Toronto Metropolitan University
Department of Electrical, Computer, & Biomedical Engineering
Faculty of Engineering & Architectural Science

GK02: Smart Reminder and Management System for a Work Environment

Capstone Design Project Final Report

Authors: Anthony Valenti, Paolo Dimaano, Aisha Yunus Navivala, Cavin Sabesan

FLC: Gul Khan

April 14, 2023

Acknowledgements

We would like to extend our sincere gratitude and appreciation to our supervising professor, Dr. Gul Khan, for his time commitment and valuable feedback throughout our entire project design and implementation. His guidance and mentorship have been much appreciated through our final two semesters.

We would also like to acknowledge the entire Computer Engineering department at Toronto Metropolitan University for their continuous support and dedication towards providing students with the best possible environment to learn and grow as upcoming engineers.

Certification of Authorship

We, Anthony Valenti, Paolo Dimaano, Aisha Navivala and Cavin Sabesan, certify that the work presented in this report is our own, and any ideas or assistance received has been properly cited and acknowledged.

Index

Abstract

This report examines the design, implementation and results of a smart office reminder and management system. The smart system consists of two components, an embedded system built on the Raspberry Pi referred to as the “central hub” and a cross-platform (IOS and Android) mobile application. Users of the central hub are able to view/create reminders, see weather updates, control smart lights, and can track employees in the office based on bluetooth device tracking. Users of the mobile application are able to control smart lights, create/view reminders, create/view meeting events, message other employees, and finally receive up to date weather and maps information. The central hub system was designed using Python while the mobile application was developed using Flutter to enable cross-platform capabilities. A variety of APIs (Application Program Interface) such as Google Maps and WeatherAPI, along with public Python libraries were utilized to implement some of the functionalities. Finally, Google's Firebase platform was used to provide key functionalities such as Database storage and User Authentication. The implementation presented in this report are based upon a previous implementation plan in which design schematics and technical specifications were created. Comparing with the initial design plan, all features with the exception of two have been successfully implemented. The aforementioned two features that were not implemented as outlined in the design specifications are geo-fencing feature which has been changed to bluetooth tracking and smart-thermostat control which was dropped due to cost constraints. This report provides the final implementation of our designs along with a concise analysis of our designs, testing procedures, final results, alternative designs and performance.

Introduction and Background

The use of technology has revolutionized the way organizations operate, and the workplace is no exception. In recent years, there has been an increasing demand for smart systems in the office environment, with the aim of improving productivity and efficiency. This report examines the design, implementation, and results of a smart office reminder and management system. The system comprises two components: an embedded system built on the Raspberry Pi referred to as the “central hub,” and a cross-platform (iOS and Android) mobile application. The central hub allows users to view and create reminders, control smart lights, see weather updates, and track which employees are in the office based on Bluetooth device tracking. Meanwhile, the mobile application enables users to control smart lights, create/view reminders and meeting events, message other employees, and receive up-to-date weather and maps information. The system's design was implemented using Python for the central hub and Flutter for the mobile application to enable cross-platform capabilities. APIs such as Google Maps and WeatherAPI, along with public Python libraries, were utilized to implement some functionalities, while Google's Firebase platform was used to provide key functionalities such as database storage and user authentication.The main objective of this project was to design, develop and implement an embedded system and mobile application that work together as a smart reminder and management system for a work environment. The goal was to create an all-in-one system that satisfies the common needs of employees in an office environment. The central hub, placed in a central location within the office, allows users to interact with it via a touch screen interface to use all available features. The central hub also uses Bluetooth location data of verified devices to display which employees are currently in the office, while the mobile application is accessible through a login portal that only verified users can access. This report provides the final implementation of the designs and a concise analysis of the testing procedures, final results, alternative designs, and performance. The implementation presented in this paper is based on a previous implementation plan, in which design schematics and technical specifications were created. All features of the system were successfully implemented, except for two features that were not implemented as outlined in the previous design specifications: the geo-fencing feature was changed to Bluetooth tracking, and smart-thermostat control was dropped due to cost constraints. The following sections will provide a detailed description of the specific objectives for the central hub (embedded system) and mobile application.

Objectives

The objectives of this project were to design, develop and implement an embedded system and mobile application that work together as a smart reminder and management system for a work environment. The goal was to create an all in one system that satisfies common needs of employees in an office environment. The embedded system referred to as the central hub should be placed in some central location within the office and users can interact with it via a touch screen interface to use all the available features. The central hub should also use bluetooth location data of verified devices to display which employees are currently in the office. Meanwhile the mobile application should be able to run on both IOS and Android devices and be accessible through a login portal that only verified users can access. The specific objectives for the central hub and mobile application are as follows.

Embedded System:

1. Check Weather
2. Create/Remove/View reminders for employees
3. View employees currently in the office
4. Create timestamps of entry of employees in the office stored in database
5. Create/ View meeting events on a calendar
6. Smart light control

Mobile Application:

1. Cross platform capability
2. Check weather
3. Maps information
4. Create/Remove/View reminders
5. Create/Remove/View meeting events on a calendar
6. Employee messenger
7. Smart light control

Theory and Design

System Architecture

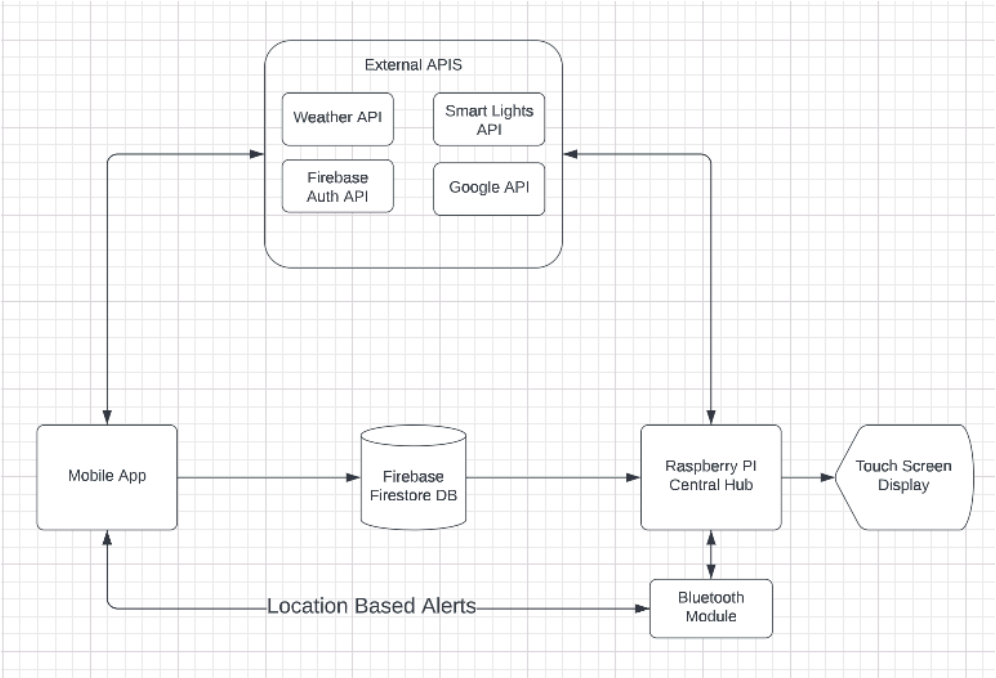


Figure 1: System architecture block diagram

Figure 1 above shows the block diagram representation of our system architecture, the diagram depicts how our system as a whole interacts with external services as well as between the hub and mobile application. The mobile application utilizes external APIs such as Firebase's Google Auth to provide user authentication via gmail for our system, this ensures that only permitted users have access to the application. Furthermore, communication between the devices is facilitated through a Firebase Firestore Database, this allows for synchronous data across all connected devices as well as reliability from Google through features such as offline access and user authentication. Lastly our location based notifications between the mobile application and central hub is accomplished through the raspberry pi's bluetooth module which can be used to determine how close/far a specific device is from the hub's location.

Mobile Application Overview:

The mobile application is the most prominent form of interaction within our system and has a multitude of features available to the users. We designed the application using Flutter due to its cross-platform capabilities allowing development for both IOS and Android devices, this was critical for our team as not all members own the same smartphones. Flutter is also created by Google meaning that majority of our technologies used are from Google and work very well together (Flutter, Firebase, GoogleAPI). Upon opening the mobile application the user must sign in using their Gmail to continue to the home page. From the homepage the user is able to select between the features offered by the app such as, checking the weather, viewing/scheduling meetings, controlling the smart lights, viewing commute times/routes, and setting reminders. All data being displayed/transmitted between devices is stored in its own collection on our Firebase Firestore which we query or add/delete data from as needed.

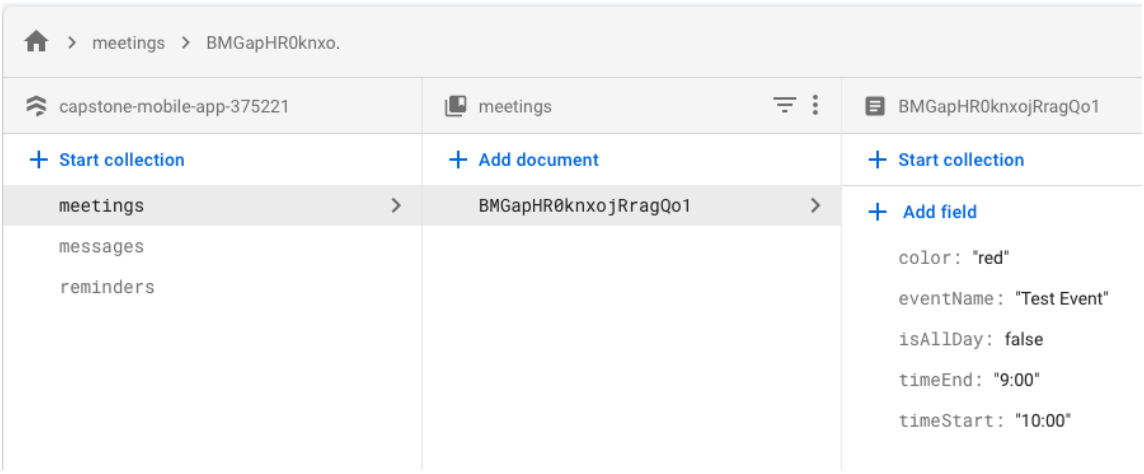


Figure 2: An overview of our FireStore Database with example data fields.

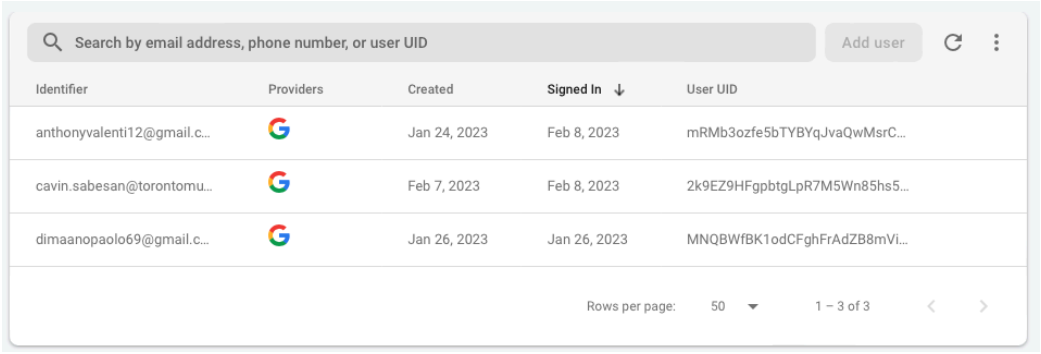


Figure 3: User authentication management via Firebase Authentication.

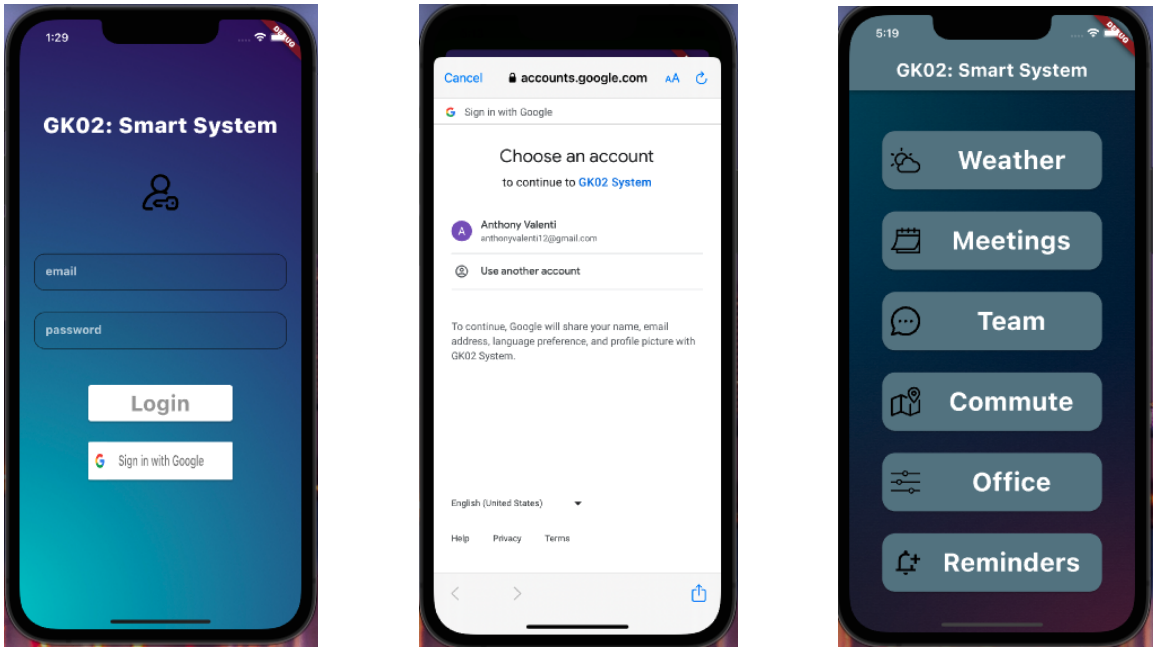


Figure 4: Mobile application login screen (left), google sign-in (middle) and homepage (right)

Shown in Figure 4 above is the program flow from opening the mobile application to logging in to finally having access to the homepage of the system where the user can access all of its features. It is important to note that the user interface depicted above is not final and many visual improvements are to be implemented for the final implementation of our mobile application.

Mobile Application: Smart Light Control

A key feature of our system is the ability to interact with other smart devices that might live within an office environment. We decided to implement smart light control due to the popularity of mobile controlled lights and also the cost associated with implementing them into an office environment is fairly low. The lights our system is able to interact with are the SONOFF B05-BL-A19 smart bulbs, which were chosen due to being one of the only brands to provide an open API for developers. The lights are controlled via sending HTTP POST requests to an address unique to each bulb, within this POST request we can turn on/off the lights, change the color and change the brightness. The API will respond with a JSON body containing an error indicator in which ‘0’ means the request has been successfully received. Within our system we utilize the API requests to bring full control of the lights within the mobile. It is important to note that in order to control these lights the device must be connected to the same wifi network as the bulbs, this is not ideal but was necessary since no other bulbs for reasonable prices provided an open API.

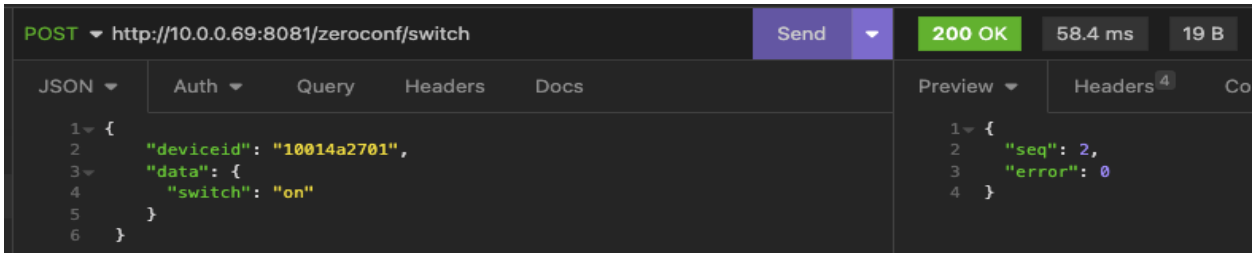


Figure 5: HTTP POST request to turn the lights on (left) and the response(right)

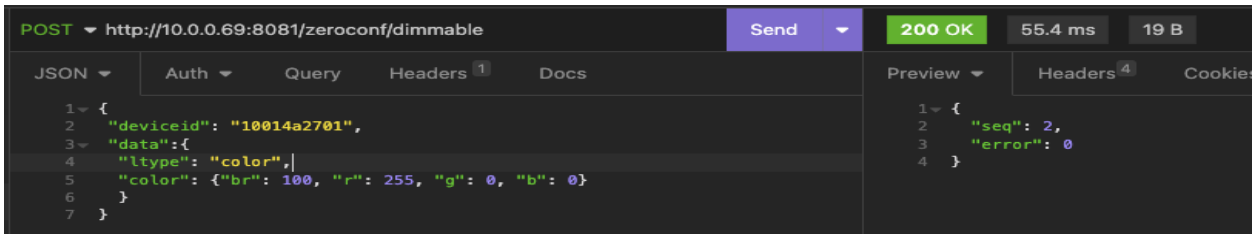


Figure 6: HTTP POST request to change light color to red (left) and the response(right)

Figure 7: Light control implementation switches trigger an on/off API request while the dropdown menu triggers a color change request.

Mobile Application: Meetings

We also are implementing a meetings tab which has two main components: the calendar view and the meeting creation. This tab is important for employees to stay informed of their upcoming meetings. When you enter the meetings tab you'll see a calendar view that lists all the scheduled meetings including the name, start time, end time, and members invited. When you are creating a meeting you can specify these details in the creation process. These are the details usually found in other calendar apps. We are also using the Firebase database from Google to serve as the backend to store and receive meeting events. When a user creates a new meeting, the data is pushed to the database. Then the database will signal the calendar, which will then immediately update to show the newly added meeting on the app. Now it will work the same way, when you edit an existing meeting, the updated information is pushed to the database, and the calendar is updated. The user is also able to scroll through the calendar, which will allow users to easily view all the upcoming meetings for today and days to come. There are two main buttons on the main screen which are the Add, and Edit buttons. When you tap on the Add button, you will see a pop-up window where you can enter the meeting name, start time, end time, and invite members. Once the user has filled in all the information, they will have created a new meeting that others can view. This will then get pushed into the Firebase database, and then the calendar view is updated to show the added meeting. The user can also delete a meeting event by tapping on the edit button, in which a list of all the meetings will appear and the user can select the one to be removed. The Meetings tab provides an easy to use interface for managing meetings and the Firebase database will ensure that all the information is stored and retrieved efficiently. The design of the meetings tab is a simple and straightforward way to create and manage meetings.

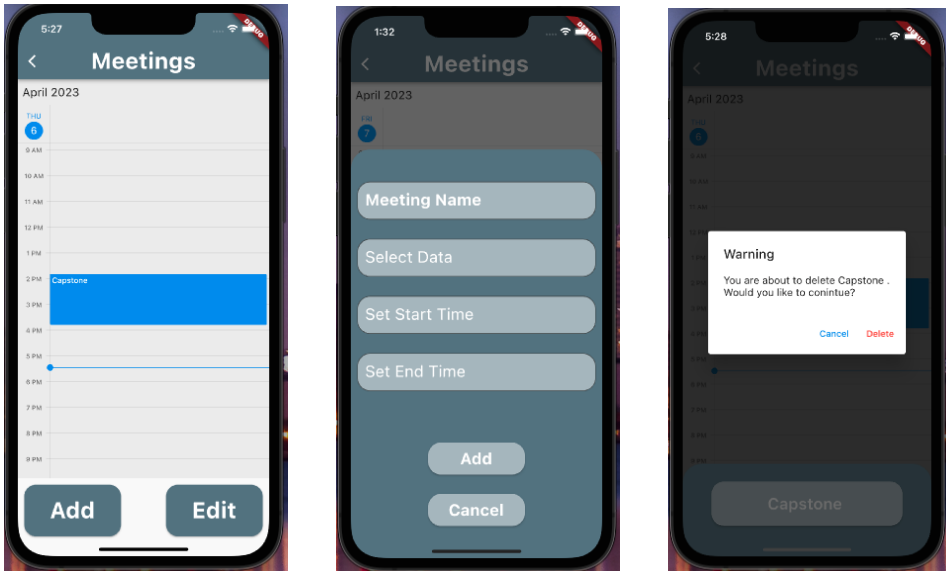
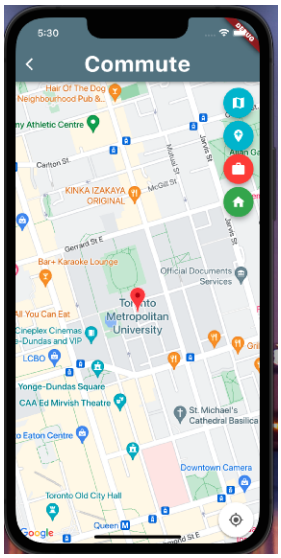


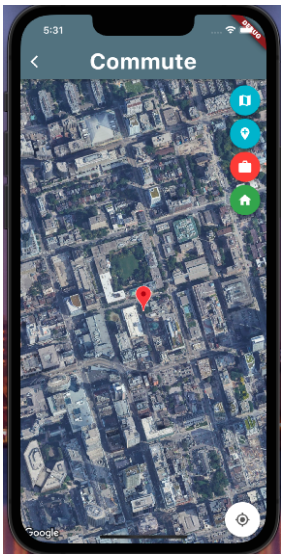
Figure 8: Meetings page on the mobile application

Mobile Application: Commute

An important feature implemented in our system is the Commute feature. The demand for convenient and efficient navigation features has grown, we want our users to be able to quickly and easily navigate to either their home or work location. Our system is built for a workplace setting so having a commute feature built into the application will make it easy and fast for the user to get directions to their workplace or their place of residence. We are using the Google Maps API in order to implement this feature into our application. Google Maps API is a widely used platform for developing location-based applications. It provides a variety of services, including maps, geolocation, and routing, that can be integrated into a mobile application. One of the most commonly used features of Google Maps API is its routing capabilities, which allow developers to provide users with turn-by-turn directions to a destination. Which makes it an ideal platform for the implementation of our Commute feature, as it provides a reliable and accurate means of navigating to a user's home or work location.

The Commute feature consists of four main buttons, home, work, satellite and marking locations, which is displayed on the main screen when you go into the Commute tab. When the user taps either the home or working button the map will move over to the location with a marker to show the location, where then you are able to get directions. There is also a satellite button that can be used to switch back and forth between satellite view and google maps view. To ensure that the Commute feature is user-friendly and efficient, we follow established practices for user experience design. This includes using clear labeling for the buttons, using familiar icons, and providing visual feedback to the user when a button is tapped.





It is expected to increase productivity for users. By allowing them to navigate to their home or work location quickly and easily, they are able to spend less time searching for directions and more time focused on their tasks.

Mobile Application: Weather

The weather functionality within the mobile application is made possible through a free public API called WeatherAPI. In order to use the API you need to sign up using an email and generate a free API key. From the mobile application a get request is sent using the API key and the program receives back a JSON body containing different types of current and predicted weather information such as temperature, precipitation, wind speed and more. The JSON body is parsed for the desired information and then displayed to the users in a visually appealing manner. This feature also utilizes a package called WeatherIcons by flutter which contains various different weather related icons. A new get request is sent every time the user enters the weather page ensuring that up to date information is always displayed.

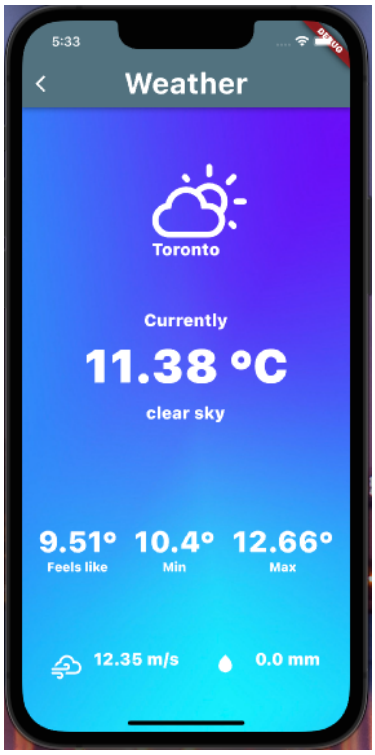


Figure 9: Weather page in mobile application

Mobile Application: Reminders

The reminders functionality within the mobile application is very simple and straightforward. Operating similarity to the calendar feature, users are able to create, view and remove reminders that sync with our Firebase database. These reminders are visible to mobile application users as well as on the central hub system. When creating a reminder the two necessary elements are an employee name, and then the actual reminder content. Reminders are stored in a global remindersList variable that is synchronously updated with the Firebase so that reminders remain consistent between the mobile application and central hub. If a user creates a reminder, the app will first create a local Reminder object and append it to our current list of reminders, next the contents of the created reminder is pushed to the Firebase database. Within the mobile app reminders are displayed using a scrollable table format, this provides a very clean and organized user interface as well as easy to understand code.

Figure 10: Reminders page in mobile application.

Mobile Application: Messenger

The messaging feature is an essential part of our mobile application, designed to allow for efficient communication between colleagues. The UI is quite simple and straightforward similar to what you would see from any application that allows messaging. It shows you a list of different people that you can communicate with. Then in the conversation view you are able to see past messages, and type new messages to send to them. To support the messaging feature we are using Firebase database to serve as the backend to store all the messages. It is important to be able to communicate instantly in the modern day and this feature helps achieve that, while increasing productivity by having all your colleagues in one place which leads to everything being more organized.

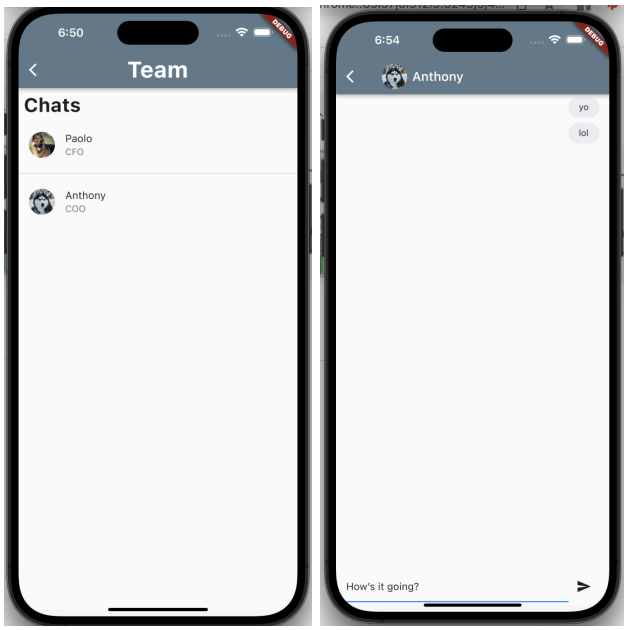


Figure 11: Messenger page in mobile application.

Central Hub-Overview:

The Central Hub Hardware consists of a housing enclosure, the Raspberry Pi 3 Model B+, and a touch screen interface.

Figure 12: Front, side, and read view of the Central hub

The development platform chosen for the application within the Central Hub employs python 3 tkinter to build the GUI. Python3 Tkinter was chosen because it is simple, beginner-friendly, and it is the GUI builder with the most online resources for learning. Similar to the mobile platform, Firebase is heavily integrated with the system because it is through Firebase that the Central Hub and Mobile devices stay connected and exchange information.

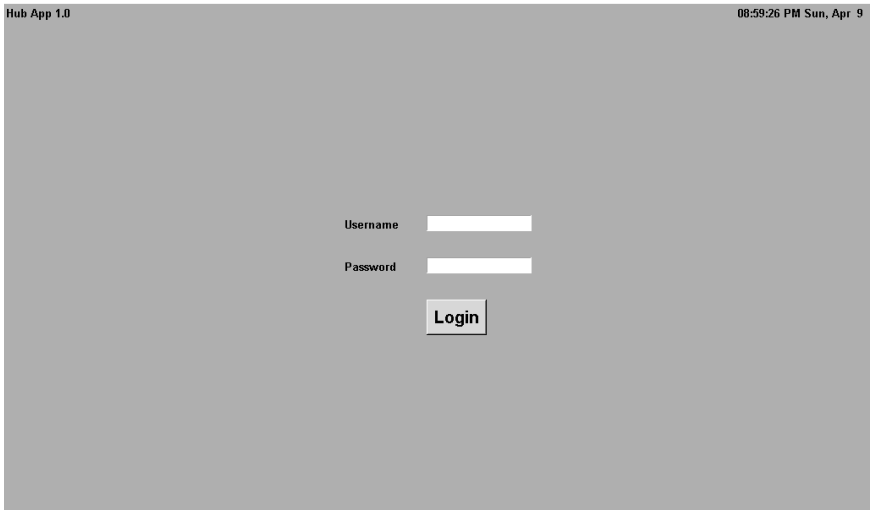


Figure 13: Login screen for Central Hub

To access the central hub, first the user is brought to the login page and input the username and password as shown in Figure 13. This is not tied to a google account, just locally created and stored information in the central hub just to add another layer and security and so that the central hub can be used and accessed without the cloud.

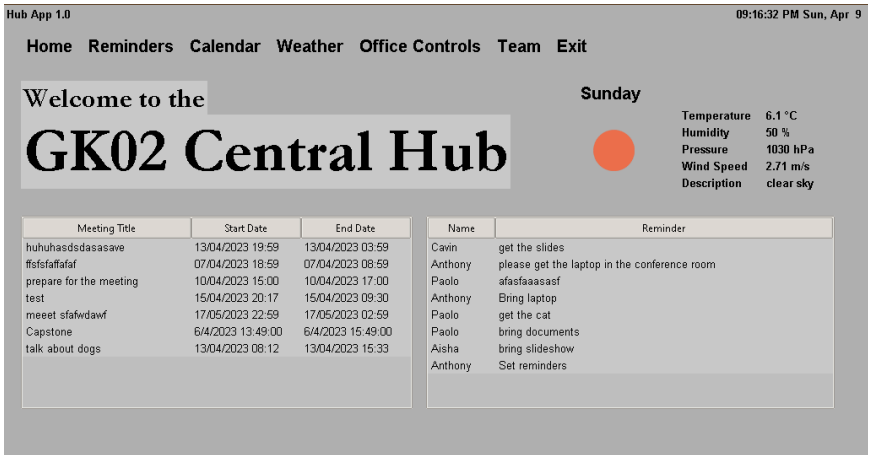


Figure 14: Homescreen for Central Hub

In the homescreen, the user is greeted with a welcome message and they will immediately be able to view the most important information needed in the office such as the weather, the meeting schedule, and the reminders posted. A clean and grayscale theme is used to achieve a professional look and it isa design that values functionality over aesthetics.

Central Hub: Reminders

The Reminder feature in the central hub will allow users to create personalized reminders for any person in the team. The primary purpose of this project is to be employed in a work environment and so being able to create reminders that are displayed in the hub and that can be sent to specific users is important. In the Central Hub, a user can select which member he/she would like to send a reminder to, and then he/she would be able to type in a message in a textbox to send them. Once the user sends the reminder, this information is stored in the Firebase Firestore database. The Mobile app users view the reminders created from the Central Hub and these reminders can also be viewed in the Central Hub home screen. Deleting reminders can only be done in the Reminders page however.

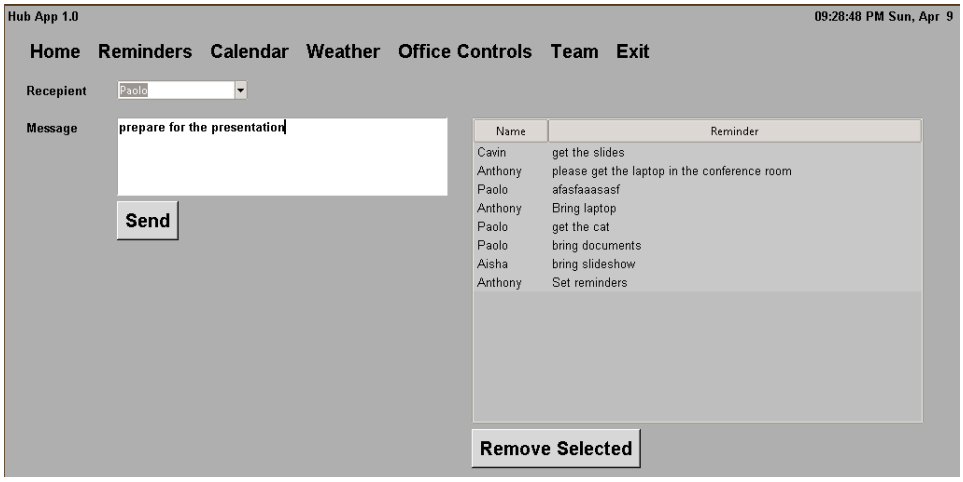


Figure 15: Reminders page for Central Hub

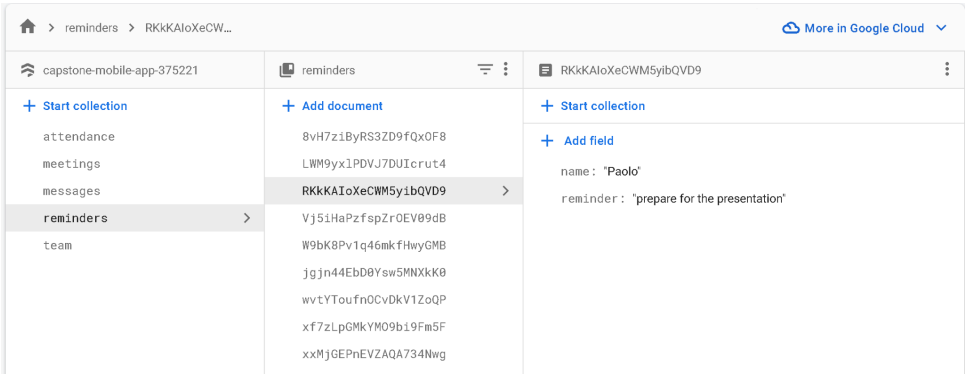


Figure 16: Created reminder was received in the database

Central Hub: Meetings/Calendar

The Calendar and Meetings event creator feature of the Central Hub has a similar function to that of the mobile side. The user is able to select a Date from a calendar and a Start Time and End Time for a meeting from a clock, create a title for the meeting event, and then send it to the database. To accommodate better flow of the program and to optimize the application without needing to open more pop up windows, it was decided that to view the meetings organized, the user would instead have to move away from the Meetings/Calendar page and go to the homescreen where they view the meetings created. In the screenshots below we can see the fully interactive calendar and clock for setting up meetings, we can also see the meeting we created displayed in the homescreen, and we can also see it successfully received in the database.

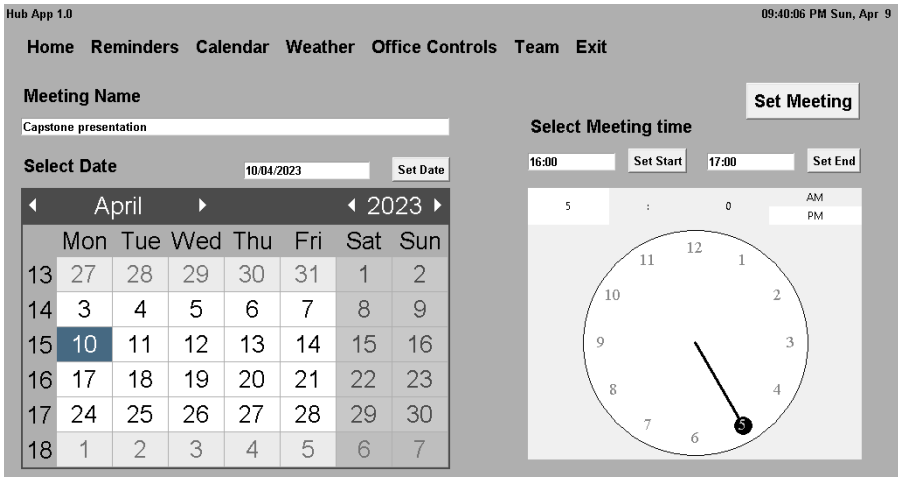


Figure 17: Meetings/Calendar page for Central Hub

Meeting Title	Start Date	End Date
huhuhasdsdasasave	13/04/2023 19:59	13/04/2023 03:59
ffsfsfaffafaf	07/04/2023 18:59	07/04/2023 08:59
prepare for the meeting	10/04/2023 15:00	10/04/2023 17:00
test	15/04/2023 20:17	15/04/2023 09:30
meeet sfafwdawf	17/05/2023 22:59	17/05/2023 02:59
Capstone	6/4/2023 13:49:00	6/4/2023 15:49:00
talk about dogs	13/04/2023 08:12	13/04/2023 15:33
Capstone presentation	10/04/2023 16:00	10/04/2023 17:00

Figure 18: Meetings view in the home screen of Central Hub

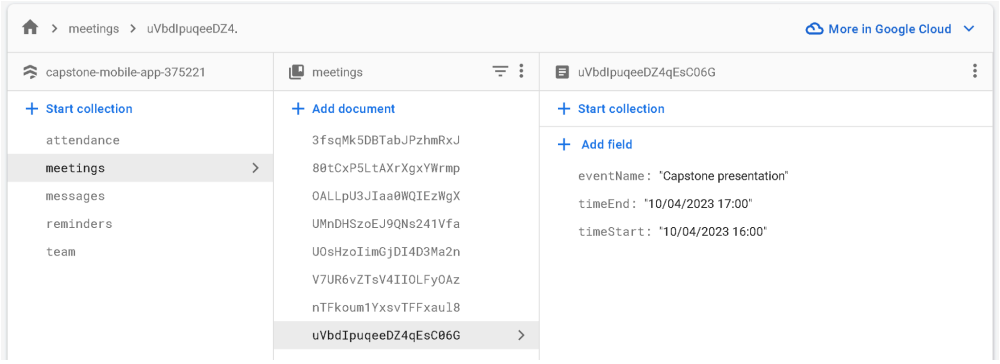


Figure 19: Created meeting event received in the database

Central Hub: Bluetooth mobile sensor tracking and Notifications

In our preliminary design proposal, we planned to integrate a geo-fencing feature using software to compare the current longitude and latitude of authenticated mobile devices within our system relative to the position of the central hub. However, after consultation with the FLC, it was recommended to decrease some reliance on public API's and to start utilizing the hardware components of our embedded system to their full potential. Thus we have pivoted

away from “geo-fencing” and instead integrated location tracking using the raspberry pi’s built in bluetooth module. Why Bluetooth? Bluetooth is a feature that all modern devices have, it is a very simple data sharing protocol, for baseline hardware it is low-cost and low-power. The Raspberry Pi 3 Model B+ is equipped with Bluetooth 4.1 like most modern devices. The idea was that since the central hub is placed within an office, we can scan for authenticated devices' bluetooth addresses and determine if they are within the office or not. This information is used to generate a daily list of when each employee has arrived or left the office which can be critical information for tracking productivity or even logging/validating employee hours worked. The bluetooth data necessary for the operation of this function would simply be the desired bluetooth device's name/address and RSSI which stands for Received Signal Strength Indicator. RSSI levels can be used to determine the distance between a specific device and the central hub's bluetooth module. The RSSI threshold can also be set so tagging office devices can be done in a specific distance radius from the hub in a more advanced setting with a powerful bluetooth hardware adapter inside the Central Hub. Since the Central Hub does not have an expensive and powerful external bluetooth module adaptor, it has a limited range where it can scan for devices, however with more funds it is a scalable way of detecting when an employee enters the office. In this implementation, once a user has been verified to have entered the radius for the Central Hub to scan them, a notification is sent to the database indicating the name of the employee and the timestamp for when they entered the range of the bluetooth authenticator scan and so the company will know if an employee has went away from the office and reentered it recently. In the screenshots below we can see the Team view page where detected and authenticated users have their names displayed and we can also see the timestamps sent to the database.

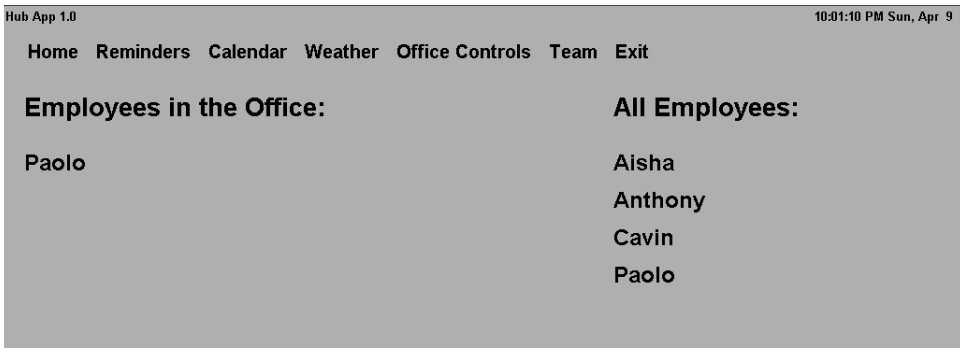


Figure 20: Employee attendance view page in Central hub

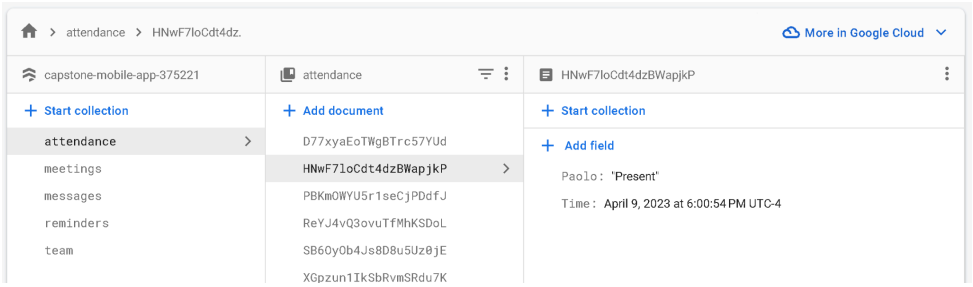


Figure 21: Database employee timestamp notification with universal time

Central Hub: Smart Light Control

The Central Hub offers users the ability to control smart light devices through the hub itself, which allows for a more convenient and accessible method of managing the lighting in a space. This feature is made possible by sending an HTTP post request from the hub to the connected light bulb devices, which is the same approach used by the mobile application.

By using the Central Hub as a central point of control, users can manage multiple smart light devices at once, without the need for separate control devices or apps. This not only simplifies the user experience but also enables greater flexibility in managing the lighting in a space. Moreover, the Central Hub's control over smart light devices can be integrated with other smart home devices, such as sensors, cameras, and door locks in the future with more time and money invested in the project. This integration can potentially create powerful automation scenarios that enhances the overall user experience. For example, when a sensor detects motion at the front door, the lights can automatically turn on to illuminate the area, while the camera records the event and sends a notification to the user's phone. Additionally, the Central Hub's ability to control smart light devices can also improve the energy efficiency of a space. Overall, the ability to control smart light devices through the Central Hub is a key feature that adds significant value to the user experience. It simplifies the management of smart lighting devices, and allows for greater flexibility and control over the lighting in a space.

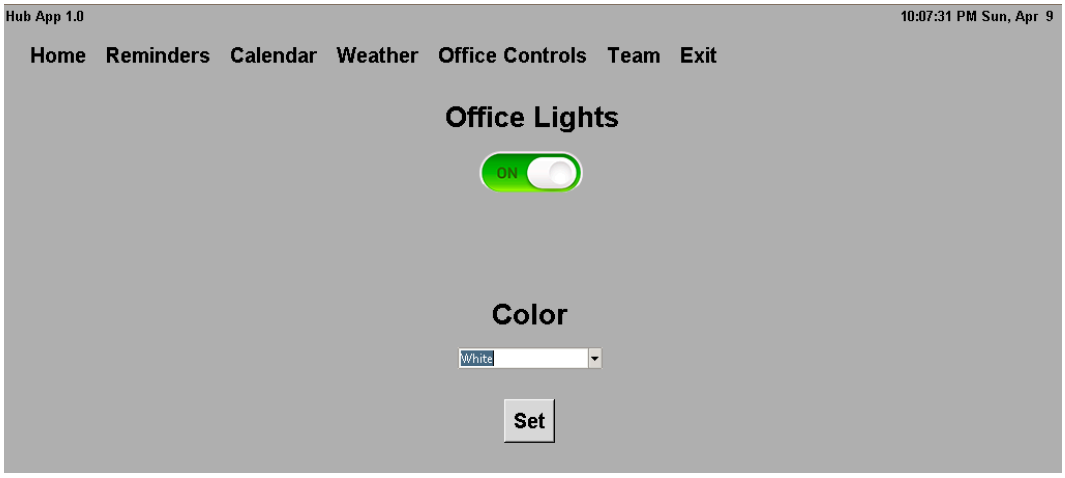


Figure 22: Office control page in the Central Hub

Central Hub: Weather

The ability to view the weather in the vicinity of the office is another essential feature of the Central Hub, which provides valuable information for the office employees. This feature is implemented by displaying a 7-day weather forecast report in the Weather page of the Central Hub, which can help employees plan their work and activities accordingly. Additionally, the current weather of the day can also be seen on the homepage of the hub, as it is a critical piece of information for office employees to have. To implement this feature, the Central Hub utilizes the openweathermap API to retrieve all the relevant weather information. This API provides up-to-date weather data for any location based on geographic coordinates, which makes it an excellent choice for the Central Hub to obtain accurate weather information for the office's vicinity. The weather information obtained from the API includes temperature, humidity, wind speed, and atmospheric pressure, among other data points. Depending on the weather description, an image stored in the local directory is matched with this description, and the image is then displayed in the weather page of the hub. This image matching adds a visual element to the weather display, making it more intuitive for the office employees to understand and interpret. Furthermore, the 7-day weather forecast report provides a detailed view of the expected weather conditions for the coming week, allowing office employees to plan their schedules and activities accordingly. This information can be especially useful for planning outdoor activities, meetings, or events that are sensitive to weather conditions. Overall, the ability to view the weather in the vicinity of the office is a valuable feature that adds to the convenience and usability of the Central Hub. By using the openweathermap API and image matching, the Central Hub provides accurate and visually appealing weather information to office employees, making it easier for them to plan and prepare for the day's activities.

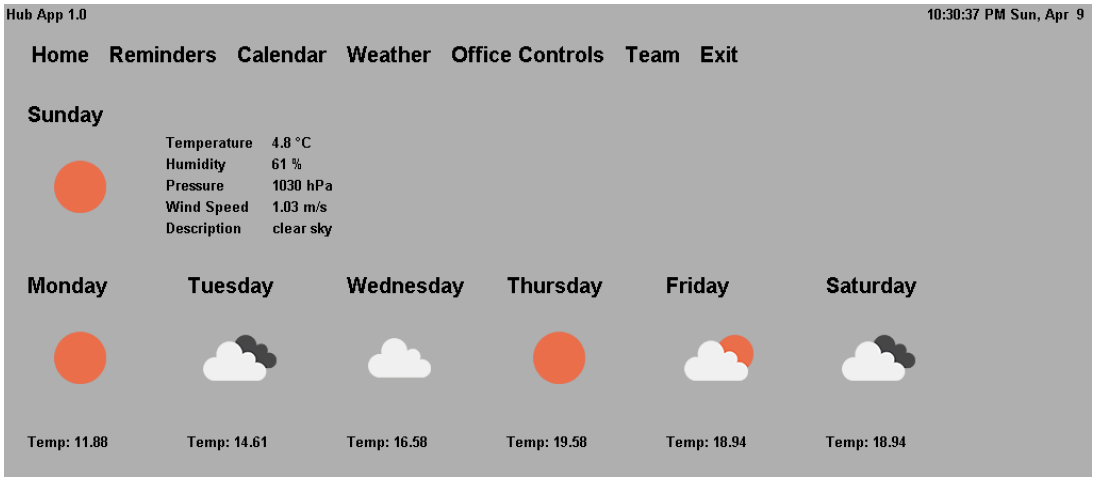


Figure 23: Weather page in the Central Hub

Alternative Designs

Mobile Application:

Multiple alternative designs were considered for the mobile application during our design and development phase. Initially the group considered using React Native instead of flutter to develop the app. Ultimately we decided against using React Native due to the syntax of Flutter being pretty similar to Java, a language all group members were familiar with. Also during the early stages of the project we considered using MongoDB as our database instead of Firebase. Firebase was ultimately chosen because it could perform user authentication, sign-in with Google, as well as store all of our data in one platform, which would not have been possible if we used MongoDB. Some alternative design features that we decided against included interacting with more IoT devices such as a smart thermostat. We decided against the implementation of this feature for a multitude of reasons. One being that smart thermostats are very expensive ranging from $120-$250, along with the fact that online it is made very unclear which brands actually have an open API for integrating their devices into custom applications. Another reason we decided against integrating a smart thermostat was that it's a very hard feature to demo, requiring more than just a simple “plug & play” setup that we get from using smart bulbs. Finally there were multiple alternative design choices for the mobile user interface, things such as the color scheme, button style and fonts changed throughout the entirety of the implementation phase, until we finally reached a unanimous decision on a darker blue theme. Seen below is a comparison between the alternative design (left) and the implemented login interface (right).

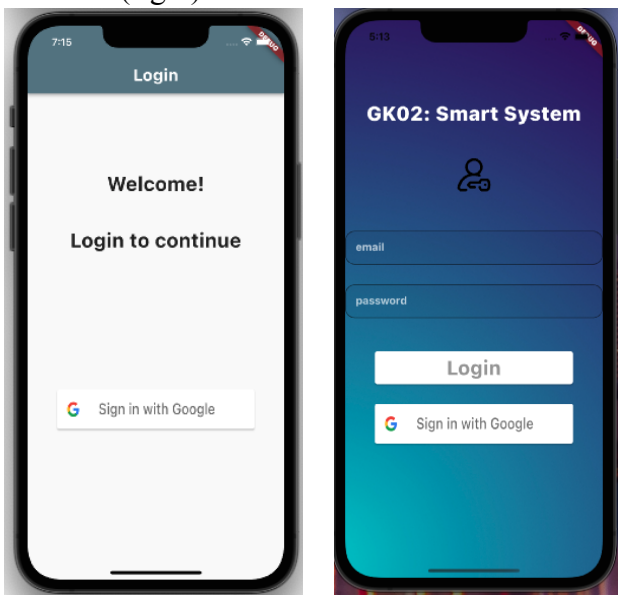


Figure 24: Before and After login interface

Central Hub:

Throughout the development phase, numerous changes to the implementation of the design of the Central Hub have been made. The first was at the start we decided the Central Hub would have speakers but during the development, we found that none of the features we had needed a speaker implementation, we also considered that in an office setting it would be detrimental to employee focus if the hub was constantly ringing from alert notifications. The second was to use a temperature sensor in the Central Hub to monitor the office temperature and have connectivity with a smart thermostat to control the office temperature but it was expensive to purchase, setup, and connect to the Central hub so we decided to move away from that idea. Another change that was made is in the implementation of the reminder feature for the Central Hub. The first implementation had it where the Central Hub is connected to the Realtime Database of Firebase because of difficulties in using the Firestore Database but as the development went on and we became more familiar with using the database, we were able to change it so that both the mobile application side and the Central Hub were using the same Database in Firebase.

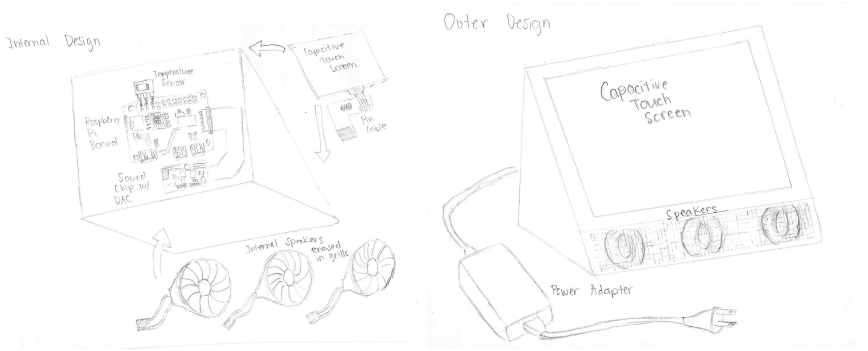


Figure 25: Initial Central Hub design

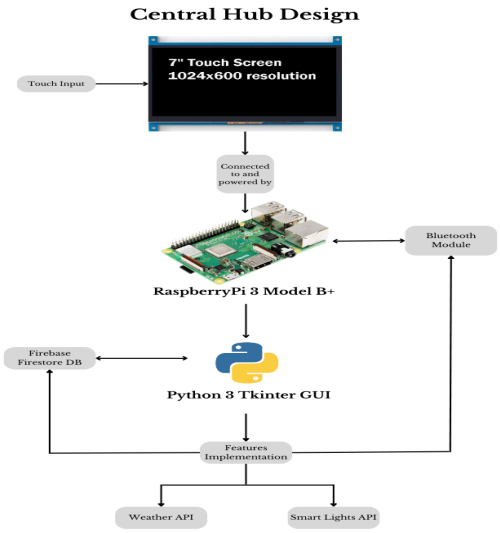


Figure 26: Updated Central Hub Design

Material/Component List

Item	Cost	Description
Raspberry Pi 3 B+	Free - group member already owned one	The embedded device used to power our central hub. Cortex A53 CPU, 1GB of Ram, built-in Wi-Fi and bluetooth, 40 GPIO pins, Micro SD support.
Raspberry Pi touchscreen (https://www.amazon.ca/SunFounder-Raspberry-Touchscreen-1024%C3%97600-Capacitive/dp/B07Y889J3X/ref=mp_s_a_1_3?crid=3FVDIQIMMCGHI&keywords=raspberry+pi+touch+screen&qid=1673908396&sprefix=raspberry+pi+tou%2Caps%2C115&sr=8-3)	$80	7 inch touchscreen display used to interact with our central hub system. 1024×600 display powered by USB, connected via HDMI.
Sonoff Smart Bulbs (https://www.amazon.ca/SONOFF-B05-BL-A19-Wi-Fi-Smart-Variable/dp/B09TNPZ2CS/ref=mp_s_a_1_3?crid=1LQ2XVW1N29RQ&keywords=sonoff+bulb&qid=1674666681&sprefix=sonoff+bulb%2Caps%2C123&sr=8-3)	$35	Wifi enabled smart LED bulbs, one of the only brands with an open API for developer use. Color changing and adjustable brightness.
Wood planks	Free	Wood planks were cut and attached together to create the housing for the Central Hub.
Total:	$120	

Measurement and Testing Procedures

This section sets forth the pre-requirements as well as the expected outcome of each of the functionalities implemented on the mobile application and central hub. The purpose of this table is to keep track of how each functionality should operate and can be used to repeatability and consistently test each feature.

Mobile Application:

Functionality	**Requirements**	**Expected Outcome**
Login - email	● Internet connection ● Verified user within our firebase	Upon entering an already verified email and corresponding password into the corresponding fields, the user can press the login button and should successfully move to the home screen. If incorrect the incorrect info text should appear.
Login - Gmail	● Internet connection ● Verified user within our firebase	Upon clicking the sign in with google button, an in-app web page should open where a user can enter their gmail information. If correct the user will move on to the home screen.
Weather	● Internet connection	Upon clicking the weather page, up to date weather information should be displayed to the user. The information should refresh upon each entry into the page.
Meetings	● Internet connection	The user should be brought into the calendar view with the current day displayed, and be able to swipe left/right into past or future days.
Meetings - Add	● Internet connection	Upon clicking the ‘Add’ button a modal should appear from the bottom of the screen allowing the user to enter meeting information. After confirming the Add the created meeting should now be visible on the calendar.
Meetings - Edit	● Internet connection ● Existing meetings	Upon clicking the ‘Edit’ button a modal should appear from the bottom of the screen displaying all Meetings in the database. Selecting one of the meetings will allow a user to remove it upon confirmation from a pop-up.
Messenger	● Internet	When first entering the messenger function the

	connection ● Verified users in the database	user is presented with conversations between all registered employees. The user can select a conversation and send/receive text messages. The user should also be able to see a profile picture and title for each employee.
Smart Lights Control	● Internet connection ● Lights must be connected to wifi ● device connected to same network as lights	The user can toggle the on/off switch and the lights should respond within 2 seconds. The user can also control the color of the lights using the dropdown menu and the light should respond accordingly.
Reminders	● Internet connection	The user should be brought into the reminder view with all reminders shown in table format.
Reminders - Add	● Internet connection	Upon clicking the ‘Add’ button a modal should appear from the bottom of the screen allowing the user to enter reminder name and information.
Reminders - Edit	● Internet connection ● Existing reminders	Upon clicking the ‘Edit’ button a modal should appear from the bottom of the screen displaying all reminders in the database. Selecting one of the reminders will allow a user to remove it upon confirmation from a pop-up.
Map	● Internet connection via cellular or WIFI	Upon launching the map the user should be able to scroll through an interactive map, drop location pins, see the saved office address, and toggle between satellite and

Central Hub:

Functionality	Requirements	Expected Outcome
Log in	● Verified user stored in system	Once the stored username and password is entered the user is brought to the homescreen
Weather	● Internet connection	1. In the homescreen, the current weather is displayed along with information 2. In the weather page, a 7-day weather forecast is shown with visuals and the temperature information
Calendar/Meetings	● Internet connection	1. In the homescreen, the table of event meetings with start and end times are displayed. 2. In the Calendar/Meetings page, the user can select a date from the calendar, start and end times from the clock, write a title for the meeting and when the user sets the meeting it is sent immediately to the database and updated in the meetings list displayed in the hub.
Smart Light Controls	● Internet connection ● Lights must be connected to wifi ● device connected to same network as lights	The user can toggle the switch for the lights on/off and they also have the option to select the color of the light.
Reminders	● Internet connection	1. In the homescreen, the list of reminders along with who it is addressed to are displayed in a table 2. In the Reminders page, the user can select a recipient, type in any reminders they want to send, and when they press send, it is immediately sent to the database and updated in the reminders table. 3. The user also has the option to delete

		any selected reminder from the table and press on the delete button.
Team view/Bluetooth sense	• Internet connection • Central hub has bluetooth turned on • Mobile device has bluetooth turned on and set to visible mode	1. In the Team view page, the names of the registered employees scanned via bluetooth in the office is displayed. 2. Once an employee has been confirmed to have been scanned and authorized, a timestamp of when they were scanned is sent to the database which can be used to monitor office attendance. 3. Reentries by the same employee in the office zone will also have their reentry timestamp sent to the database.

Performance Measurement Results

Mobile Application:

The results were overall positive for our mobile application, we used use cases to gather these results that we are going to go over. First, we tested our smart light control by observing the ability to send HTTP POST requests which turn the lights on/off, and change the color and brightness. We found that we were able to control the lights with our system with little to no delay, with the API response time being fast. For our meetings tab, we tested it to efficiently store and receive meeting event data by using the Firebase database. To Add a new meeting the process is straightforward, the user will click on “Add” and proceed to fill in the information for “Meeting Name”, “Select Data”, “Set Start Time”, “Set End Time”, and then click on “Add” to successfully add the meeting to the calendar. The user can follow the same steps to edit an existing meeting on the calendar, and then it is immediately pushed to the database. We also went and tested the speed it would take to update the calendar view when you add, edit or delete a meeting and we found that the average time was less than two seconds which is fast.

The commute feature integrating the Google Maps API went smoothly and was a reliable method for our users to be able to see their workplace and home and an estimated time to reach those destinations. We have icons in place for users to quickly go from Home to Work, and this has shown to be fast and responsive. The weather feature was implemented using the WeatherAPI, with the Celsius being displayed with also the real feel, minimum temperature, maximum temperature, and wind speed. The information displayed is accurate as we checked alongside several weather applications to make sure. Our testing for our reminder feature consisted of creating, viewing, and removing reminders as well as making sure the data is synced with Firebase. To add a reminder the user will click “Add” , fill in the respective “Name”, and “Reminder”, and then click Add. The time it takes for it to be created is basically instantaneous, and no different as it synchronizes with Firebase. For our messaging feature we tested using different networks, and sent messages with different lengths. We found that it was able to handle large amounts of data, and sending and receiving messages were very fast allowing users to communicate in real-time.

Central Hub:

The performance results of the Central Hub were overall positive, this conclusion was achieved through testing the performance of each of the implemented features. The login feature that greets the user when they boot the central hub software in the raspberry pi works without any hitches or delays once a successful or unsuccessful login is done. The homepage loads almost instantly with the current weather displayed, current reminders stored displayed, and the current event meetings displayed. Opening the weather page, the 7-day forecast instantly loads up and displays without any delays. The pictures are displayed and scaled correctly and the data information is also displayed correctly. Next, when we open the Calendar/Meetings page, the interactive calendar is displayed correctly with no lag along with the interactive clock. Setting the start time and end times by clicking the associated button also works correctly. When the user clicks on the set meeting button, the information is immediately sent to the database and the page reloads with no noticeable delay. The next test was opening the Reminders page where the textbox, dropdown menu box, and the reminders list table are all displayed without delay. Typing in the remainder and pressing the send button instantly sends the data to the database and reloads the page without any hitches. Deleting a reminder through pressing the delete button also pushes the data to the database immediately and once the page reloads, the table is already updated accordingly. The next test was the Office Controls page where the on and off switch button along with the dropbox to change the color of the lights loads immediately. The response time for the changes pushed to the light bulb via HTTP post requests showed little to no delay in ideal conditions with great internet connections. The last test done was the Team view/bluetooth sense feature test which showed the worst performance out of all the tests. Pressing the button that leads to the team view page, there is a delay of up to 700 ms before the list of employees present is displayed in the page. The scanning range for the bluetooth sensing was measured to be about 5 meters in radius of the Central Hub. More work needs to be done to improve this performance since it is the only feature that has a slow response time.

Analysis of Performance

Mobile Application:

To analyze the performance of the mobile application we can compare the expected outcomes of each function listed in the previous section to the actual results of execution within our app. To start, both login features perform exactly as expected, with both logging in using email and from Google. The invalid email/password message properly displays if an unauthorized user attempts to login. Next, the weather, map and messaging feature all work exactly as described with responsive updates to new requests experiencing no issues at all. However, there are some problems with performance when it comes to the reminder and meetings features. The reminders feature works as described, however there is a slight visual bug that occurs when first entering the page in which the background image takes a slight delay to load. The same bug occurs when first opening the light control feature. Along with this visual delay the meetings and reminders features also experience an error when deleting meetings/reminders from the database. The objects are successfully and instantaneously removed from the database, however the changes are not reflected on screen unless you exit the page and re-enter. Despite numerous attempts to fix this bug no solutions could be reached within our time constraints. This error is especially unusual considering the add and delete features operate so similarly yet the add feature experiences no problems instantly displaying new elements.

Central Hub:

The Central Hub hardware and software system is designed to be user-friendly and efficient for use in a work environment. The hardware components consist of a Raspberry Pi 3 Model B+ and a touch screen interface housed in a wooden enclosure. Hardware wise, the Raspberry Pi 3 Model B+ is equipped with Bluetooth 4.1 which is an older generation hardware, it is 5 years old but through most of the features of the hub, it performed great without noticeable hiccups and delays aside from the bluetooth sense feature. Through our analysis of the cause of the delay of the bluetooth sense feature talked about in the previous section, we concluded that the hardware bluetooth module was the limiting factor since the delay primarily came from the long device scanning times while the process of matching the bluetooth address from different devices to the registered devices took little to no time. Software wise, the Central Hub software system was designed with simplicity and functionality in mind, with a clean and grayscale theme that values functionality over aesthetics. The software development platform used was Python 3 Tkinter, which is beginner-friendly and has numerous online resources for learning. The program overall ran as expected but Tkinter proved to not be the best choice when creating more aesthetically pleasing widgets because of its limited customization tools. The Central Hub is integrated with Firebase, which allows for seamless communication and information exchange between the hub and mobile devices and analysis of this integration shows that it is very well integrated since all data pushed from the Central Hub to the database all happens instantly and retrieval of data also happens instantly without any delays. Overall, the Central Hub system was designed to be user-friendly, efficient, and functional for use in a work environment. The integration of various features and components, including reminders, meetings, smart lighting controls, and Bluetooth sense tracking all worked seamlessly with only the bluetooth feature needing more significant work to be done for optimization.

Our current solution for the limiting factor of the outdated bluetooth module hardware in the Raspberry Pi 3 model B+ causing delays in bluetooth device scanning and having a small scanning range of approximately 5 meters in radius is to place the central hub in the entrance of the office rather than the middle to accommodate for the small scanning range. This solution will temporarily solve the problem but if we want to place the Central Hub in the middle of the office as we originally planned, then we would need to invest more money to purchase a better bluetooth adapter that we can integrate with the system which will allow us to increase the effective range and the scanning speed for office devices.

Conclusions

In conclusion, the objective of this project was to design and implement a smart reminder and management system for a work environment. Our system consists of an embedded system using a Raspberry Pi3 and a cross-platform mobile application that work together to provide users features such as creating calendar events, creating reminders, instant messaging, smart light control, in-office device tracking as well as current weather and maps information. Our system uses Google's Firebase to support the majority of its functionalities including data storage, messaging and user authentication, while we use the bluetooth module on the Raspberry Pi to provide location data on verified devices. Utilizing our proposed design specifications created in the past semester, we were able to carry out the successful implementation of the desired system, with the exception of modifying two previously proposed features. In our initial proposal we planned to use geo-fencing to determine which users were present in the office, however this feature has been changed to use bluetooth RSSI to determine proximity of devices to our central hub. We also originally planned to implement smart thermostat control, but due to cost constraints we did not proceed with this feature. All implemented features operate as expected with the exception of a minor error that occurs when deleting reminder/calendar events in the mobile app. Upon deletion of an event/reminder the changes are instantly reflected in the database, however the calendar/list does not visually correspond to the changes unless the user manually refreshes the page by returning to home and then re-entering. Unfortunately due to time constraints this error could not be removed, however in the future this error should be fixed. Looking forward, some additions that could be made to our project are implementing further IoT device control such as smart plugs, cameras or door locks. Another addition that could be made is better utilization of our bluetooth location data of devices to turn on specific lights upon entering/leaving a room. Overall, we are proud of the successful implementation of our specified design and recognize the potential improvements that could be made in the future.

References

[1] https://docs.flutter.dev/

[2] https://firebase.google.com/docs

[3] https://docs.python.org/3/library/tk.html

[4] Baxi, P. S. (2012). Android application of quick organizer. https://krex.k-state.edu/dspace/bitstream/handle/2097/13800/PoonamBaxi2012.pdf?sequence

Appendices

Mobile Application Source Code: https://github.com/AnthonyValenti/Capstone_MobileApp

Central Hub Source Code: pdimaano88/capstone-hub (github.com)