Study Spot Finder

Perla Del Castillo, Maria Ines Herbas Gutierrez, Andrew Kwong, and Darwin Motato
Dept.of Electrical Engineering and Computer Engineering
University of Central Florida
Orlando, Florida, United States

***Abstract* — As college students, we sometimes find ourselves struggling to find a study space. At the University of Central Florida, we might have it a bit harder due to the large number of students on campus and how far apart study spaces are from each other. This ends up wasting valuable study time, gives us frustration, and even can ruin a perfect day for studying.**

Our solution is an interconnected network of hardware devices attached to study tables that allow users to find out which spots are available. These devices can be attached to different study tables in different buildings around campus. This device has an app that students can download to reserve a spot as well as view all available spots. The device has lights that represent different states such as: available in green, awaiting confirmation in yellow, and taken in red, in order to easily communicate its status to students. The user can easily find available spots on the app, tentatively reserve it, and lastly confirm by inputting a keycode. We believe this solution makes finding a study spot a breeze and we hope that students spend more time studying than trying to find where to study.

I. Introduction

Everyday students lose valuable time looking for a place where they can sit down and study for their classes. They will waste precious hours over the course of their entire education searching multiple floors and countless buildings for that perfect desk, the one with a comfy chair, a nearby outlet, etc. Instead they turn away, disappointed to find that their spot has already been taken. This process repeats again and again for many students - that wasted time could be utilized for additional studying time for that student's upcoming exam, or for preparation time before studying (e.g. getting food, water, materials)

Our team is proposing a reservation system for study spaces that will be easily scalable and accessible to all students. It streamlines the process of finding a proper space and instead allows the user to focus their time more on what matters: their education. Using a combination of a hardware device and a mobile app, students can locate their preferred spot with their desired amenities, reserve it, and arrive to find their spot reserved and ready for their use.

This project was inspired by countless first and secondhand experiences. On far too many occasions we found ourselves searching floor after floor for spaces to accommodate our needs, subtracting valuable time from our studying experience. We want to eliminate this useless, ancient process of trial and error and bring it to the 21st century, as so many things have been. More importantly, we want to improve the experience of all students through the use of technology and demonstrate that any problem can be solved with ingenuity and creativity.

II. System Overview concept

This section describes how the Study Spot Finder works as a complete system. Study Spot Finder behavior is cyclical, and this process can be summarized in 3 stages: available, awaiting, and reserved.

A. Available

This is the initial state of the device after it is first set up through the admin side of the application, and whenever a new reservation is about to occur. In this state, the device is continually checking for an update in the Firebase database under its unique ID, while having green LEDs on to signal its availability. It appears as available on the mobile application for users to select, along with the other devices. Once a device is selected, a popup containing specific characteristics of the device appears, along with the option to place a reservation. These characteristics are set by the admin during the initial setup.

B. Awaiting

Once a user has selected a suitable spot in the Mobile Application, it will update that device ID's status the Firebase database. This update, once the device has recognized it, will change the device to awaiting confirmation. The LEDs will light yellow and the device will begin to accept an input on its keypad. Meanwhile, on the mobile application, the popup will update with a randomly generated keycode, along with an option to release the spot prematurely. Once the user arrives at the study spot, they must input the keycode on the device in order to verify that the correct user is confirming. The device will also have access to the generated keycode, and, after successfully comparing the code, switches to the Reserved state.

C. Reserved

Once reserved, the Mobile Application will simply provide the option to release the reservation, while the device LEDs light up red and continually check for an update in the Firebase database under its unique ID. Once the user has finished with the study spot, they will release the reservation, which will update the device's status in the Firebase and show the spot as available in the app. The device will return to the Available step once it recognizes this update, and the process repeats for another user.

III. System components

This section briefly describes the individual components or modules that were either designed or bought to integrate. Each component is necessary for realizing the capabilities and functions of Study Spot Finder.

A. Casing

In order to conform with the specifications for a lightweight, easy to use device, the team decided to create a 3D printed enclosure using ABS plastic. The shape of the device was chosen to be cylindrical for aesthetic purposes, and to hold the components properly. The original design had four stages: the battery casing, the main PCB housing, the clear

LED PCB housing and the push button housing. However, due to unforeseen circumstances, the team decided to create a handmade back-up design using PVC tubing to hold the main PCB, and a clear polypropylene tube to hold the two LEDs. The push button will be located on top of the PVC, and a cap will be located below to secure the components.

B. Wi-Fi Module

For the communication between the device and the mobile application, we needed a wireless communication component. We decided to use Wi-Fi over Bluetooth, since it had longer range and a more efficient means to communicate to the application and its database. Some WI-FI modules discussed were the CC3100 from Texas Instruments and ESP 8266 from Espressif. ESP8266 was the ideal Wi-Fi module for this device due to its low cost and power consumption. This device was also compatible with the Arduino IDE for programming, which allowed for faster prototyping and useful APIs concerning Firebase communication as well as serial communication to our microcontroller.

C. Microcontroller

Our original choice for the microcontroller was the MSPEXP430-G2ET due to its popularity, processing power, and our own experience with it. However, because of the efficiency of the Arduino IDE, and the overwhelming documentation concerning Firebase communication with Arduino, we decided to use the ATMEGA328P chip. This chip has enough GPIO pins for all our external devices as well as a 16MHz clock to quickly check for keypad inputs while simultaneously sending and receiving data packets over serial communication.

D. LED

The requirements for our LED was that it needed to be able to notify the user of changes to the status (available, awaiting confirmation, and reserved). For this we chose to use an RGB LED, mainly because of its ability to produce multiple colors, including the colors we needed: green, yellow, and red. In order to reduce power consumption yet maintain status awareness, we decided to use 2 RGB LEDs.

E. Supply Voltage & Regulation

For the supply voltage, we decided to move to a 9 Volt Lithium Battery in order to cover the dropout voltage for our regulators, to provide a high enough voltage to supply the devices, and for its large 1200mAh capacity. To provide the adequate and stable voltage for our microcontroller and Wi-Fi module, we used two fixed voltage regulators, for 5 Volts and 3.3 Volts. Decoupling capacitors were also included to stabilize the supply voltage leaving the regulators.

F. Printed Circuit Boards

To accommodate for the size restrictions and LED placement, we decided to use two separate types of PCBs. The first is the main PCB, which will house our main components; the power supply, microcontroller, Wi-Fi module and on/off switch, as well as connections to our external devices. The second type of PCB will be LED PCBs, which will house the components for the functionality of the LED, allowing it to be compact and separate from the main PCB.

G. Keypad & Pushbutton

In order to complete the reservation confirmation process and validate the correct user, a keypad and pushbutton were implemented into the design. We decided to use a 4-digit keypad to produce 255 combinations to maintain a secure confirmation while also making the process as easy as possible for users. The pushbutton acts as the ‘send’. We decided to use a large, 10 cm pushbutton to make it noticeable for users.

H. Mobile Application

For the mobile application, we needed a way to provide the users with a seamless connection between them and the hardware device. This applies to both admins, the user who can add devices/study spots and make changes to it, and students, the user who can reserve a study spot. To make this possible, we approached this connection in two different ways:

a) Front-End: For this part, we needed a system that would handle the user interface in an untroubled way. We looked at different technologies to build the front-end on. These are Android, iOS, and React Native. Android uses Java as their primary programming language, which is a language we are mostly comfortable with. For the next techonolgy, iOS uses Swift as their primary programming language. Unfortunately, we were not as familiar with this language at all, making it a big learning curve for us. For our last technology, React Native is a framework that uses JavaScript as their primary programming language. We are more familiar with this language since we have used it previously in other web development projects. In addittion, iOS and Android are native applications, while React Native is considered a cross-platform application. This is an important key decision because we want to be able to launch this application to multiple mobile devices without having the limitation to just one operating system. In the end, React Native was chosen to be the framework that handles our Front-End. This decision was made based on updates/maintenance, cost, support, development, and team knowledge.

b) Back-End: In order to use the database and make it work with our application and hardware device, we needed to make several API calls. With the API call, the information is sent and processed back to the user. There are a few technologies we considered for this such as PHP, Node.js, and Python. PHP is simple to get started with and not much knowledge is needed in order to use it for the project. However, there are scalability issues and it is limited by memory to the number of connections it can support. For Python, the syntax is simple to understand and to use. It also supports asynchronous coding. Unfortunately, Python is not a popular language for mobile app development; therefore, there is less support for this. Lastly, Node.js is great for building RESTful APIs for NoSQL database support. The possible biggest drawback of Node.js is its inability to process CPU bound tasks. Node.js is not generally recommended for heavy computation; however, this would not affect us as much since we are not using heavy computation in our project. In the end, Node.js was chosen to support our Back-End. This decision was made based on mobile compatibility, performance, and IoT compability.

I. *Firebase*

Google Firebase was chosen since it was very simple to implement and low cost. There is a built-in authentication system and supports Robust APIs for JavaScript. It also provides us with a Realtime Database, which is what we are using for our project. In the end between AWS and Azure, Firebase was chosen to support our database. This decision was made based on cost, ease of use, performance, and updates and maintenance support.

IV. HARDWARE DESIGN

For the hardware design we took into account constraints related to power efficiency, size, ease of use, and response time. After creating a successful initial prototype, we created a fully integrated Hardware design, described in detail below.

A. *Hardware Block Diagram*

The hardware design block diagram in Figure 1 describes the overall flow and critical components used in this device. It also demonstrates how components are distributed.

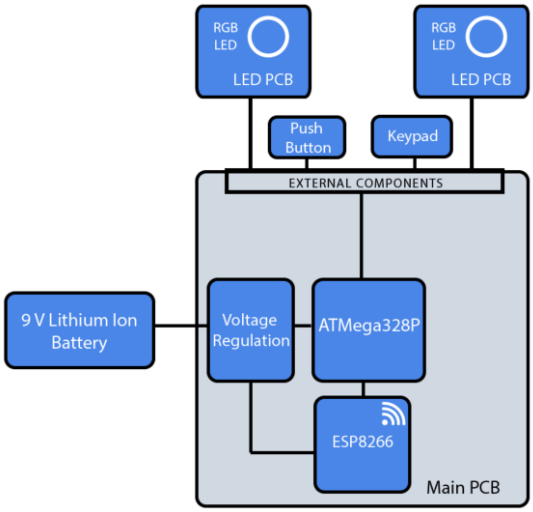


Fig. 2. Hardware Block Diagram

The 9-Volt Lithium Ion battery is attached to the PCB through JST 2-Pin connectors. It enters voltage regulation and then passes to our two main components: the ATMega328P microcontroller and the ESP8266 Wi-Fi module. The Microcontroller and Wi-Fi module communicate with each other, and the rest of the external components are controlled by the ATMega chip. Below, we will discuss each component in more detail.

B. *Main PCB Design*

1) Power Regulation: As mentioned previously, two voltage regulators were used in the main PCB to supply the microcontroller and the ESP Wi-Fi Module. These regulators were chosen due to their low dropout voltage (500 mV). They are also extremely small and have an ultra-low quiescent current of ~10uA. We decided to run the ATMega328 at 5V in order to take advantage of its 16 MHz processing capabilities. The ESP Module has many curent spikes during normal operation, especially when transmitting and receiving information through Wi-Fi, so we included decoupling capacitors of various capacitances to handle any noise/ voltage spikes, as shown in Figure 2.

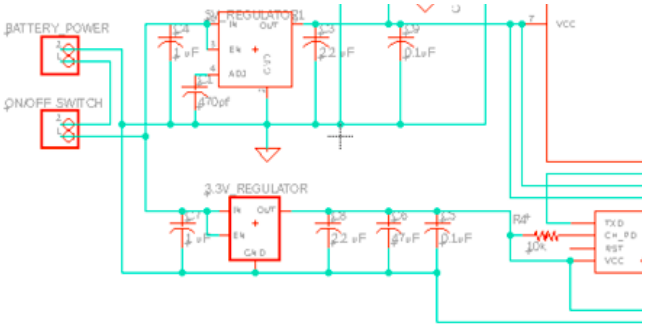


Fig. 1. Power Regulation Schematic

2) ESP8266-01 Wi-Fi Module: The ESP Wi-Fi Module, as shown on Figure 3, has a high enable pin, which requires a 10k pullup resistor. Since this module is a populated ECA, we have attached the traces to female headers on the PCB which the ESP will mate to. This allows for easy removal and replacement of the module. As for communication to the module, we are using UART serial communication to the ATMega328P chip through the TX and RX lines available on the module. This ECA comes fully equipped with a internal 2.4 GHz antenna, so no external antenna is required.

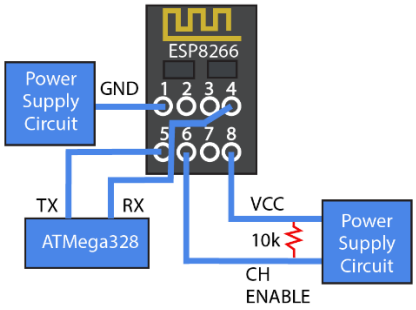


Fig. 3. ESP 8266 Hardware Block Diagram

3) ATMega328P Microcontroller: The microcontroller is located on an IC holder for easy removal and replacement if necessary. However, we have also placed a set of 3 female headers that can be used to upload code onto the microcontroller without removing it from the PCB, for quick debugging. We have also attached a 16MHz crystal for faster processing. The ATMega328P contains 6 PWM channels, 4 of which we are using for LED voltage control, and 2 of which are used for serial communication. We are not using the ADC on the chip for energy consumption purposes.

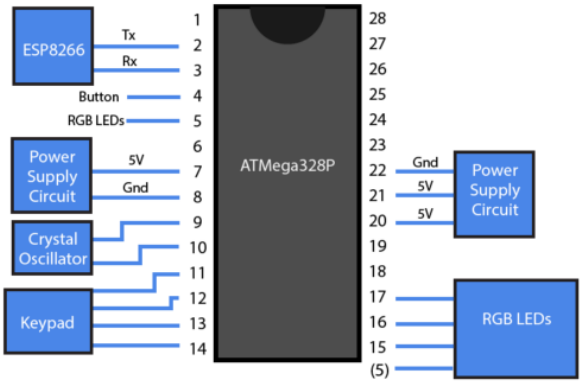


Fig. 4. ATMega328P Microcontroller Schematic

4) Final PCB Design: The final PCB was designed in Eagle and contains external connections to the keypad, pushbutton, on/off switch, and LED PCBs.

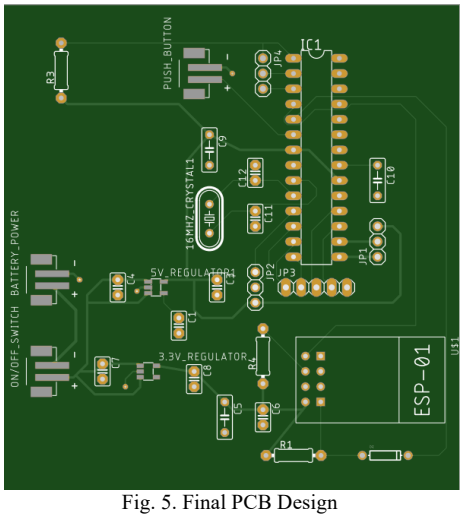


Fig. 5. Final PCB Design

C. LED PCB Design

The LED PCB was created to be separate from the main PCB for it to be placed in the see-through area. The PCB only consists of the necessary components for the functionality of the LED: 2 resistors and header pins to mate to the main PCB. We used the following formula to find the value of the resistors:

$$20\ mA = [5V(Supply)\text{-}1.8V(Drop)] / \text{x}\ \Omega$$

$$\text{x}\ \Omega = 3.2V / 20\ mA$$

$$\text{x}\ \Omega = 180\ \Omega$$

Eqn. 1 LED Resistance Calculations

The closest commercially available resistor was 195 Ohms. The board was designed to be as small as possible. Two of these LED boards are used in the final design, and are located inside of the clear tube, allowing the LED light to shine through from inside the enclosure.

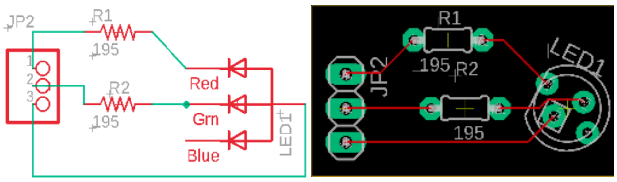


Fig. 6. LED PCB Schematic

Figure 6 shows the PCB schematic. The Blue pin is left floating because it is not used in this project. The necessary colors are green, red, and yellow. yellow was created by mixing green and red together. The final dimensions of this PBC are 3.7 cm by 2.8 cm.

D. Serial Communication:

Serial communication was used for the transmission of data between the microcontroller and the ESP8266. In order to send multiple values of information at once, a data packet was created for both the ATMega chip and the ESP Module. Table 1 demonstrates the data packet for each device.

Device	*Byte 1*	*Byte 2*	*Byte 3*	*Byte 4*	*Byte 5*
Arduino => ESP	Header	Keycode High Byte	Keycode Low Byte	Status	Footer
ESP => Arduino	Header	Valid	Status	Footer	

Table 1. Data Packet for UART Communication

The keycode's maximum value available, using a 4-digit keypad, would be 4444, which is 13 bits, too much to fit into one Byte of transferred data. To resolve this, the integer containing the 4-digit keycode was sent using the *highByte* and *lowByte* command in the Arduino IDE, which divided the integer into two parts. Once received by the ESP Wi-Fi module, the high byte was shifted by 8 and an OR operation was performed with the low byte to obtain the full Keycode. The header and footer were also included in the data packet to be able to properly decode the packets and organize the information.

E. Keypad and Pushbutton

For the confirmation and authentication process, we used a 4-digit keypad and a large 10cm pushbutton. Both components were externally attached to the device and pull its respective GPIO pin to ground when pressed. The keypad presses are only registered when the device is in the "Awaiting" state, through continuous polling. This allows the device to also perform other actions such as checking for updates in the database in the case that the user prematurely releases the reservation. Once 4 digits have been received, the keypad ceases to be polled and the button is polled instead. The press of the push button sends the code to the Wi-Fi module which performs code verification and proceeds accordingly.

F. Hardware Assembly

The enclosure was assembled as 3 separate sections, with the pushbutton and keypad on the top part, the LED PCBs in the middle, and the remaining components on the bottom section. The middle section is covered in reflective tape to increase the effectiveness of the LEDs. A tube covered in reflective tape runs through the middle section, allowing the wires from the top part to reach the PCB without being seen. A mockup can be seen below in figure 7.

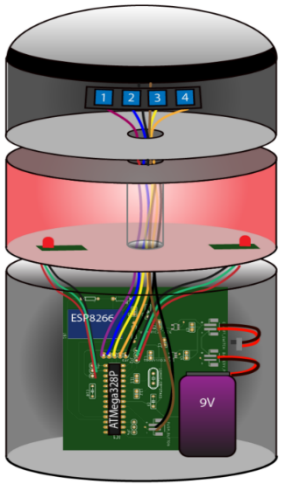


Fig. 7. Hardware Design Mockup

V. SOFTWARE DESIGN

For the software of our project, we took into account efficiency, user experience, and seamless connection technology. In the end, we built a successful mobile application for both types of users: students and admins. The software design is described in detail below.

A. *Use Case Diagram*

The following diagram in Figure 8 represents the use case diagram for both users: students and admins. This figure allows us to explain how the user will be interacting with the device. Since our main source of information is our database, the user never has to interact directly with the hardware device, but with the database itself.

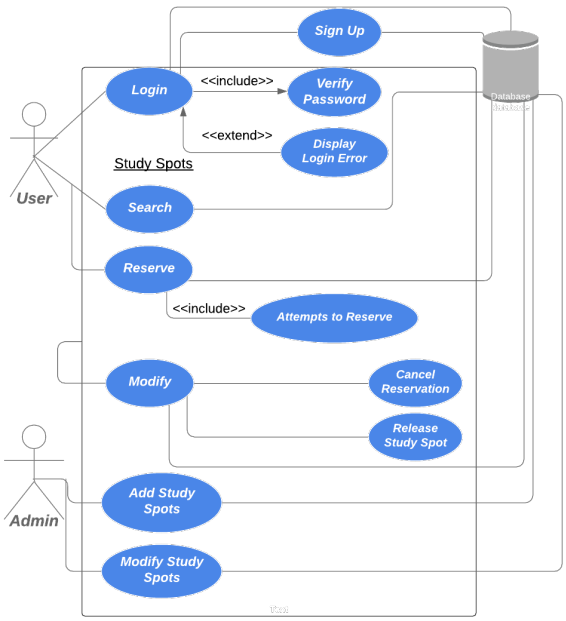


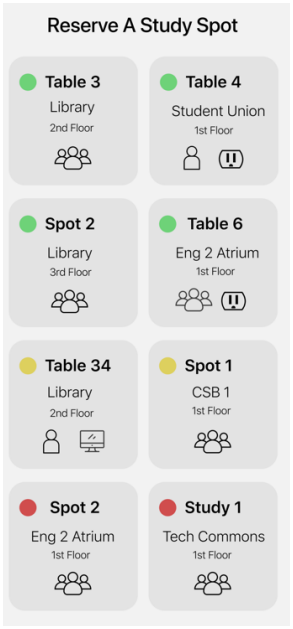
Fig. 8. Use Case Diagram for User Student and Admin

The user student is able to login into the application, and if they have never used it before, they are able to sign up with their university emails. “Verify password” will be a state that will always happen every time the user logins into the application. A display login error message will be brought back to the user if the password is not correct.

For the study spots, the user student is able to search, reserve, and modify study spots. Whenever the user student reserves a spot, the “attempts to reserve” state will be automatically applied. Lastly, the user is able to modify a study spot. That is, release the study spot for someone else to use or cancel their reservation. For the admin side of the application, they are able to add new devices/study spots and modify study spots’ information. IT officials from the university would be ideally the admins for this application.

B. *User Student Interaction*

The user student is our client for this whole system. They are able to search, reserve, and modify study spots all in one single place. Information will be provided to them before they make a decision, and we make the process simple for them to interact with. Figure 9a and figure 9b demonstrate this process.



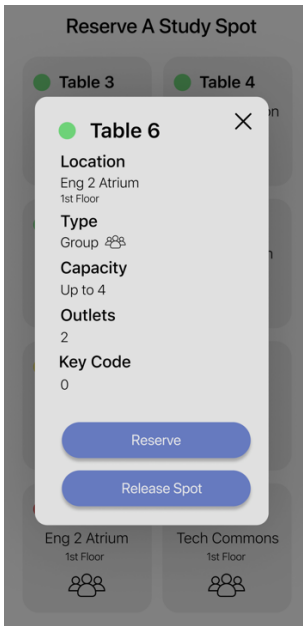


Fig. 9a. Study Spots List UX Design
Fig. 9b. Single Study Spot ready to be reserved UX Design

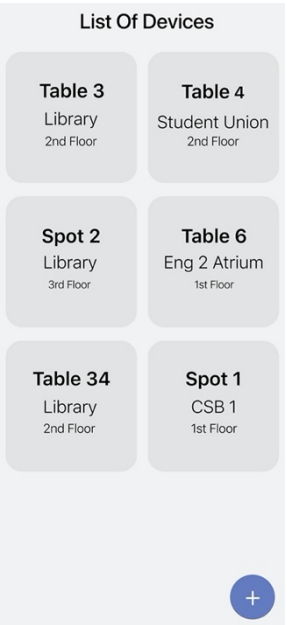
In Figure 9a, the user student is provided with a small amount of information on the study spots/devices. The following fields are currently displayed: name of the study spot, building name, floor, and small icons. These icons represent whether the study spot has outlets, a desktop computer available, or if the spot is for an individual or for a group. We believe this is the right information to be shown at first since the user student can make a decision based on these factors.

In Figure 9b, the user student reaches the second state of the reservation process. This modal shows more information on the device they have selected. We also have a new important field, the key code. This key code is currently set to 0. When the user presses the reserve button, this key code will now have a new random generated 4-digit code value from one to four. This is the key code that they will have to enter in the hardware device keypad in order to finish completing their reservation. They also have the option to release their study spot in case they have made a mistake or do not need the spot anymore.

C. *Admin Interaction*

The admin will have their own application system when interacting with the hardware device. When a new hardware device is manufactured, this device will have their own unique ID. The admin will have to register/add this new ID along with the other information for the device. The following figures 10a and 10b demonstrate this process.

In figure 10a, the admin will have the list of devices currently already set up. The fields that are currently displayed are: name of device, location, and floor number. We believe this is the necessary information to be shown to the admin before they decide to modify a study spot. Whenever they want to add a new device, they can simply press the add button on the bottom right, and that action will take them to the next screen which is figure 10b.



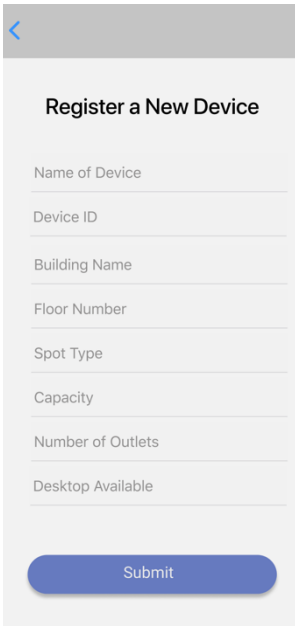


Fig. 10a: UX Design for the List of Devices in the admin side
Fig. 10b. UX Design for the registration of a new device

In figure 10b, the admin is able to register/add a new device. They will need to provide information on the study spot. First, they will need to write the name of the device. This name can be placed on the hardware device as a label, and users will be able to find the study spot this way. The device ID will be the unique ID that identifies each device. The hardware device will need this field in order to make the device functional. The rest of the fields are general information of the device such as where it will be located, spot type, capacity, number of outlets, and if there is a desktop available.

D. Reserving a Study Spot

After the user logins into the mobile application, the user is greeted, as seen in Figure 9a. This figure shows the entire overview of the study spot finders - with their statuses, locations, and the type of study spots (outlets, tables, computers). When a user is able to see a spot with available, as marked with a green dot, only then the user will be able to reserve a spot. Once they press on the spot that reserves their status they are able to view the entire details of that spot – location, type of study spot (group or individual), capacity, number of outlets, and the key code, which will be used to type into the study spot device. When the user is ready to reserve a spot, they will press reserve, which will randomly generate a 4-digit code, each digit being between 1-4, and the device will turn yellow. An alert modal will pop up informing the user that the reservation is pending and must enter the code in the specified study spot. When the user types in the code on the study spot device, the device will turn red, meaning that the spot has been reserved. During the pending reservation phase, is the option to cancel your reservation, by tapping the release spot button, which will reset the status to available and the study spot device will turn back green again.

E. Database and API

The firebase real-time database holds all the study spots and users. The study spots have been registered by the admin and the users have been registered by the users themselves when they sign up for an account in our application.

In Figure 11, we present an entity relationship diagram for the application overall. In this figure we have users as our main part of the application. These users will have a username, which is the email they signed up with at the beginning, and a password of their choice. These passwords will be protected in the database thanks to the built-in authentication system provided by Firebase.

Admins and students are users of the application. That is, they inherit all the attributes of users. For this case, many admins are able to set up many devices. In addition, many students are able to reserve study spots, but only one study spot can be reserved per student. Students will have their student ID, first name, last name, school email, phone number, and university name as part of their profiles.

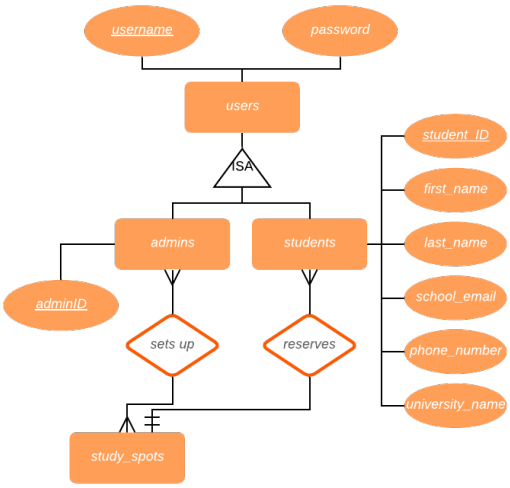


Fig. 11. Entity Relationship Diagram for the application

In Figure 12, we continue the entity relationship diagram from Figure 11. In this figure, we describe the study spots. One or many study spots can have properties such as name, location, spot type, capacity of study spot, number of outlets, and desktop computer available. These study spots will also have a unique ID value which will identify each study spot in the database. These ID will be used by the hardware device and mobile application to modify study spots.

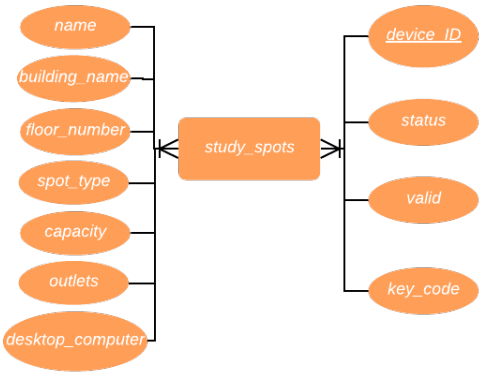


Figure 12: Continued Entity Relationship Diagram for the application focused on study spots

VI. Hardware and Software Integration

Figure 13 demonstrates the hardware and software integration during the reservation process. The hardware and software communicate updates through the database, using a "status" field. Status first start as 1, which means the device is available and the light on the device is green. Then, when the user attempts to reserve a study spot, the status field will change to 2, which means the device is now awaiting confirmation, and the light on the device will be bright yellow. In addition, the code will be generated here for verification purposes. This code will be updated in the database for the application and hardware device to use. Once the user goes to the study spot location, they use the keypad on the device to enter the code. Once this verification is completed by the hardware device, the status is now changed to 3 and the light will be now red, which means the spot has been confirmed and reserved.

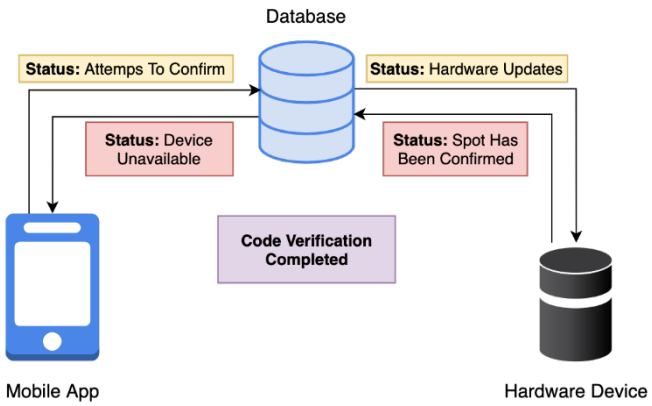


Fig. 13. Software and Hardware Integration

Whenever we want to send a value to the Firebase from the Microcontroller, we send a data packet to the ESP Module. Using functions from the Firebase-Arduino library, the ESP Module sets the corresponding value in the firebase, which updates in real-time for the mobile application.

VII. Conclusion

While the study spot finder project was overall successful, we faced many changes and challenges during the last 4 months. In the beginning, our team did extensive research about new software technologies as well as hardware components, and the most efficient way to integrate them. Then, we decided what features needed to be prioritized in order to complete the project within the time constraint. After deciding on our components and goals, we set out to design our device.

There have been certain challenges that the team encountered during the prototyping stage of this project. On the hardware side, the biggest hurdle was integrating the Wi-Fi module, in terms of communication, voltage stabilization, and flashing software. We also experienced some trouble with lead times on products that were ordered, as well as PCB design. On the software side, we experienced the most difficulties in combining multiple software frameworks and working with the React Native framework in conjunction with Firebase databases. Some other software issues for this project were with debug errors- the use of Google was a major help in this project. Emulation of this app was a bit tricky as one of us had Windows, and the other had MacOS. We had plenty of flexibility when it came to developing, thanks to the React Native Framework, but when it came to testing, we used Expo for the majority of testing in order to make sure our application was working correctly. For integration between Firebase and React Native, we had to implement APIs, which involved reading the Firebase Documentation.

While this group is formed by three computer engineers, the hardware team and software team has been divided into two equal teams where programming skills are required in both sides. By the end of this project, the team have developed new skills in both software and hardware levels. For the hardware aspect, the team will have gained programming skills, hardware design involving power supply, hardware components integration, PCB schematics, design and assembly. On the software aspect, out team has gained experience in multiple programming languages, database environments, and libraries, along with other software development skills. Furthermore, the entire team has gained managerial skills and teamwork.

VIII. The Team

Perla Del Castillo is a senior in Computer Engineering at the University of Central Florida. She has worked on the back end and admin portal for this project. After graduation, she will be joining Deloitte as a Solutions Analyst and work towards getting her MBA.

Maria Ines Herbas Gutierrez is a senior Computer and Industrial Engineering major at the University of Central Florida. She worked on the hardware design of the project. After graduation, she will be working at ASML as a Production Engineer using both majors.

Andrew Kwong is a senior Computer Engineer at the University of Central Florida. He has worked on the front-end mobile development of the project. After graduation he will be investing in the stock market and deciding between multiple job offers.

Darwin Motato is a senior in Electrical Engineering at the University of Central Florida. He has worked on the hardware design and the enclosure for this project. After graduation, he will begin a full-time position as a Test Engineer at Lockheed Martin.

ACKNOWLEDGMENT

The authors wish to acknowledge and thank Dr. Lei Wei and Dr. Samuel Richie for their constant support over the courses of Senior Design I/II.

REFERENCES

[1] "The 16 Most Important Pros and Cons of Using Python for Web Development." Django Stars Blog, 17 Sept. 2019, djangostars.com/blog/python-web-development/.

[2] "5V 40 LEDs Digital WS2812B Programmable Pixel LED Light Ring." Kutop International Limited, kutop.com/5v-40-leds-digital-ws2812b-programmable-pixel-led-light-ring.html.

[3] "ECFR - Code of Federal Regulations." Electronic Code of Federal Regulations (ECFR), www.ecfr.gov.

[4] Gamma, Smart. "What Are the Pros and Cons of Using PHP?" Medium, Medium, 1 June 2016, medium.com/@smartgamma/what-are-the-pros-and-cons-of-using-php-490553ed8ff2.

[5] "The Good and the Bad of Swift Programming Language." AltexSoft, www.altexsoft.com/blog/engineering/the-good-and-the-bad-of-swift-programming-language/.

[6] "The Good and the Bad of .NET Framework Programming." AltexSoft, 28 June 2019, www.altexsoft.com/blog/engineering/the-good-and-the-bad-of-net-framework-programming/.

[7] "IEEE Standards." IEEE, www.ieee.org/standards/index.html.

[8] "LED 101: Identifying Different Types of LEDs." Electronic Products, 19Apr.2018,www.electronicproducts.com/Optoelectronics/LEDs/LED_101_Identifying_different_types_of_LEDs.aspx.

[9] "LED STRIP LIGHTS." Everything You Need to Know About LED Strip Lights | Waveform Lighting, www.waveformlighting.com/led-strip-lights

[10] Marrakchi, David. "Top 5 PCB Design Guidelines Every PCB Designer Needs to Know." Altium Resources, 30 Sept. 2019, resources.altium.com/pcb-design-blog/top-pcb-design-guidelines-every-pcb-designer-needs-to-know.

[11] "Native vs. Cross-Platform Apps: The Startup Dilemma." Skelia, 25 Sept. 2019, skelia.com/articles/the-startup-dilemma-native-vs-cross-platform-apps/.

[12] Smith, W.A. "ESP8266 Testing." Starting Electronics, Electronics for Beginners, Hobbyists and Beyond, startingelectronics.org/articles/ESP8266-testing