RAJALAKSHMI ENGINEERING COLLEGE

RAJALAKSHMI NAGAR, THANDALAM – 602 105

RAJALAKSHMI
ENGINEERING COLLEGE
An AUTONOMOUS Institution
Affiliated to ANNA UNIVERSITY, Chennai

CS19442 SOFTWARE ENGINEERING CONCEPTS LAB

Laboratory Record
Note Book

Name :……. ……...

Year / Branch / Section : …….. …….

University Register No. : …….. ……

College Roll No. : …….……

Semester : ……..……

Academic Year : …….

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 Concepts Laboratory during the year 2023- 2024

Signature of Faculty-in-charge

Submitted for the Practical Examination held on ----------

Internal Examiner External Examiner

INDEX

STUDENT'S KIT

BHARATH D 220701042
CAROLINE SUJA J S 220701048
CHITHRA k 220701054
DEVDHARSHAN S R 220701060
DHINISIYA M 220701066
SAI SRAVANTHI J P 220701902

OVERVIEW OF THE PROJECT:

The Student Kit App aims to provide a comprehensive tool for students to manage their academic and personal tasks effectively. This app integrates essential features such as a calendar for scheduling classes, exams, and extracurricular activities; subject-wise folders for organizing notes and study materials; customizable to-do lists for managing and prioritizing tasks; and an expense tracker for managing personal finances. The app ensures a seamless user experience by authenticating users through an efficient authentication system, allowing them to access and utilize these features securely. The calendar component handles scheduling actions, the to-do list helps students stay on top of their workload, the folder system organizes academic materials, and the expense tracker promotes financial responsibility. By consolidating these functionalities into one application, the Student Kit App enhances productivity, improves organization, and supports better financial management for students. This comprehensive approach not only simplifies the management of academic and personal responsibilities but also fosters a more organized and efficient student life.

SOFTWARE REQUIREMENTS SPECIFICATION (SRS)

EXP.NO: 1 **DATE: 20.2.2024**

Table of Contents

STUDENT'S KIT

1. Introduction

1.1 Purpose:

The purpose of the Student Kit App is to provide a comprehensive solution designed to help students organize and manage their academic and personal tasks efficiently. By integrating various essential tools into a single platform, the app aims to streamline students' daily routines, enhance productivity, and ensure they can keep track of their responsibilities and goals effortlessly.

1.2 Scope:

The Student Kit App will include a variety of features tailored to meet the diverse needs of students. These features encompass an integrated calendar for scheduling classes, exams, and extracurricular activities; subject-wise folders for organized storage of notes and study materials; customizable to-do lists to manage tasks and prioritize activities; and an expense tracking tool to help students manage their finances. By offering these functionalities, the app aims to streamline students' daily routines and enhance their overall productivity.

2. Overall Description

2.1Product Perspective:

The Student Kit App will operate as a standalone mobile application, accessible on both Android and iOS platforms. It will integrate various features to enhance the overall student experience.

2.2Features

2.2.1 Calendar:

- Users can view and manage their academic and personal events.
- Ability to set reminders for deadlines, exams, and other important dates.
- Intuitive navigation for daily, weekly, and monthly views.

2.2.2 Subject-Wise Folders:

- Organize study materials, notes, and resources according to subjects.
- Easy file upload/download functionality.

• Search and filter options for efficient retrieval.

2.2.3 To-Do List:

• Create and manage tasks with due dates.

• Prioritization and categorization of tasks.

• Progress tracking and completion notifications.

2.2.4 Expense Tracking:

• Record and categorize expenses related to education and personal life.

• Visual representation of spending patterns through charts.

• Budgeting features to help students manage finances effectively.

2.2.5 User Classes and Characteristics:

• Students: Primary users who will utilize the app for academic and personal organization.

• Administrators: Responsible for managing system settings and user accounts.

3. Specific Requirements

3.1 Functional Requirements:

3.1.1 Calendar:

• Users should be able to add, edit, and delete events.

• Set customizable reminders for each event.

• Different views (daily, weekly, monthly) should be available.

3.1.2 Subject-Wise Folders:

• Users can create folders for each subject.

• Upload, download, and delete files within the folders.

- Implement search and filter options for efficient organization.

3.1.3 To-Do List:

- Create, edit, and delete tasks with due dates.
- Categorize tasks by priority and subject.
- Receive notifications for upcoming deadlines.

3.1.4 Expense Tracking:

- Record expenses with details such as date, category, and amount.
- Generate expense reports and visualizations.
- Set budget limits and receive alerts for exceeding limits.

3.2 Non-Functional Requirements:

3.2.1 Usability

- The interface should be user-friendly and intuitive.
- Cross-platform compatibility for both Android and iOS devices.

3.2.2 Performance

- Quick response time for loading events, files, and tasks.
- Scalability to accommodate a growing user base.

3.2.3 Security

- Implement secure user authentication and authorization.
- Encryption for sensitive data, such as financial records.

4. External Interface Requirements

4.1 User Interfaces:

- Intuitive design with easy navigation.

- Consistent theme and layout for a seamless user experience.

4.2 Hardware Interfaces:

- Compatible with standard Android and iOS devices.

4.3 Software Interfaces:

- Integration with calendar APIs for event synchronization.
- File storage integration for subject-wise folders.

5. Conclusion

The Student Kit App aims to provide a comprehensive solution for students' organizationalneeds, combining calendar functionality, subject-wise folders, to-do lists, and expense tracking. This SRS document serves as a guideline for the development team to create a robust and user-friendly application.

SCRUM METHODOLOGY

EXP.NO: 2 **DATE: 01.3.2024**

1. Introduction

The Student Kit App aims to provide an intuitive and comprehensive solution for students to organize their academic and personal lives efficiently. The Agile Scrum framework will guide the development process, ensuring iterative enhancements and timely delivery of key features.

2. Objectives

- Develop a user-friendly application to manage academic events, study materials, tasks, and expenses.
- Prioritize user needs through iterative development cycles.
- Ensure seamless collaboration between students and administrators.

3. Product Backlog Introduction

The product backlog is a dynamic list of features, enhancements, and fixes prioritized by the product owner. It serves as a roadmap for the development team.

4. Product Backlog

4.1 For Students

- **Calendar Features:**
  - Event management with reminders.
  - Views for daily, weekly, and monthly schedules.

- **Subject-Wise Folders:**
  - File upload/download within folders.
  - Search and filter options for efficient organization.

- **To-Do List:**
  - Task creation, prioritization, and progress tracking.

- **Expense Tracking**:
  - Record creation, budgeting features.

4.2 For Administrators

- **User Management:**
  - Create and manage user accounts.
  - View and moderate shared resources.

- **System Settings:**
  - Customize app settings and preferences.
  - Access analytics for user engagement.

5. User Stories

- **As a Student:**
  - I want to add, edit, and delete events in the calendar to manage my schedule.
  - I want to organize my study materials in subject-wise folders for easy retrieval.
  - I want a to-do list to prioritize and manage my tasks effectively.

- **As an Administrator:**
  - I want to be able to manage user accounts and permissions.
  - I want to customize system settings to meet the specific needs of our educational institution.

6. Sprint

- A time-boxed iteration during which a set of user stories is implemented and tested.

7. Sprint Backlog

The sprint backlog is a list of tasks selected from the product backlog for a specific sprint. In other terms Sprint Backlog could also be defined as the subset of Product backlog which is chosen for a specific sprint. In general a sprint backlog allows the development team to work on the tasks necessary to implement the User Stories withinthe selected sprint.

8. Sprint Review

A meeting held at the end of each sprint to review and demonstrate the completed work.

Sprint 1 Review:

- The sprint review is a meeting that includes the demonstration of implemented

featuresat that particular sprint that is conducted at the end of each sprint.

- In the Student kit app the sprint backlog for the first week is the Use case Diagram andthe sprint backlog for the second use case is Software Requirement Specification document.

9. Software Used

- **Development Platform:** Flutter application, SQL,CALENDAR API'S

10. Conclusion

The Agile Scrum framework for the Student Kit App ensures a systematic approach to development, focusing on user needs and iterative improvements. By breaking down the project into manageable sprints, the framework allows for continuous feedback, resulting in a robust and user-friendly application for students and administrators alike.

USER STORIES

EXP.NO: 3 **DATE: 12.3.2024**

As a Student:

1. Manage Calendar Events
User Story: I want to add, edit, and delete events in the calendar to manage my schedule.
Acceptance Criteria:
- The student can add new events with title, details, date, and time.
- The student can edit event details, date, and time.
- The student can delete events.
- Events are displayed on the calendar.

2. Organize Study Materials
User Story: I want to organize my study materials in subject-wise folders for easy retrieval.
Acceptance Criteria:
- The student can create subject-wise folders.
- The student can upload files to folders.
- The student can move files between folders.
- Folders and files are displayed in a hierarchical view.

3. Maintain a To-Do List
User Story: I want a to-do list to prioritize and manage my tasks effectively.
Acceptance Criteria:
- The student can add tasks with title, details, and due date.
- The student can edit task details and due date.
- The student can mark tasks as complete or delete them.
- Tasks are displayed in a list format.

4. Track Expenses
User Story: I want to record and categorize my expenses to manage my budget.
Acceptance Criteria:
- The student can add expenses with amount, category, and date.
- The student can edit expense details.

- The student can delete expenses.
- Expenses are displayed in a list with categories.

5. Set Reminders for Tasks
User Story: I want to set reminders for my tasks to ensure I don't miss important deadlines.
Acceptance Criteria:
- The student can set reminders for tasks.
- Reminders trigger notifications at the set times.

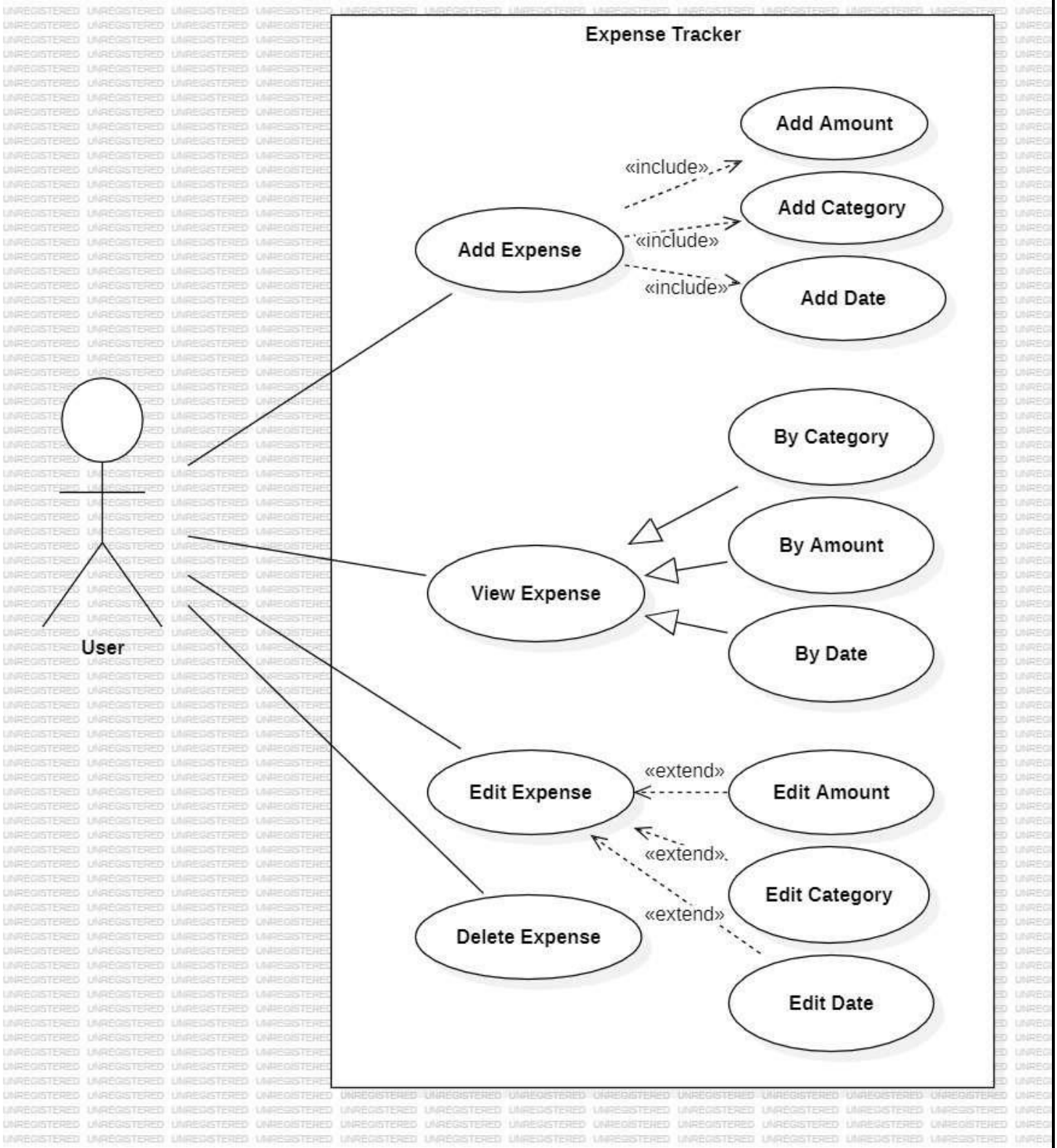
As an Administrator:

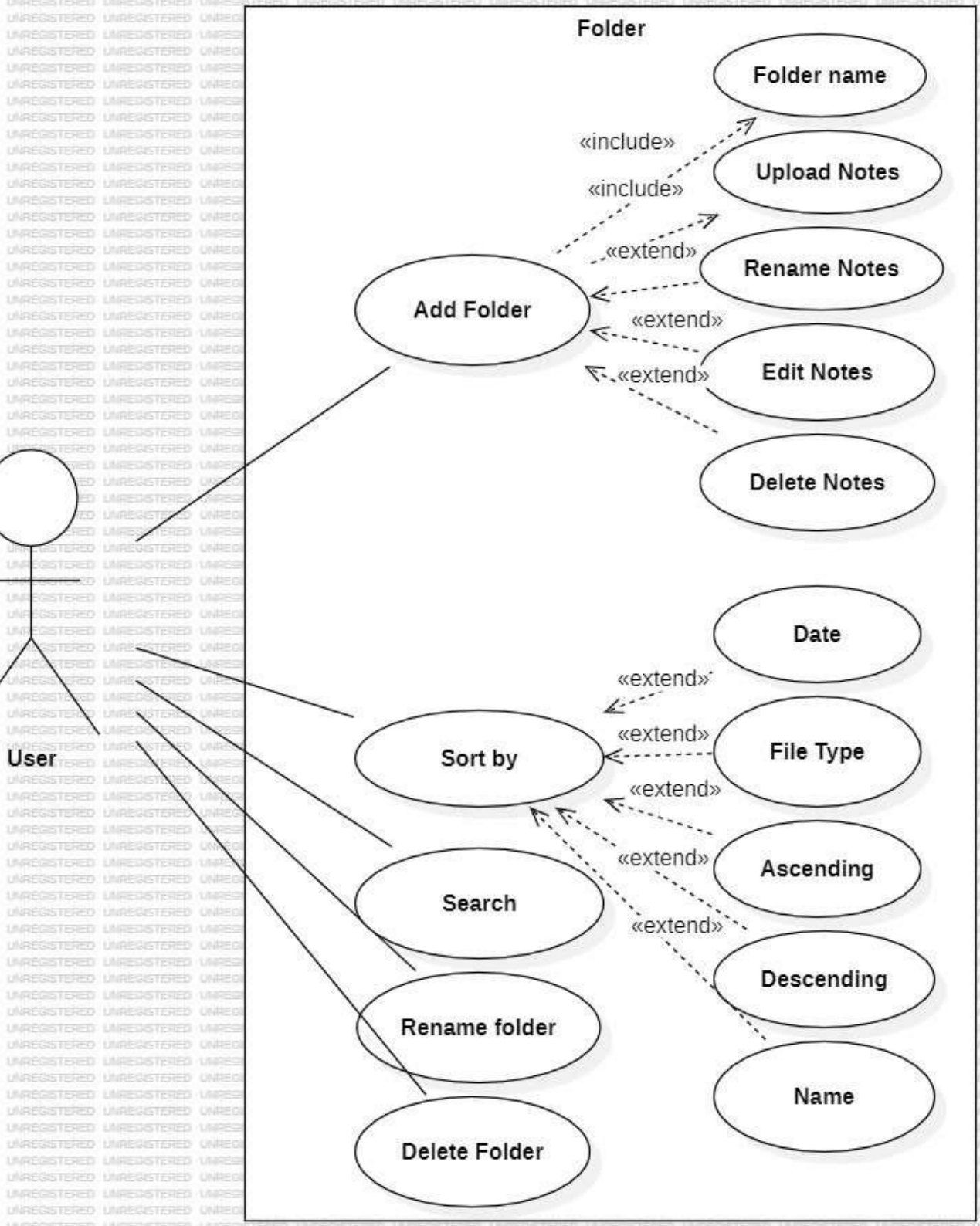
6. Manage User Accounts and Permissions
User Story: I want to be able to manage user accounts and permissions.
Acceptance Criteria:
- The administrator can add, edit, and delete user accounts.
- The administrator can set user permissions.

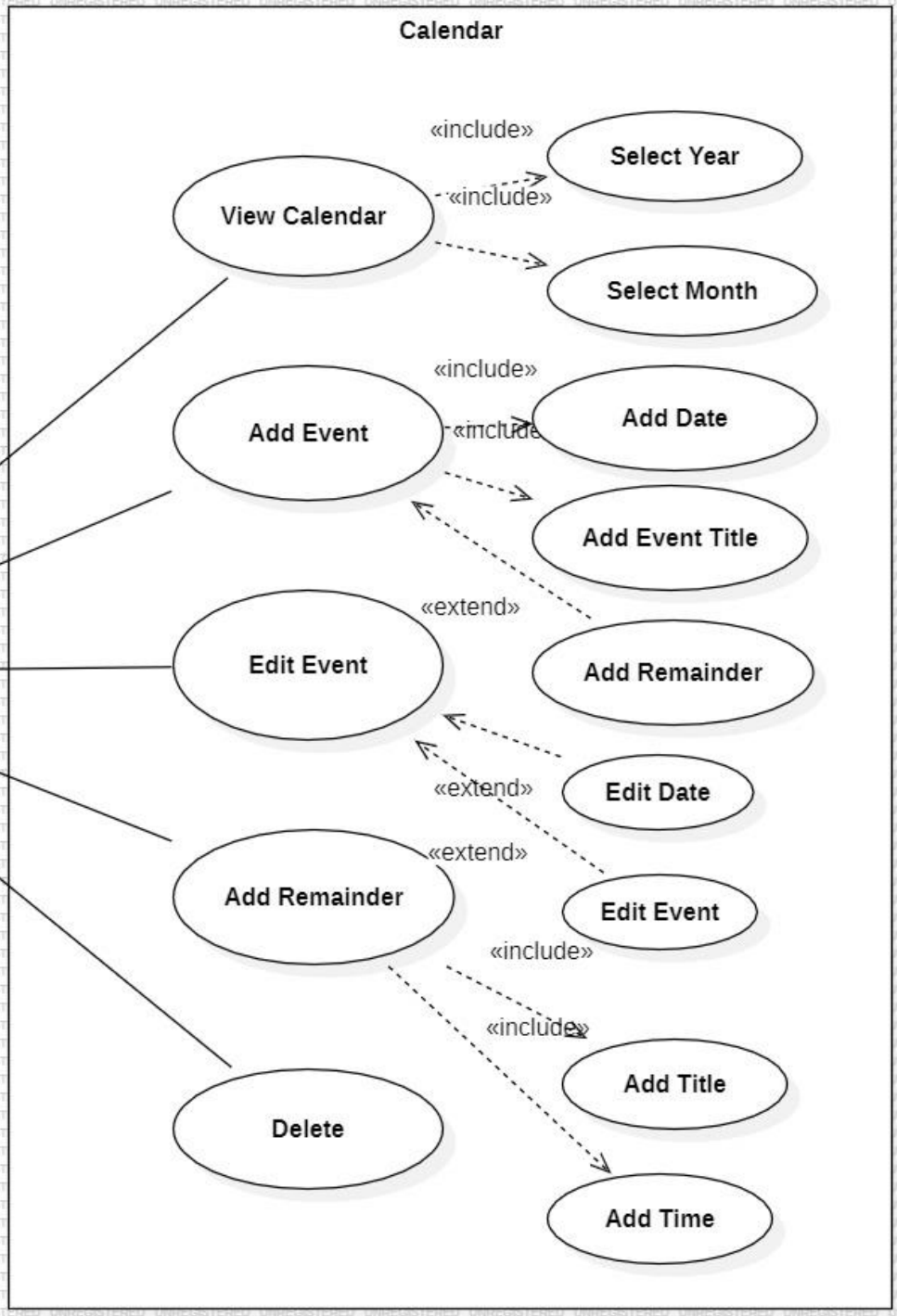
7. Customize System Settings
User Story: I want to customize system settings to meet the specific needs of our educational institution.
Acceptance Criteria:
- The administrator can update system settings.
- Changes to settings are applied and saved.

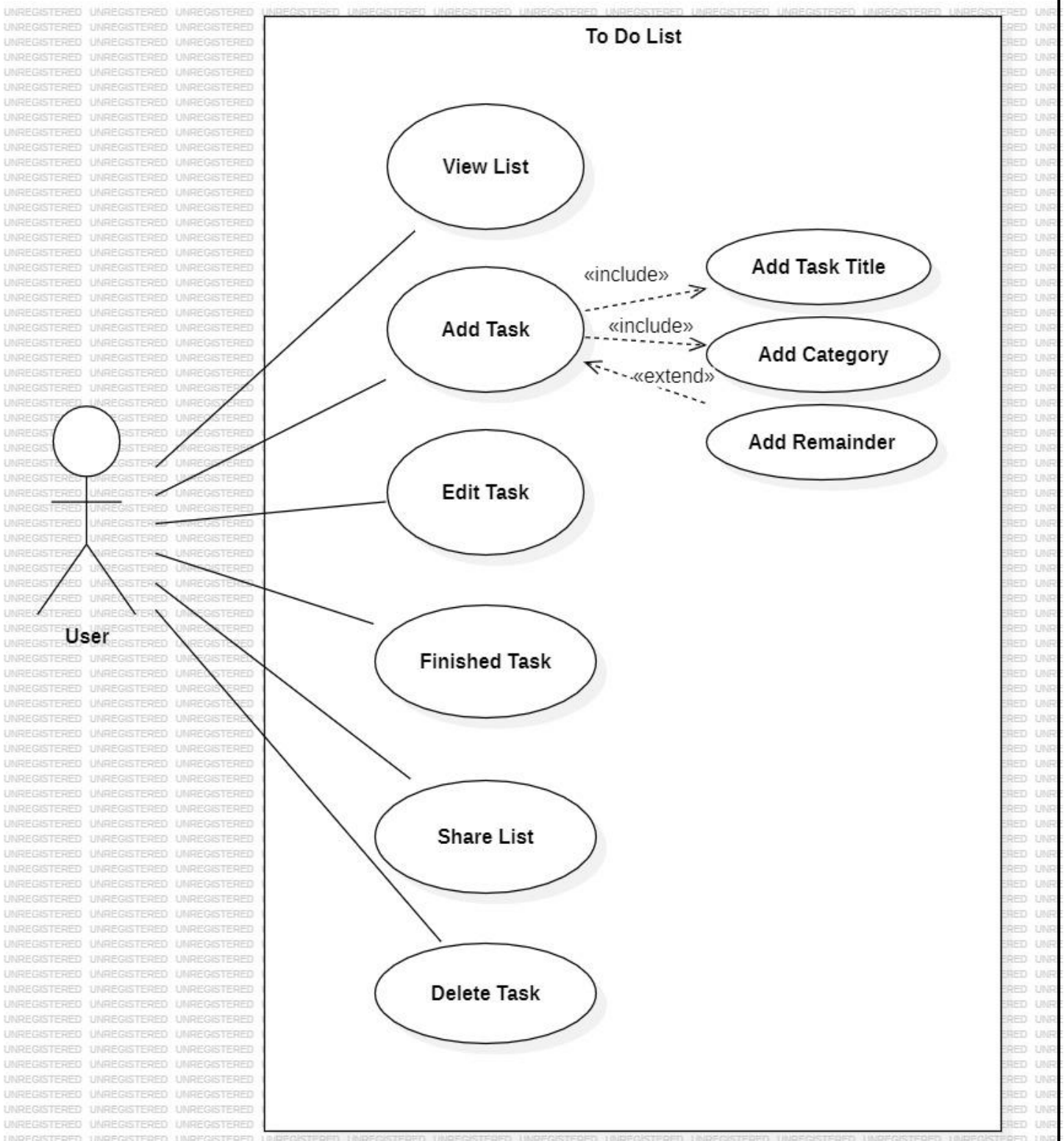
USE CASE DIAGRAM

EXP.NO: 4 **DATE: 19.3.2024**




Folder
Folder name
«include»
Upload Notes
«include»
«extend»
Rename Notes
Add Folder
«extend»
Edit Notes
«extend»
Delete Notes
Date
«extend»
«extend»
File Type
Sort by
«extend»
«extend»
Ascending
Search
«extend»
Descending
Rename folder
Name
Delete Folder
User


Calendar
«include»
Select Year
View Calendar
«include»
Select Month
«include»
Add Event
«include»
Add Date
Add Event Title
«extend»
Edit Event
Add Remainder
Edit Date
«extend»
«extend»
Add Remainder
Edit Event
«include»
«include»
Add Title
Delete
Add Time
User


To Do List
View List
Add Task
Edit Task
Finished Task
Share List
Delete Task
Add Task Title
Add Category
Add Remainder
«include»
«include»
«extend»
User

NON-FUNCTIONAL REQUIREMENTS

EXP.NO: 5 **DATE: 29.3.2024**

1. Performance

The Student Kit App should be highly responsive, with all major functions such as loading the calendar, to-do list, and expenses screen taking no more than 2 seconds to load. The app should handle up to 500 concurrent users without performance degradation, ensuring smooth and efficient user experience during peak times such as the beginning of a semester.

2. Security

The app must implement robust security measures to protect user data. All user information, including login credentials, personal details, and stored notes, must be encrypted both in transit and at rest. Access to the app should be controlled via secure authentication mechanisms, and administrators should have the ability to set and enforce password policies. The app should also include role-based access controls to ensure that users only have access to the features and data appropriate to their role.

3. Usability

The Student Kit App should provide an intuitive and user-friendly interface that is easy to navigate for all users, including those with limited technical skills. It should adhere to common usability principles, providing clear labels, logical flow, and accessible help documentation. The app should support both desktop and mobile platforms, ensuring a consistent and pleasant user experience across devices.

4. Reliability

The app must be reliable, with an uptime of at least 99.9%. It should include mechanisms for automatic failover and recovery to minimize downtime and ensure continuous availability. Regular backups should be performed to prevent data loss, and in the event of a failure, the system should recover without data corruption or significant downtime.

5. Scalability

The Student Kit App should be scalable to accommodate future growth in the number of users and the volume of data. The system architecture should support the addition of new features and integration with other educational tools without requiring significant redesign. The app should be able to handle an increasing number of simultaneous users and data entries with minimal impact on performance.

6. Maintainability

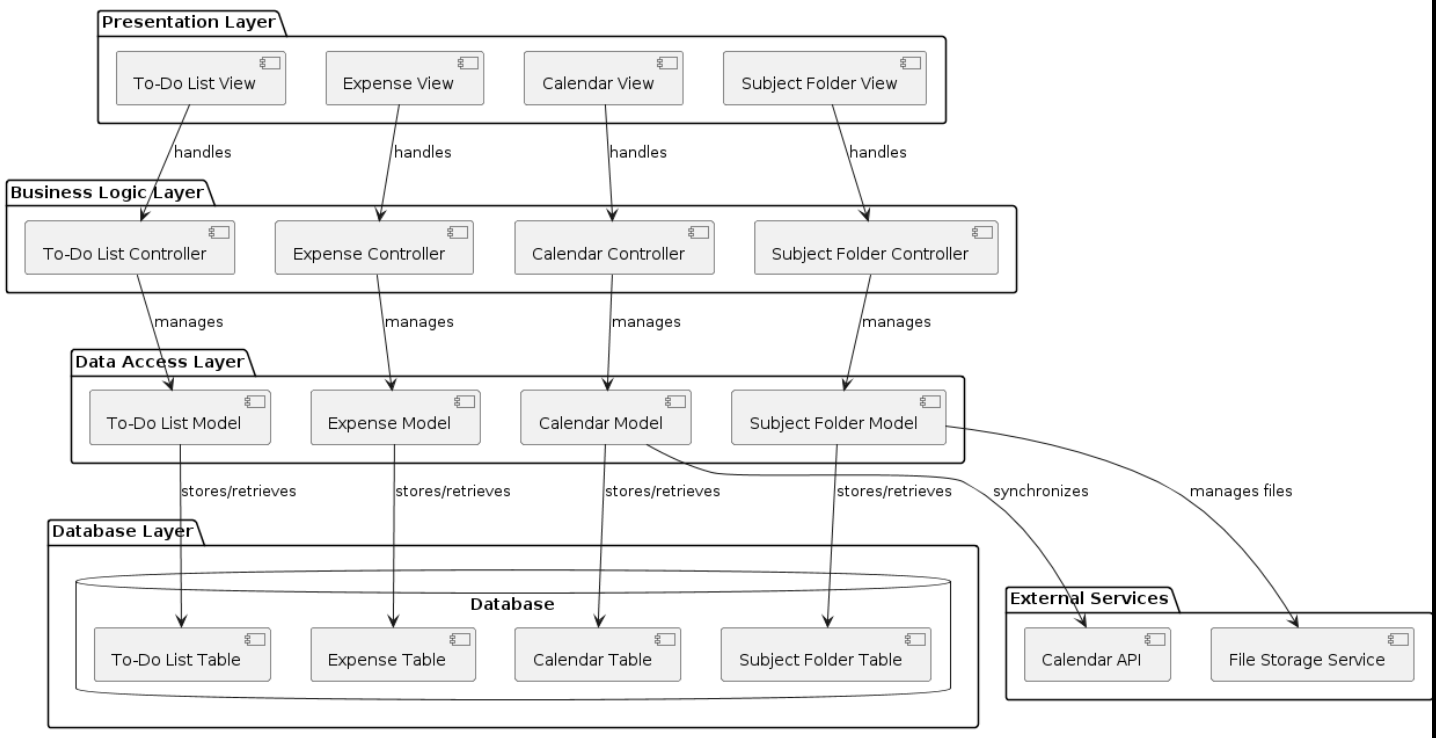
The app should be designed with maintainability in mind, enabling easy updates and bug fixes. The codebase should follow standard coding practices and be well-documented to facilitate understanding and modifications by different developers. Modular design principles should be applied, allowing individual components to be updated or replaced without affecting the entire system.

7. Compliance

The Student Kit App must comply with relevant regulations and standards, including data privacy laws such as GDPR for users in Europe. The app should ensure that all data handling practices meet legal requirements for data protection and user privacy. Regular audits should be conducted to verify compliance, and any necessary adjustments should be made promptly to address new regulatory changes.

OVERALL PROJECT ARCHITECTURE

EXP.NO: 6 **DATE: 09.4.2024**



Layers:

- **Presentation Layer:** This layer handles how information is presented to the user. It's responsible for gathering user input and displaying processed information.
- **Business Logic Layer:** This layer handles the core logic of the application. It receives user input from the presentation layer, performs necessary operations on the data model (interactions with the data access layer), and prepares the processed information for presentation.
- **Data Access Layer:** This layer interacts with the data storage system (database) to retrieve, store, and manage data.

Components:

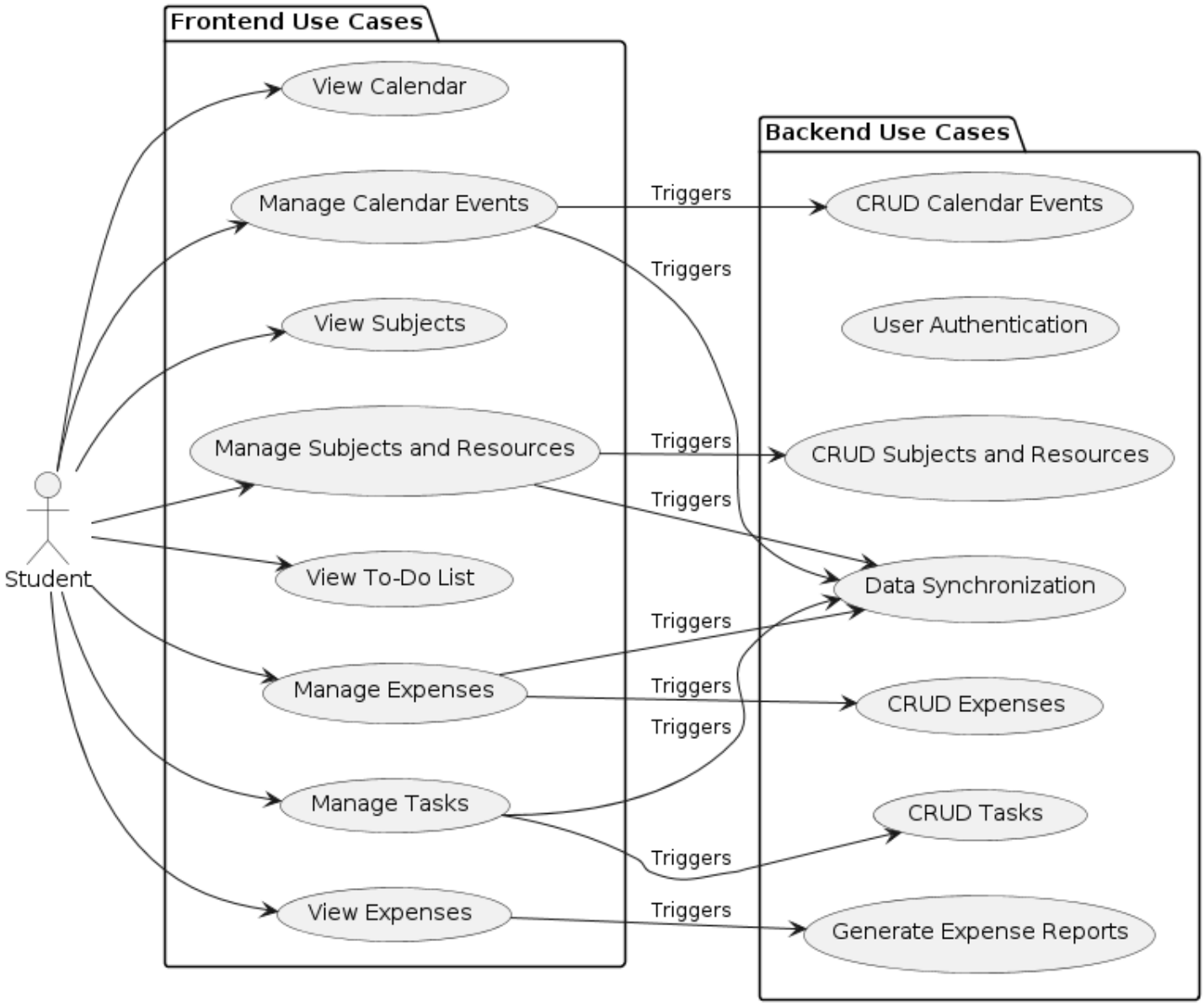
- **Presentation Layer:**
  - **To-Do List View:** This component is responsible for displaying the student's to-do list.
  - **Expense View:** This component is responsible for displaying the student's expenses.
  - **Calendar View:** This component is responsible for displaying the student's calendar events.
  - **Subject Folder View:** This component is responsible for displaying the student's subjects likely categorized in folders.
- **Business Logic Layer:**
  - **To-Do List Controller:** This component handles user interactions related to the to-do list. It receives user input from the To-Do List View, performs operations on the To-Do List Model (likely adding, editing, or deleting tasks), and retrieves data from the To-Do List Model to update the view.
  - **Expense Controller:** This component handles user interactions related to expenses. It interacts with the Expense Model and Expense View in a similar manner as the To-Do List Controller.
  - **Calendar Controller:** This component handles user interactions related to the calendar. It interacts with the Calendar Model and Calendar View in a similar manner as the To-Do List Controller.
  - **Subject Folder Controller:** This component handles user interactions related to subjects and folders. It interacts with the Subject Folder Model (not shown explicitly) and Subject Folder View in a similar manner as the To-Do List Controller.
- **Data Access Layer:**
  - **To-Do List Model:** This component stores and retrieves data related to the student's to-do list. It likely interacts with the To-Do List Table in the database.
  - **Expense Model:** This component stores and retrieves data related to the student's expenses. It likely interacts with the Expense Table in the database.
  - **Calendar Model:** This component stores and retrieves data related to the student's calendar events. It likely interacts with the Calendar Table in the database.
  - **Subject Folder Model:** This component stores and retrieves data related to the student's subjects and how they are categorized in folders (not shown explicitly in the diagram). It likely interacts with the Subject Folder Table in the database.

Additional Interactions:

- **Database:** The Data Access Layer components interact with the database to perform CRUD (Create, Read, Update, Delete) operations on the relevant data tables.
- **External Services (Optional):** The architecture mentions optional external services. These could be third-party services that the application might interact with to provide additional functionalities (e.g., a calendar API for advanced calendar features).

BUSINESS ARCHITECTURE DIAGRAM

EXP.NO: 7 **DATE: 19.4.2024**



Actor:

- Student: Represents the primary user who interacts with the Student Kit application. This actor is involved in all frontend use cases.

Frontend Use Cases:

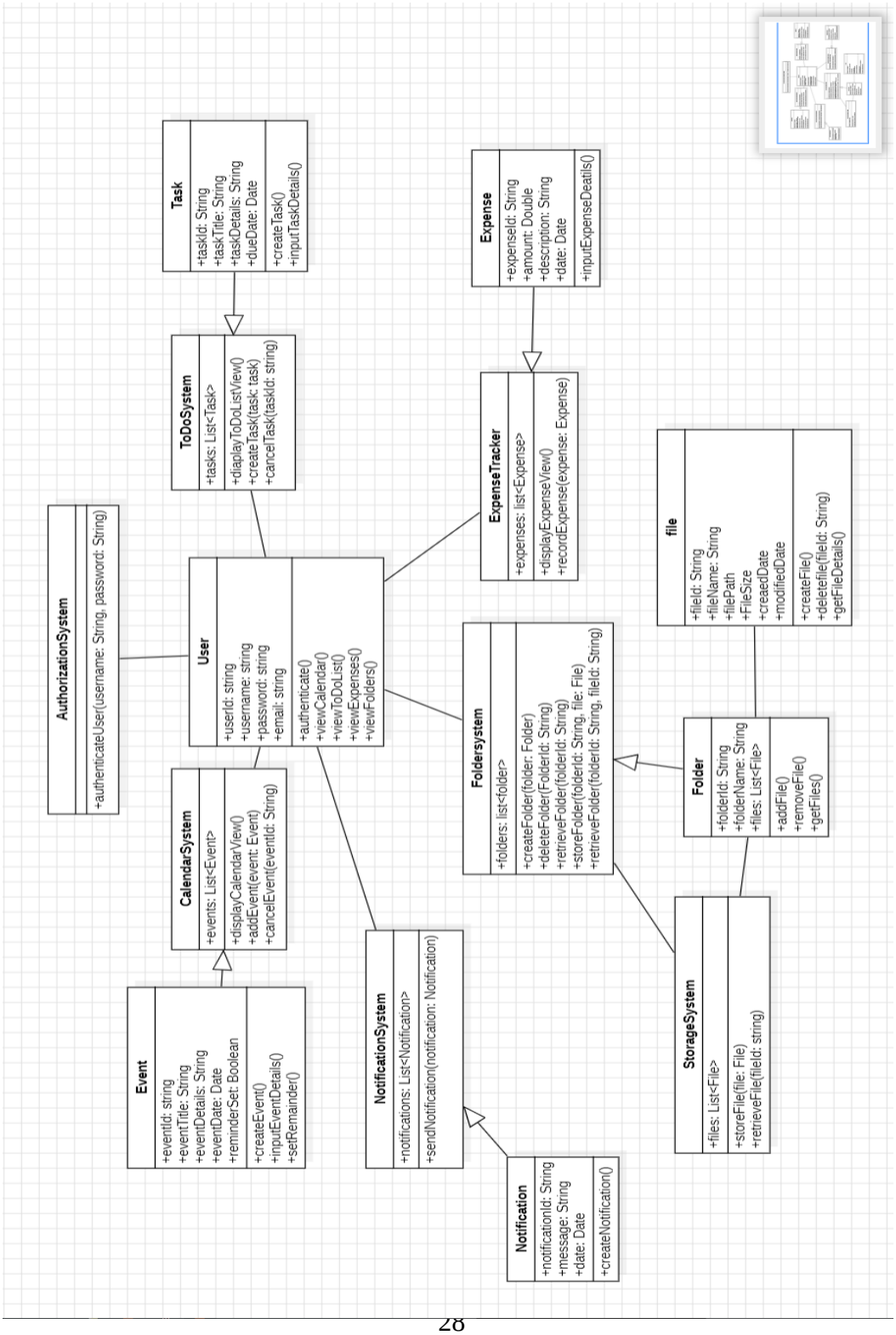
- **View Calendar**: Allows the student to view their calendar with events and reminders.
- **Manage Calendar Events**: Enables the student to add, edit, or delete calendar events.
- **View Subjects**: Allows the student to view a list of subjects and the associated study materials.
- **Manage Subjects and Resources**: Enables the student to manage subjects by adding, editing, or deleting subjects and related resources.
- **View To-Do List**: Allows the student to view their to-do list with tasks.
- **Manage Tasks**: Enables the student to add, edit, or delete tasks.
- **View Expenses**: Allows the student to view their recorded expenses.
- **Manage Expenses**: Enables the student to record, edit, or delete expenses.

Backend Use Cases:

- **Calendar Events**: Handles the backend operations (Create, Read, Update, Delete) for calendar events.
- **Subjects and Resources**: Manages backend operations for subjects and their resources.
- **Tasks**: Manages backend operations for tasks.
- **Expenses**: Manages backend operations for expenses.
- **Generate Expense Reports**: Creates reports based on the student's expense data.
- **User Authentication**: Manages user login and authentication processes.
- **Data Synchronization**: Ensures that data changes are synchronized between the frontend and backend.

CLASS DIAGRAM

EXP.NO: 8 **DATE: 30.4.2024**



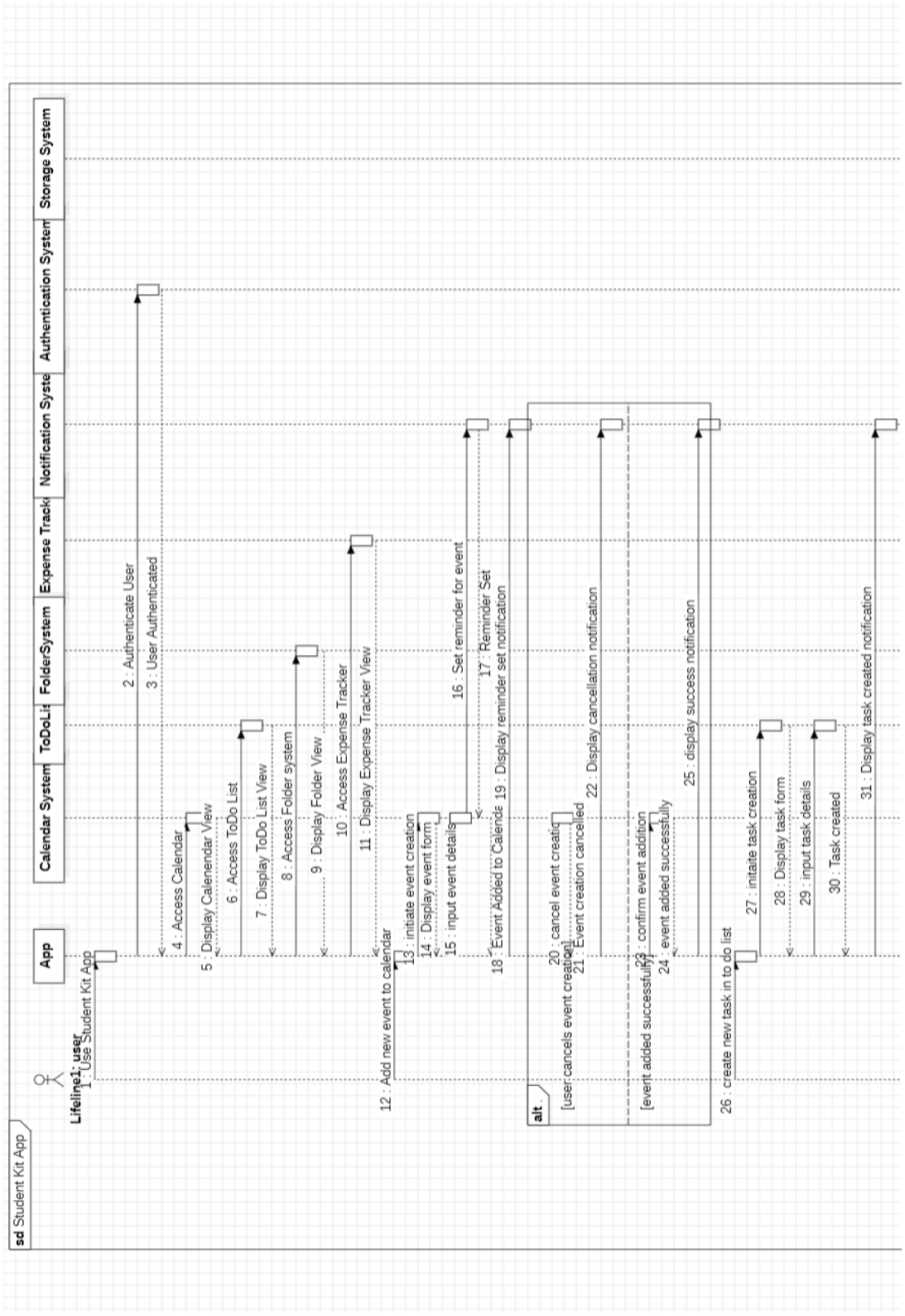
Classes:

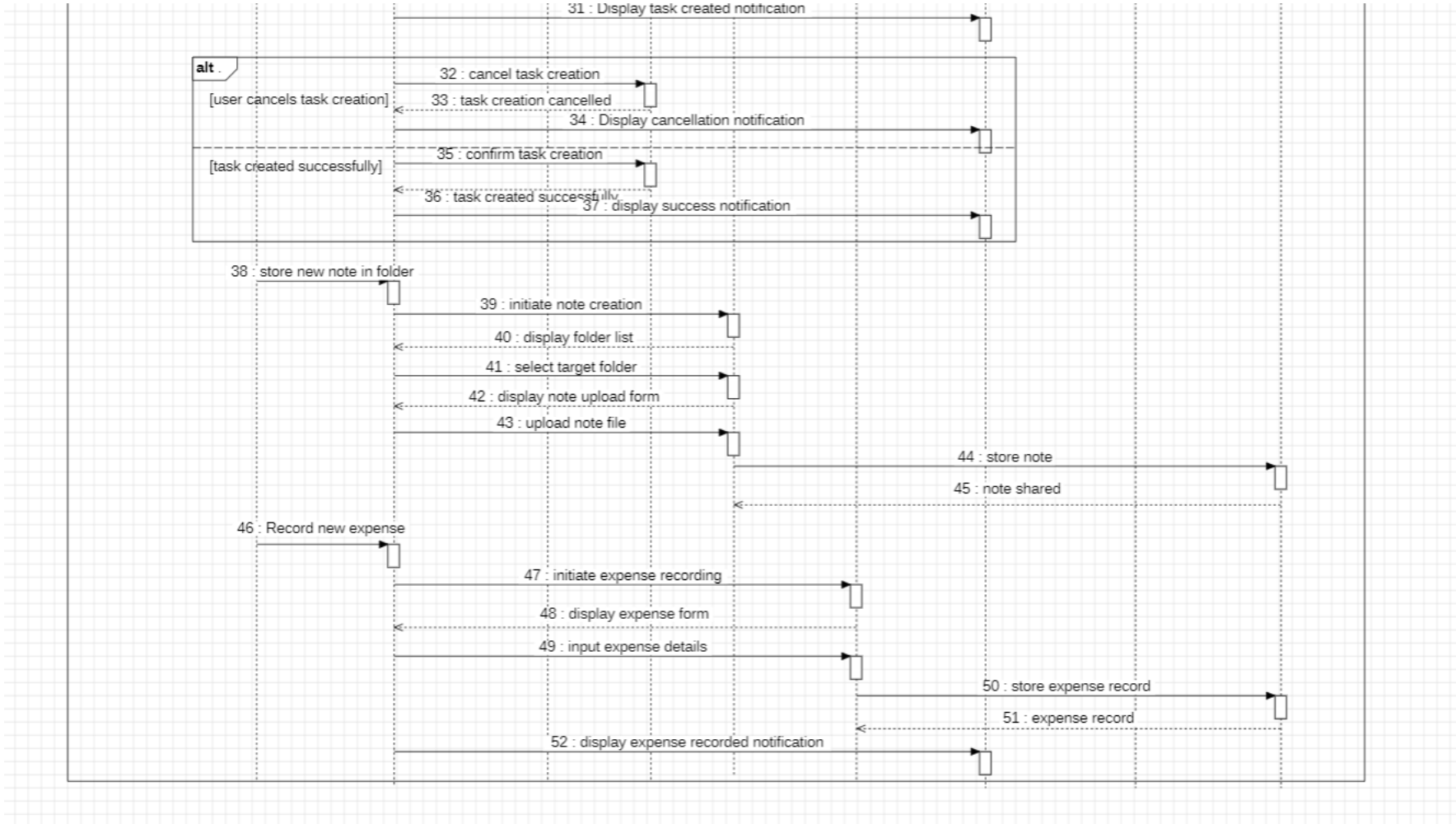
1. User: This class handles the functionality related to user login. It includes attributes such as password and mailid. It likely contains methods like verifyLogin which verifies the login credentials of the user. This class manages student information. Attributes include name (student's name).
2. CalendarSystem: This class is responsible for managing the student's calendar. It includes an attribute events, which is likely an array of Event objects. Methods might include addEvent for adding a new event, editEvent for modifying an existing event, and viewCalendar for displaying the calendar.
3. Event: Represents an event in the student's calendar. Attributes include title, date, and possibly a time. Methods for getting and setting these attributes are likely part of this class.
4. Task: Represents a task for the student. Attributes include title, completed (a boolean indicating if the task is completed), reminder (a date), and category. Methods for managing tasks may include createTask, completeTask, and methods for getting and setting task attributes.
5. TodoSystem : Manages a list of tasks. Attributes likely include tasks, an array of Task objects. Methods might involve managing the task list, such as createTask (potentially delegating to the Task class) and display for showing the list of tasks.
6. FolderSystems: Manages the student's notes. Attributes might include name and content of the note. Methods could involve addFolder for adding a new folder,.This has subclasses Folder and File that help in managing and controlling of this system easier.
7. ExpenseTracker: This class tracks the student's expenses. Attributes might include expenses, an array of Expense objects. Methods could include addExpense for recording a new expense and so on.
8. Expense: Represents an expense. Attributes may include amount, date, and possibly category. Methods for getting and setting these attributes are likely included.
9. Additional classes like NotificationSystem, AuthenticationSystem, and StorageSystem and many more help in interaction of application and device/web browser that it runs on by performing various functions.

Relationships: In the Student Kit App, the User class handles user authentication and interacts with the Profile class to manage user information. CalendarSystem for scheduling events and associates with Settings for managing user preferences. The Calendar class is composed of Event objects, indicating a strong lifecycle dependency. Similarly, the TodoSystem class contains Task objects, and the ExpenseTracker class contains Expense objects, both representing composition relationships where the lifecycle of tasks and expenses is tightly bound to their respective lists.

SEQUENCE DIAGRAM

EXP.NO: 9 **DATE: 10.5.2024**




31 : Display task created notification
alt .
32 : cancel task creation
[user cancels task creation]
33 : task creation cancelled
34 : Display cancellation notification
35 : confirm task creation
[task created successfully]
36 : task created successfully
37 : display success notification
38 : store new note in folder
39 : initiate note creation
40 : display folder list
41 : select target folder
42 : display note upload form
43 : upload note file
44 : store note
45 : note shared
46 : Record new expense
47 : initiate expense recording
48 : display expense form
49 : input expense details
50 : store expense record
51 : expense record
52 : display expense recorded notification

Actor:

- **Student:** The user interacting with the application.

System Components:

- **App:** The main student kit application.
- **Authentication System:** A system that verifies the student's identity.
- **Calendar:** The component responsible for managing the student's calendar events.
- **ToDoList:** The component responsible for managing Todo List events of the student.
- **Notification System:** The component responsible for showing notifications to the student.
- **Folder System:** This component manages the Notes storing and folder storing system of the student.
- **ExpenseTracker:** This component manages the Expenses of the student.
- **Storage System:** This component manages the storage access of the system and application.

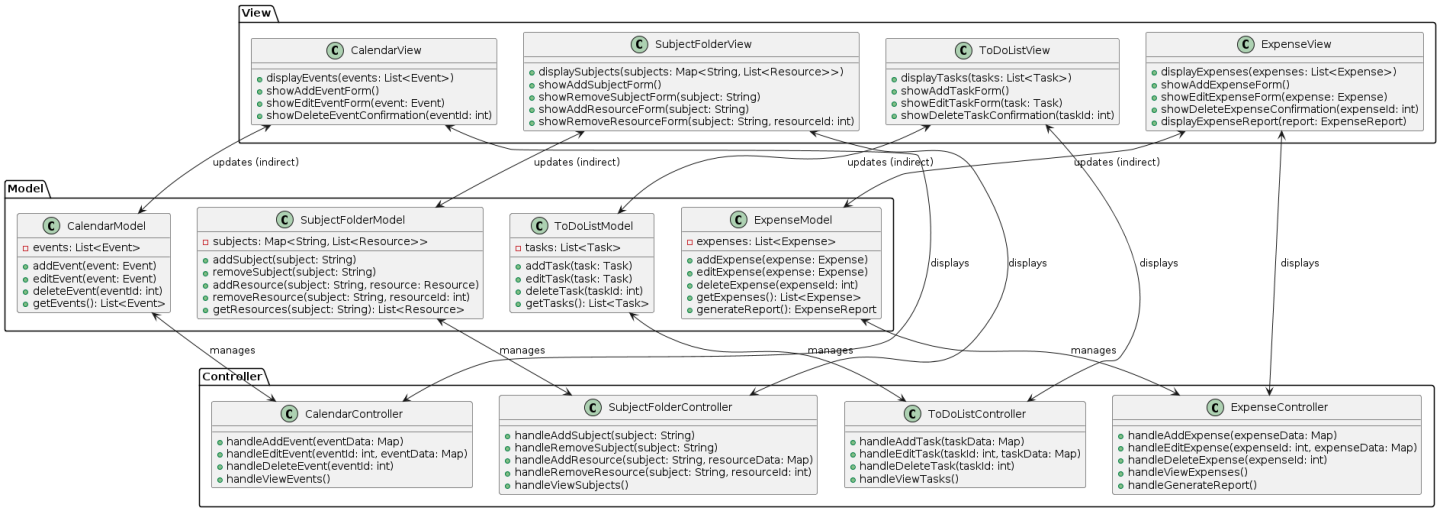
Sequence of Events:

- Student Uses App: The student interacts with the Student Kit application (App) to perform various tasks such as accessing the calendar, to-do list, folders, or expense tracker.
- App Authenticates User: The application initiates the authentication process by calling the Authentication System to verify the user's identity.
- Authentication System Authenticates User (Positive Outcome): The Authentication System successfully verifies the student's identity and sends a success message back to the App.
- User Authenticated: The App receives the authentication success message, allowing the user to proceed with their intended action.
- Access Calendar: The App interacts with the Calendar component to perform the requested calendar action, such as adding a new event, editing an existing event, or viewing the calendar.

- Calendar Processes Request: The Calendar component processes the request based on what the user is trying to do, whether it involves saving a new event, updating an existing event, or retrieving calendar data to display.
- (Optional) Notification System Shown: If the action results in a notification for the user (e.g., confirmation of a new event), the Calendar component might interact with the Notification System to display the message to the student.
- Access To-Do List: The student can interact with the To-Do List component to manage their tasks, including adding new tasks, updating existing tasks, or marking tasks as completed.
- To-Do List Processes Request: The To-Do List component processes these actions and updates the task list accordingly.
- Access Folders: The student can access the Folder system to organize their notes and other academic materials into subject-wise folders.
- Folders Process Request: The Folder system allows the user to create, update, or delete folders and manage the contents within them.
- Access Expense Tracker: The student can utilize the Expense Tracker to manage their personal finances by adding, updating, or viewing their expenses.
- Expense Tracker Processes Request: The Expense Tracker processes the financial data and updates the expense records accordingly.

ARCHITECTURAL PATTERN (MVC)

EXP.NO: 10 **DATE: 17.5.2024**



Model:

- **CalendarModel:** This likely represents the data and logic related to the student's calendar events. It has methods for getEvents, addEvent, editEvent, and deleteEvent.
- **SubjectModel:** This likely manages data and logic related to the student's subjects. It has methods for getSubjects, addSubject, removeSubject, and potentially addResource (to add learning materials to a subject).
- **ToDoListModel:** This likely manages the student's task list. It has methods for getTasks, addTask, editTask, and deleteTask.
- **ExpenseModel:** This likely manages the student's expenses. It has methods for getExpenses, addExpense, and editExpense.

View:

- **CalendarView:** This component displays the student's calendar events. It has methods for displayEvents, showAddEventForm, showEditEventForm, and showDeleteEventConfirmation.

- **SubjectView:** This component displays information related to the student's subjects. It has methods for displaySubjects, showAddSubjectForm, showRemoveSubjectConfirmation, potentially showAddResourceForm, and showEditResourceForm.
- **ToDolistView:** This component displays the student's task list. It has methods for displayTasks, showAddTaskForm, showEditTaskForm, and showDeleteTaskConfirmation.
- **ExpenseView:** This component displays the student's expenses. It has methods for displayExpenses, showAddExpenseForm, showEditExpenseForm, and showDeleteExpenseConfirmation.

Controller:

- **CalendarController:** This component handles user interactions related to the calendar. It has methods for handleAddEvent, handleEditEvent, handleDeleteEvent, potentially handleViewEvents, and methods for handling adding and removing events (not fully shown in the diagram).
- **SubjectController:** This component handles user interactions related to subjects. It has methods for handleAddSubject, handleRemoveSubject, potentially handleAddResource, handleEditResource, and methods for handling viewing subjects and resources (not fully shown).
- **ToDoListController:** This component handles user interactions related to the task list. It has methods for handleAddTask, handleEditTask, handleDeleteTask, and potentially handleViewTasks.
- **ExpenseController:** This component handles user interactions related to expenses. It has methods for handleAddExpense, handleEditExpense, handleDeleteExpense, and potentially handleViewExpenses.

Relationships:

The Model, View, and Controller components interact as follows:

- The Model provides data and logic to the Controller.
- The Controller receives user input from the View and manipulates the Model accordingly (e.g., adding a new event).
- The Controller instructs the View to update itself based on changes in the Model (e.g., displaying a newly added event).
- The View doesn't directly interact with the Model. It communicates with the Model only through the Controller.

TEST CASES

EXP.NO: 11 **DATE: 26.5.2024**

To-Do List:

- Add Task Test adding a task with all fields filled. Test adding a task with only mandatory fields filled. Test adding a task with invalid data (e.g., past due date). Verify that the new task appears in the task list.
- Edit Task Test editing a task's name, description, due date, and priority. Test editing a task to mark it as completed. Verify that changes are reflected in the task list.
- Delete Task Test deleting a task. Verify that the task is removed from the task list. Test deleting a task that does not exist (error handling).
- Mark Task as Completed Test marking a task as completed. Verify that the task's status is updated.
- View Task List Test viewing all tasks. Test filtering tasks by status (completed, pending). Test sorting tasks by due date, priority.

Calendar:

- Add Event Test adding an event with a title, date, time, and description. Test adding an event with invalid data (e.g., past date). Verify that the new event appears on the calendar.
- Edit Event Test editing an event's details. Verify that changes are reflected on the calendar.
- Delete Event Test deleting an event. Verify that the event is removed from the calendar. Test deleting an event that does not exist (error handling).
- Set Reminders Test setting reminders for events. Verify that reminders are received at the correct times.

Subject-wise Folder:

- Create Folder Test creating a new folder with a valid name. Test creating a folder with invalid data (e.g., empty name). Verify that the new folder appears in the list.
- Upload Files Test uploading a PDF file to a folder. Test uploading multiple files at once. Test uploading a file with an unsupported format. Verify that uploaded files appear in the folder.

- Organize Content Test renaming a folder. Test moving files between folders. Test deleting a folder and verifying that its contents are also deleted. Verify the folder structure is maintained correctly.

Finance Tracker:

- Add Expense Test creating an expense with a valid amount and description. Test creating an expense with an invalid amount (e.g., negative or zero).Test creating an expense with a missing description Verify that the new expense appears in the list after creation.
- Edit Expense Test editing an existing expense's amount and description. Test editing an expense with an invalid amount (e.g., negative or zero).Verify that the edited expense appears correctly in the list.
- Delete Expense Test deleting an expense. Test deleting a non-existent expense: Verify that the expense list is updated correctly after deletion.