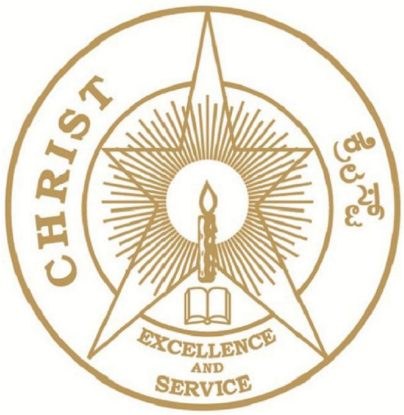


CHRIST

(DEEMED TO BE UNIVERSITY)

BANGALORE · INDIA

PROJECT TECHNOBOARD

Dehar Nath Bhattacharya 1940211

Ronald Wilson 1940224

Bhavya Kukkar 1940261

Guided by

Dr. Monisha Singh

Department of Computer Science

Project TechnoBoard

Introduction

Imagine conducting online classes, where different classes are supposed to be conducted on different platforms, back to back classes, from 9 in the morning, to 4 in the evening, all of which need to be scheduled, and sent to the attendees.

This process of scheduling the classes, sending the details to the attendees, switching between platforms, etc. can be brought down to one particular application, which will serve as a dashboard, accumulating every other resource required to thrive through conducting classes (online and offline), making the entire process sustainable for the user.

The application can be used to complete all of the tasks (that one might have to take care of in order to conduct classes more efficiently).

For example, a particular user might have to announce an upcoming class. And after letting the attendees know about the class, the user will be required to make the link for the class, on any particular platform. Following this, the user will need to send the invite link to all the attendees. During the class, the user will need to take attendance of the attendees, and might want to conduct some quiz for the attendees. After the class is over, the user might wish to upload an assignment for the students to work on. And then the entire process all over again, for a different set of attendees, with a different set of specifications.

So, here, the application will provide the user with an option to make an announcement in platforms like Google Classroom, or via Google Mail, with all the participant list, or the particular classroom details prepared beforehand. Next, when the user will need to create a meeting link, he/she will have meeting platforms like Google Meet, Cisco Webex, Microsoft Teams, etc. one click away in the application itself. All the user needs to do is choose the platform and confirm his/her action. And right after this, they can forward the invite to the participants in a simple move, just like the announcement was made. The platform will have an

option for the user to take/record the attendance of the attendees in one go, without using any other external software/extension, or without taking in the attendance manually. Now, during the class, the user can access other platforms like Kahoot (for example) to engage the attendees in any quiz session, or any other activity, which can be accessed directly from the application itself. After the class/meeting is done, the user has the option to schedule an assignment for the attendees, through platforms like Google Meet, but by using the same application alone. As simple as that. The application will also facilitate pre-scheduling recurring events like weekly classes.

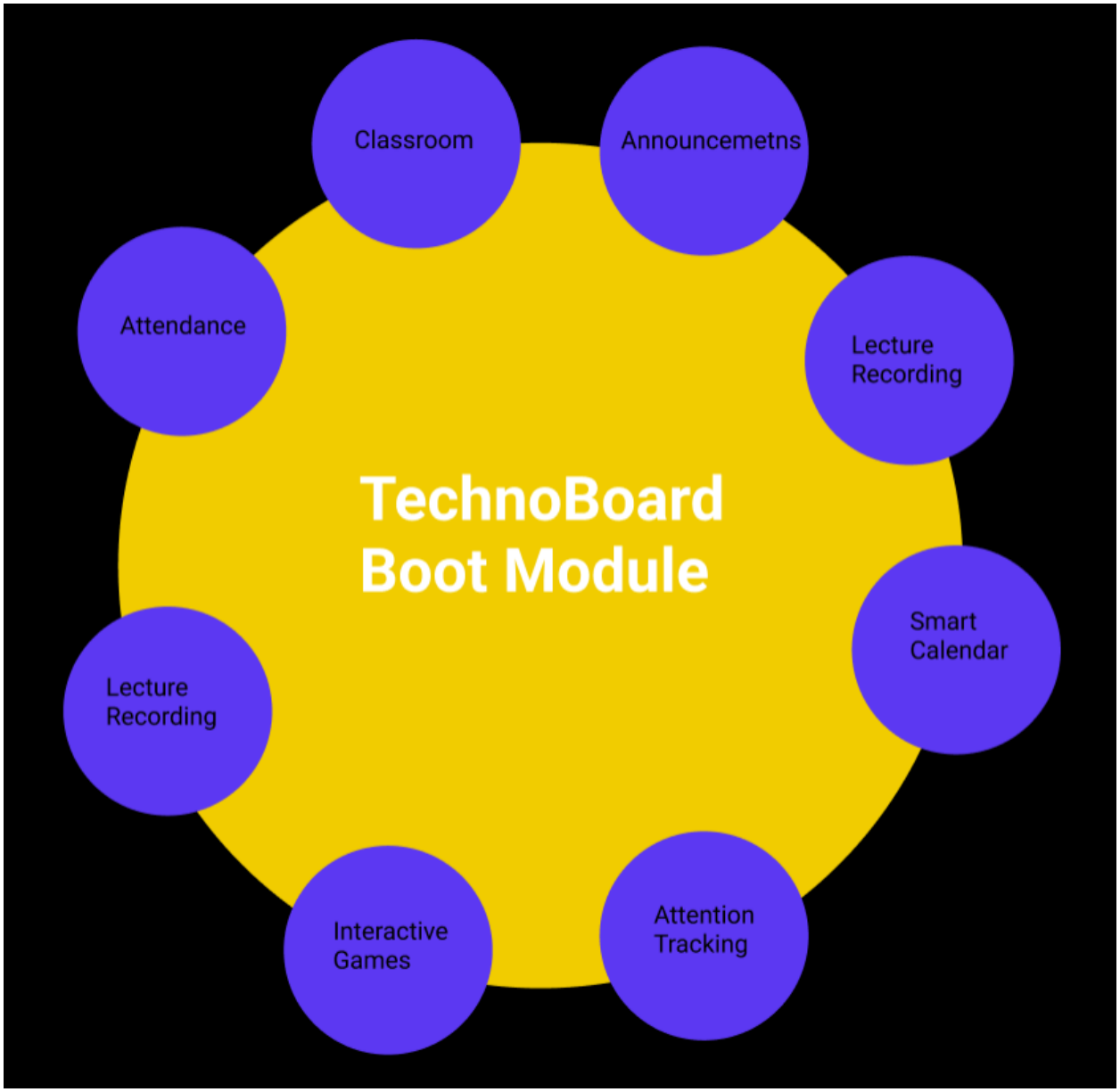
This structure has the capacity to reduce the burden of maintenance of different platforms/screens/windows/tabs by providing one master key to govern all of them. And since this master key has the potential to be tailor-built according to the user's choices, it is going to be even more convenient than the existing system.

Statement of Goals: -

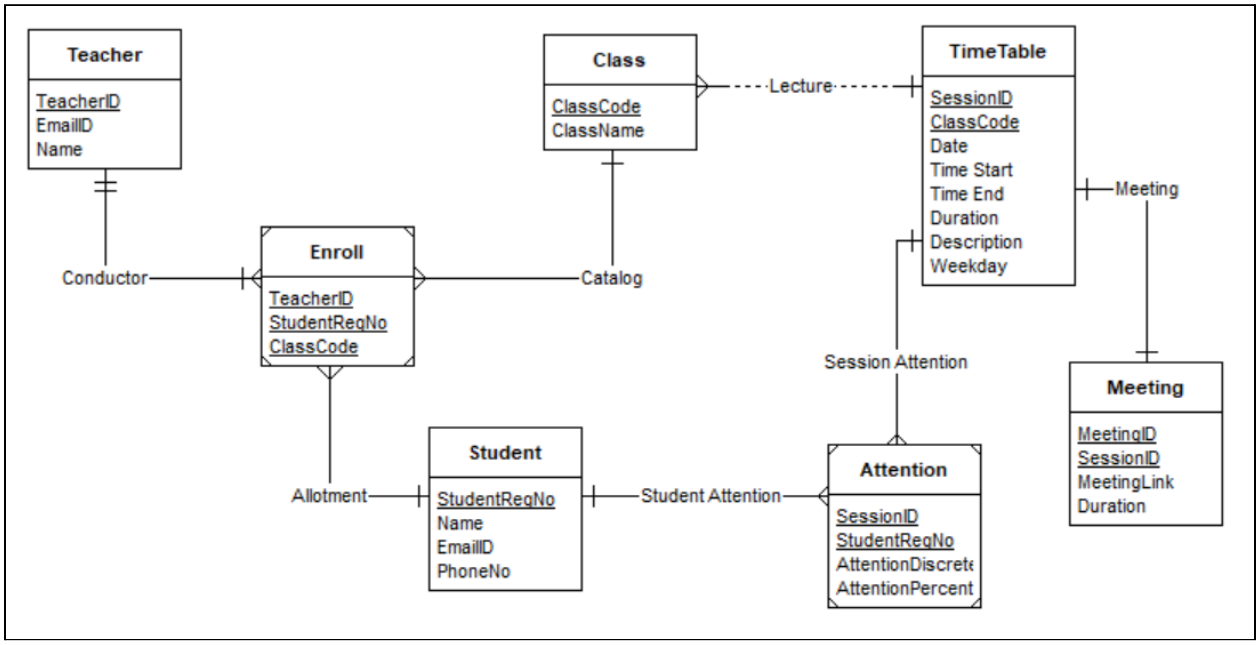
1. To enable easier conducting of online sessions for the hosts
2. To enable easier attending of online sessions for attendees
3. To increase productivity for both the kinds of participants, the hosts and the attendees
4. To increase online activity of attendees
5. To help in the creation of more engaging sessions and hence, increase the quality of online sessions
6. To ensure smoother reminders and updates for upcoming sessions, hence, making a technologically sound system
7. To help with attendance monitoring and better administration
8. To ensure better interoperability for the host and the attendees
9. To increase the speed of operation, and not at a great cost either

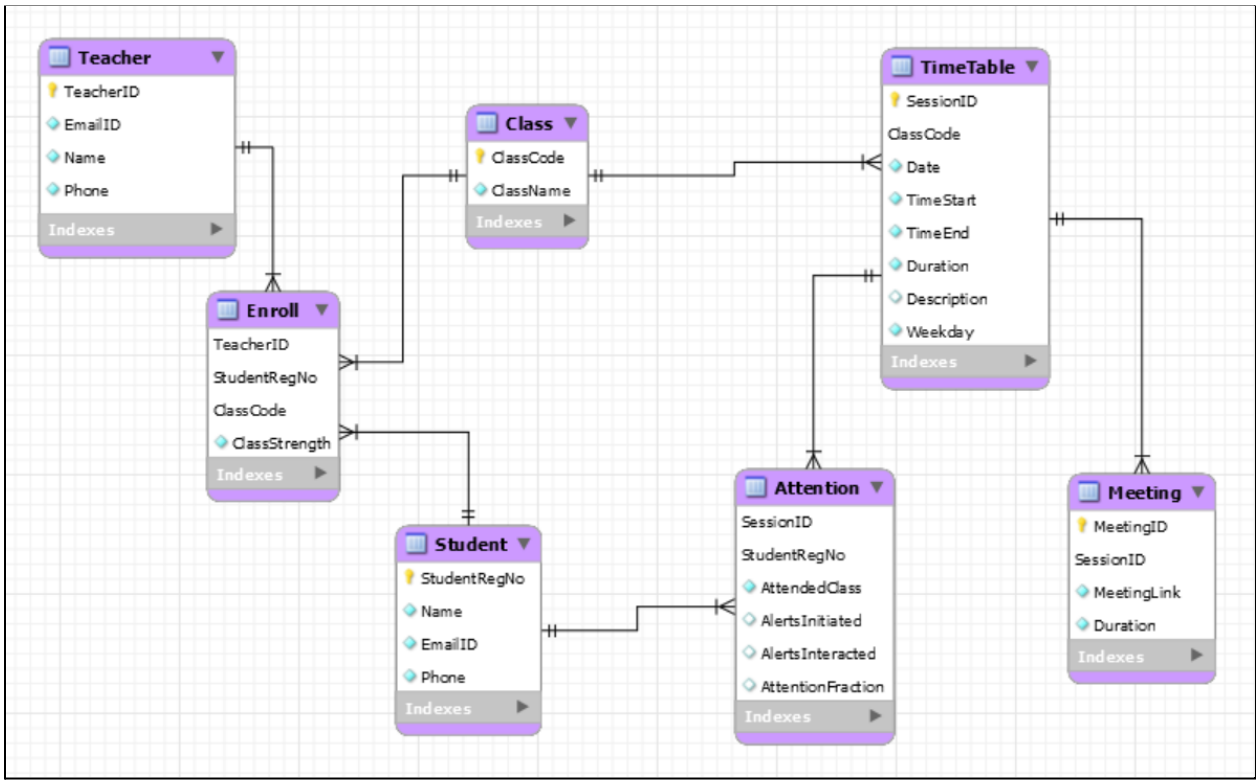
System Models: -

Block diagram for the system



DFD / ER Diagram



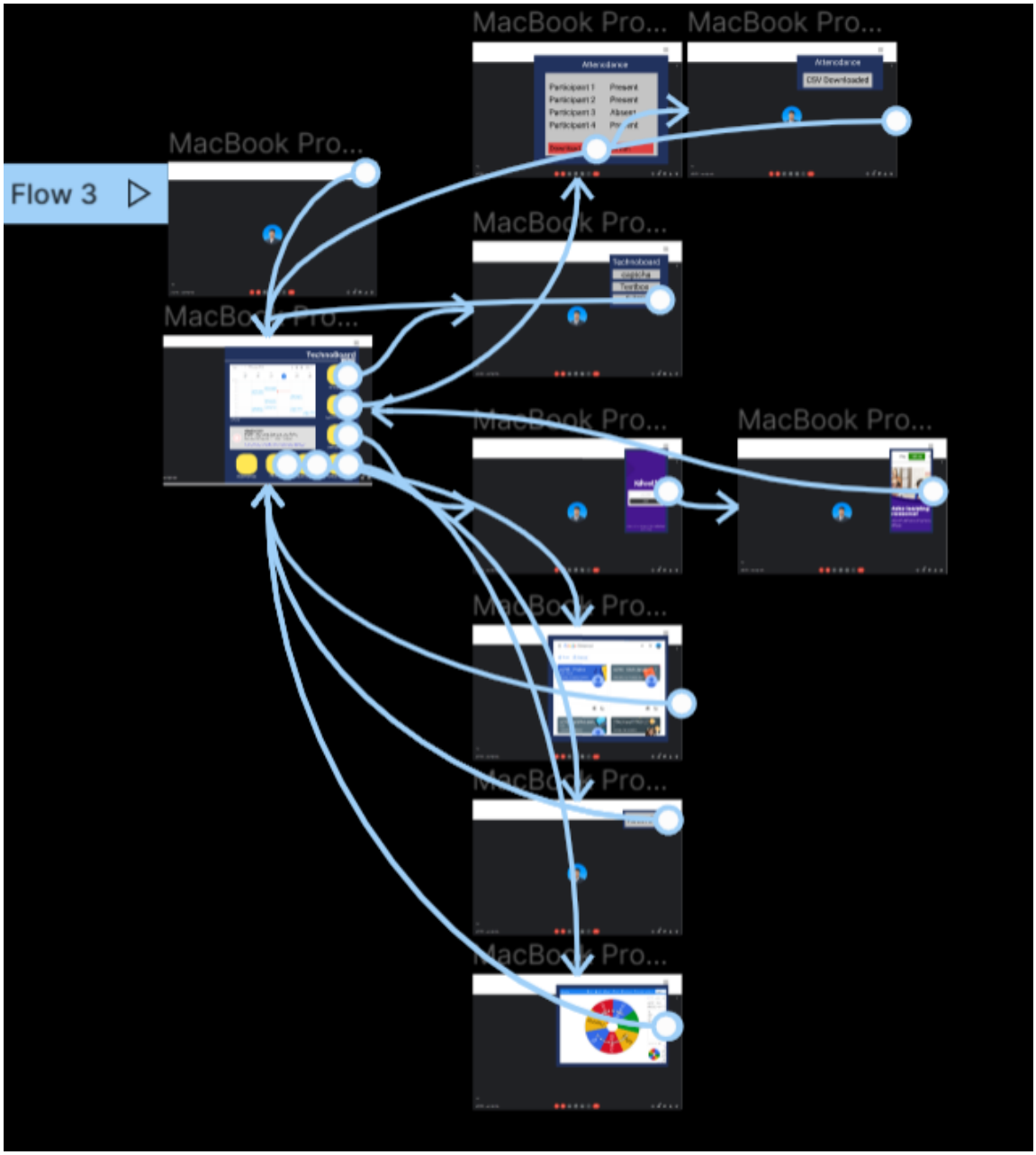


Database Design:

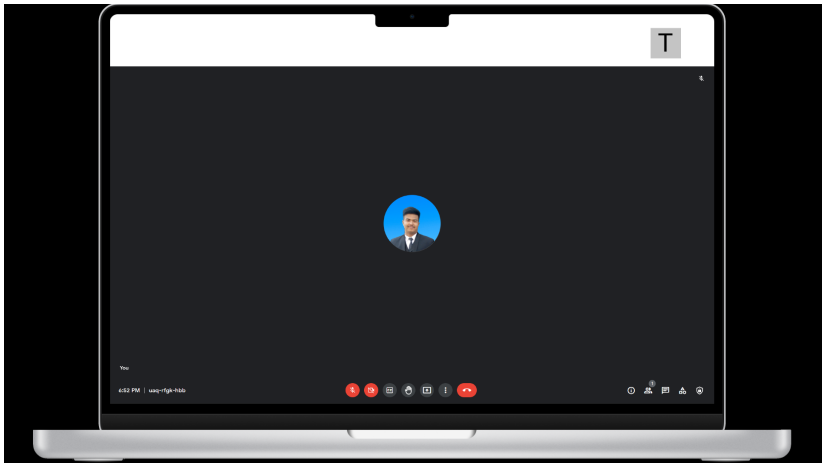
Database Tables must have Primary keys and foreign, keys information with normalized forms.

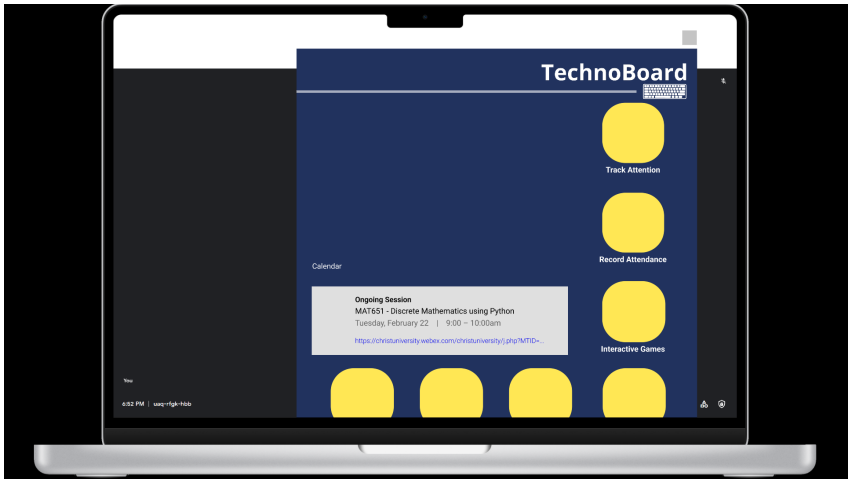
Data Dictionary – {Organized listing of all data tables with fields that are pertinent to the system with clear definitions to get better understanding of inputs, outputs, components and intermediate calculations. Check conditions & default values should be used if required}

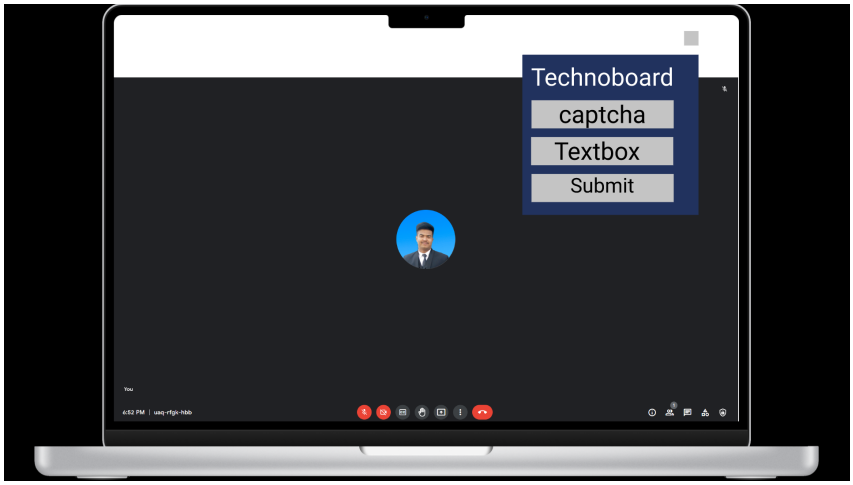
User Interface Design Interface between the subsystems/modules

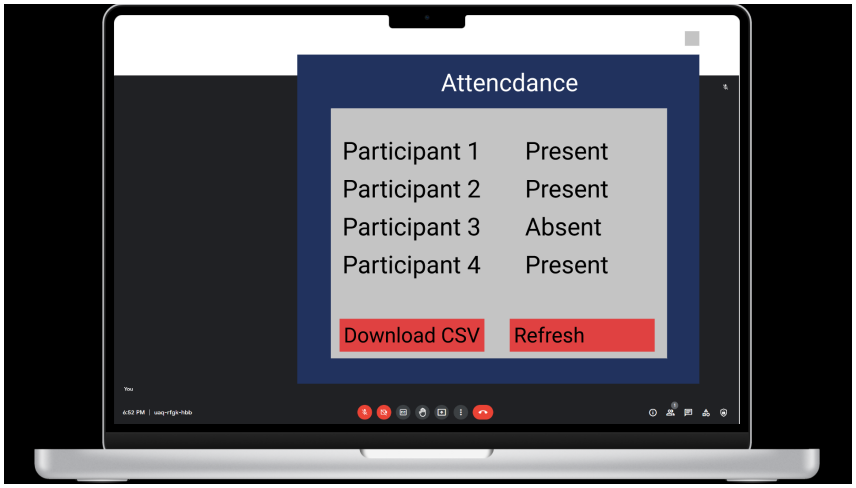


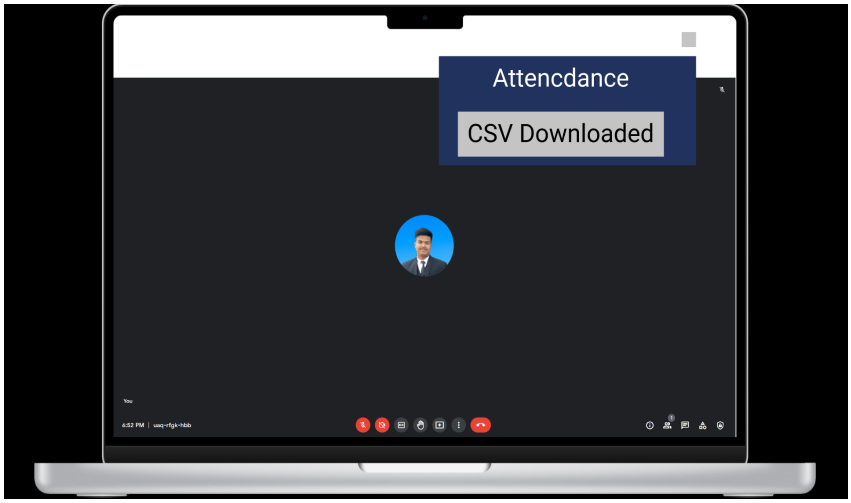
Interface Design : between user and system

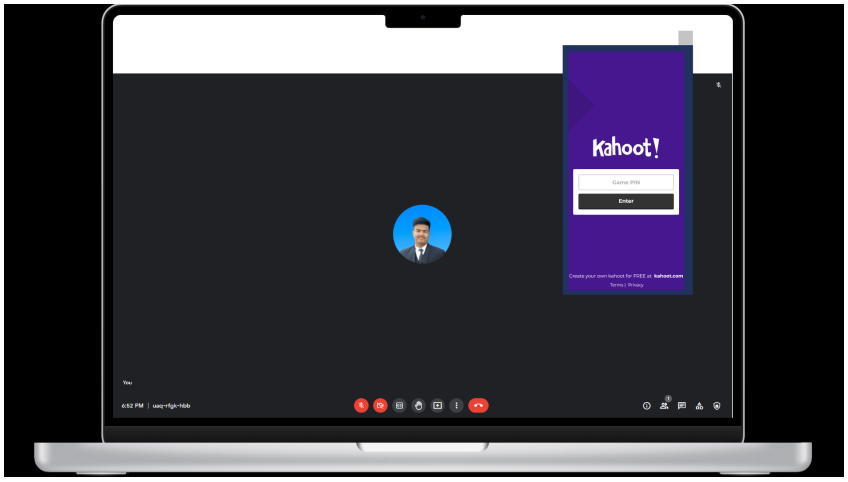


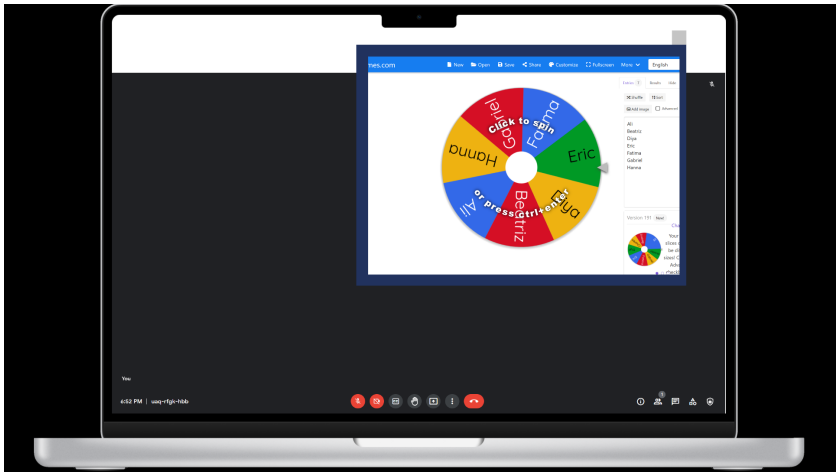


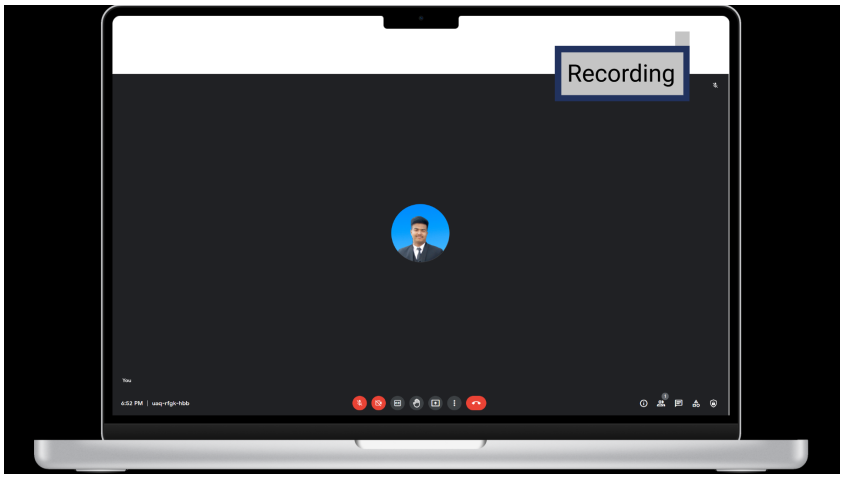



Attencdance
Participant 1 Present
Participant 2 Present
Participant 3 Absent
Participant 4 Present
Download CSV
Refresh


Attencdance
CSV Downloaded


Kahoot!
Game PIN
Enter





Functionality:

Functionalities

The application is able to perform many operations which is able to enhance the experience of the user and provides productive usage of time. The application is able to do the following of the tasks:

1. **Smart Calendar:** The Smart Calendar component of the application resides as a standalone software class internal to the application, and intuitively uses the memory of past meetings created by the teacher to predict the times at which the teacher is expected to create or join a meeting in the future. Based on the abundance or mereness of memory criteria, specifications of a certain number of meetings from the present into the past are recorded and analyzed , on the variables meeting date , meeting start time , meeting end time , meeting frequency (if specified, else weekly) , meeting participants amount.

2. Attention Tracking System: The attention tracking system tracks the attentiveness of the users and lets the meeting host know if the users are attentive or not. Based on the attention of the whole meeting the system will recommend the host to do little fun activities like playing a game on Kahoot or other interactive games.

3. Automatic Attendance Recording: This feature allows the host to maintain attendance record of the attendees as well as maintains the record of attention span of the attendees.

4. Automatic Lecture recording and upload: This feature allows the host to automatically record the lecture and upload the recorded lectures to a central server using which the attendees can use this to access the previous lectures.

5. Interactive Games feature: During the meeting the user will be able to play games with other attendees. This allows the host and the attendees to have a much more interactive and fun learning experience.

6. Announcements options through Whatsapp, SMS, etc: Powered by an interactive interface to manage APIs, announcements can be made via SMS/Whatsapp//Automated Calls and more.

7. Classroom Integration: The interface will be able to develop an integration between classrooms and google meet, simplifying the work of the host to

Sequence of Execution

To install the extension the user can either install the extension from the chrome extension store or download the extension from github and load the extension using the standard process of installing packed extensions.

Installation

Once the extension is installed the user can use the extension by creating a google meet and then working on it and using the various features of the extension

Specify Algorithms

Smart Calendar

The Smart Calendar can be defined as an encapsulated class with an extensive complex of arrays and a method to recommend/assign through fuzzy logic an expected session at any time.

```
class SmartCalendar {
	File calendarHistoryFile;
	char[][] thisWeek;
	int weekLength = 7;

	SmartCalendar(File calendarHistory, Date currentDate) {
		this.calendarHistory = calendarHistory;
		importHistory();
	}

	void currentSession(Date currentDate, Time currentTime) {
		int n = importHistoryLength();
		int fuzzyIndex = 0;

		int k = 3.0f; //Positive Probability Multiplier
		int c = 3.0f; //Negative Probability Multiplier
		int sn = 50.0f; //Maximum Index Possible

		for(int date = currentDate; date > currentDate-n; date -= weekLength) {
			if(browseHistory(date, currentTime)[date] == true)
				fuzzyIndex += k*(1/(currentDate - date));
			else
				fuzzyIndex -= c*(1/(currentDate - date));
		}
		if(fuzzyIndex > 0.6)
			notifySession();

		exportHistory(currentDate, session, duration);
	}


	//Imports the length of available data which is written to the end of the file
	void importHistoryLenth();

	//Imports exact session in past
	void importHistorySession(Date date);

	//Exports verified session to history
	void exportHistory();

	//Browse imported history
	void browseHistory(Date date, Time time);

	//Notify host that session may take place
	void notifySession();
}
```