Develop by Diego R.R.

Cometical

group # *24*

Diego R.R.
dar220007@utdallas.edu

Gael Romero
jgr230000@utdallas.edu

Emily Rouse
zxr220003@utdallas.edu

Alan Roybal: leader
aer220004@utdallas.edu

Rishabh Sabnavis
rds230002@utdallas.edu

November 13, 2023

1 Project Idea

We have selected the *Assignment and Study Tracker* app idea for our project. This app is meant as an integrative solution, designed for students who are struggling with multiple responsibilities and deadlines.

2 Scheduling and Planning Subsystem

The Scheduling and Planning Subsystem is a core component of the *Assignment and Study Tracker* app, designed to provide a comprehensive solution for managing academic responsibilities. Its primary function is to integrate seamlessly with eLearning platforms, harnessing a variety of data points such as assignment due dates, quiz schedules, test timings, and extracurricular

event dates. This integration allows for the creation of a dynamic, customizable scheduling planner that goes beyond mere reminders, actively assisting students in optimizing their time based on their academic and personal commitments.

3 Design

In the *Assignment and Study Tracker* app, the Scheduling and Planning Subsystem is integrated with other subsystems, adhering to the Input, Process, Output design paradigm. The **Input** phase, led by the *Data Retrieval Function*, works closely with the Data Integration and Synchronization Subsystem to ensure accurate data collection from eLearning platforms needed for the *Scheduler*. The **Process** phase is named *Scheduler* and is split between the *Personalized Scheduler Function* and *Dynamic Scheduler Adjustment Function*. *Personalized Scheduler Function* works with Security and Privacy Subsystem and User Profile Management and Customization Subsystem to provide a customizable experience and adhere to the privacy contracts. Finally, the **Output** phase is encompassed by *User Interaction and Interface Function*. This cohesive structure ensures that each subsystem not only performs its specific role but also collaboratively contributes to a harmonious and effective overall system.

3.1 Visual Prototype

To provide a more comprehensive explanation of why the Scheduling and Planning Subsystem in *Assignment and Study Tracker* app lacks a visual prototype, it's crucial to understand the nature of its operations and design philosophy. This subsystem is fundamentally a backend functionality, focusing on processing and managing academic data retrieved from eLearning. Its primary responsibility is to handle complex tasks such as parsing assignment deadlines, quiz schedules, and other academic events. This emphasis on data processing is a clear indication of its backend orientation, where the main objective is to ensure accurate, efficient, and reliable data handling rather than visual presentation.

3.2 Pseudocode

3.2.1 Data Retrieval Function

Retrieving the ELearningAPI is handled by the Data Integration and Synchronization Subsystem. The Scheduling and Planning Subsystem only needs to call the Data Integration and Synchronization Subsystem to retrieve the ELearningAPI. The ELearningAPI is then used to retrieve the calendar and preferences for the user, later used by the hollistic algorithm.

```
1: function RetrieveDataForScheduling(user)
2:     eLearn ← newELearningAPI
3:     calendar ← eLearn.getCalendar(user)
4:     preferences ← user.getPreferences()
5:     schedule ← newSchedule
6:                 ▷ New Schedule based on calendar, preferences, and user
7:         ▷ This schedule is just the list of events retrieved from eLearning
8:     return schedule
```

3.2.2 Personalized Scheduler Function

This function received the data from the Data Integration and Synchronization Subsystem and generates a personalized schedule for the user. The initialization of the scheduler consist of algorithm that checks multiple conflicts such as privacy hints and personal preferences.

```
1: function PersonalizedScheduler(user)
2:     schedule ← retrieveDataForScheduling(user)
3:     personalizedSchedule ← new Schedule
4:     for all event in schedule.getEvents() do
5:         if IsConflict(event, personalizedSchedule) then
6:             event ← ResolveConflict(event, personalizedSchedule)
7:         ApplyPreferencesToEvent(event, user.getPreferences())
8:         personalizedSchedule.addEvent(event)
9:     return personalizedSchedule

10: function ApplyPreferencesToEvent(event, preferences)
11:     for all preference in preferences do
12:                                 ▷ Modify the event based on the preference
```

```
        ▷ This would be a family of functions that modify the event
    return event

function IsConflict(event, schedule)
    for all scheduledEvent in schedule.getEvents() do
        if IsOverlapping(event, scheduledEvent) then
            ▷ Implementation of IsOverlapping exluded for brevity
            return true
    return false

function ResolveConflict(event, schedule)
        ▷ Ask user for input on how to resolve the conflict
        ▷ Resolve problem and retrieve new event
    return event
```

3.2.3 Dynamic Scheduler Adjustment Function

Machine intelligence used for this function was simplified to a hollistic algorithm. Take into account that this is the most independent function from other subsystems and it assumes that the Personalized Scheduler Function initialized the hints correctly.

```
function DynamicScheduler(user, schedule, ELearnAPI)
    userData ← ELearnAPI.getData(user)
    peerData ← ELearnAPI.peerData()
    while not IsEndOfDay do
        performance ← userData.getInteger()
        adjustments ← Hollistic(performance, peerData, schedule)
        ApplyAdjustments(schedule, adjustments)
        OptimizeSchedule(schedule)
        wait for a predefined interval
    return newSchedule

function Hollistic(performance, peerData, schedule)
        ▷ Perform analysis on the performance and peer data
        ▷ Generate adjustments to the schedule
    return adjustments
```

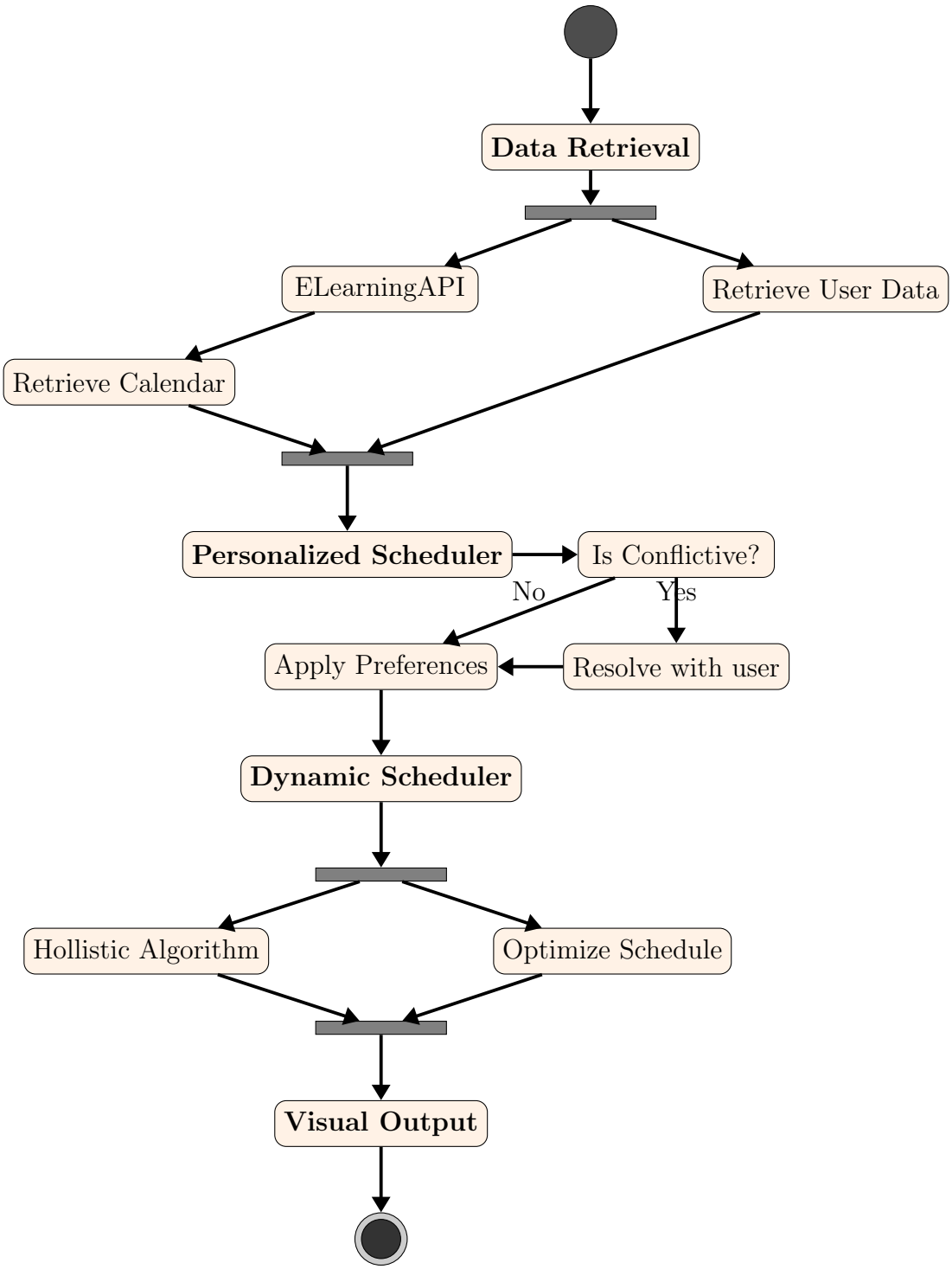
15: **function** ApplyScheduleAdjustments(*schedule*, *adjustments*)
16: ▷ *Apply generated adjustments to the schedule*

17: **function** OptimizeSchedule(*schedule*)
18: ▷ *Use machine intelligence to further optimize the schedule*

3.2.4 User Interaction and Interface Function

Since the Scheduling and Planning Subsystem is a backend component, it does not have a user interface. However, it does interact with the User Profile Management and Customization Subsystem and Security and Privacy Subsystem to ensure that the user has a customizable experience and that their privacy is protected.

3.3 UML Activity Diagram



3.4 Data Used

The Scheduling and Planning Subsystem uses data that is easily identified by how classes interact between each other. Thus, the following UML class chart will help understand the data needed for this subsystem: