Student Event Tracking System

Group name - pip install project520

Aaron Sun | Deepa Rukmini Mahalingappa | Deepika Sinagri Velu | Karthik Ravichandran

Github Repo Link: https://github.com/karthikRavichandran/students-event-tracking-system

Presentation (pptx): 520 midpoint ppt.pptx
Presentation Video link : video3531240850.mp4

1. Requirements

1.1. Overview

Our project is called the Student Event Tracking System, which is targeted towards students. Our objective is to create a system for students who can view critical and non-critical alerts in the form of a one-liner and a short summary. The primary reason for developing this system is to bring events pertaining to each user in various categories into a single view.

The project will consist of an interactive UI (an initial design will be developed in Figma) that will be developed in JAVA, a simple database connection that stores user data and event information (like a one-liner, summary, and raw data), and LLM-based information extraction from different sources such as Mail, Piazza discussion, GradeScope, and Spire. Since it will be difficult to get permission to assess those APIs, we will synthesize data in those forms and emulate the scenario for our demo.

1.2. Features

The main 5 components of our project are critical deadline alerts, grade summaries, important piazza discussions, upcoming exam alerts, and providing an LLM-based summary of the information above.

1.3. Functional Requirements (Use cases)

Karthik

1. **User information:** Display a basic summary of student information, such as courses enrolled, student ID, etc.
    a. A student wants to know which classes they have enrolled on Gradescope to know their schedule.

b. Other information like Student ID and their information will help them in quick reference

2. **Critical deadline alerts.**

c. Here it will show 1-2 days submission deadline with course details.
d. Critical deadline like 6-12 hr submissions with course details
e. Alerts to mark attendance to classes

3. **Grade Summary**

f. A small analysis of grade and its progression from time to time .
g. Score to Grade conversion and estimated effort to push the grade to next level
h. IF possible we will allow the user to set the targeted grade and the estimation will show how you can achieve this

4. **Average load in hours per day**

i. From the Grade score we can collect the deadline for each assignments, by using this we can estimate the workload per hour, per day and per week
j. Based on the previous submission we can even suggest the effort we have to put to complete it before the deadline
k. Clashes of quizzes, Exams and assignments deadline can viewed shows some suggestion to handle the conflict

5. **Instructor's announcements**

l. Since Instructor Announcement is important, we will show the summary for the instructors announcement with respect to the courses

6. **Critical Piazza discussions**

m. Piazza discussion will be filtered based on recent assignments and quizzes
n. Response for your discussion and summary will be posted

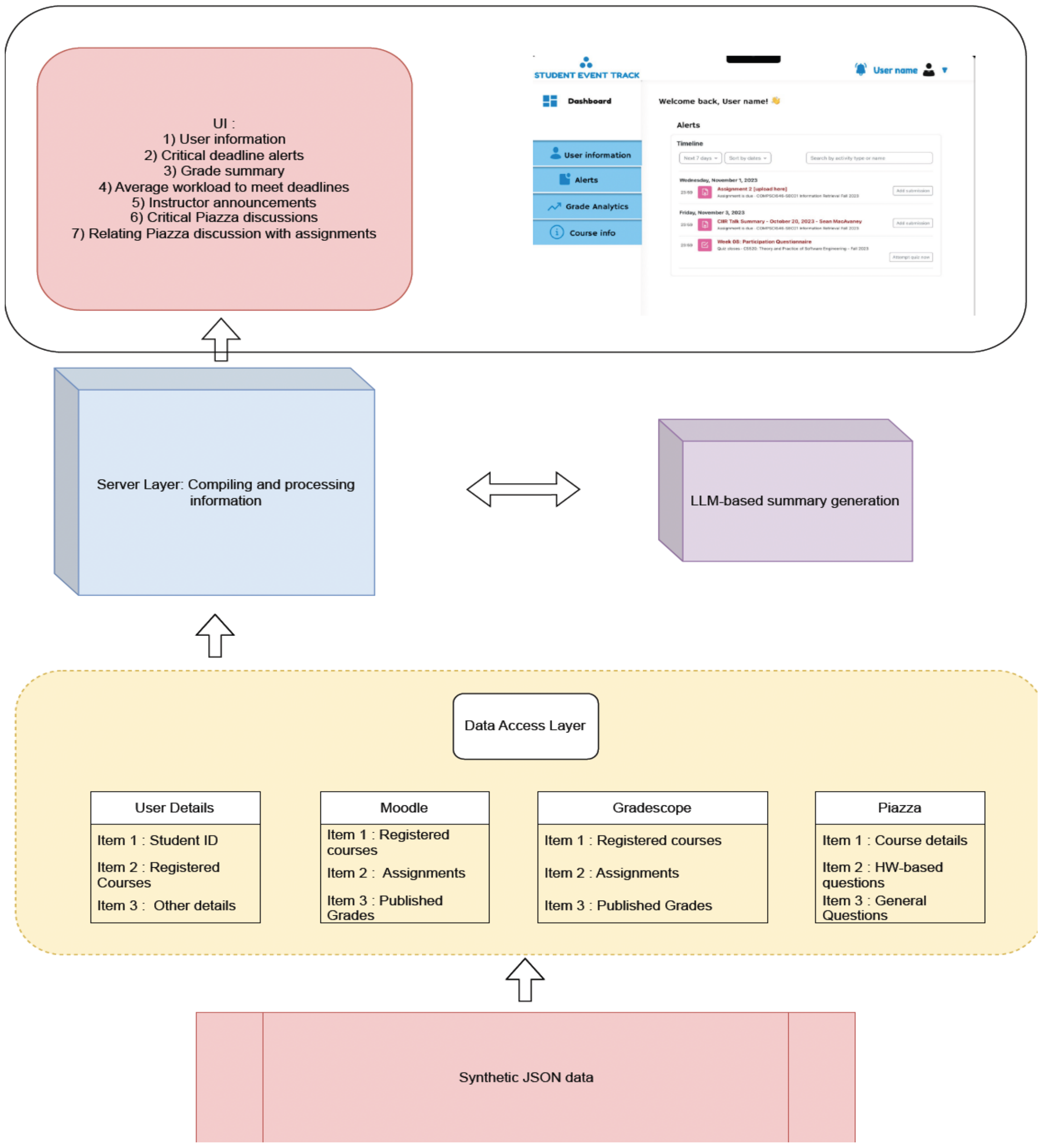
1.4. Non-Functional Requirements

- Security: Grades should remain private for each student.
- Usability: The UI should be easy to understand and digest.
- Performance: The system should be fast at responding to new grades and posts.

2. Design

2.1. Architecture Diagram

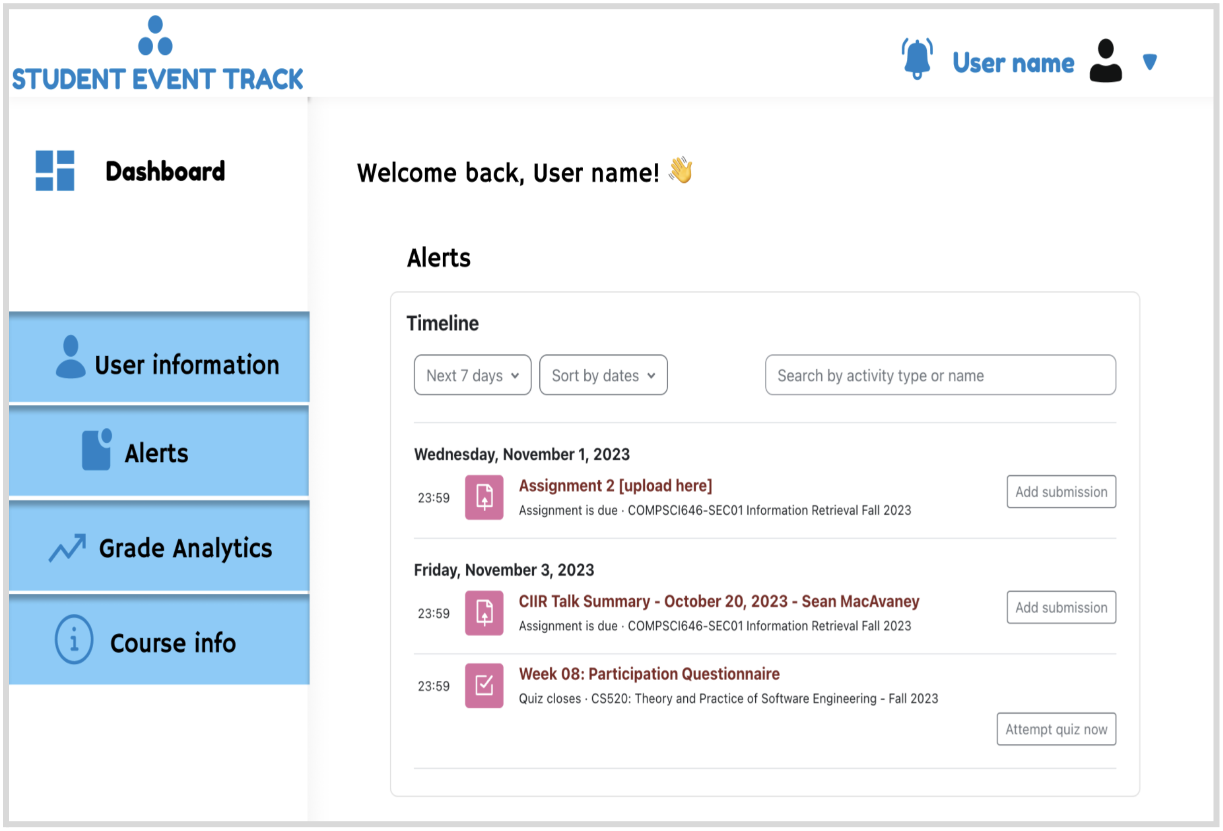
For our project, we are choosing to use two different languages based on their different strengths. For most of the project, including the UI, server, and database, we will use Java since it has strong object-oriented frameworks such as interfaces and abstract classes. However, for the LLM-based part of the project, we will use Python because of the powerful libraries available for deep learning, most notably Pytorch and Tensorflow.


UI :
1) User information
2) Critical deadline alerts
3) Grade summary
4) Average workload to meet deadlines
5) Instructor announcements
6) Critical Piazza discussions
7) Relating Piazza discussion with assignments
STUDENT EVENT TRACK
User name
Dashboard
User information
Alerts
Grade Analytics
Course info
Welcome back, User name!
Alerts
Timeline
Next 7 days
Sort by dates
Search by activity type or name
Wednesday, November 1, 2023
Assignment 2 [upload here]
Assignment is due · COMPSCI646-SEC01 Information Retrieval Fall 2023
Add submission
Friday, November 3, 2023
CIIR Talk Summary - October 20, 2023 - Sean MacAvaney
Assignment is due · COMPSCI646-SEC01 Information Retrieval Fall 2023
Add submission
Week 08: Participation Questionnaire
Quiz closes · CS520: Theory and Practice of Software Engineering - Fall 2023
Attempt quiz now
Server Layer: Compiling and processing information
LLM-based summary generation
Data Access Layer
User Details
Item 1 : Student ID
Item 2 : Registered Courses
Item 3 : Other details
Moodle
Item 1 : Registered courses
Item 2 : Assignments
Item 3 : Published Grades
Gradescope
Item 1 : Registered courses
Item 2 : Assignments
Item 3 : Published Grades
Piazza
Item 1 : Course details
Item 2 : HW-based questions
Item 3 : General Questions
Synthetic JSON data

2.2. Technology Stack with Justification

- **Figma:** We chose Figma for designing and developing the UI. Figma is simple for developing a prototype for the UI and can also be used to export the code.
- **LLM:** We are using Pytorch and pre-trained models in Python for the large language model section of the project.
- **Database:** We are writing the database in Java using Firebase.
- **Security:** We are using a built-in Java library for authorization and authentication within our service. Java already has many tools for our purpose, meaning we do not need to implement cryptographic functions ourselves. Right now, we are looking at using JAAS (Java Authentication and Authorization Service), but this choice is not yet finalized.
- **Server:** We will use basic Java classes and libraries in order to implement the primary logic of the server. Since we will already be dealing with Java objects from the backend database, it will be straightforward to use Java to manipulate these objects and process the information.

2.3. UI Mockup



Welcome back, User name! 👋

Alerts

Timeline

Wednesday, November 1, 2023

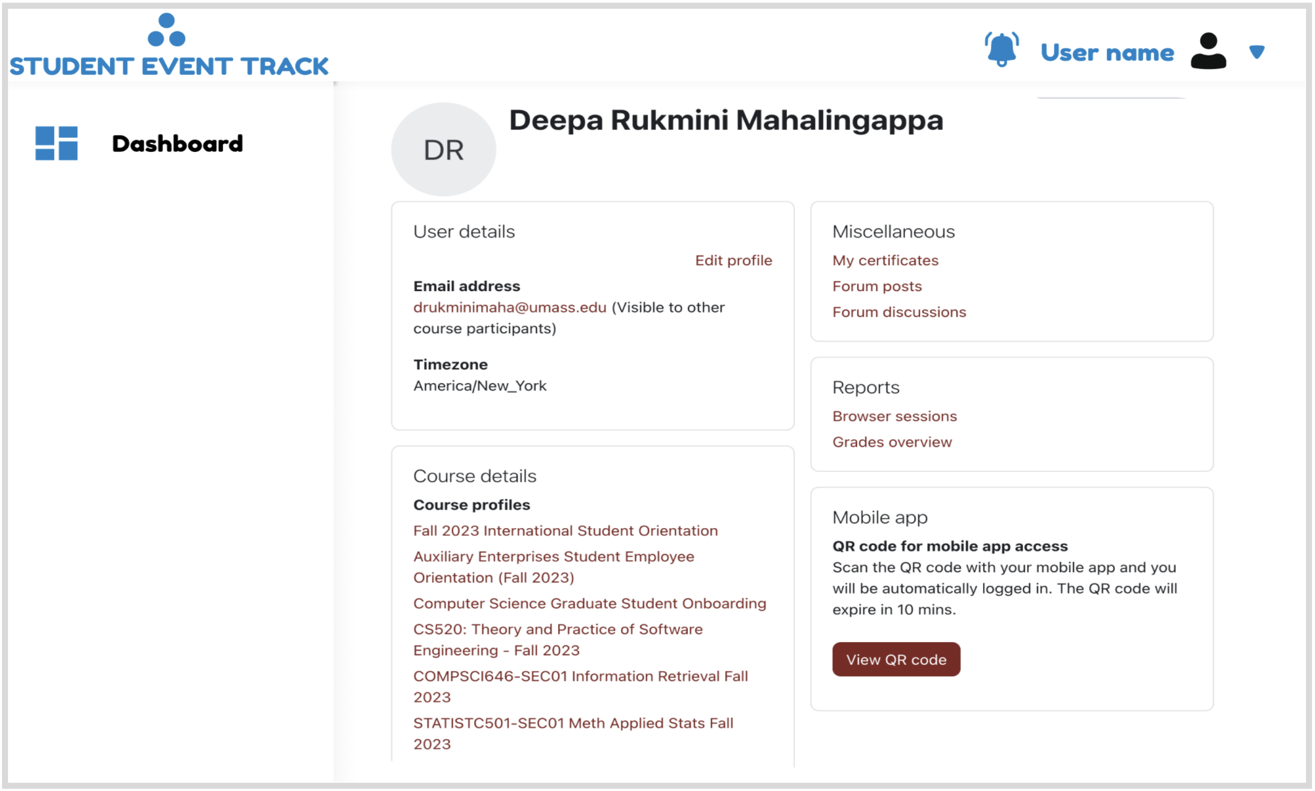
23:59 Assignment 2 [upload here]
Assignment is due · COMPSCI646-SEC01 Information Retrieval Fall 2023

Friday, November 3, 2023

23:59 CIIR Talk Summary - October 20, 2023 - Sean MacAvaney
Assignment is due · COMPSCI646-SEC01 Information Retrieval Fall 2023

23:59 Week 08: Participation Questionnaire
Quiz closes · CS520: Theory and Practice of Software Engineering - Fall 2023

UI-1 : ***Dashboard contains components such as User information, Alerts from Moodle, Piazza, and Gradescope, Grade analytics, and Course information.***



Deepa Rukmini Mahalingappa

User details

Edit profile

Email address
drukminimaha@umass.edu (Visible to other course participants)

Timezone
America/New_York

Course details

Course profiles
Fall 2023 International Student Orientation
Auxiliary Enterprises Student Employee Orientation (Fall 2023)
Computer Science Graduate Student Onboarding
CS520: Theory and Practice of Software Engineering - Fall 2023
COMPSCI646-SEC01 Information Retrieval Fall 2023
STATISTC501-SEC01 Meth Applied Stats Fall 2023

Miscellaneous
My certificates
Forum posts
Forum discussions

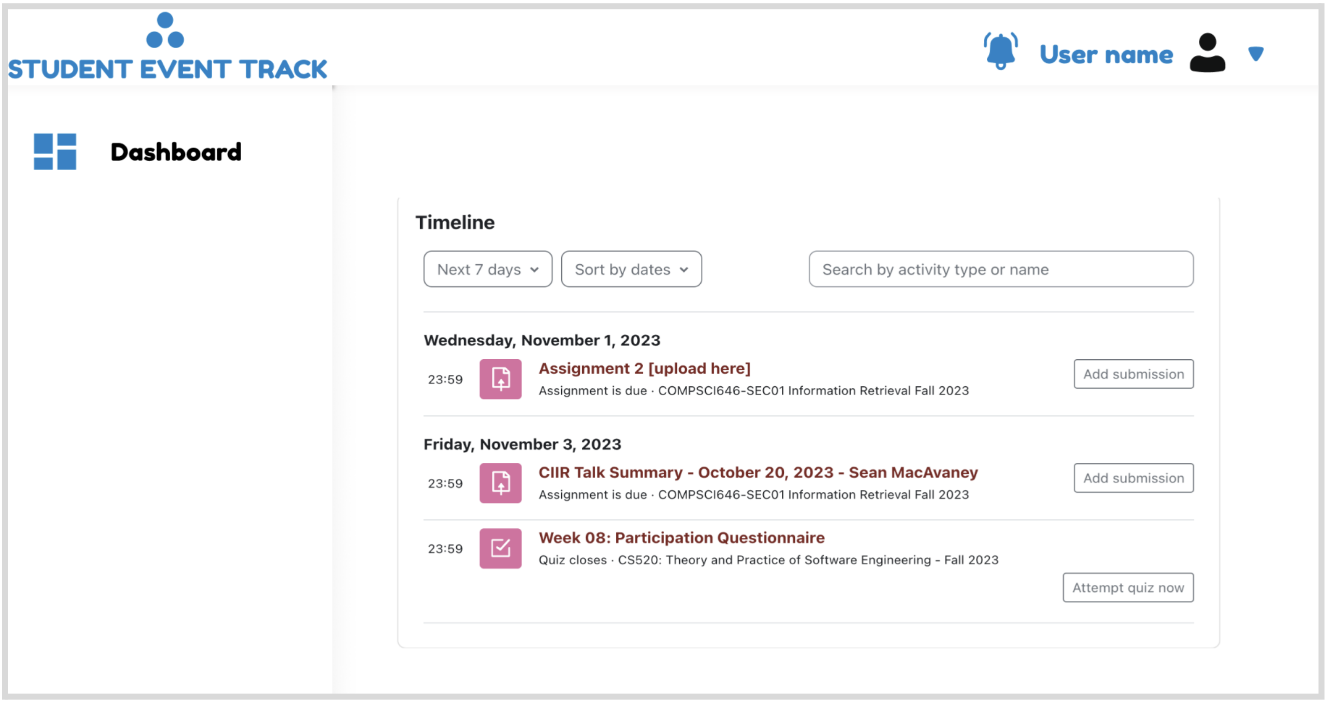
Reports
Browser sessions
Grades overview

Mobile app

QR code for mobile app access
Scan the QR code with your mobile app and you will be automatically logged in. The QR code will expire in 10 mins.

View QR code

UI-2 : ***User information displays User Information, Course details, and Grades overview, and posts and discussions as per Moodle.***



Timeline

Next 7 days | Sort by dates | Search by activity type or name

Wednesday, November 1, 2023

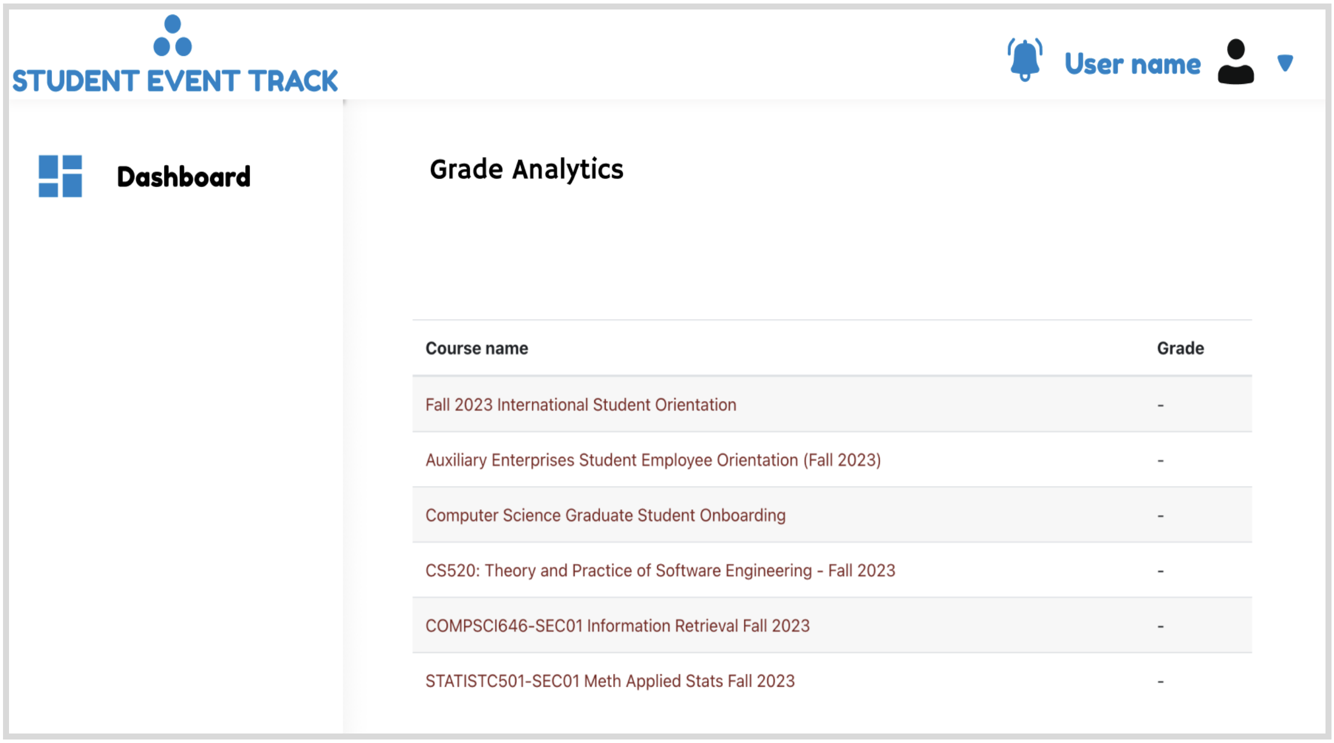
23:59 **Assignment 2 [upload here]**
Assignment is due · COMPSCI646-SEC01 Information Retrieval Fall 2023
Add submission

Friday, November 3, 2023

23:59 **CIIR Talk Summary - October 20, 2023 - Sean MacAvaney**
Assignment is due · COMPSCI646-SEC01 Information Retrieval Fall 2023
Add submission

23:59 **Week 08: Participation Questionnaire**
Quiz closes · CS520: Theory and Practice of Software Engineering - Fall 2023
Attempt quiz now

UI-3 : ***Alerts pop-ups the notifications to mark attendance, take quizzes, submit assignments and exams in the given timeline.***


STUDENT EVENT TRACK

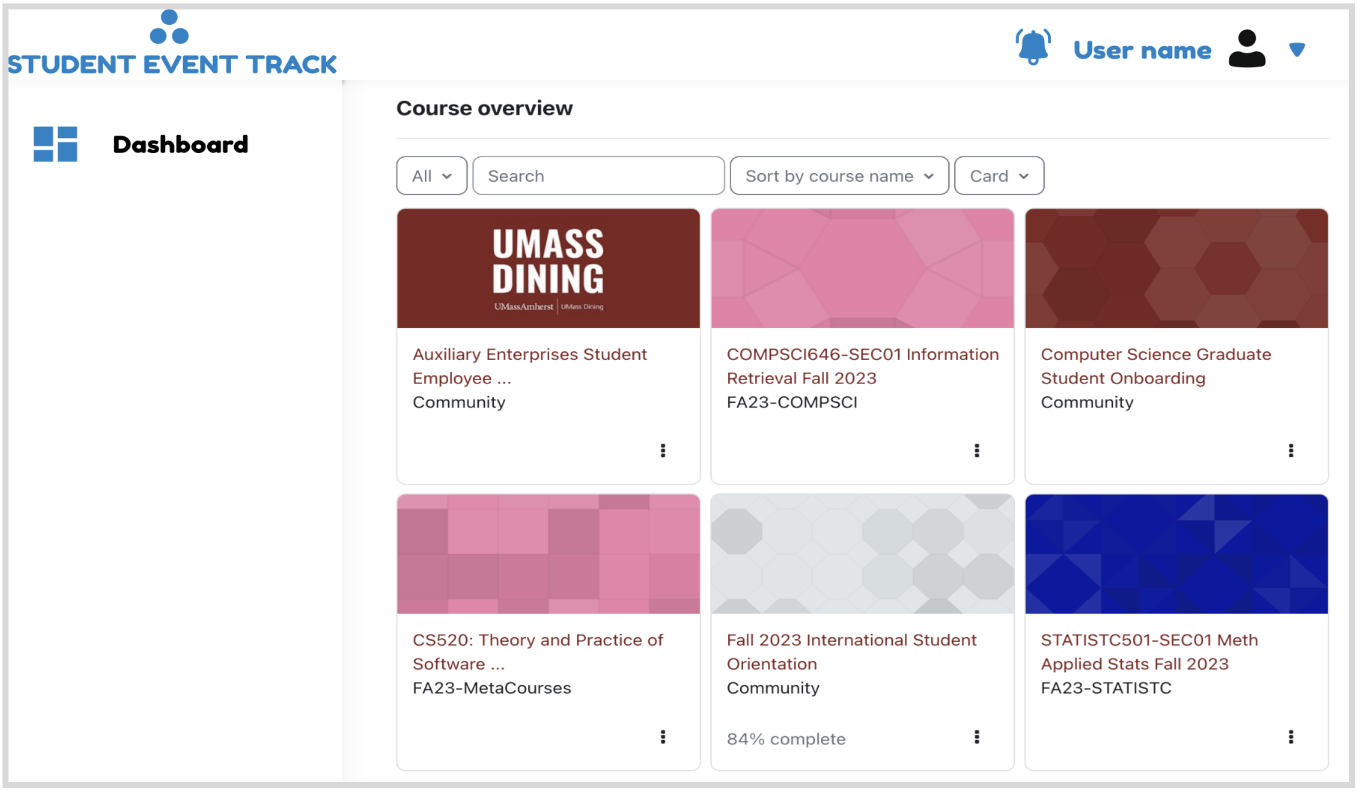
User name

Dashboard

Grade Analytics

Course name	Grade
Fall 2023 International Student Orientation	-
Auxiliary Enterprises Student Employee Orientation (Fall 2023)	-
Computer Science Graduate Student Onboarding	-
CS520: Theory and Practice of Software Engineering - Fall 2023	-
COMPSCI646-SEC01 Information Retrieval Fall 2023	-
STATISTC501-SEC01 Meth Applied Stats Fall 2023	-

UI-4 : *Grade analytics displays the grades of individual courses taken as per Gradescorpe and Moodle.*


STUDENT EVENT TRACK

User name

Dashboard

Course overview

Auxiliary Enterprises Student Employee ...
Community

COMPSCI646-SEC01 Information Retrieval Fall 2023
FA23-COMPSCI

Computer Science Graduate Student Onboarding
Community

CS520: Theory and Practice of Software ...
FA23-MetaCourses

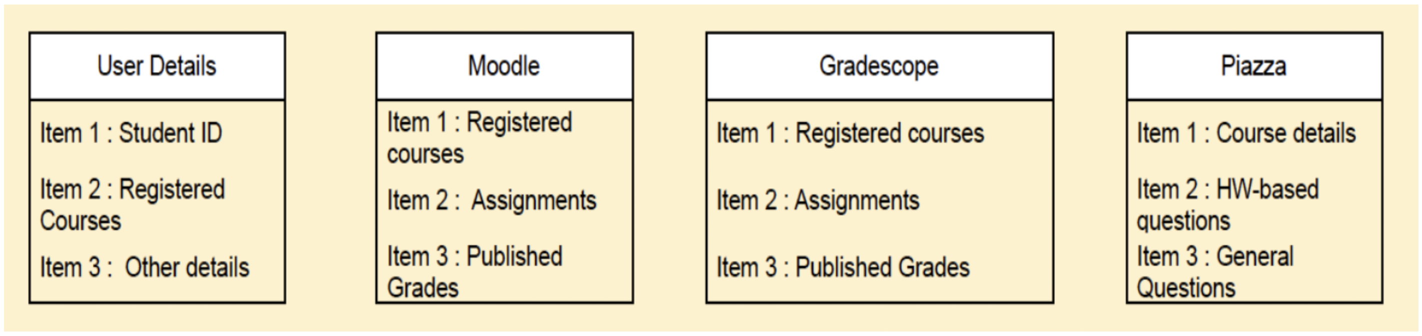
Fall 2023 International Student Orientation
Community
84% complete

STATISTC501-SEC01 Meth Applied Stats Fall 2023
FA23-STATISTC

UI-5 : Course information displays all the courses each user has taken in their respective semester.

2.4. Data Model

Database Schema diagram for Student Event Tracking system :


User Details
Item 1 : Student ID
Item 2 : Registered Courses
Item 3 : Other details
Moodle
Item 1 : Registered courses
Item 2 : Assignments
Item 3 : Published Grades
Gradescope
Item 1 : Registered courses
Item 2 : Assignments
Item 3 : Published Grades
Piazza
Item 1 : Course details
Item 2 : HW-based questions
Item 3 : General Questions

Entity Relationship(ER) diagram for Student Event Tracking system (Many - Many relationship)

- User table:
    - Student ID (PK)
    - Registered Courses (FK)
    - First Name
    - Last Name
    - Date of Birth (DOB)
    - Email ID
- Moodle
    - Registered courses : Course 1, Course 2, Course 3 (Foreign Key to Course) (1: N relationship)
    - Assignments : Deadlines, Names, and Descriptions 1 to many
    - Published Grades :S , A, B or C
    - Announcements : Assignments, projects, quizzes
- Gradescope
    - Registered Courses: Course 1, Course 2, Course 3 (Foreign Key to Course)
    - Assignments : Course Name, Status, Course Released, Course due deadline
    - Published Grades : Marks
- Piazza
    - Posts: poster, contents, follow ups, etc
    - Course details : Course 1, Course 2, Course 3 (Foreign Key to Course)
    - HW based questions : Live Q & A, Instructor post, lecture
    - General Questions: Events, logistics, participation.

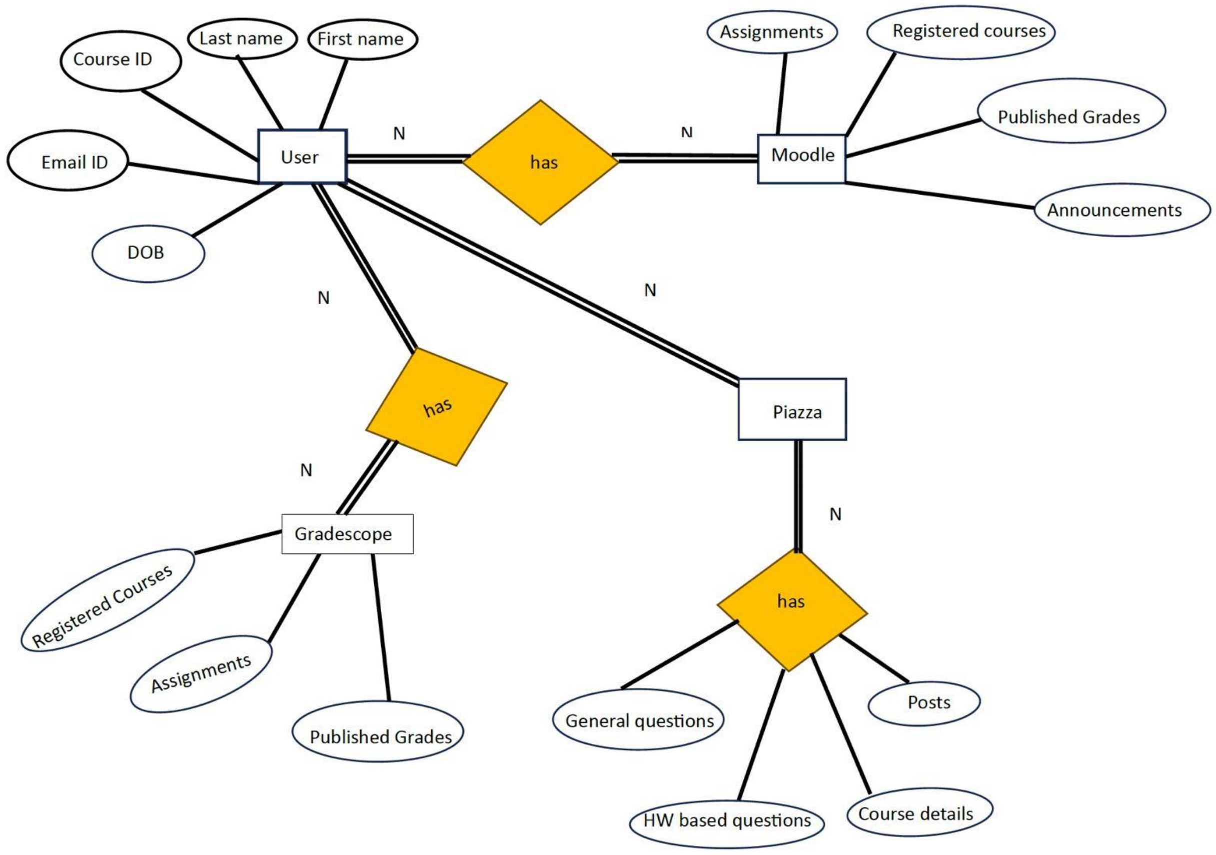


Figure: The ER diagram for Student Event Tracking System (Many to Many)

3. Implementation

For our project, we intend to use version controls with git and split the project into a few distinct parts. By making our development process modular, we hope to make our program more debuggable and easy to expand. Below, we outline some of the considerations and processes we will follow for implementing each part of the project.

3.1. Security and Risks

- **Risks:** We anticipate that our risk analysis mostly revolves around the level of sensitivity to the information we are working with. Since users are all students with their own set of courses, grades, and schedules, we want to restrict each user to only being able to see their own information while restricting access to the information of others.

    - Sensitive data: Personal information like course schedules and student IDs, as well as any grades, should stay private to only the owner. Also, private piazza posts should only be readable by the student who posted it.
    - One other risk we need to consider when using LLMs in our project is ensuring that our information isn't leaked to the public when we use open source software.
- **Prevention:** We want our system to use both authentication and authorization to ensure that the user's identity can be verified and that they are only granted access to information that they are permitted to see.
    - Authentication: We will implement a username/password based login in order to authenticate users. While we haven't finalized the exact implementation yet, we will not store the password directly, but rather a hash of the password to prevent other users from reading the raw authentication data to obtain the information of other users.
    - Encryption: We are also planning to implement basic encryption in order to ensure users are not able to read the information of other students without proper authorization.

3.2. LLM Implementation

- ➢ The reason for having LLM in this project is because the data we are looking at is from discussion and announcement forums such as piazza and Moodle . The data is mostly in the form of text (Natural language) and
- ➢ We are planning to sue either a API provided by the Open Ai or a LM model that can be downloaded and used in a local machine
- ➢ Using LLM we can Collate the information from different fields in json and make a meaningful summary
- ➢ We are limiting the usage of LLM only for summary and collation of the information because it demands a lot of work in basic implementation and integration

3.3. Database implementation

- ➢ We are using Firebase as a database that stores data in JSON format, which allows to synchronize data in real time for each user.
- ➢ To implement a database, firstly need to create and setup Firebase project, then import libraries and configure them. Firebase's real-time database is a JSON tree, we need to construct a structure that meets the data requirements of your application.

➢ Real-time data synchronization is provided by Firebase. So any changes are reflected immediately.

3.4. UI implementation

<u>Tool</u> - Figma

1. We used Figma, which is a cloud-based design and prototyping tool used in User interface (UI) and User experience (UX) design.
2. We opted for Figma due to its collaborative and real-time design capabilities, making it a versatile tool for designing web and mobile applications, websites, and more.

<u>User Interface(UI) Components</u>

❖ Dashboard contains components such as User information, Alerts from Moodle, Piazza, and Gradescope, Grade analytics, and Course information.

❖ User information displays User information, Course details, and Grades overview, and posts and discussions as per Moodle.

❖ Alerts pop up with notifications to mark attendance, take quizzes, submit assignments, and take exams within the given timeline.

❖ Grade analytics displays the grades of individual courses taken as per Gradescorpe and Moodle.

❖ Course information displays all the courses each user has taken in their respective semester.

3.5. Server implementation

We will break down the server implementation into a few parts, which we may add to or update over time.

● **Login:** This part of the server will deal with user login, authentication, and authorization. Additional details can be found in Section 3.1.

● **Piazza analyzer:** This section of the server will read through all the piazza posts for courses in which the student is enrolled and sort them based on their prominence (assignment deadline, number of upvotes, etc.) to send to the frontend for display.

● **Deadline alerter:** The deadline alerter will be tasked with compiling information from Moodle and Gradescope about assignments and sorting them based on time-sensitivity.

● **Grade summary:** This part of the server will simply compile grades across moodle and gradescope in order to provide a short summary to the user for each class they are enrolled in.

3.6. Generating synthetic data

- For each type of data (user information, Moodle course, gradescope grades, piazza post, etc.), we will design and outline a fixed scheme to store information as JSON. Depending on the type of data, the fields in the JSON may vary.
- For instance, a Piazza post may be stored using the fields {Course name: String, Visibility: String, Date posted: int, Poster name: String, Topic: String, Post title: String, Post body: String, Num upvotes: int, Views: int, Thread number: int}.
- For some fields, such as course name and visibility, the possible values may be fixed to a small set, such as "private" or "public." We may add additional JSON files that enumerate the possible values for the fixed fields.
- Generating data via LLM: We will synthetically generate three courses and three students enrolled in some subset of the courses, where each course has its own assignments, announcements, and grades.
- We will first create the student data and basic course data by hand, using made-up names and courses, as we want to avoid exposing any personal data for our mock prototype and testing.
- Next, using ChatGPT or some other publicly available language model, we will generate further information about the course and data which accompanies it, such as piazza posts, assignment information, announcements, etc.
- For course grades, we will generate scores randomly for each student based on a normal distribution. We will likely assign each student a mean and variance for their scores per class and sample accordingly.

4. Work Plan

4.1 Task Breakdown

Below is a rough outline of the members in charge of leading each part of the project, but throughout our development process, we expect members to review, test, and contribute to each other's work.

- Karthik: LLM + Server
- Deepika: Database implementation and design
- Deepa: UI

- Aaron: Security + synthetic data

4.2 Timeline

- Week of 10/30: Finish design
- Week of 11/06: Generate synthetic data, implement UI, begin implementation of database, LLM
- Week of 11/13: Finish implementation of database, LLM, link with server
- Week of 11/20: Refine server, finish implementation of functionality, add security components
- Week of 11/27: Testing
- Week of 12/04: Testing