# I. Introduction

## Project Title: Scheduler application

In today's fast-paced world, effective time management is essential for both individuals and organizations. The *Scheduler Application* project aims to provide an all-in-one solution for managing various schedules, including personal, academic, and celebratory events. The application is designed to enhance productivity, reduce manual scheduling errors, and streamline the organization of daily activities. With features such as adding/modifying personal schedules, automated crawling of school timetables, customizable view options, reminders, alarms, and event filtering, the system will cater to a wide range of user needs.

The project adopts an **AGILE methodology**, ensuring iterative development, regular updates, and adaptability to changing requirements. The system will focus on delivering a user-friendly interface, automation capabilities, and notification systems to improve time management efficiency. Through this application, users will gain better control over their schedules, allowing them to organize their lives more effectively.

# II. Planning

**Reason to select the project:**
We want to create an application that enhances productivity, reduces manual scheduling errors, and serves as an all-in-one solution for managing multiple types of schedules, including personal, academic, and celebratory events.

**Initial System Request:**
Add/Modify/Delete Personal Schedule: Users can manually manage their own schedules such as appointments, tasks, or events by editing
Crawling School Timetable Data: The system will automatically retrieve the data and update the school’s timetable whenever changes occur for the users
Setting to Change View Option (Week/Month Mode): Users can switch between different viewing modes (weekly or monthly) for better clarity of their schedules over different time frames.
Filter by type of event: Users can filter specific types of events (e.g., work, academic, personal) to easily arrange their time based on the type of event and gain a clearer understanding of their schedules
Reminders: Notification system to remind users of upcoming events as their start time approaches
Celebrations/Anniversaries: Support for tracking special days like birthdays or anniversaries.
Daily News Display: Add-on feature to show daily news articles.

Alarm: An alarm system to alert users of specific times, events, or reminders.

**Feasibility Analysis:**

- **Technical feasibility:**

+Requires expertise in UI/UX design for a clear timetable layout and knowledge of scheduling algorithms to handle overlapping events or classes: Feasible

+Access to System Clock, Sound, Notifications, and Files: Feasible

+Crawling School Timetable and Accessing Daily News: Feasible

- **Operational feasibility:**

+Easy to use

+Adaptable for different types of schedules

+Update and upgrade: regularly updating the tool with new features or bugs fixed

+Technical support: support system, whether it's via FAQs,...

- **Initial Project Plan:**

- Methodologies: AGILE

- Estimate project scope (size), time:
  - Size: medium
  - Time: 2-2.5 months

- Staffing Plan: 7 developers

- CASE tools:
  Github: Store version history
  Astah UML: UML diagramming
  Draw.io: Demo UI

# III. Analysis

## System Request:

**Add/Modify/Delete Personal Schedule:** Users can manually manage their own schedules such as appointments, tasks, or events by editing

**Crawling School Timetable Data:** The system will automatically retrieve the data and update the school's timetable whenever changes occur for the users

**Setting to Change View Option (Week/Month Mode):** Users can switch between different viewing modes (weekly or monthly) for better clarity of their schedules over different time frames.

**Filter by type of event:** Users can filter specific types of events (e.g., work, academic, personal) to easily arrange their time based on the type of event and gain a clearer understanding of their schedules

**Reminders:** Notification system to remind users of upcoming events as their start time approaches

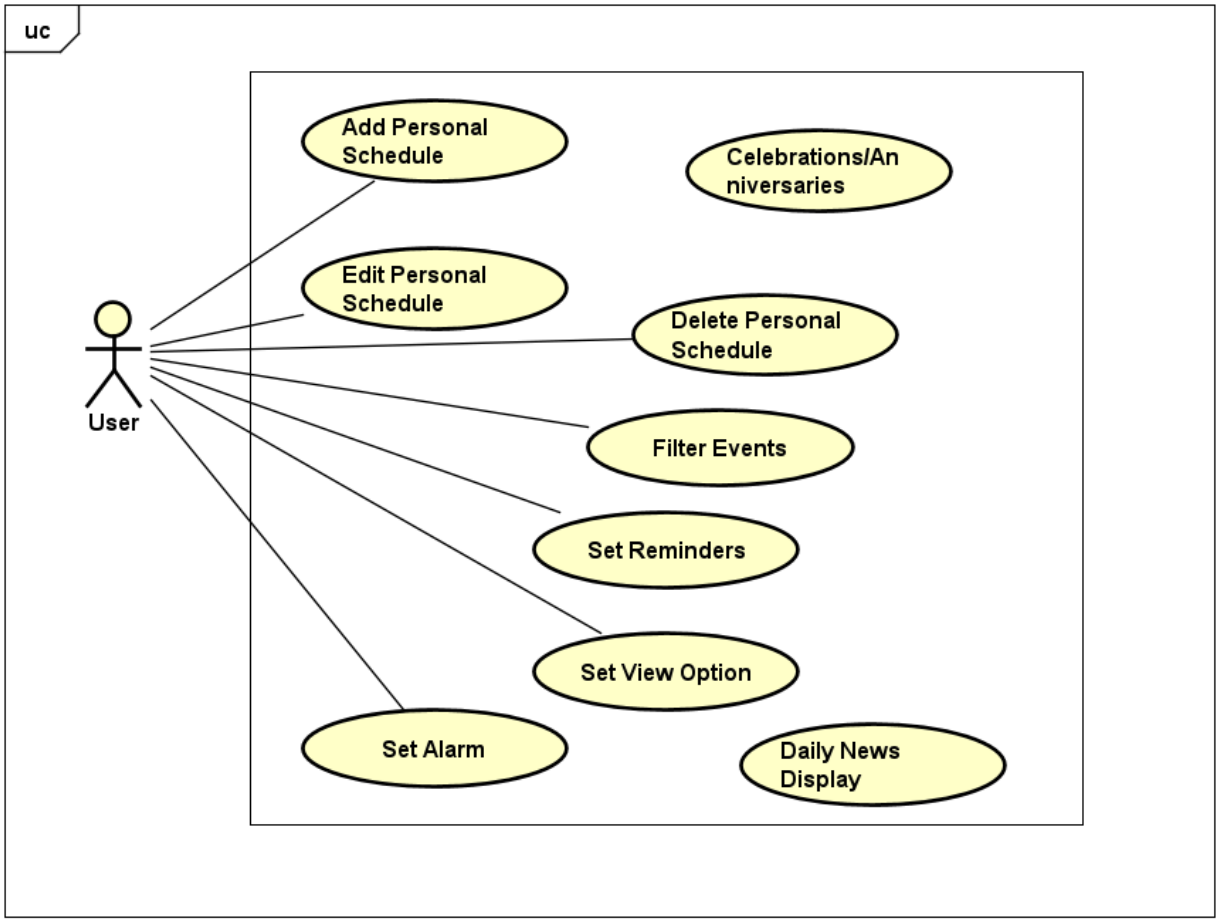
**Celebrations/Anniversaries:** Support for tracking special days like birthdays or anniversaries.

**Daily News Display:** Add-on feature to show daily news articles.

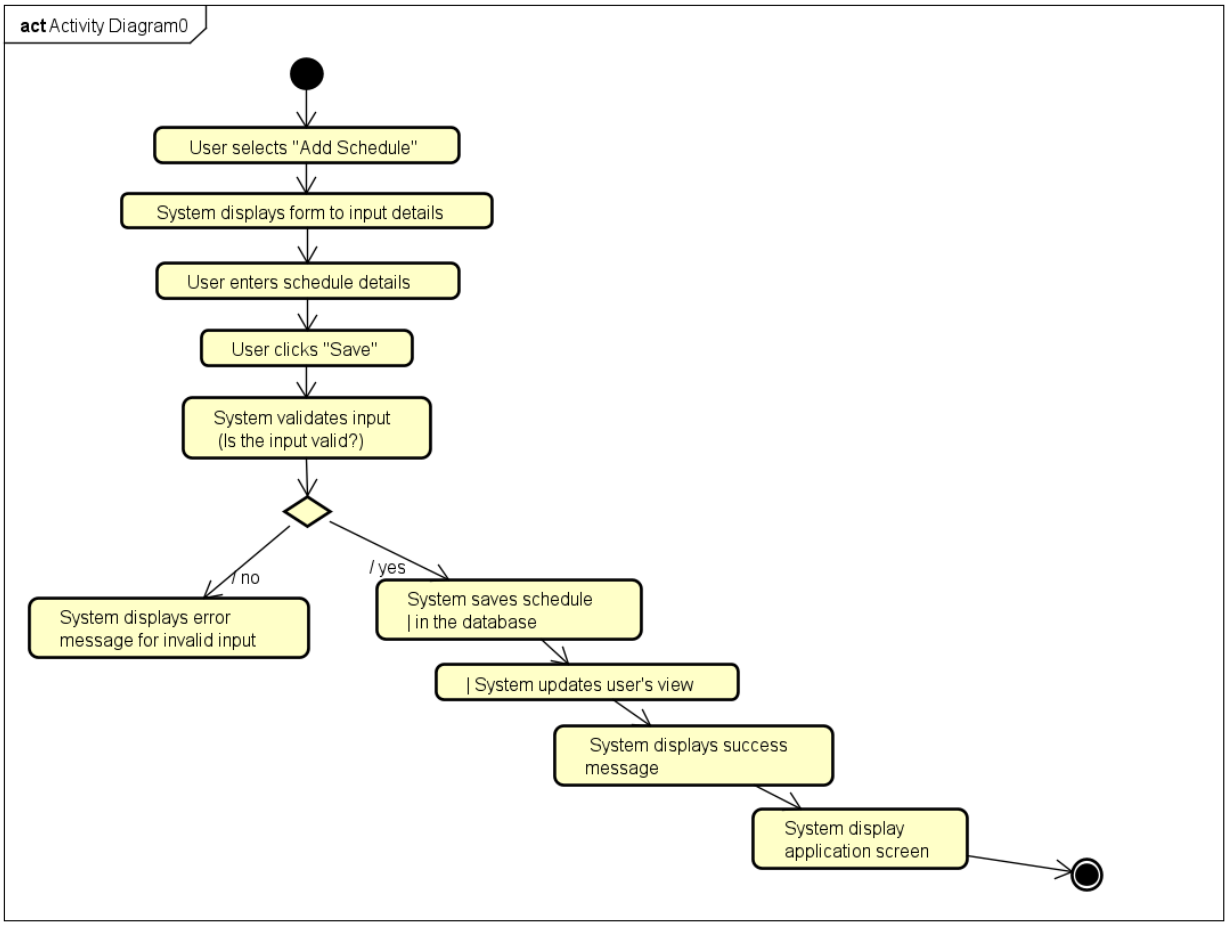
**Alarm:** An alarm system to alert users of specific times, events, or reminders.

# IV. Design

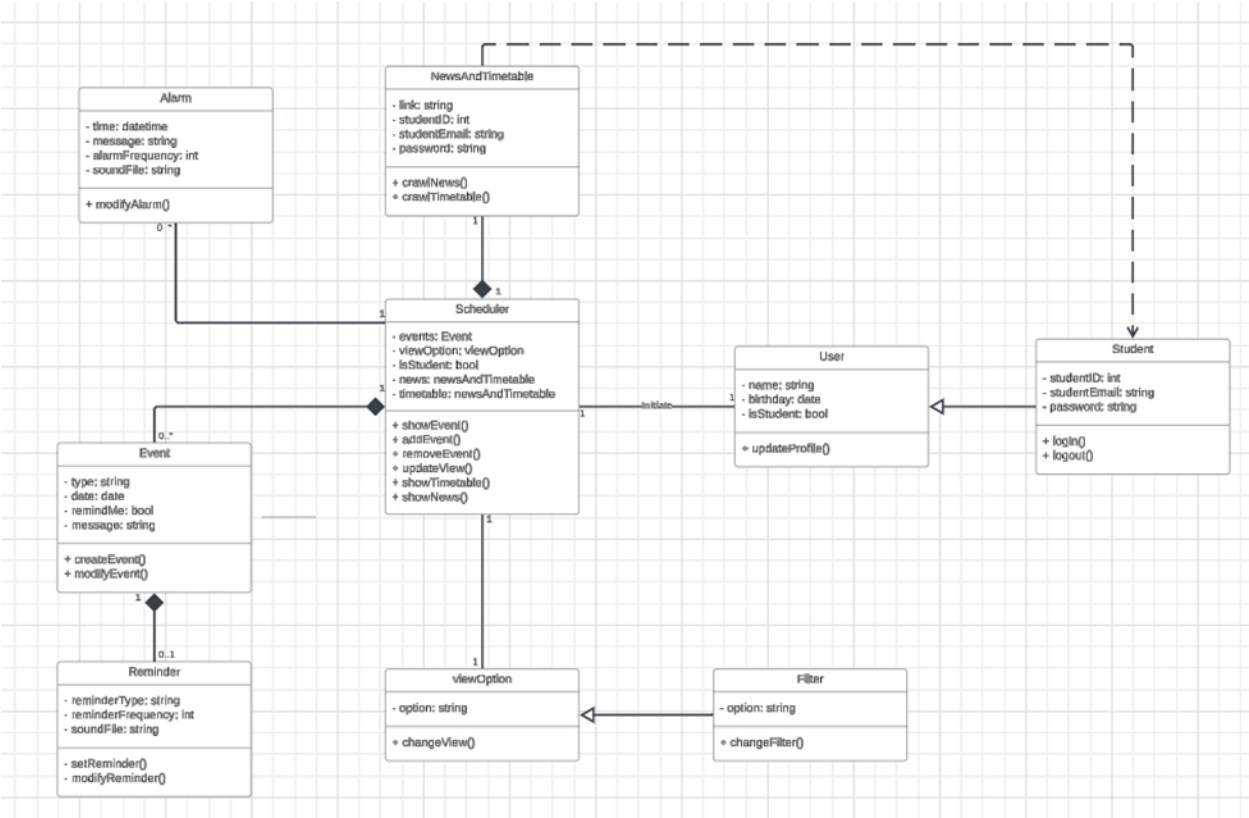
## Use case diagram



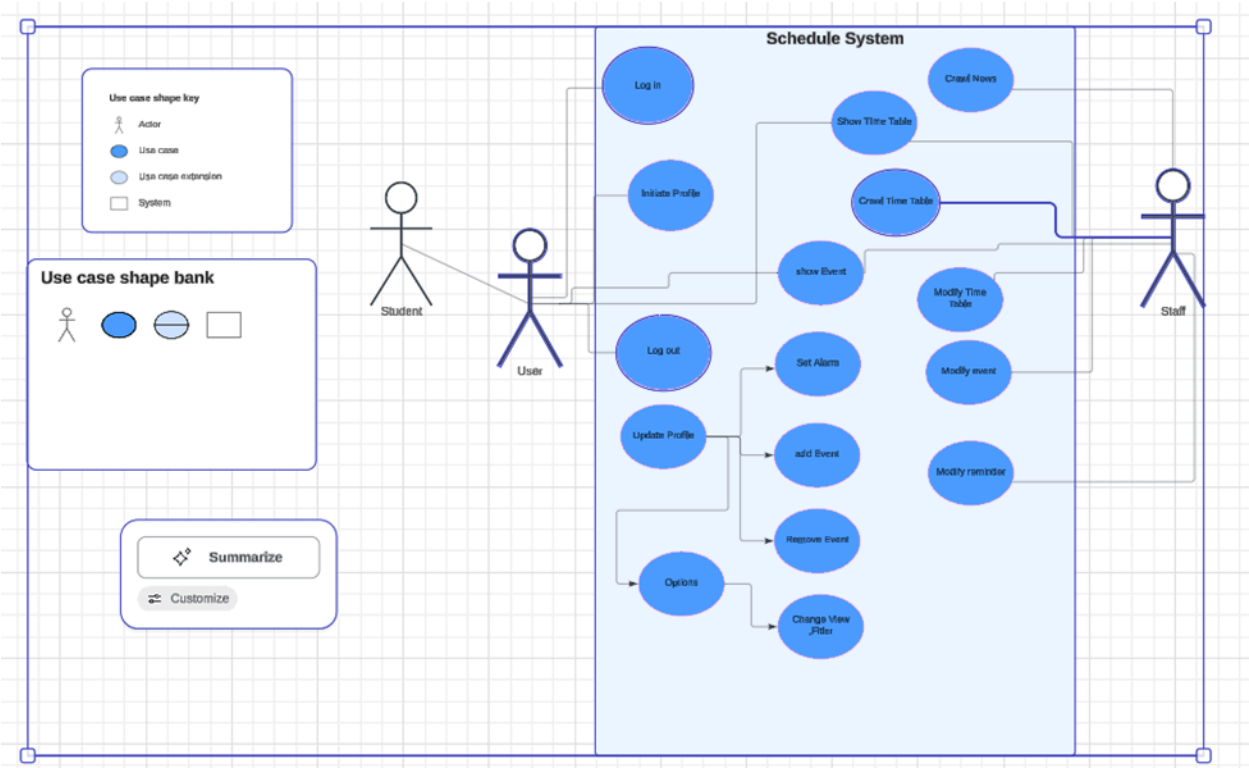
# Activity Diagram



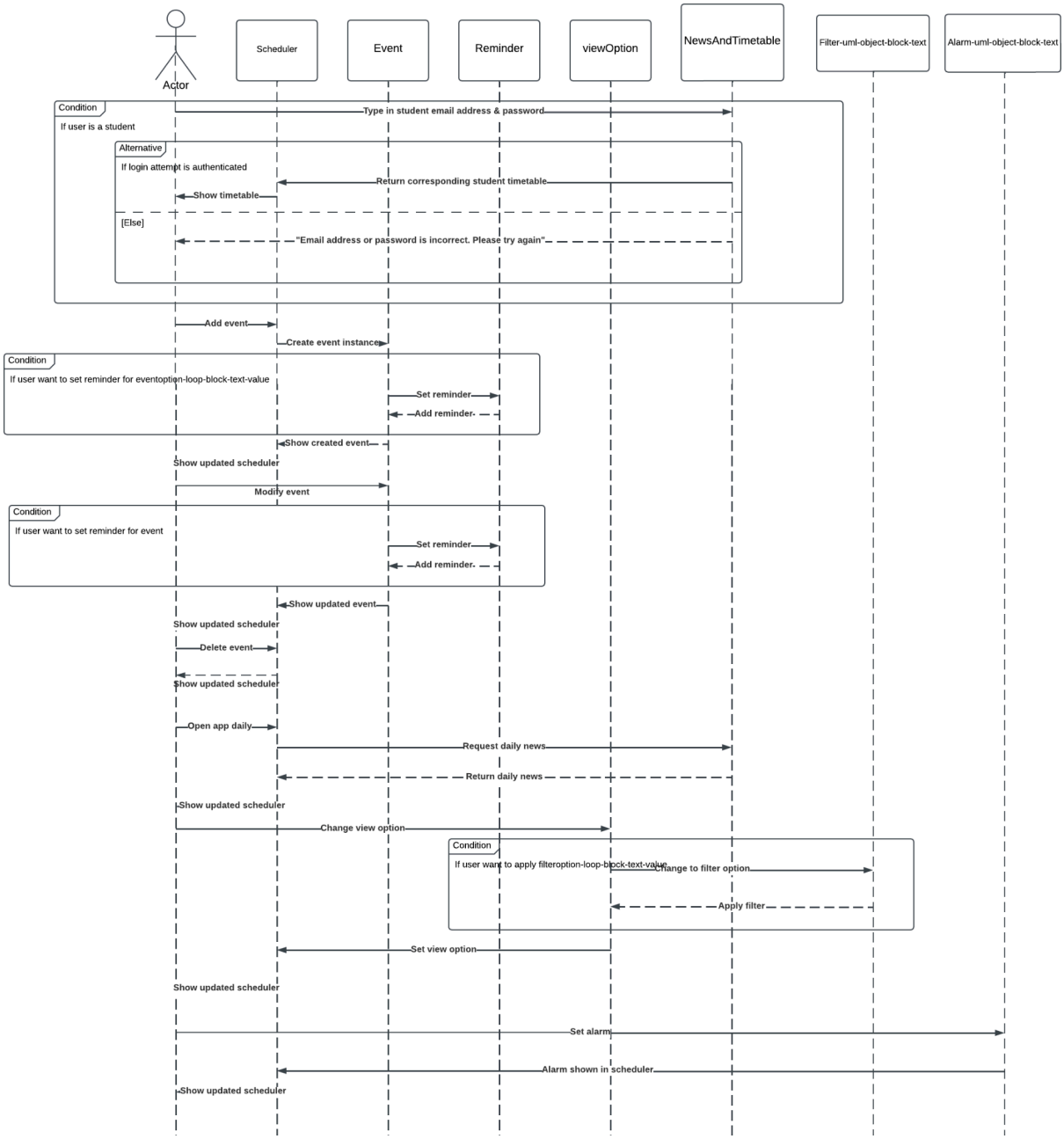

## Class diagram:



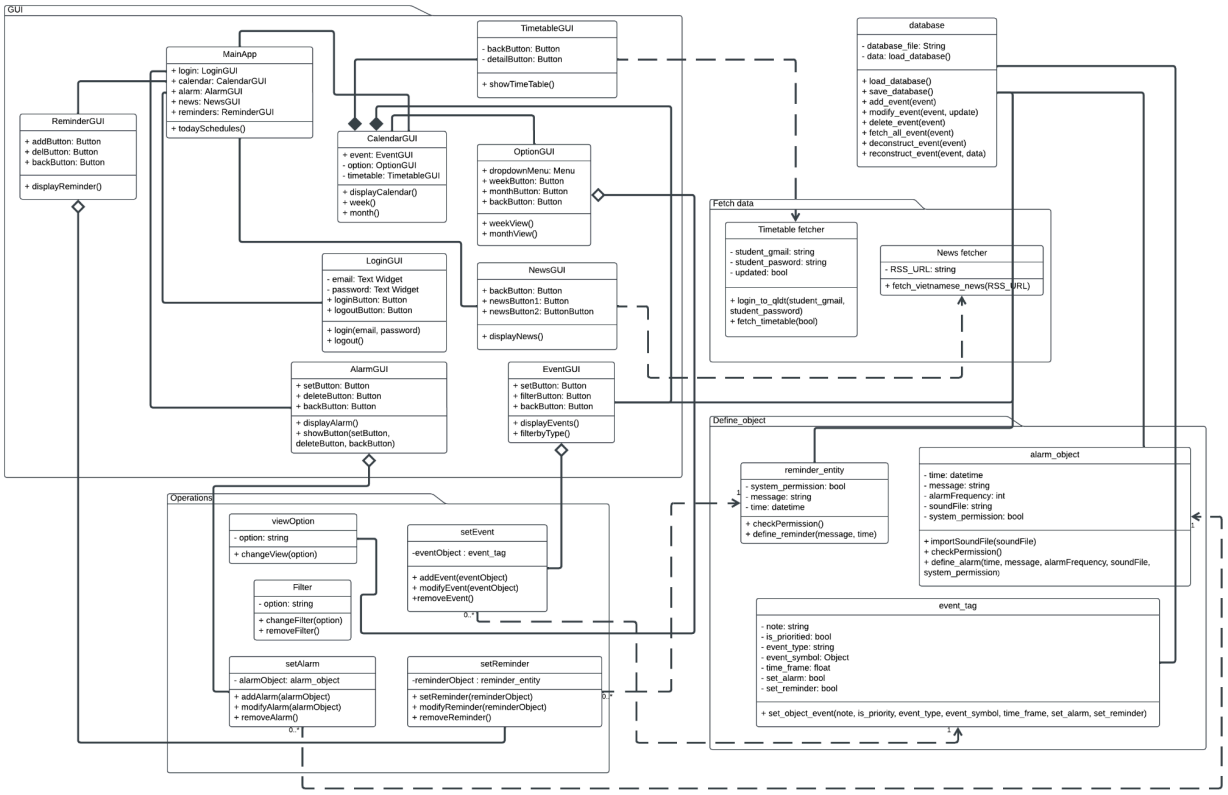

## Communication diagram:




## Sequential diagram:




## Class diagram:




### Database table:

If allow_reminder = false:

| Attribute name | Data type | Description |
|---|---|---|
| event_id | string | Unique identifier for each event |
| event_type | string | Type of the event |
| enent_name | string | Name of the event |
| event_timeframe | string | Timeframe of the event |
| allow_reminder | boolean | Indicates if a reminder is enabled |
| event_note | string | Additional notes about the event. |

If allow_reminder = true & reminder = “alarm”:

| Attribute name | Data type | Description |
|---|---|---|
| alarm_sound_file_directory | string | Path to the alarm sound file |
| alarm_repetition | integer | Number of times the alarm repeats. |

If allow_reminder = true & reminder = “notification”:

| Attribute name | Data type | Description |
|---|---|---|
| notify_in_advance | integer | How long the system will notify in advance (Minutes) |

# Use Case Detailed Description

**Use Case: Add/Edit/Delete Personal Schedules**

- **Actor:** User
- **Goal:** Allow users to manage their personal appointments, tasks, or events.
- **Preconditions:** The user must be logged into the application.
- **Postconditions:** The personal schedule is updated successfully (added, edited, or deleted).

**Main Flow (for Add Personal Schedule):**

1. The user selects the "Add Schedule" option.
2. The system displays a form to input schedule details (e.g., date, time, description, type).
3. The user fills out the form with the schedule details.
4. The user clicks "Save."

5. The system saves the new schedule to the database and updates the user's view.
6. The system displays a success message.

**Alternate Flow (for Edit Personal Schedule):**

1. The user selects an existing schedule and clicks "Edit."
2. The system displays the schedule details in a form.
3. The user modifies the necessary details.
4. The user clicks "Save."
5. The system updates the schedule in the database and the user's view.
6. The system displays a success message.

**Alternate Flow (for Delete Personal Schedule):**

1. The user selects an existing schedule and clicks "Delete."
2. The system asks for confirmation to delete the schedule.
3. The user confirms the deletion.
4. The system removes the schedule from the database.
5. The system updates the user's view and displays a success message.

**Use Case: Filter events**

- **Actor:** User
- **Goal:** Allow users to efficiently view only the events that are relevant to their current needs, reducing clutter and enhancing usability.
- **Preconditions:** The user must be logged into the application and there is at least 1 event created.
- **Postconditions:** The user sees a list of events that match the selected filter criteria.

**Flow:**

1. The user selects "Filter" from the events page**.**
2. The system displays filtering options. The user selects or inputs the desired criteria.
3. The user confirms the selected filter options by clicking "Apply".
4. The system queries the event database and retrieves events that match the selected criteria.
5. The system displays the filtered events on the user's screen, hiding events that do not meet the criteria.
6. The filter remains active until the user resets or modifies it.

**Use Case: Set reminders**

- **Actor:** User

- **Goal:** Ensure that users are notified about important events ahead of time, helping them stay organized and on schedule.

- **Preconditions:** The user must be logged into the application and there is at least 1 event created.

- **Postconditions:** The system schedules a notification based on the time and reminder settings.

**Flow:**

1. The user opens an event for which they want to set a reminder.

2. The system prompts the user to input reminder settings: Time before the event (e.g., 10 minutes, 1 hour, 1 day), notification type, recurrence option.

3. The user confirms the reminder settings by clicking "Save".

4. The system schedules the reminder based on the user's input and adds the reminder details to the event.

5. The system shows a confirmation message indicating that the reminder has been set successfully.

6. The reminder is stored and will trigger the notification at the specified time.

**Use Case Name**: Set Alarm

**Actors**:

- **User**: The person setting the alarm.
- **System**: The Scheduler application.

**Goal**: The user wants to set an alarm for a specific time to be reminded of an event or task.

### Preconditions:

- The user is logged into the application.
- The user has a schedule or event created (though alarms can be set for any time, regardless of events).

### Postconditions:

- The alarm is successfully set, and the user will be notified when the alarm time arrives.
-

### Main Flow:

1. **User Action**: The user navigates to the alarm settings.
2. **System Response**: The system displays an interface where the user can set the alarm time, date, and optional message or label for the alarm.
3. **User Action**: The user enters the time, selects the date, and sets an optional label/message for the alarm.
4. **User Action**: The user confirms by pressing the "Set Alarm" button.
5. **System Response**: The system stores the alarm in the database.
6. **System Response**: The system displays a success message indicating that the alarm has been successfully set.
7. **System Response**: At the specified time, the system triggers the alarm notification.

**Use Case Name**: Set View Option (Week/Month Mode)

**Actors**:

- **User**: The person changing the view settings.
- **System**: The Scheduler application.

**Goal**: The user wants to change how their schedule is displayed, switching between a weekly view and a monthly view.

### Preconditions:

- The user is logged into the application.
- The application is displaying the user's schedule.

### Postconditions:

- The view mode is successfully changed, and the schedule is displayed either in week mode or month mode based on the user’s preference.

### Main Flow:

1. **User Action**: The user navigates to the view settings (or toggles a button directly on the schedule view).
2. **System Response**: The system displays the available options: **Week View** and **Month View**.
3. **User Action**: The user selects the desired view option (either **Week** or **Month**).
4. **System Response**: The system updates the schedule view based on the selected option.
5. **System Response**: The system refreshes the display, showing either the week’s events or the entire month’s events.
6. **System Response**: The system confirms that the view has been changed by showing a success message or visually indicating the new view mode (e.g., highlighting the selected view option).

# V. Conclusion

The *Scheduler Application* is a robust and innovative solution designed to tackle the challenges of modern time management by providing a centralized platform for

scheduling various events and tasks. The application offers essential features such as manual scheduling, automated crawling of school timetables, reminders, alarms, and event filtering, ensuring users can seamlessly organize and track their personal, academic, and celebratory schedules. With additional capabilities like customizable view modes, daily news display, and support for celebrations or anniversaries, the system provides a holistic approach to time management, enhancing productivity and minimizing scheduling conflicts.

The feasibility analysis demonstrates the project's strong technical and operational foundation. The integration of advanced scheduling algorithms, automated data crawling, and a notification system ensures the system's reliability and efficiency. The inclusion of user-centric design principles and adaptability to diverse scheduling needs makes this application practical and accessible for a wide range of users.

By adopting an **AGILE methodology**, the development process is designed to be iterative, flexible, and collaborative, ensuring that user feedback is incorporated at each stage to refine the system. With a dedicated team of seven developers and tools such as **GitHub** for version control, **Astah UML** for system modeling, and **Draw.io** for UI design, the project is structured to deliver a high-quality product within the estimated 2-2.5 months timeframe.

In conclusion, the *Scheduler Application* will not only streamline scheduling processes but also empower users to manage their time more effectively and efficiently. By reducing manual errors, automating updates, and providing a user-friendly interface, the application aims to become a reliable and indispensable tool for individuals managing complex schedules in their personal, academic, and professional lives. With its scalability and potential for future enhancements, the system is well-positioned to evolve into a comprehensive productivity solution that meets the ever-growing demands of time management in today's dynamic world.