EE 547 – Applied and Cloud Computing

Final Report

Divya Nandlal Sahetya, Akshata Jahagirdar, Krishna Dheeraj Krovi

December 14, 2022

Project Title: My Personalized Dashboard

Summary and Description

In this project, we have built a web application for students to check their day-to-day schedules, read emails, see their calendar events, see their academic progress and resources with a single click. All these applications shown are mutable, and each user will be able to personalize their dashboard with features they would like to have. This application is targeted to be used by Students and Teachers. Currently, we are providing integration with 3 applications namely, Gmail, Google Calendar, and Academics. Checking incoming emails and calendar events is same for all the users. However, in terms of academic feature, the student will be able to view his/her grades, the Professor will be able to modify and view each students' grades. With the help of this project, the user will be able to handle multiple tasks that usually require opening many applications at one place.

1 Architecture

Our project consists of a front-end web application, a back-end web server, and a MongoDB database as source of academic data that can be modified or viewed through interaction with the front-end application. Since, our front-end and back-end application run on different ports, we have also used CORS to enable cross domain requests.

The back-end server is implemented in Node.js with express as middleware and GraphQL server will be used for the API request handler. Apollo serves as a fully-featured GraphQL client for the backend server. We have also included Bootstrap to make the application more responsive. The users interact with the application through the front-end client (web application) by logging in with Google OAuth2.0. We also store the access token, and refresh token(needed to generate the token if expiration date has passed) in local cached session storage.

- Web pages:

  Dashboard The home landing page displays the student's personalized dashboard containing recent emails, academic details, and a calendar of events. This will look different for teacher as well as student.

  Login This page provides an option for the user to select if he is a student or teacher, and log in using Google Authentication.

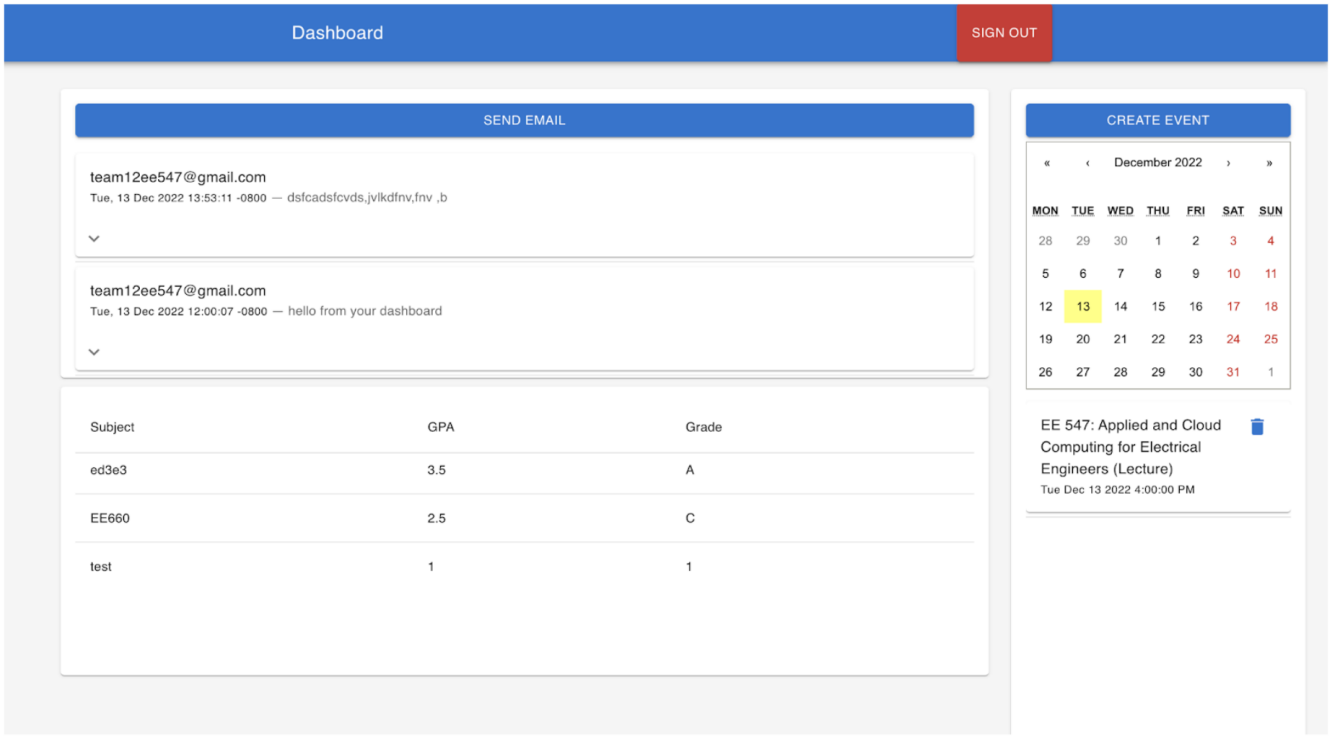

Figure 1: Dashboard

The goal of this project is to create a personal dashboard that can be customized to include various widgets, some of which include Google Calendar, Gmail, and academic records of students. There will be two levels of access, namely, student and teacher levels.The teacher would have read and write access to records in the database of all the students in the class. A student would have access pertaining to the user's record with read access only.

We used a NoSQL database like MongoDB to store academic data since it does pretty well for hierarchical data. The Gmail and Google Calendar functionalities are implemented using the corresponding APIs from Google. We have deployed the application in Google App Engine after testing it locally to make it more scalable. As the future scope of this project expands, this can be further extended to include more widgets, such as Google Photos, Google Tasks, and Spotify music.

2 Technologies

- ExpressJS and Node.js:

  Node.js is an open source and cross-platform runtime environment for executing JavaScript code outside of a browser. Express is a small framework that sits on top of Node.js's web server functionality to simplify its APIs and add helpful new features. It makes it easier to organize your application's functionality with middleware and routing. It adds helpful utilities to Node.js's HTTP objects.

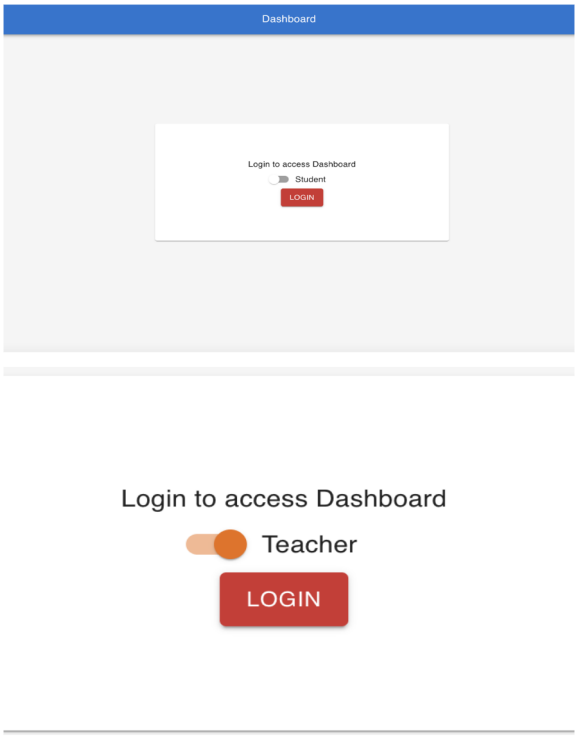

Figure 2: Login Page

We have used Node.js in our project to implement backend server as it has lot of supporting npm packages readily available to use. This helped us in speedy development in the project. We have also used Express.js as the middleware, it helped us in handling API requests from the Google Emails and Calendar API.

- ReactJS:

  ReactJS is a declarative, efficient, and flexible JavaScript library for building reusable UI components. It is an open-source, component-based front end library that is responsible only for the view layer of the application. We have used reactjs to render individual UI pages at a time, thus speeding up development.

- Bootstrap :

  Bootstrap is one of the most popular HTML, CSS, and JavaScript framework for developing responsive, mobile first projects on the web. We have used bootstrap to wrap our HTML, CSS and JS objects in our project. We have used reactjs to render individual UI pages at a time, thus speeding up development.

- GraphQL:

  GraphQL is an open-source data query and manipulation language for APIs, and a runtime for fulfilling queries with existing data. We have used GraphQL as a API request handler from our database storing academic records.

- Apollo:

Apollo is a fully-featured, production ready caching GraphQL client for every UI framework and GraphQL server. We have used this for get and POST GraphQl queries across the server.

- MongoDB:

  MongoDB is a document database with the scalability and flexibility that you want and the querying and indexing that you need. MongoDB uses JSON-like documents with optional schemas. This is very useful for storing hierarchical data. Due to this reason, we have used MongoDb for QUERYING academic records pertaining to students as well as MUTATING the same in the case of professors.

- Google Cloud Platform:

  Google Cloud Platform is a suite of cloud computing services that runs on the same infrastructure that Google uses internally for its end-user products. Since, we are hitting Google APIs as our backend, we had to create Google Cloud account and generate auth credentials for us. Therefore, we decided to use Google Cloud as our deployment platform. We have used Google Cloud App Engine to deploy our app and Atlas to deploy MongoDb.

3 Application Flow

Figure 4 below shows the route from one page to another in our application. The user selects the toggle button according to his or her role, and then signs in with Google. On successful login, the dashboard will ask for permissions related to Gmail, and Calendar (since data from these apps will be fetched later). The access token for authorizing these Google APIs is stored locally in session storage. After every hour, a new refresh token is generated and stored so that the user remains logged in. After successful login, The dashboard will show recently received emails, and recent calendar events and also a view of grades per subject enrolled in. User will also have an option to send emails, or create calendar events using a relevant popup form. All these options are available with a single click! If the user has the role of a professor, he/she will also be able to modify the scores of each student. User can sign out of the application once his work is done. This functionality is implemented by simply removing the local session-storage files.

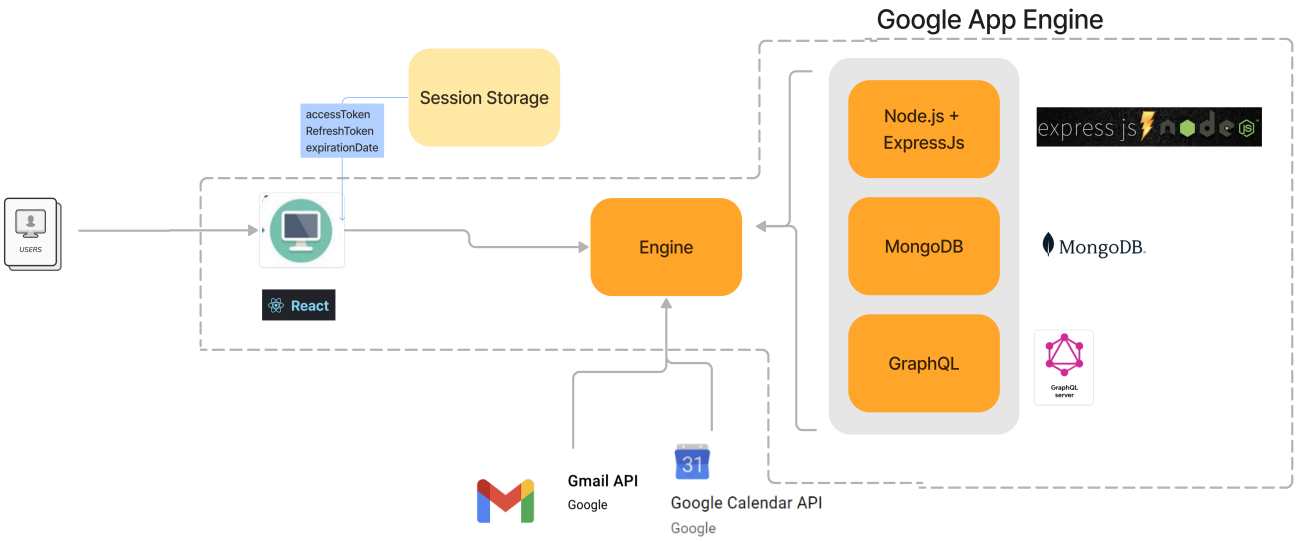

Figure 3: Data flow diagram

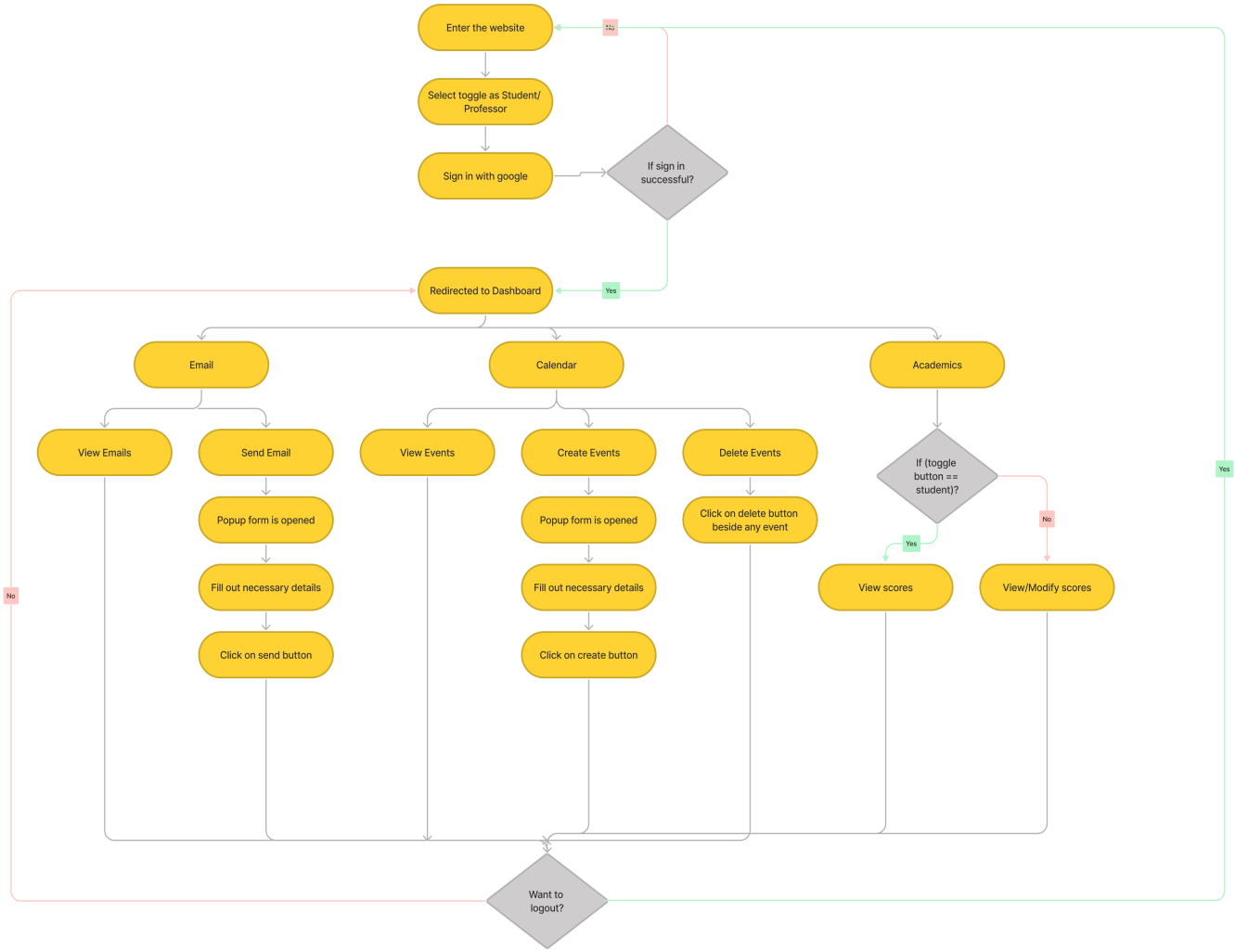

Figure 4: High-level flowchart of dashboard application

USER INTERACTION

- Login Page:
  - The user will select the toggle as "Student/Professor":
    - Sign in with Google.
- Dashboard:
  - Toggle as Student:
    - The user can view his/her academic records.
    - The user can view his/her recent emails and send a email using an inline form on the dashboard.
    - The user can view his/her calendar events for a particular day and create an event using a button click.
  - Toggle as Professor:
    - The user can view and modify respective students' academic records.
    - The user can view his/her recent emails and send a email using an inline form on the dashboard.
    - The user can view his/her calendar events for a particular day and create an event using a button click.
- Sign out:
  - The user will be able to log out of the application.

3.1 DATABASE SCHEMA:

Table 1: Schema

personUpdateInput	Type
fname	String
lname	String
role	roleEnum
gpa	Float
personCreateInput	Type
fname	String
lname	String
role	roleEnum
gpa	Float
emailid	String
Person	Tyoe
fname	String
lname	String
id	ID
role	roleEnum
gpa	Float
emailid	String
gradebookUpdateInput	Type
subject	String
grade	String
emailid	String
gpa	Float
gradebookCreateInput	Type
subject	String
grade	String
emailid	String
gpa	Float
gradebook	Type
emailid	ID
subject	roleEnum
gpa	Float
emailid	String
Subject	Type
code	String
name	String
Instructor	String

4 APIs Used

For Gmail and Google Calendar, we follow RESTful API architecture style, since we will fetching data from Google APIs. For Academic details, we will be fetching data from MongoDb databse using GraphQL. Below figure represent the usage of gmail, google calendar and graphql respectively.

- Gmail: There are 2 major operations in gmail: sending emails and viewing recent ones. For sending, we directly use a POST call with the destination address, subject and message as input. For viewing messages as a list, we first fetch list of Ids of recent emails, and for each id, we get the email details.

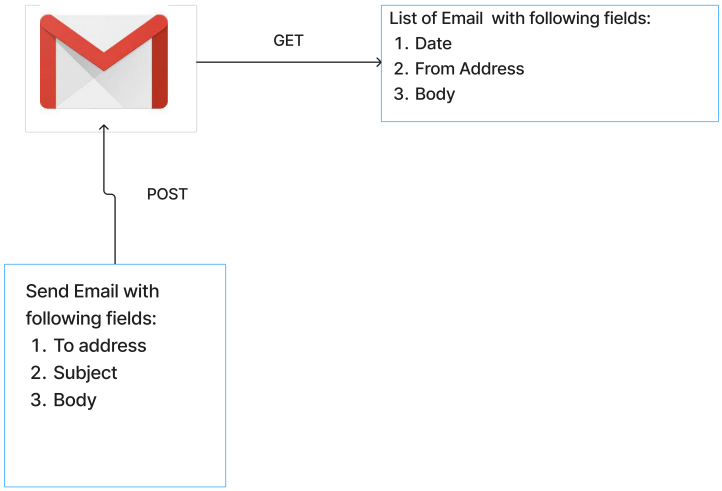

Figure 5: APIS From Gmail Service

- Google Calendar: There are 3 major operations in Google calendar: listing the existing events, creating one based on user inputs, and deleting the selected event. These are achieved with following google Calendar APIs used as shown in the figure.

- GraphQL: There is only one URL handler in graphql that is taking all post and get requests. The schema of the handlers is included in the appendix. It has three major operations. One of it is fetching the gradebook of a particular student that logged in and the other is fetching grades of the entire class when an instructor is logged in. The instructor has a special option to add or modify grades to entire class so there are mutations to perform creation and updation of gradebook. This project can be extended further for which additional mutations and queries are present in it. If a new student has to be entered into the records, the mutation personCreate would handle it. Similarly there are other mutations, queries written and shown in the Table 1.

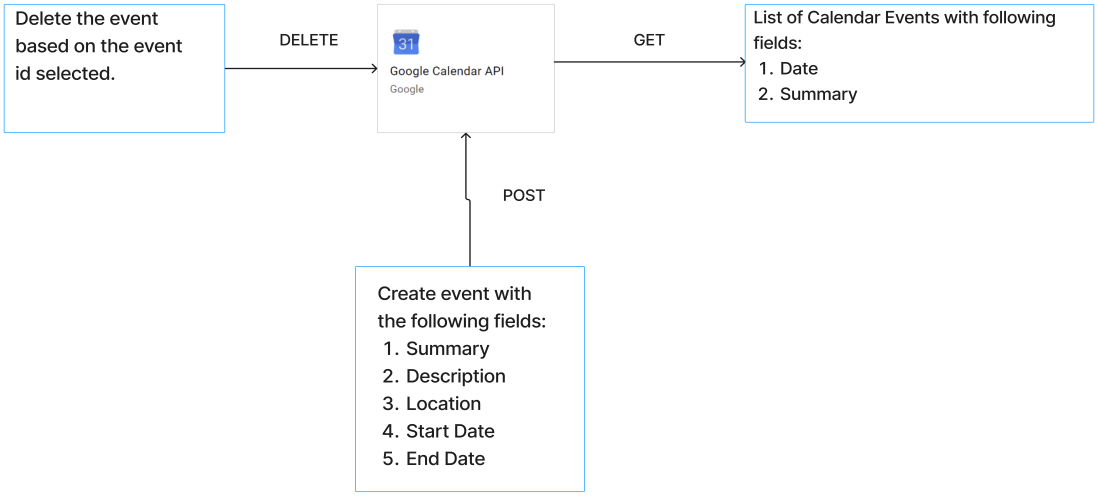

Figure 6: APIs from Google calendar service

5 Datasources

- Google Calendar: The Calendar API enables the user to integrate the application with Google Calendar. Some of the methods of this API we have used include:
  - CalendarList:get, list and insert - https://developers.google.com/calendar/api/v3/reference/events These perform tasks on the user's calendar such as getting the events using the get method and creating the event using insert method.
  - Calendar:get, list and insert - https://developers.google.com/calendar/api/v3/reference/events Similar to the calendarList, we have methods to create and fetch the events.
- Gmail: The data for the gmails reside on the server side of Google and are accessed or entered using Rest API calls. The Gmail API enables the user to integrate the application with Gmail. Some of the methods of this API used include:
  - Gmail: get, list: - https://developers.google.com/gmail/api/reference/rest/v1/users.messages These perform tasks on the user's email list such as getting the list using the get and list method.
  - Gmail:send - https://developers.google.com/gmail/api/reference/rest/v1/users.messages/send Above method is used to send a email.
- Atlas MongoDB: For the academics data, MongoDB by Atlas is chosen and resides on the cloud. MongoDB has unmatched performance compared to S3, Azure and GCP. The collections in the Database include
  - **person** It has all details about all the people in the database such as students, teachers. The information includes subjects taken by them or taught by them.
  - **gradebook** It includes the grade of each student.
  - **subject** It has more details about the subject.

6 Implementation details

6.1 User Login

exNodejs with Google client OAuth package are used to handle user authentication. The user unique access token and refresh token is stored in **session storage**. The refresh token can be used to fetch a new access token on expiry for the same user. Once a user selects the role and logins using google sign in, the user is provided to the dashboard application as displayed in Figure 7.

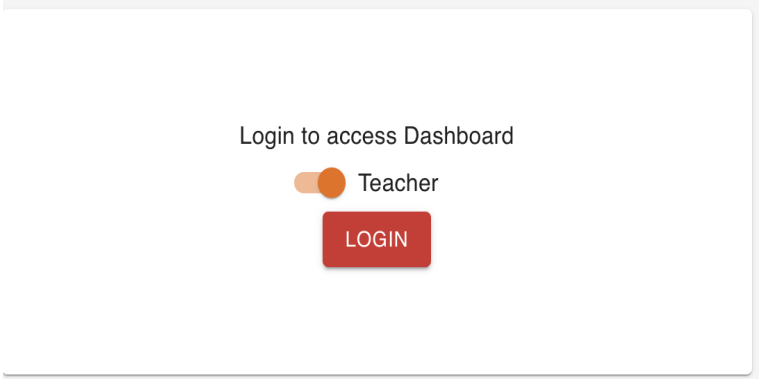

Figure 7: Login

6.2 Dashboard

The Dashboard is developed using **React and Material UI package** is used to develop the front end view of the application.

The Dashboard displays the different widgets available for the user to use. CRUD functionality has been implemented for each one of them which are available to the user based on the type of the role.

The widgets that are currently available are Google Mail, Google Calendar, and student academic records.

For example, when a user logins as a student, the user will only be able to view the subject grades but in case of the teacher role, one can enter the details of each students records can be entered and can view all the students records.

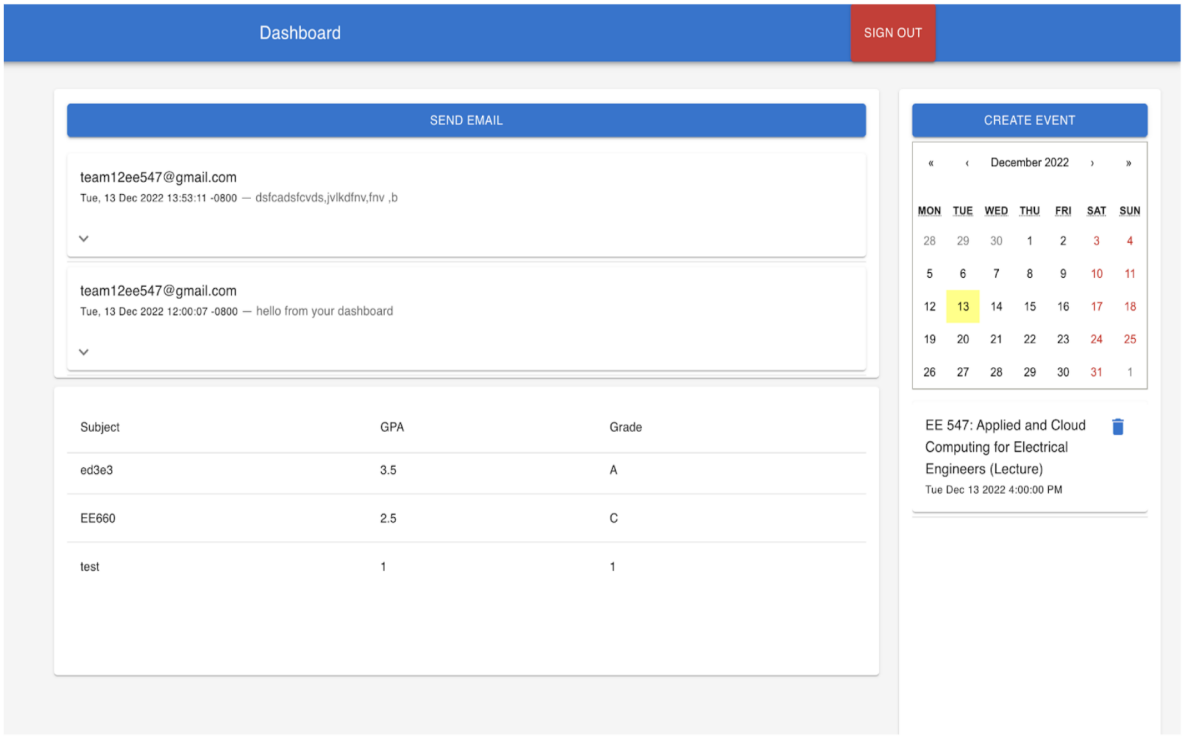

Figure 8: dashboard application

6.2.1 Teacher view

Once a user logins as a teacher, the role is saved in the session storage for that user associated account. The teacher has access to the calendar widget (list events, create events, delete events), email widget (list emails, send email), enter each student's academic record and view the previous records.

Material UI package API is used to design the view and specifics related to display.

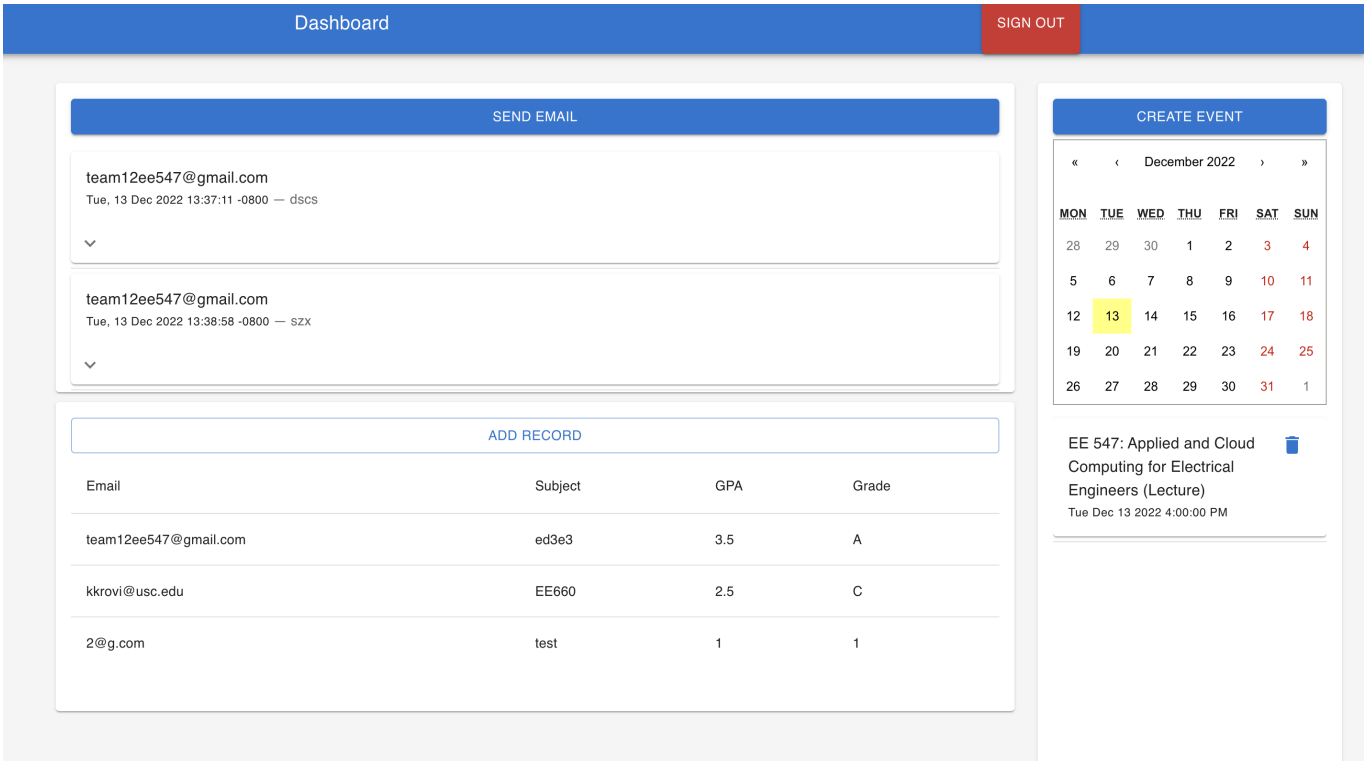

Figure 9: Teacher dashboard view

6.2.2 Student view

Once a user logins as a student, the role is saved in the session storage for that user associated account. The teacher has access to the calendar widget (list events, create events, delete events), email widget (list emails, send email) and view their gradebook.

Material UI package API is used to design the view and specifics related to display.

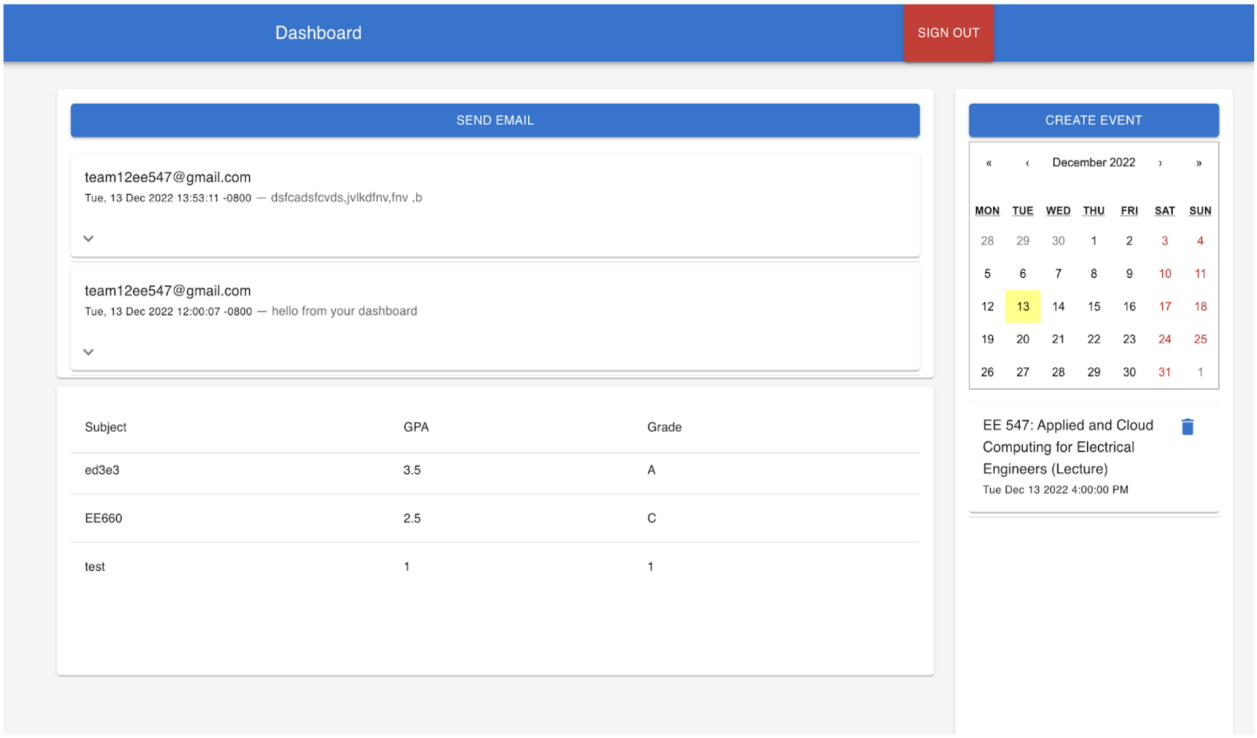

Figure 10: Student dashboard view

6.3 Calendar

The Calendar widget is implemented using the Google Calendar API using REST (GET, POST, DELETE) operations in NodeJS (backend) and React (frontend). The user can view the calendar in month view. The user the can view the list of events, create a new event and delete a event.

6.4 Email

The Email widget is implemented using the Google Mail API using REST (GET, POST) operations in NodeJS (backend) and React (frontend). The user can view the emails in a paper view and a dropdown to see the entire message when the number of characters are more than 75 characters. The user the can view the list of emails, and can send a email directly from this dashboard.

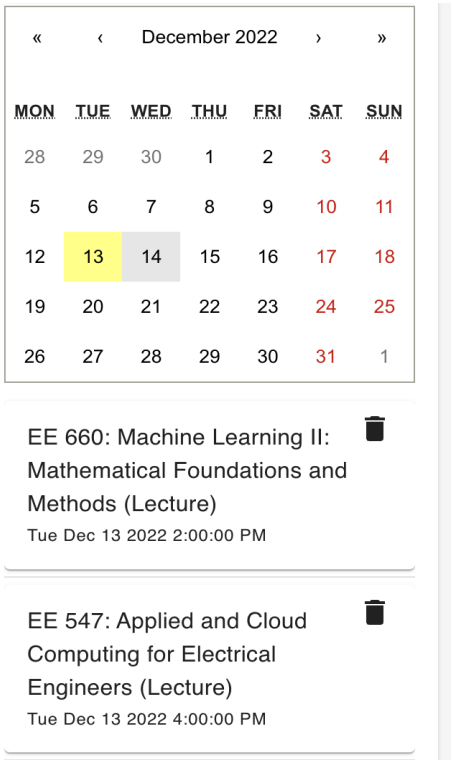

Figure 11: Calendar

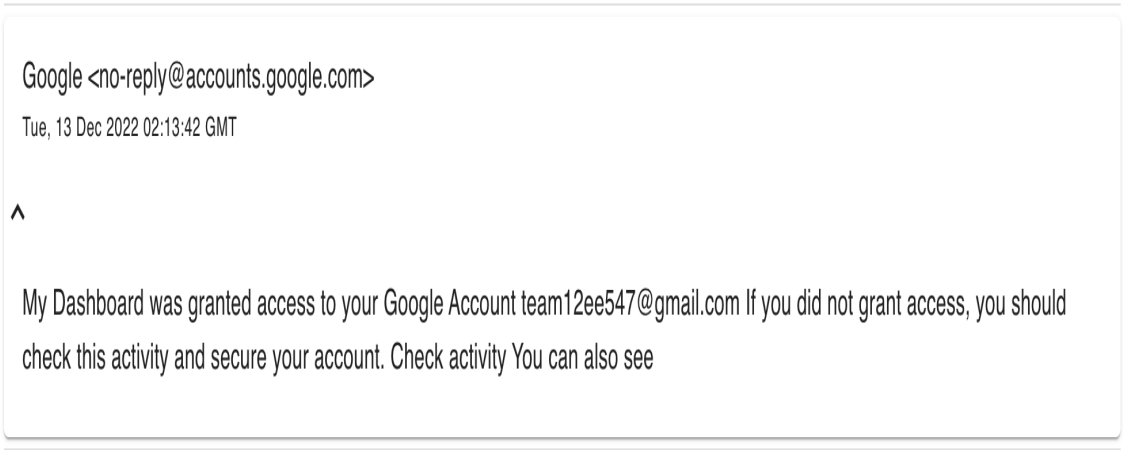

Figure 12: Calendar

7 Future Scope/ Planned

- Verification of the toggle value before login

  This idea can be extended into the future scope where we can verify if the toggle value is same as the permissions of the user. This will ensure that a user won't be able to proceed by selecting an unintended role.

- Integration of more applications into the dashboard

  Currently, the dashboard has 3 applications integrated with it, gmail, google calendar and academics. Later on, this application can be extended to include other apps, such as Spotify, Google Tasks, Google Photos, etc.

8 References, Tutorials, Codebases, Documentation, and Libraries

- NPM
- ExpressJS
- Gmail API
- Google Calendar API
- Google Calendar Node JS Tutorial
- React Tutorial