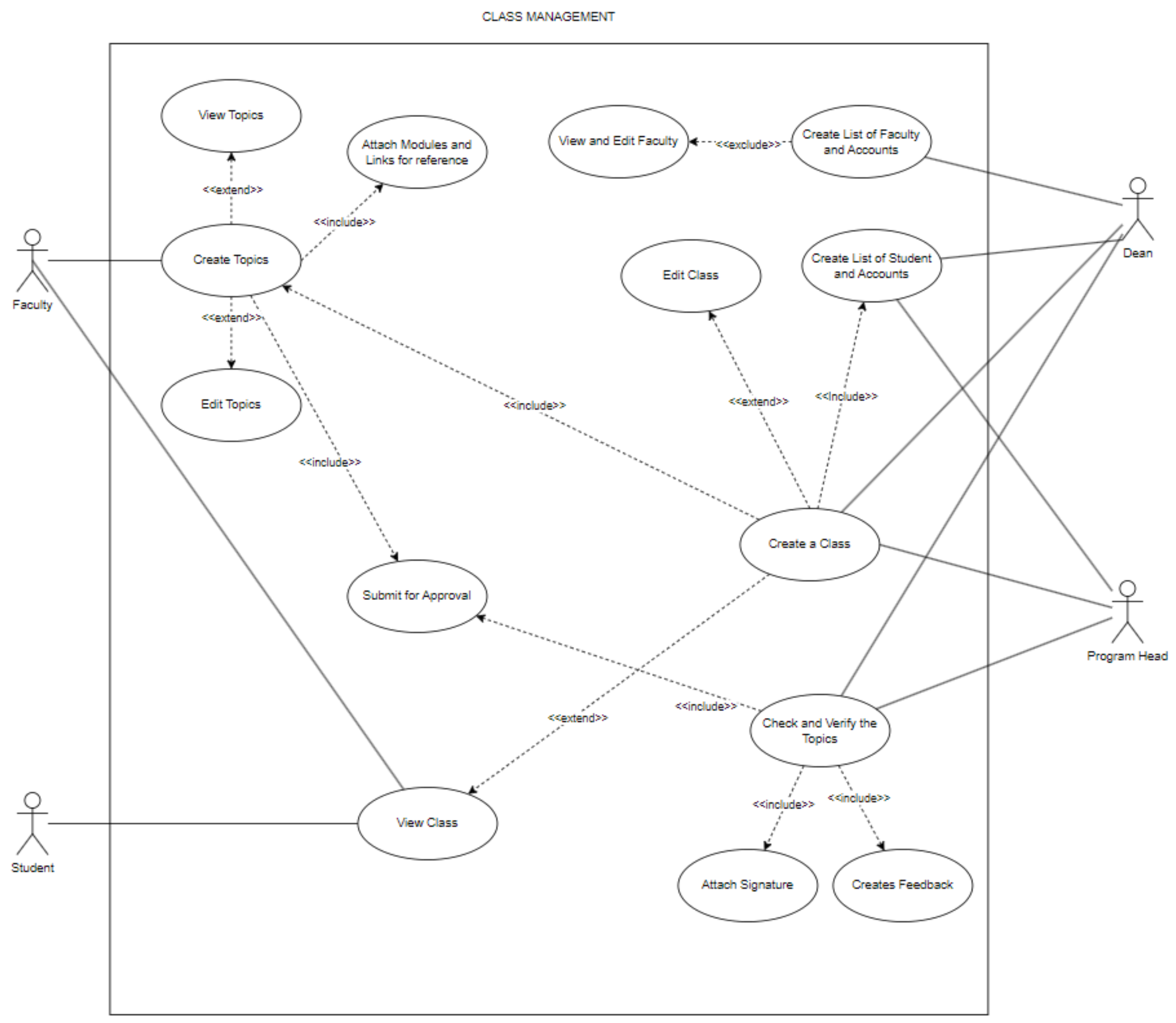
NON-FUNCTIONAL REQUIREMENTS

- The system shall ensure data integrity with appropriate measures in place to prevent data loss or corruption of documents.
- The system shall use role-based access control to ensure only authorized users (Dean, Program head, faculty, students) can perform specific actions.
- The system shall encrypt sensitive data, including user credentials and personal information.
- The system shall be designed with modular components to facilitate easy maintenance and updates.
- The system shall include comprehensive logging and error reporting to assist in troubleshooting and maintenance.
- The system shall provide a reliable backup and recovery mechanism to prevent data loss.

- The system shall be compatible with major web browsers (Chrome, Firefox, Safari, Edge).
- The system shall support access from various devices, including desktops, laptops, tablets, and smartphones.

SOFTWARE DESIGN

USE-CASE DIAGRAM



USE-CASE SCENARIO & SEQUENCE DIAGRAM

Use case name: Create Topics	**UniqueID:** FC_001
Area: Topics Development	
Actor(s): Faculty	
Description: The faculty members create Topics for students. These topics needs approval of the program head for implementation	
Triggering Event: Faculty members need to upload a new Intervention plan or manage existing Intervention plans for a course or module.	
Trigger Type: ✓ ~~External~~ ☐ Temporal	
Steps Performed (Main Path)	**Information for Steps**
1. Create a Topic and Include the Title and description of the topic	Topic Creation Form
2. Upload Title of the module and sub-modules, Attachments for any file related to the topic and Links for references	Module Creation Form
3. Save the Topic Created	Saving Page
4. Submit for Approval of the Program Head	Submission Page
Preconditions: - Faculty members must have a valid account and appropriate permissions to access the panel. - Topics should be prepared and ready for upload in digital format.	
Postconditions: Existing Topics are managed and updated within the Accountancy Review Management System.	
Assumptions: Existing Topics are stored securely within the system and can only be accessed by authorized users.	
Requirement Met: - Topics are organized and accessible to faculty members within the system. - Changes made to Topics are saved and reflected in the system's database.	
Outstanding Issues: Version control for Topics may need to be considered to track revisions and updates over time.	
Priority: Medium to High	
Risk: High	

Use case name: View Topics	**UniqueID:** FC_002
Area: Viewing of the Topics	

Actor(s): Faculty	
Description: The faculty members should be able to view the topics created	
Triggering Event: faculty use ARMS and login using their account, Goes to Topic List, and view a certain topic	
Trigger Type: ☑ ~~External~~ ☐ Temporal	
Steps Performed (Main Path)	**Information for Steps**
1. Viewing of the Topics	Topic List View
2. Viewing the content of the created modules and submodules.	Module View
Preconditions: - Faculty members must have a valid account and appropriate permissions to access the panel. - Topics should be prepared and ready for upload in digital format.	
Postconditions: Existing Topics are viewed.	
Assumptions: Existing Topics that are on the database can only be accessed by professors.	
Requirement Met: - View the Topics available from the system's database.	
Outstanding Issues: Appropriate access control for privileged users.	
Priority: Medium to High	
Risk: High	

Use case name: Attach Modules and Links for reference	**UniqueID:** FC_003
Area: Attaching of different Modules and Links	
Actor(s): Faculty	
Description: The faculty members should be able to attach modules and Links for reference of a certain module or sub-module.	
Triggering Event: faculty use ARMS and login using their account, Goes to Topic List, and view a certain topic	
Trigger Type: ☑ ~~External~~ ☐ Temporal	
Steps Performed (Main Path)	**Information for Steps**
1. Create Title of the module and sub-modules	Module Creation Form
2. Upload Attachments for any file related to the topic	Module Creation Form

3. Attach the Links for references needed for the certain module	Module Creation Form
4. Save the modules.	Saving Page
Preconditions: - Faculty members must have a valid account and appropriate permissions to access the panel. - Topics should be created and available for attachment of modules	
Postconditions: Topics are updated with the module, sub-modules and links for reference.	
Assumptions: Existing Topics that are on the database can only be accessed by professors.	
Requirement Met: - Create Modules and sub-modules for the certain topic. - Include links or attachments needed for the topic.	
Outstanding Issues: Appropriate access control for privileged users.	
Priority: Medium to High	
Risk: High	

Use case name: Edit Topics	**UniqueID:** FC_004
Area: Edit of the topics created	
Actor(s): Faculty	
Description: The faculty members should be able to edit the topics before submission for approval	
Triggering Event: faculty use ARMS and login using their account, Goes to Topic List, and Edit a topic	
Trigger Type: [✓ ~~External~~] ☐ Temporal	
Steps Performed (Main Path)	**Information for Steps**
1. Open existing topic and edit for any updates	Topic Edit Form
2. Change and update the modules.	Topic Edit Form
5. Update the Links for references needed for the certain module	Edit Module Form
6. Save the modules.	Saving Page
7. Submit for Approval of the Program Head	Submission Page
Preconditions: - Faculty members must have a valid account and appropriate permissions to access the panel. - Topics should be created and available for editing.	

Postconditions: Topics are updated with the new or refined module, sub-modules and links for reference.
Assumptions: Existing Topics that are on the database can only be accessed by professors.
Requirement Met: - Update the modules, sub-modules and links. - Change or revise the required revision for the Topics
Outstanding Issues: Version control for Topics may need to be considered to track revisions and updates over time.
Priority: Medium to High
Risk: High

Use case name: Submission for Approval	**UniqueID:** FC_005
Area: Edit of the topics created	
Actor(s): Faculty	
Description: The faculty members should be able to submit the topics created.	
Triggering Event: Faculty use ARMS and login using their account, Goes to Topic List and submit the topics for approval.	
Trigger Type: [✓ ~~External~~] ☐ Temporal	
Steps Performed (Main Path)	**Information for Steps**
1. Final check and revision of the topic	Topic Overview
2. Submit the document for approval	Approval Page
Preconditions: - Faculty members must have a valid account and appropriate permissions to access the panel. - Topics should be available for approval.	
Postconditions: Program head or the Dean should evaluate the submitted topics.	
Assumptions: Existing Topics that are on the database can only be accessed and used in the evaluation.	
Requirement Met: - Submit the Topics for approval	
Outstanding Issues: Version control for Topics may need to be considered to track revisions and updates over time.	
Priority: Medium to High	
Risk: High	

Use case name: View Class	**UniqueID:** UC_001
Area: View the Class Overview	
Actor(s): Student, Faculty, Dean and Program Head	
Description: The Users should be able to view the class information	
Triggering Event: Student, Faculty, Dean and Program Head use ARMS and login using their account, Goes to the class information	
Trigger Type: [✓ ~~External~~] ☐ Temporal	
Steps Performed (Main Path)	**Information for Steps**
1. Open Existing Class	Class Viewer
2. View the Students List Information	Students Information View
3. View topics assigned to the Class	Topics View
4. View the Modules available for the class	Module View
Preconditions: - Faculty members must have a valid account and appropriate permissions to access the panel. - Classes should be available and Students should be enrolled in the subject.	
Postconditions: Class should be registered and can be retrieved from systems database	
Assumptions: Existing classes that are on the database can only be accessed.	
Requirement Met: - View the Students enrolled in the class - View the topics included in the class - View the modules present in the current class	
Outstanding Issues: Version control may need to be considered to track revisions and updates over time.	
Priority: Medium	
Risk: Low	

Use case name: Check and Verify Topics	**UniqueID:** DP_001
Area: Check and verification of the topics created	
Actor(s): Dean and Program Head	

Description: The Dean and Program Head members should be able to check and verify the submitted topics of the faculty.	
Triggering Event: Dean and Program Head use ARMS and login using their account, Goes to Topic List and undergoes verification of topics.	
Trigger Type: [✓ ~~External~~] ☐ Temporal	
Steps Performed (Main Path)	**Information for Steps**
1. Open Submitted Topics by Professor	Topic List Viewer
2. Evaluate and check for errors the Topic Submitted	Evaluation Form
3. Disapprove topics should be returned to the professor and Have the feedback for revision.	Evaluation Form
4. Approve Topics will be Implemented attach with signatures	Implementation Form
5. Save Changes to the topics	Saving Page
6. Update the status of the submitted topic	Update Page
Preconditions: - Faculty members must have a valid account and appropriate permissions to access the panel. - Topics should be created and available for evaluation.	
Postconditions: Program head or the Dean should check and verify the submitted topics.	
Assumptions: Existing Topics that are on the database can only be accessed and used in the evaluation.	
Requirement Met: - Topics are checked and verified	
Outstanding Issues: Version control for Topics may need to be considered to track revisions and updates over time. New Updates on the approved topics cannot be change or modify	
Priority: Medium to High	
Risk: High	

Use case name: Attach Signature	**UniqueID:** DP_002
Area: Dean or Program head Attachment of Signature	
Actor(s): Dean and Program Head	
Description: The Dean and Program Head members should be able to attach signature on approved topics	

Use case name: Creates Feedback	**UniqueID:** DP_003
Area: Dean or Program head creates feedback for disapprove topics	
Actor(s): Dean and Program Head	
Description: The Dean and Program Head members should be able to attach feedback that needs to address in the created topics of the professor	
Triggering Event: Dean and Program Head use ARMS and login using their account, Goes to Topic List and clicks a topic and creates feedback for disapproved topics.	
Trigger Type: [✓ ~~External~~] ☐ Temporal	
Steps Performed (Main Path)	**Information for Steps**
1. Create Feedback from the evaluation of the topics	Feedback Form
2. Return the document to the professor for revision	Return Page

Preconditions: - Faculty members must have a valid account and appropriate permissions to access the panel. - Topics should be created and undergo checking and verification. - The topic should be approved for feedback.
Postconditions: Program head or the Dean should attach a feedback for revision of topics.
Assumptions: Existing Topics that are on the database can only be accessed and used in the evaluation.
Requirement Met: - Disapprove Topics are attached with feedback. - The topic is returned to the professor to have modification.
Outstanding Issues: Version control for Topics may need to be considered to track revisions and updates over time.
Priority: Medium to High
Risk: High

Use case name: Create a class	**UniqueID:** DP_004
Area: Dean or Program head creates a new class	
Actor(s): Dean and Program Head	
Description: The Dean and Program Head members should be able to create a new Class.	
Triggering Event: Dean and Program Head use ARMS and login using their account, Goes to Class List and starts to create a new class	
Trigger Type: [✓ ~~External~~] ☐ Temporal	
Steps Performed (Main Path)	**Information for Steps**
1. Create a new batch of Class	Class Creation Form
2. Enroll the Students Information	Students Enrollment Form
3. Include the topics covered in the Class	Class Creation Form
4. Save the Class	Saving Page
Preconditions: - Faculty members must have a valid account and appropriate permissions to access the panel. - Topics should be approved for implementation - Student should be enrolled to the review system	
Postconditions: - The new class is successfully created and added to the class list.	

- Students are enrolled in the class with their information recorded. - Each student has an account generated for accessing class materials and resources. - The topics covered in the class are included in the class details.
Assumptions: Students are enrolled to all the topics included for review
Requirement Met: - A new class is successfully created and added to the class list within the ARMS system. - Students are enrolled in the class with their information recorded accurately. - The topics covered in the class are specified and included in the class details.
Outstanding Issues: Validation required for ensuring that all necessary permissions and access controls are properly implemented to restrict unauthorized access to class creation functionality.
Priority: Medium to High
Risk: High

Use case name: Create List of Student and Accounts	**UniqueID:** DP_005
Area: Dean or Program head create the list of student and accounts	
Actor(s): Dean and Program Head	
Description: The Dean and Program Head members should be able to attach students information and generate accounts for students.	
Triggering Event: Dean and Program Head use ARMS and login using their account, Goes to Class List and adds the enrolled students.	
Trigger Type: [✓ ~~External~~] ☐ Temporal	
Steps Performed (Main Path)	**Information for Steps**
1. Initiate the process of creating a new list of students.	Student List Form
2. Adds the information of enrolled students to the list.	Student Information Form
3. System generates an account for student	Student Account generation Form
4. Save the Student List	Saving Page
5. Review the created list to ensure accuracy and completeness of student information and associated accounts.	Confirmation and Review Page
Preconditions: - Dean or Program Head must have valid login credentials and appropriate permissions to access the student management section of ARMS.	

- Enrollment information of students must be available and up-to-date. - Necessary resources and support systems for generating student accounts must be functional within ARMS.
Postconditions: - A comprehensive list of enrolled students along with their associated accounts is successfully created and saved within ARMS. - Each enrolled student has a corresponding account generated for accessing the system. - The created list is accessible for further management and administrative tasks within ARMS.
Assumptions: - The enrollment information for students is readily available and accurate. - ARMS has the capability to generate accounts for students automatically or with minimal manual intervention.
Requirement Met: - The Dean and Program Head are able to create a comprehensive list of enrolled students. - Each student on the list has an associated account generated within ARMS. - The process ensures accurate linkage between student information and their respective accounts.
Outstanding Issues: - Validation required for the efficiency and scalability of the account generation process, especially for large numbers of students. - Confirmation required on how students are notified about their generated accounts and provided with access instructions.
Priority: Medium to High
Risk: High

Use case name: Create List of Professor and Accounts	**UniqueID:** DC_001
Area: Dean	
Actor(s): Dean	
Description: The Dean should be able to attach Professor information and generate accounts for students.	
Triggering Event: Dean uses ARMS and login using their account, Goes to Professor List and adds the professor teaching the Review for CPALE.	
Trigger Type: [✓ ~~External~~] ☐ Temporal	
Steps Performed (Main Path)	**Information for Steps**
1. Initiate the process of creating a new list of professors.	Student List Form
2. Adds the information of the professor to the list.	Student Information Form
3. System generates an account for professor	Student Account generation Form

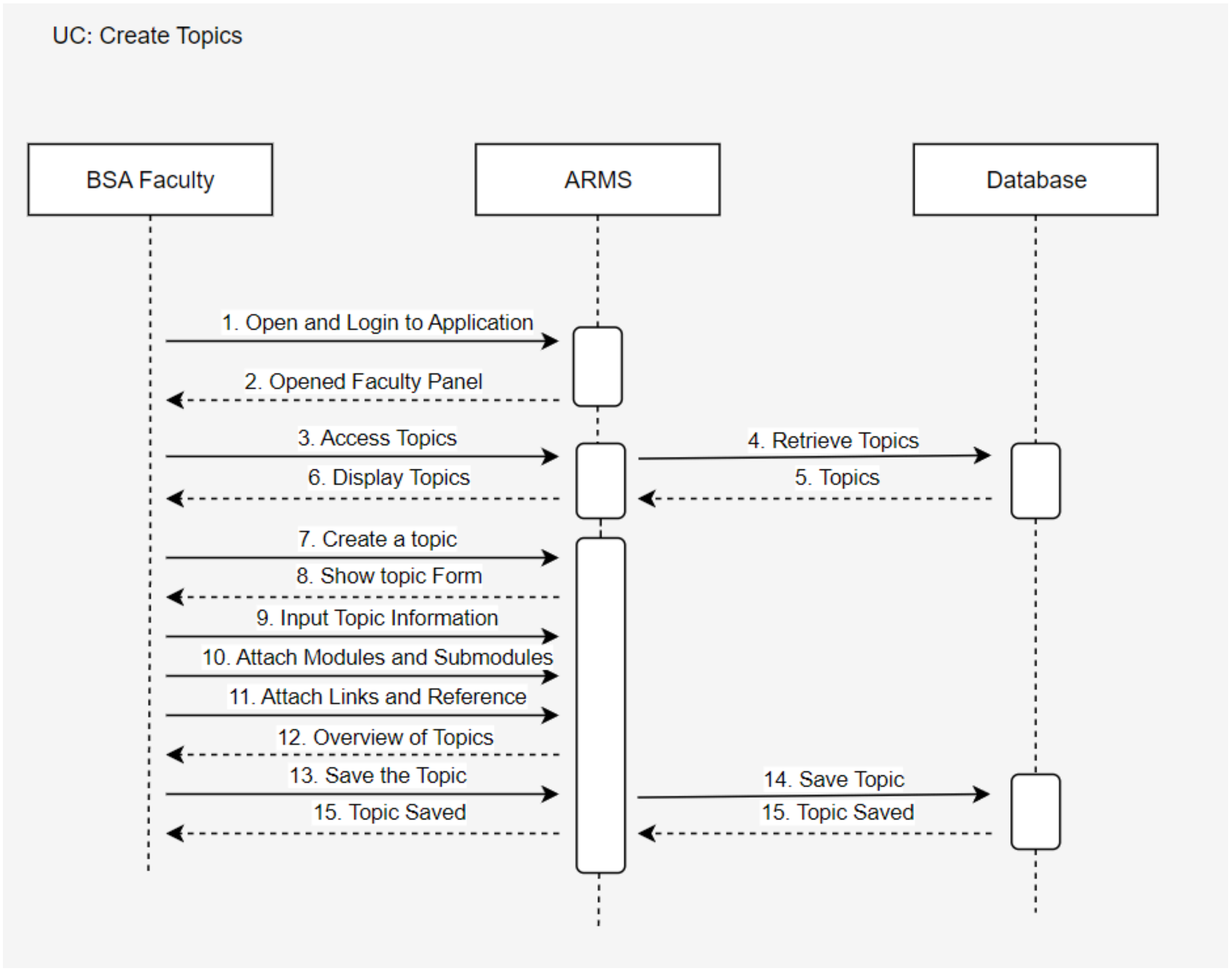
4. Save the professor List	Saving Page
5. Review the created list to ensure accuracy and completeness of professor information and associated accounts.	Confirmation and Review Page
Preconditions: - Dean or Program Head must have valid login credentials and appropriate permissions to access the student management section of ARMS.	
Postconditions: - A comprehensive list of professors along with their associated accounts is successfully created and saved within ARMS. - Each professor on the list has a corresponding account generated for accessing the system	
Assumptions: - Professor information, such as contact details and department, is readily available and accurate. - ARMS has the capability to automatically generate accounts for professors or facilitate the account creation process with minimal manual intervention.	
Requirement Met: - Dean is able to create a comprehensive list of professors and their associated accounts within ARMS.	
Outstanding Issues: - Confirmation required on how professors are notified about their generated accounts and provided with access instructions. - Clarification needed on any specific requirements or criteria for professor account credentials (e.g., password complexity, expiration policies).	
Priority: Medium to High	
Risk: High	

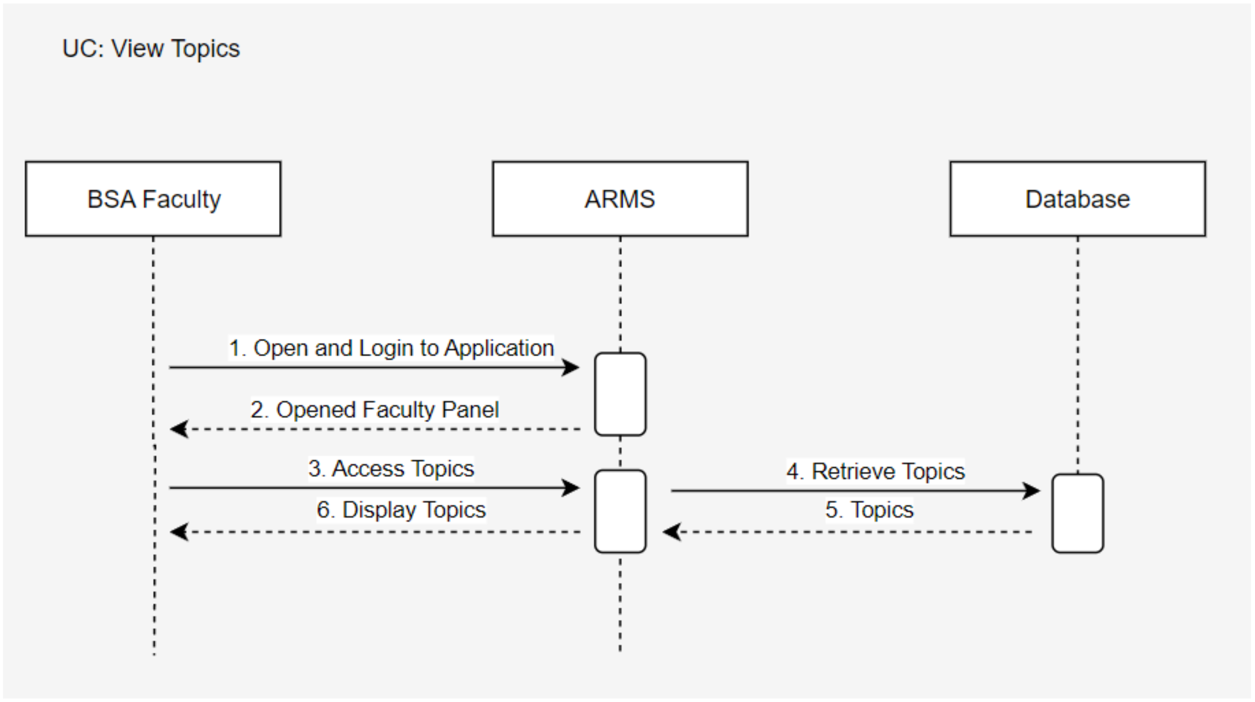
Use case name: View and Edit Faculty	**UniqueID:** DC_002
Area: View the list of professor and Edit Faculty	
Actor(s): Dean	
Description: The Dean should be able to view and edit the professors information and List of faculty available.	
Triggering Event: Dean uses ARMS and login using their account, Goes to Professor List and view the faculty enrolled to ARMS	
Trigger Type: [✓ ~~External~~] ☐ Temporal	
Steps Performed (Main Path)	**Information for Steps**
1. Display the List of Faculty Enrolled to the system	View Faculty Information
2. The Dean reviews the displayed information to gain insights into the faculty members available within the system.	Faculty List

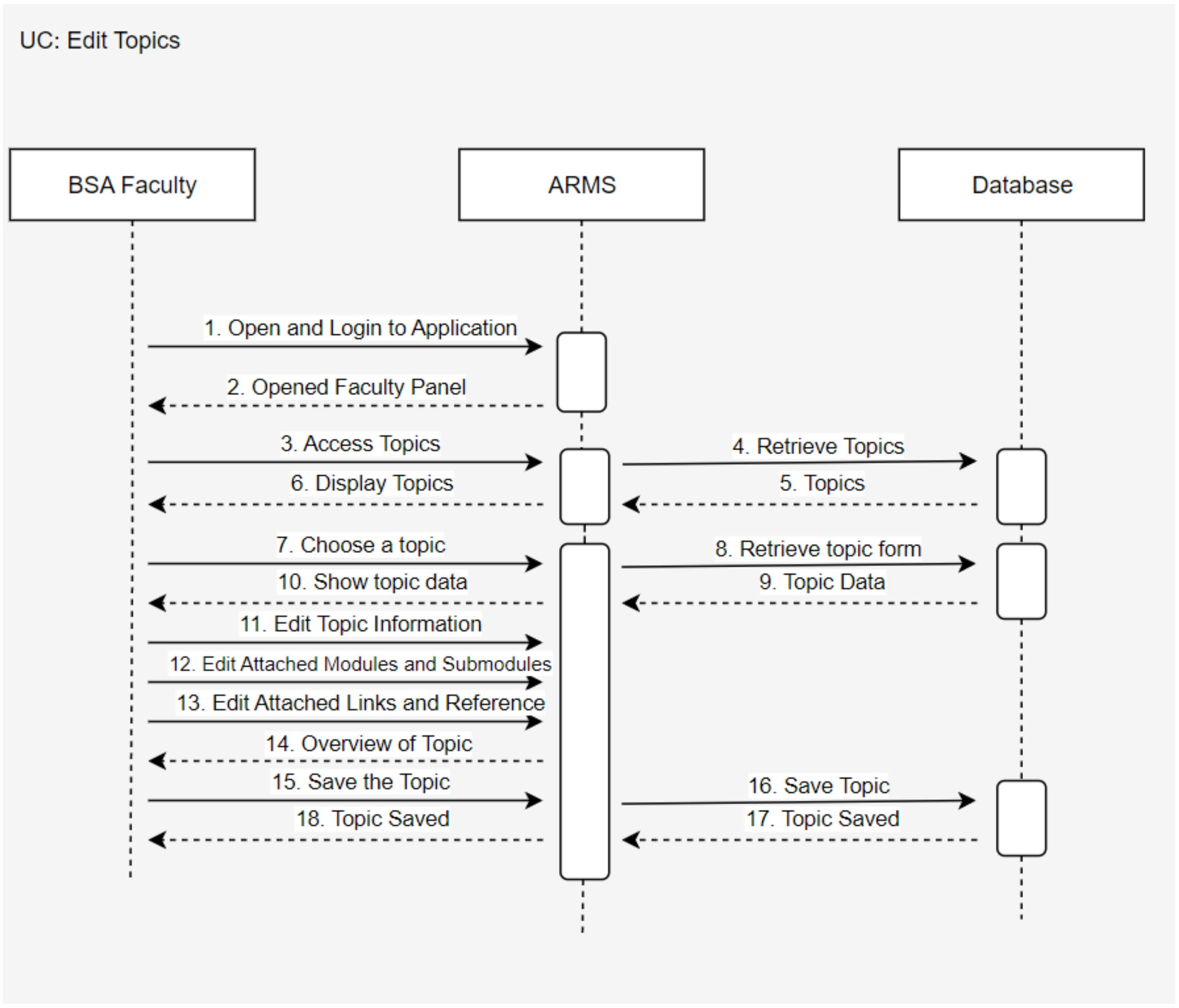
3. The Dean selects a faculty member from the list and initiates the editing process.	Editing Form
4. Save the Information	Saving Page
Preconditions: - The Dean must have valid login credentials and appropriate permissions to access the Professor List section of ARMS.	
Postconditions: - The Dean successfully views and potentially edits the information of faculty members registered in ARMS.	
Assumptions: - Faculty information stored in ARMS is accurate and up-to-date. - ARMS has the capability to display faculty information in an organized and accessible manner.	
Requirement Met: - The Dean can access, view, and edit the information of faculty members available in ARMS.	
Outstanding Issues: - Confirmation required on whether there are any restrictions or limitations on the types of edits the Dean can make to faculty information. - Consideration needed for any privacy or security measures implemented to safeguard faculty information during the editing process.	
Priority: Medium to High	
Risk: High	

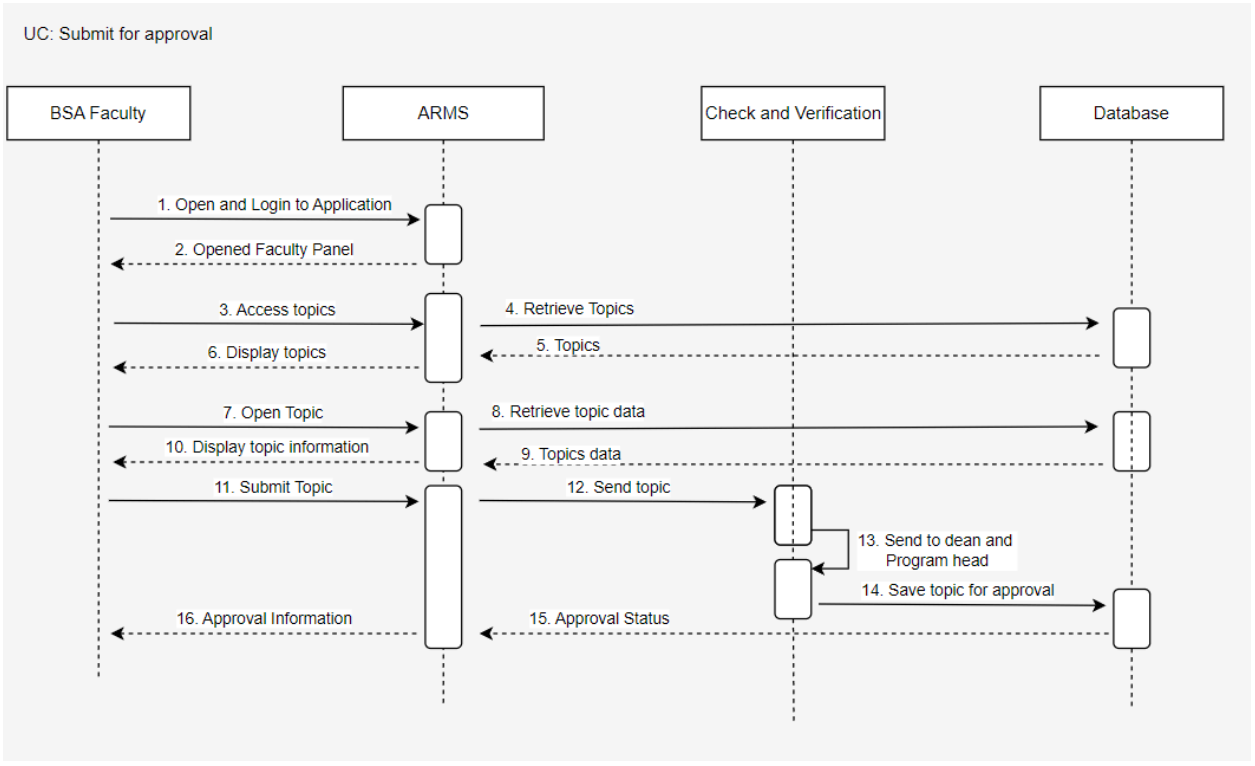
Use case name: Edit Class	**UniqueID:** DP_006
Area: Edit the Class created	
Actor(s): Dean and Program Head	
Description: The Dean and Program Head should be able to edit the class information, students and topic assigned	
Triggering Event: Dean uses ARMS and login using their account, Goes to Class List and edit the class information	
Trigger Type: ☑ ~~External~~ ☐ Temporal	
Steps Performed (Main Path)	**Information for Steps**
1. Select the Class to edit from the List	Class List
2. Modifying information such as class name	Class Information
3. Add or remove students from the class roster.	Students Modification Form
4. Modify the topics assigned to the class. They may add new topics, remove existing ones, or adjust the sequencing as needed.	Topics Modification Form

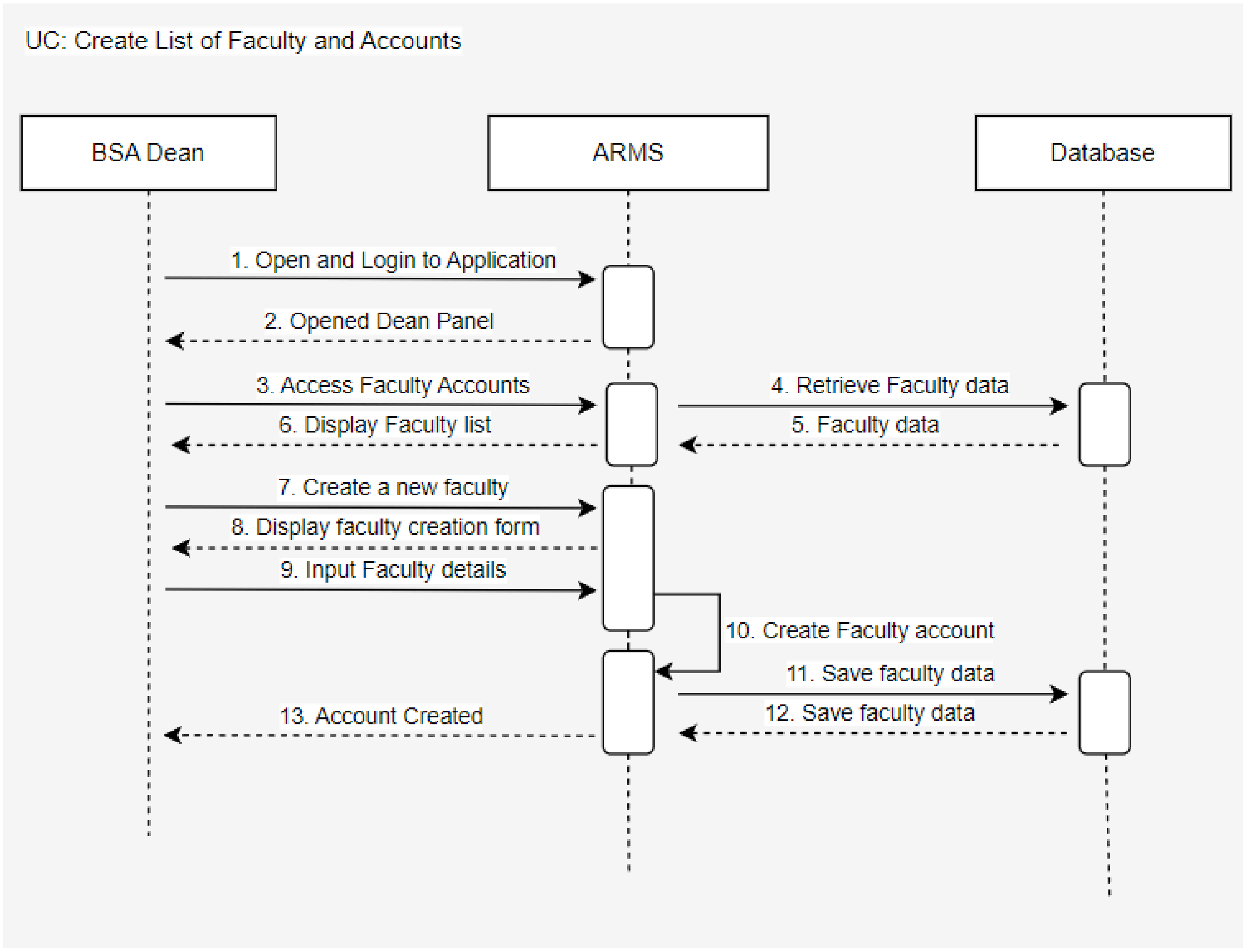
5. Save the modifications to update the class information within ARMS.	Saving Page
Preconditions: - The Dean or Program Head must have valid login credentials and appropriate permissions to access the Class List section of ARMS.	
Postconditions: - The class information, including student enrollment and assigned topics, is successfully updated within ARMS.	
Assumptions: - Class information stored in ARMS is accurate and up-to-date.	
Requirement Met: - The Dean and Program Head can access and modify class information, student enrollment, and assigned topics within ARMS.	
Outstanding Issues: - Consideration needed for any privacy or security measures implemented to safeguard class information during the editing process.	
Priority: Medium to High	
Risk: High	

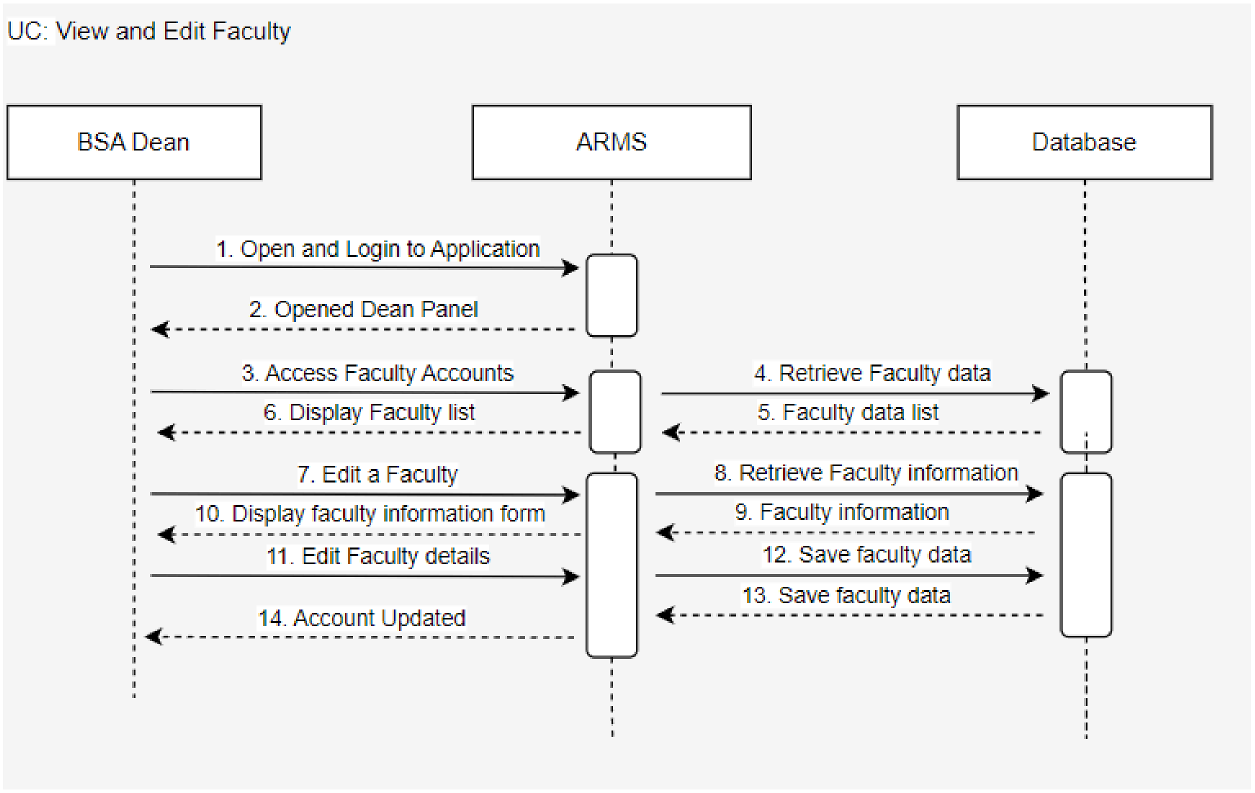

UC: Create Topics
BSA Faculty
ARMS
Database
1. Open and Login to Application
2. Opened Faculty Panel
3. Access Topics
4. Retrieve Topics
5. Topics
6. Display Topics
7. Create a topic
8. Show topic Form
9. Input Topic Information
10. Attach Modules and Submodules
11. Attach Links and Reference
12. Overview of Topics
13. Save the Topic
14. Save Topic
15. Topic Saved
15. Topic Saved

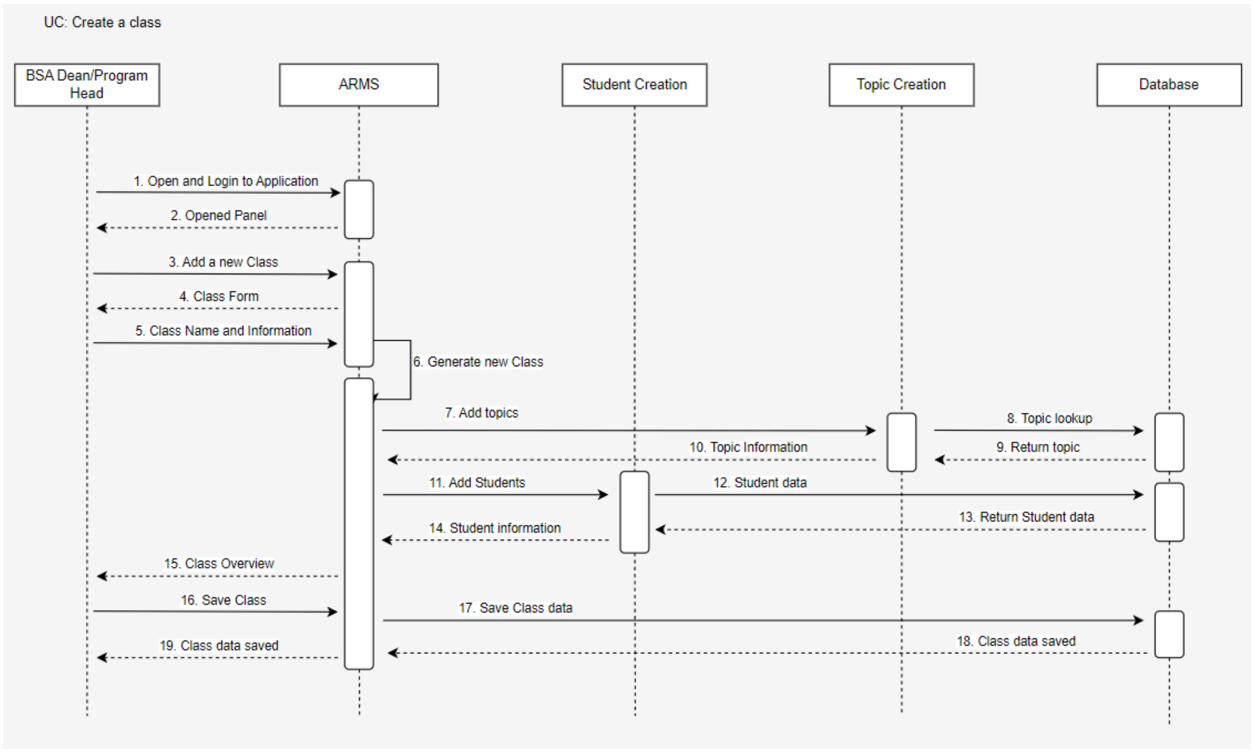

UC: View Topics
BSA Faculty
ARMS
Database
1. Open and Login to Application
2. Opened Faculty Panel
3. Access Topics
4. Retrieve Topics
5. Topics
6. Display Topics

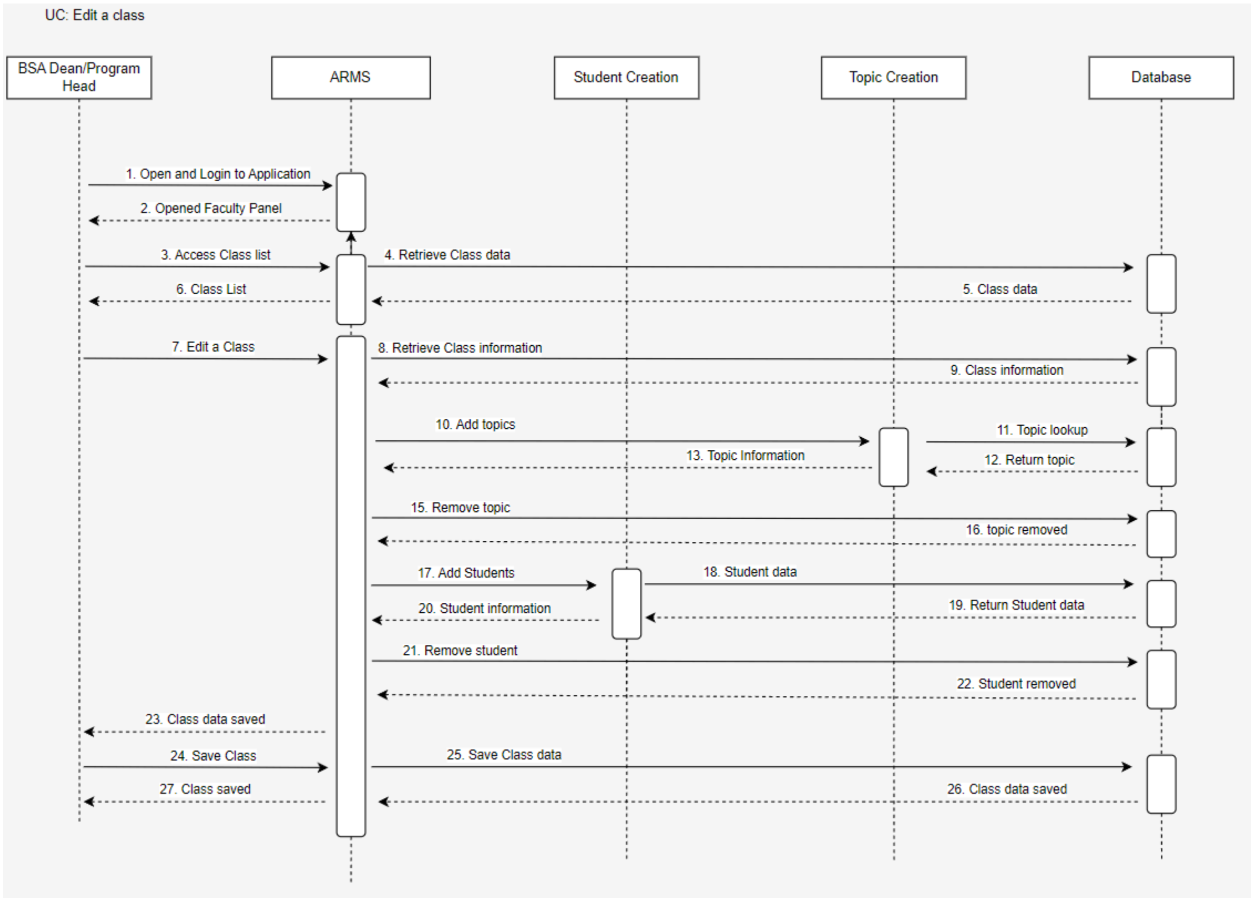

UC: Edit Topics
BSA Faculty
ARMS
Database
1. Open and Login to Application
2. Opened Faculty Panel
3. Access Topics
4. Retrieve Topics
5. Topics
6. Display Topics
7. Choose a topic
8. Retrieve topic form
9. Topic Data
10. Show topic data
11. Edit Topic Information
12. Edit Attached Modules and Submodules
13. Edit Attached Links and Reference
14. Overview of Topic
15. Save the Topic
16. Save Topic
17. Topic Saved
18. Topic Saved

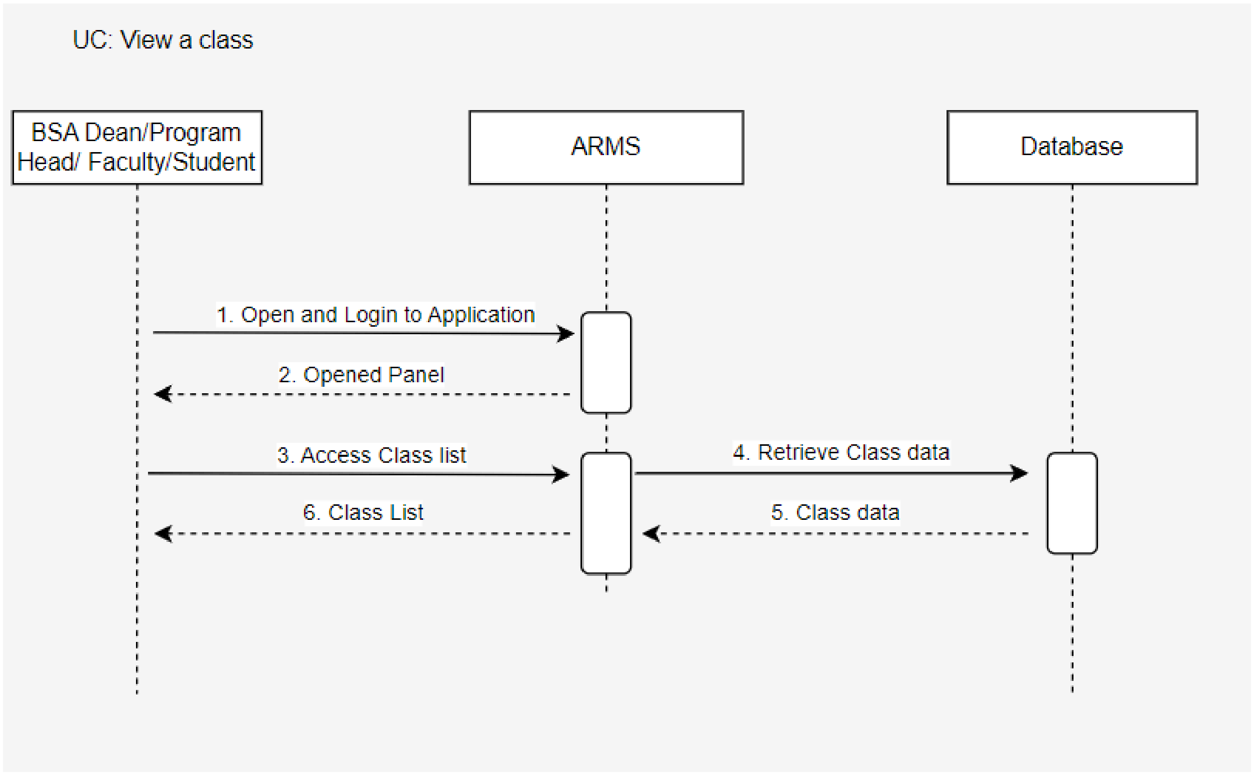

UC: Submit for approval
BSA Faculty
ARMS
Check and Verification
Database
1. Open and Login to Application
2. Opened Faculty Panel
3. Access topics
4. Retrieve Topics
5. Topics
6. Display topics
7. Open Topic
8. Retrieve topic data
9. Topics data
10. Display topic information
11. Submit Topic
12. Send topic
13. Send to dean and Program head
14. Save topic for approval
15. Approval Status
16. Approval Information

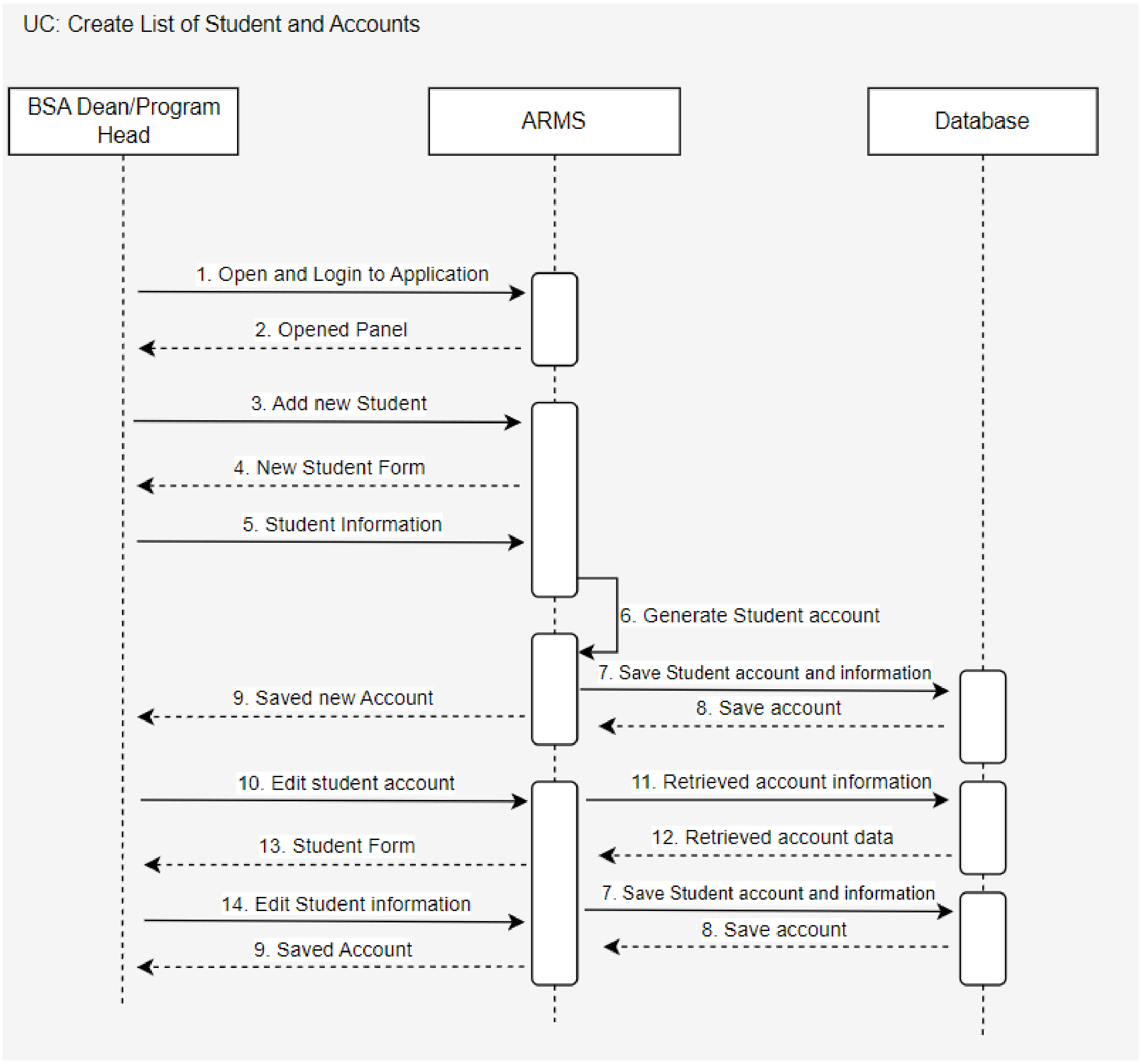

UC: Create List of Faculty and Accounts
BSA Dean
ARMS
Database
1. Open and Login to Application
2. Opened Dean Panel
3. Access Faculty Accounts
4. Retrieve Faculty data
5. Faculty data
6. Display Faculty list
7. Create a new faculty
8. Display faculty creation form
9. Input Faculty details
10. Create Faculty account
11. Save faculty data
12. Save faculty data
13. Account Created

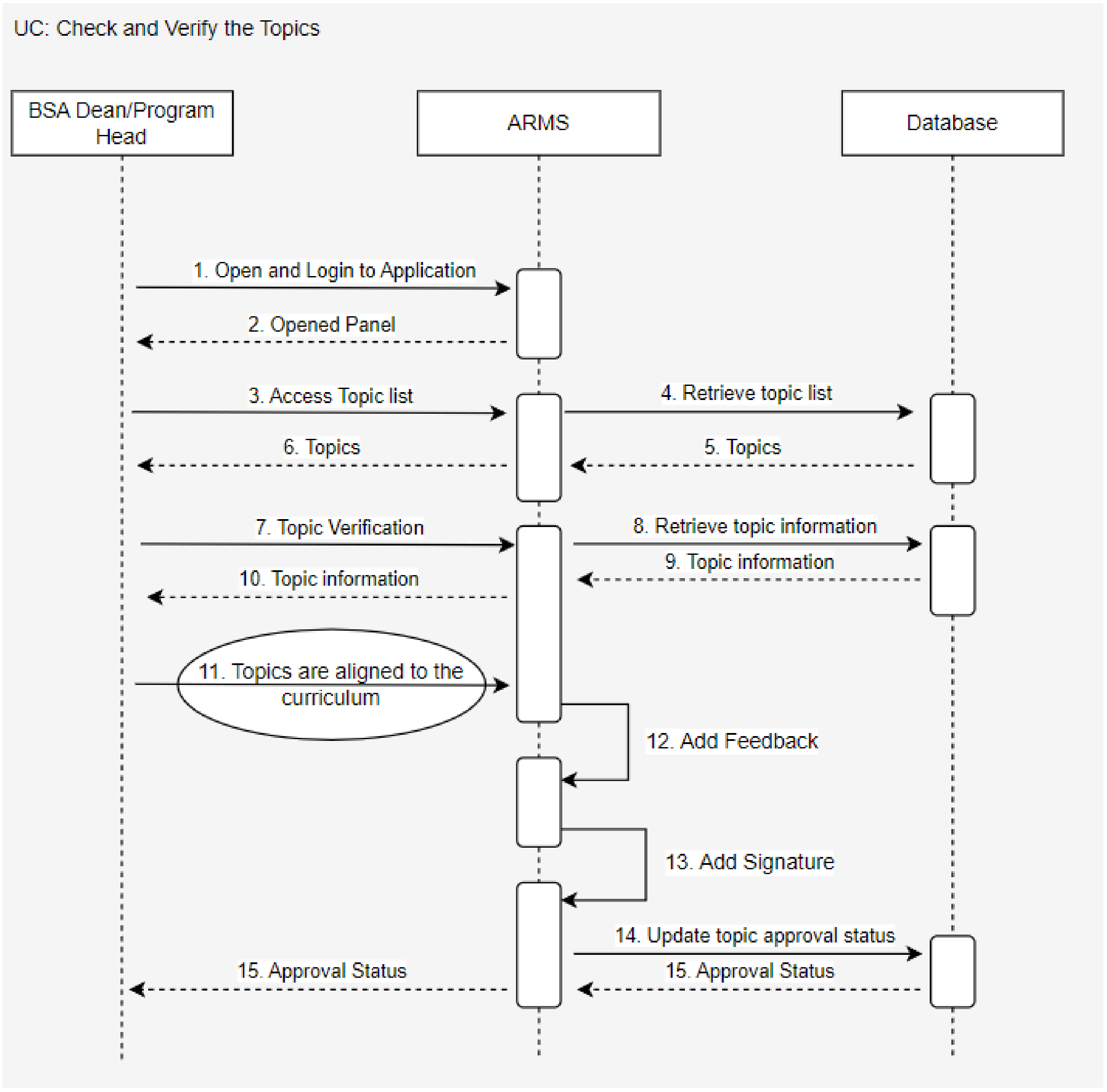

UC: View and Edit Faculty
BSA Dean
ARMS
Database
1. Open and Login to Application
2. Opened Dean Panel
3. Access Faculty Accounts
4. Retrieve Faculty data
5. Faculty data list
6. Display Faculty list
7. Edit a Faculty
8. Retrieve Faculty information
9. Faculty information
10. Display faculty information form
11. Edit Faculty details
12. Save faculty data
13. Save faculty data
14. Account Updated


UC: Create a class
BSA Dean/Program Head
ARMS
Student Creation
Topic Creation
Database
1. Open and Login to Application
2. Opened Panel
3. Add a new Class
4. Class Form
5. Class Name and Information
6. Generate new Class
7. Add topics
8. Topic lookup
9. Return topic
10. Topic Information
11. Add Students
12. Student data
13. Return Student data
14. Student information
15. Class Overview
16. Save Class
17. Save Class data
18. Class data saved
19. Class data saved

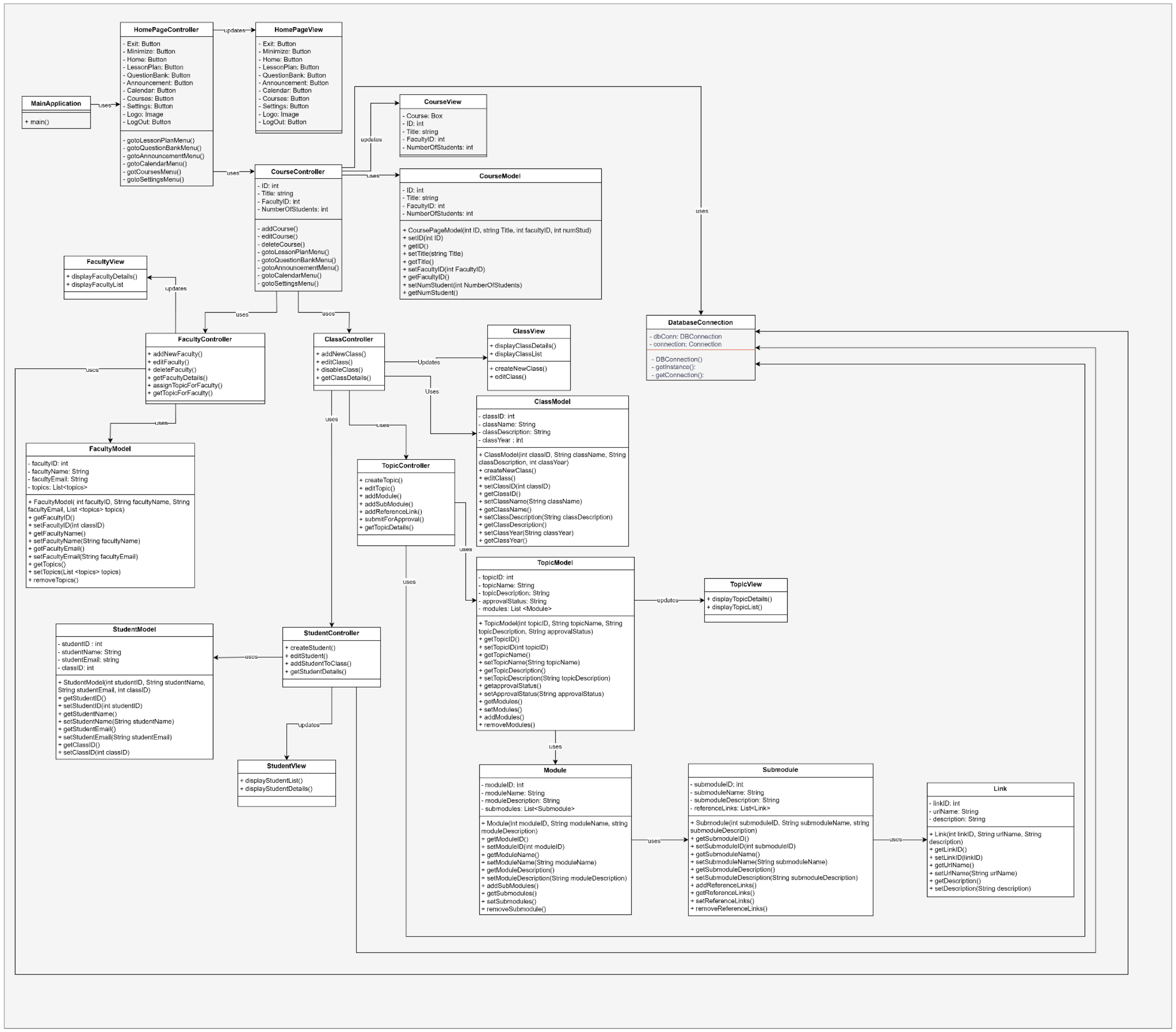

UC: Edit a class
BSA Dean/Program Head
ARMS
Student Creation
Topic Creation
Database
1. Open and Login to Application
2. Opened Faculty Panel
3. Access Class list
4. Retrieve Class data
5. Class data
6. Class List
7. Edit a Class
8. Retrieve Class information
9. Class information
10. Add topics
11. Topic lookup
12. Return topic
13. Topic Information
15. Remove topic
16. topic removed
17. Add Students
18. Student data
19. Return Student data
20. Student information
21. Remove student
22. Student removed
23. Class data saved
24. Save Class
25. Save Class data
26. Class data saved
27. Class saved


UC: View a class
BSA Dean/Program Head/ Faculty/Student
ARMS
Database
1. Open and Login to Application
2. Opened Panel
3. Access Class list
4. Retrieve Class data
5. Class data
6. Class List

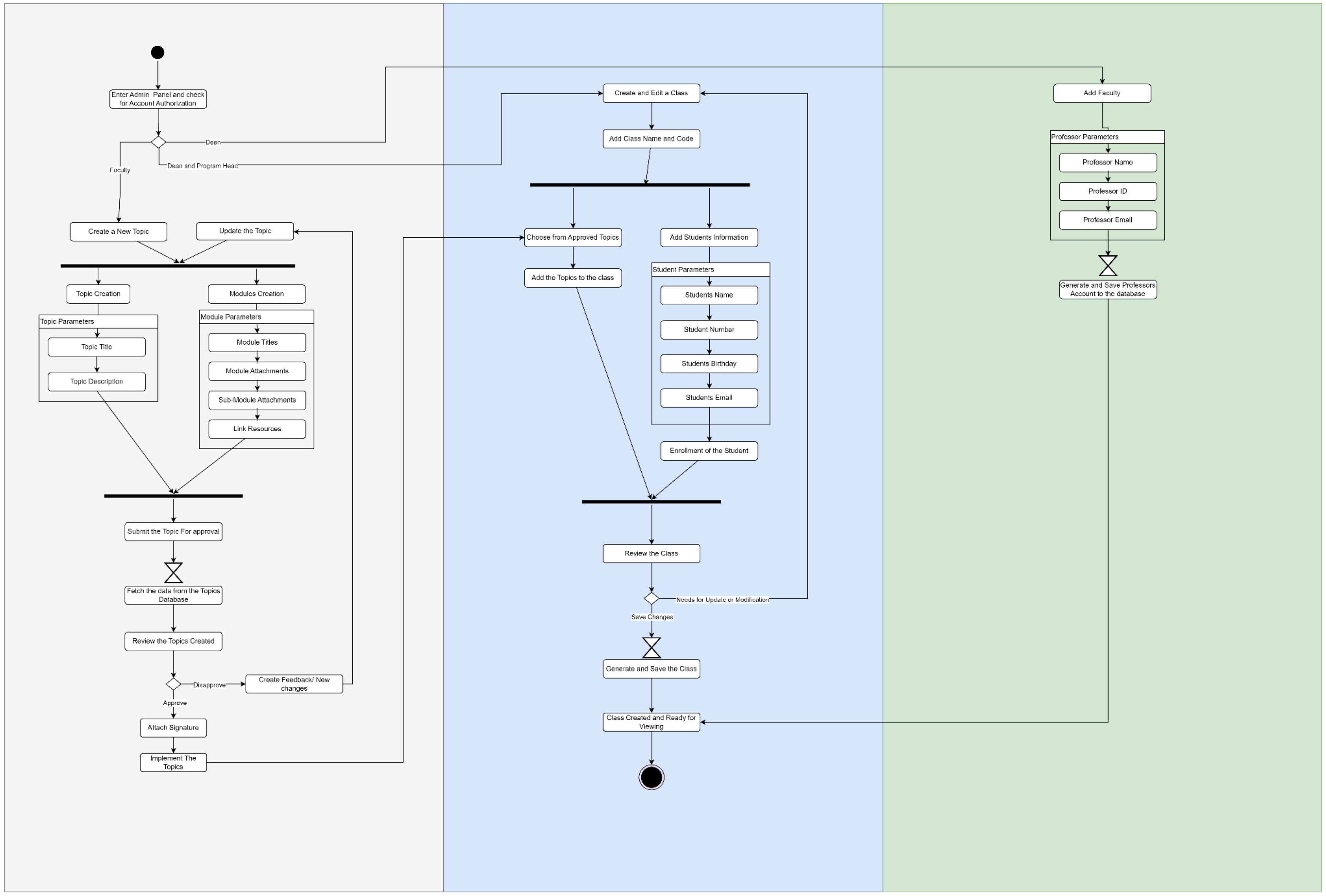

UC: Create List of Student and Accounts
BSA Dean/Program Head
ARMS
Database
1. Open and Login to Application
2. Opened Panel
3. Add new Student
4. New Student Form
5. Student Information
6. Generate Student account
7. Save Student account and information
8. Save account
9. Saved new Account
10. Edit student account
11. Retrieved account information
12. Retrieved account data
13. Student Form
7. Save Student account and information
14. Edit Student information
8. Save account
9. Saved Account


UC: Check and Verify the Topics
BSA Dean/Program Head
ARMS
Database
1. Open and Login to Application
2. Opened Panel
3. Access Topic list
4. Retrieve topic list
5. Topics
6. Topics
7. Topic Verification
8. Retrieve topic information
9. Topic information
10. Topic information
11. Topics are aligned to the curriculum
12. Add Feedback
13. Add Signature
14. Update topic approval status
15. Approval Status
15. Approval Status

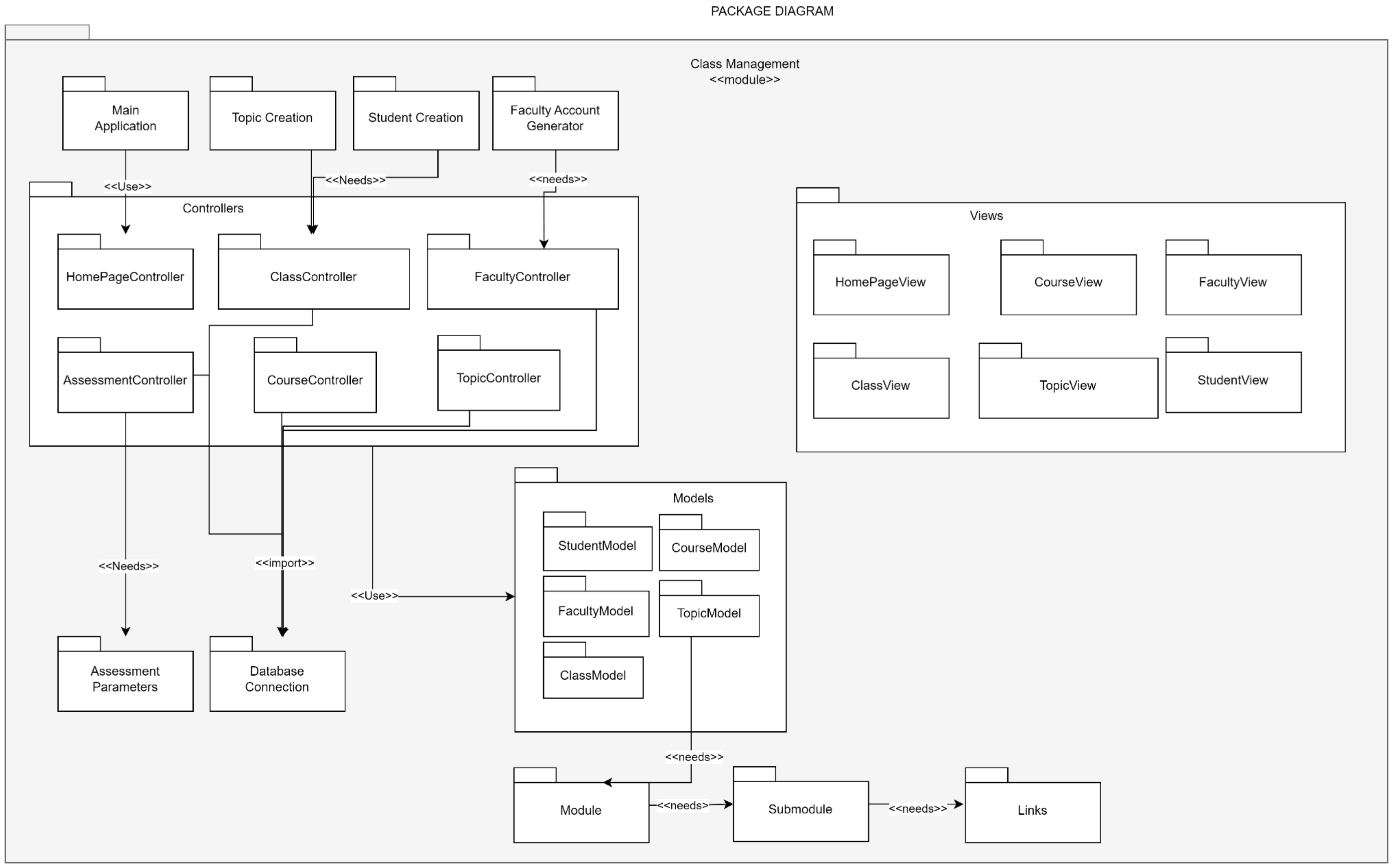
CLASS DIAGRA



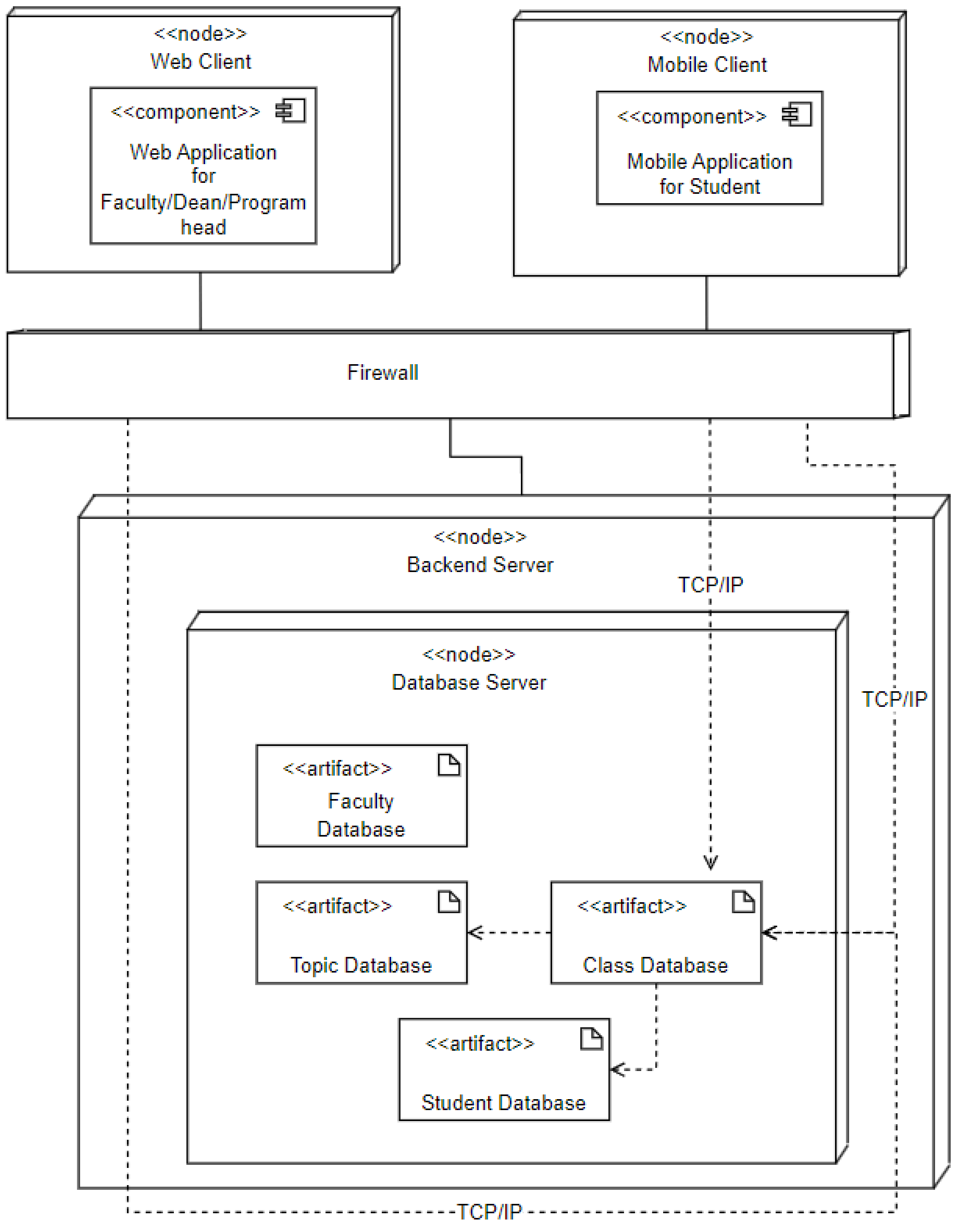
ACTIVITY DIAGRAM



PACKAGE DIAGRAM



DEPLOYMENT DIAGRAM



DEVELOPMENT PLAN

SOFTWARE DEVELOPMENT METHODOLOGY

Scrum is an excellent software development methodology for the development of an accountancy review management system, particularly given the small team size and the modular structure of our project. The group can be divided with different tasks and with scrum we can be able to do daily meetings to check on each progress.

Scrum operates on iterative development cycles called sprints, which usually last between one to four weeks. This approach allows the team to focus on delivering small, incremental updates to the system, ensuring that each module—class management, assessment creation, and question bank creation—is developed progressively. This iterative nature supports continuous improvement as feedback is incorporated into subsequent sprints, enhancing the overall quality and functionality of the system.

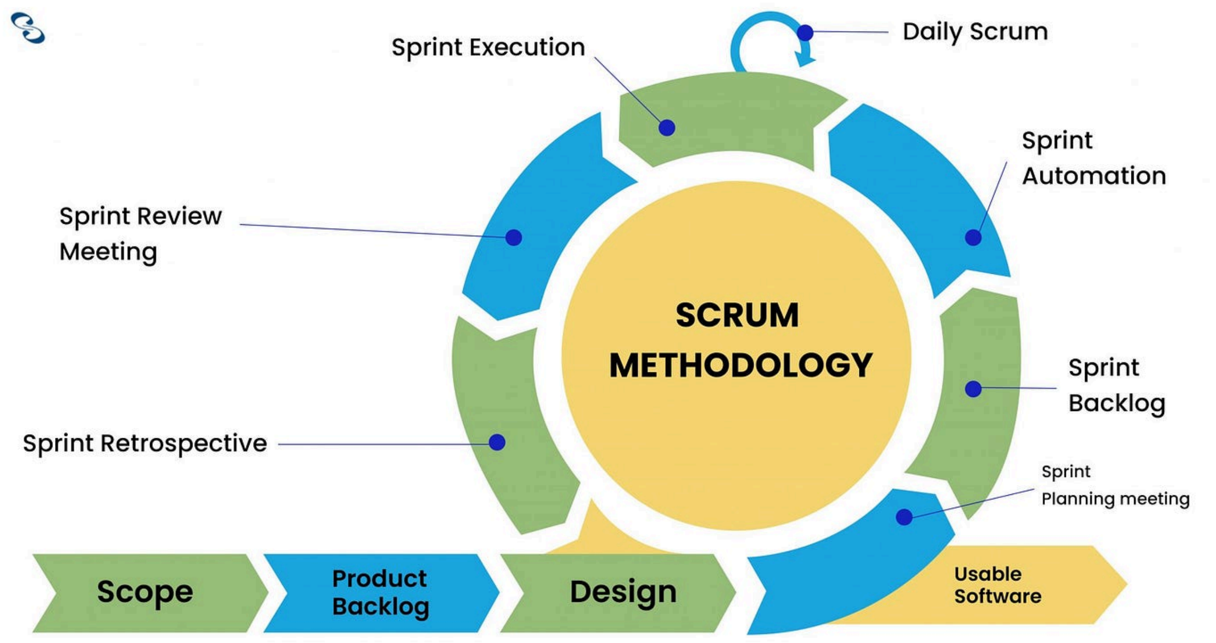
Scrum’s flexibility makes it ideal for the project where requirements might evolve. As our team gathers feedback from potential users or stakeholders, we can easily adjust the project scope, priorities, and tasks. This adaptability is crucial in a project like an accountancy review management system, where user needs and regulatory requirements may change.

Emphasizes team collaboration through regular meetings such as daily stand-ups, sprint planning, sprint reviews, and retrospectives. With a small team of three members, these meetings ensure that everyone is aligned, aware of each other’s progress, and can quickly address any issues that arise. This consistent communication helps prevent misunderstandings and keeps the project on track.

In Scrum, roles are clearly defined as Product Owner, Scrum Master, and Development Team. This clarity helps streamline decision-making and task management. In our project, one member can act as the Product Owner, focusing on defining and prioritizing the project requirements, while another can take on the Scrum Master role, ensuring the Scrum process is followed and removing any impediments. All members are required to form the Development Team, focusing on coding and development tasks. These clearly defined roles help ensure that responsibilities are appropriately allocated and managed.

Scrum prioritizes tasks based on their value to the end user, ensuring that the most important features are developed first. For your accountancy review management system, this means critical functionalities like class management and assessment creation can be prioritized, ensuring that essential features are delivered early and continuously improved based on user feedback.

The use of Scrum boards (e.g., Kanban boards) and regular updates on progress provide high transparency in the development process. Team members can see what tasks are in progress, what is completed, and what needs attention. This transparency fosters accountability and ensures that the team remains focused on their goals.



source:
https://blog.stackademic.com/excelling-in-software-development-with-scrum-methodology-part-2-e2d0b29437ce

DEVELOPMENT TOOLS

During the development phase of the Class Management module, various aspects need to be addressed to ensure a robust and effective system. This includes implementing proper authentication mechanisms to maintain data integrity and designing a responsive, user-friendly interface to provide a secure and interactive user experience.

Web Development

PHP: We will be using PHP for both the frontend and backend of the application. PHP is a widely-used, open-source scripting language that is especially suited for web development and can be embedded into HTML. Highly effective for server-side scripting, which is essential for backend development. It allows for the creation of dynamic web pages, handling form data, managing sessions, and interacting with databases. The framework that will be used during this project is **Laravel** because it offers an elegant syntax that is simple, readable, and developer-friendly. This means your team can write code that is clean and easy to understand, which speeds up development and makes maintenance easier. Laravel's expressive syntax also helps in avoiding common pitfalls and bugs, leading to more reliable code.

CSS: We will be using **Tailwind CSS** for web development because it offers a highly customizable, utility-first approach to styling, enabling rapid development and a consistent design system. Tailwind allows developers to apply styles directly in the HTML, reducing the need for writing custom CSS and thus speeding up the styling process. Its modular design promotes reusability and maintainability, making it easier to manage complex projects.

Javascript: We will be using **vanilla JavaScript** for web development because it provides the fundamental building blocks needed to create dynamic and interactive web applications without the overhead of additional libraries or frameworks. By using plain JavaScript, we ensure that our code is lightweight, fast, and compatible across all browsers. This approach allows for complete control over the functionality and performance of our application, making it easier to optimize and debug.

Mobile Development

Flutter/Dart: We will be using Flutter, a powerful open-source UI toolkit developed by Google, for mobile development due to its ability to create natively compiled applications for both iOS and Android from a single codebase. Flutter’s rich set of pre-designed widgets and its highly customizable framework enable the creation of visually attractive, responsive, and high-performance applications. Additionally, Flutter's hot-reload feature significantly accelerates the development process by allowing real-time updates and quick iterations, improving productivity and reducing development time. By leveraging Flutter and Dart, we can ensure a consistent and seamless user experience across platforms while maintaining efficient and cost-effective development practices.

Pub: We will use **pub.dev** for mobile development because it provides a centralized repository of Dart and Flutter packages, facilitating easy discovery, management, and integration of high-quality libraries and tools. This streamlines development, enhances productivity, and ensures access to a wide range of functionalities and updates essential for building robust mobile applications.

Collaboration Tools

Github: We will use GitHub as our collaboration tool because it offers robust version control, seamless code collaboration, and powerful project management features. GitHub's pull requests, code reviews, and issue tracking enhance teamwork and code quality, while its extensive integration options streamline our development workflow, ensuring efficient and organized project progress.

Trello: We'll use Trello for collaboration because it offers a simple, visual approach to organizing tasks and workflows. Its intuitive interface and flexible structure make it easy for teams to track progress, assign tasks, and prioritize work effectively, promoting transparency and teamwork.

Testing Tools

Travis Ci: We'll use Travis CI for development because it automates the testing and deployment process, ensuring code quality and reliability. Its integration with GitHub allows for seamless continuous integration and continuous deployment (CI/CD), helping to catch bugs early and deliver updates quickly and efficiently.

ESLint: We'll use ESLint for development to enforce coding standards, identify potential errors, and maintain code quality. Its customizable rules and integration with various editors and build tools help ensure consistent and error-free code, enhancing readability and maintainability.

HARDWARE REQUIREMENTS

Web Application Requirements

- Processor: Intel Core i5 or equivalent
- RAM: 8GB or higher
- Storage: 256GB SSD or higher
- Display: Full HD (1920x1080) resolution or higher
- Network: Ethernet or Wi-Fi connectivity for internet access
- Browser: Latest versions of Google Chrome, Mozilla Firefox, or Safari for optimal performance

Mobile Application Requirements

Android:

- Processor: Qualcomm Snapdragon 660 or equivalent
- RAM: 4GB or higher
- Storage: 64GB internal storage or higher
- Display: Full HD (1920x1080) resolution or higher
- Battery: 3000mAh or higher for all-day usage
- Connectivity: 4G LTE, Wi-Fi, Bluetooth

IOS:

- Processor: Apple A12 Bionic chip or equivalent
- RAM: 3GB or higher
- Storage: 64GB internal storage or higher
- Display: Retina HD display or higher
- Battery: 2500mAh or higher for all-day usage
- Connectivity: 4G LTE, Wi-Fi, Bluetooth

These hardware requirements ensure smooth performance and optimal user experience for both the web and mobile applications. Adjustments may be needed based on specific features and performance demands of the application. Higher specifications may improve the operating performance of the application.

SOFTWARE TESTING PLAN

TEST SCENARIOS

Class Creation

Scenario 1: The Dean or Program head creates a class.

Expected Output: The Dean or Program head successfully creates the class in the system. All entered details, including the class name, code, description, schedule, and student roster (if added), are accurately saved in the system without any errors.

Scenario 2: The Dean or Program head attempts to create a class without entering mandatory details.

Expected Outcome:

- The system should display an error message.
- The Dean or Program head should be prompted to enter a unique class code or modify the existing one to proceed with class creation.

Scenario 3: The Dean or Program head attempts to create a class with a duplicate class code.

Expected Outcome:

- The system should display an error message indicating that the class code is already in use.
- The Dean or Program head should be prompted to enter a unique class code or modify the existing one to proceed with class creation.

Faculty Registration

Scenario 1: Program head registers a new faculty member.

Expected Outcome:

- The program head successfully registers the new faculty member in the system.
- All entered details, including the faculty member's name, email, department, and contact information, are accurately saved in the system without any errors.

Scenario 2: Program head attempts to register a new faculty member with incomplete information.
Expected Outcome:

- The system should display appropriate error messages indicating the missing or incomplete information.
- The program head should not be able to proceed with the registration until all required details are provided.

Scenario 3: Program head attempts to register a faculty member with an existing email address and ID-number.
Expected Outcome:

- The system should display an error message indicating that the email address or ID-number is already registered.
- The program head should be prompted to enter a unique email address or ID-number for the new faculty member or verify if the existing faculty member needs to be updated instead.

Learning Development Plan Creation
Scenario 1: Faculty creates a learning development plan with modules, topics, and materials.
Expected Outcome:

- The faculty member successfully creates the learning development plan with modules, topics, and materials attached.
- All entered details, including the plan title, description, module titles, topic titles, and attached materials, are accurately saved in the system without any errors.

Scenario 2: Faculty attempts to create a learning development plan without entering mandatory details.
Expected Outcome:

- The system should display appropriate error messages indicating the missing or incomplete information.
- The faculty member should not be able to proceed with plan creation until all required details are provided.

Approval of Learning Plan

Scenario 1: Dean approves a submitted learning development plan.

Expected Outcome:

- The Dean successfully approves the submitted learning development plan.
- The approved plan is marked as such in the system, indicating that it has been reviewed and approved by the Dean.

Scenario 2: Dean rejects a submitted learning development plan.

Expected Outcome:

- The Dean successfully rejects the submitted learning development plan.
- The rejected plan is marked as such in the system, indicating that it has been reviewed and rejected by the Dean.
- The faculty member who submitted the plan is notified of the rejection and provided with the reason for rejection for further action.

Scenario 3: Dean reviews and approves multiple submitted learning development plans.

Expected Outcome:

- The Dean successfully reviews and approves multiple submitted learning development plans.
- Each approved plan is marked as such in the system, indicating that it has been reviewed and approved by the Dean.
- The faculty members who submitted the approved plans are notified of the approval for further action.

Student Viewing of the Learning Plan

Scenario 1: Student views their assigned learning development plan.

Expected Outcome:

- The student successfully views their assigned learning development plan.
- All details of the plan, including modules, topics, and attached materials, are displayed accurately and clearly.
- The student can access all relevant information needed to understand the course curriculum and learning objectives.

Scenario 2: Student navigates through the modules and topics of the learning development plan.
Expected Outcome:

- The student successfully navigates through the modules and topics of the assigned learning development plan.
- All modules and topics are displayed in a structured and organized manner, allowing the student to easily access the information.
- The student can view and download any attached materials, such as reading materials, presentations, or assignments, as needed for their learning.

TEST CASES

Class Creation

Test Case 1: Creating a Class along with its required fields

Test Case 2: Enrollment of the students to the class.

Test Case 3: Editing of the Class information

Test Case 4: Integrating the LDP to the class.

Faculty Registration

Test Case 5: Registration process of the faculty

Test Case 6: Editing of the faculty information

Learning Development Creation

Test Case 7: Creating a new Learning development Plan

Test Case 8: Editing the current learning development Plan

Test Case 9: Deleting the Learning development Plan

Test Case 10: Creating topics and modules for the LDP

Test Case 11: Uploading the materials in the LDP

Test Case 12: Submission of the LDP to the Dean or Program Head.

Approval of Learning Plan

Test Case 13: Accessing the LDP submitted by the faculty

Test Case 14: Approval process of the LDP

Test Case 15: Disapproval of the LDP

Test Case 16: Creating feedback issues of the LDP

Student Viewing of the Learning Plan

Test Case 17: Accessing the learning plan

Test Case 18: Accessing the modules and materials of the learning plan