Desk Reservation System

ECE 445 Design Document - Spring 2020

Team 37: Pragya Aneja, Ninad Godbole, Varad Khandelwal

TA: Shaoyu Meng

Table of Contents

1. Introduction

1.1 Objective

As a student, libraries are a fundamental part of college life. It is therefore vital that the student is able to find a place to sit to be able to study effectively. However, due to overcrowding it can become very difficult to find a spot, and even more so if one has a group study session. Additionally, as different students have vastly different timetables, coming early into the library to save a spot is also hard. Furthermore, it can regularly happen that even though there are some desks available, it takes a lot of time to scan the entire library to find it. For example, in Grainger Engineering Library there are 5 floors and to find a desk to sit at can take a lot of time which then reduces the student's time to study. This problem only exacerbates during exam time, where finding alone, distraction free time becomes increasingly more of a necessity. As can be seen, there are a variety of issues that are involved when it comes to a student being able to reliably find a spot to study in a place like the Grainger Engineering Library.

Hence, our main objective and goal for this project is to reduce the inefficiency involved in a student trying to find a spot to study. Our solution consists of designing a desk reservation system where one can reserve a desk in advance at a specific time for upto a few hours. This system will increase space efficiency for a library, allow the student to more conveniently find a place to study and allow the student to make more prudent use of his/her time.

1.2 Background

Today, reservation systems are used in a plethora of settings and contexts and have been for a while. One example of this is in the hospitality industry, i.e. booking rooms for a hotel. By allowing customers to use an online reservation system to choose the type of room and duration of stay, the customers have a more streamlined experience [1]. Another example of this is setting up an appointment to get a haircut. Scheduling the haircut beforehand increases efficiency by making it so that there isn't overcrowding and large wait times in the hair salon [2]. The last example worth discussing is that of reserving study rooms for libraries for universities, which is probably the most related to our project. Similar to the first example regarding the hospitality industry, giving the students the option to reserve study rooms improves their experience.

All these examples are inspiration for the desk reservation system project, which incorporate aspects from the examples mentioned above like improved efficiency through reduced overcrowding and waiting times as well as a better overall experience for the student. The concept of reserving a specific desk in a library isn't that popular most probably because of the cost required due to each desk requiring some sort of communication subsystem. Therefore, our project is trying to look into creating this system while keeping the cost as low as possible.

1.3 Visual Aid

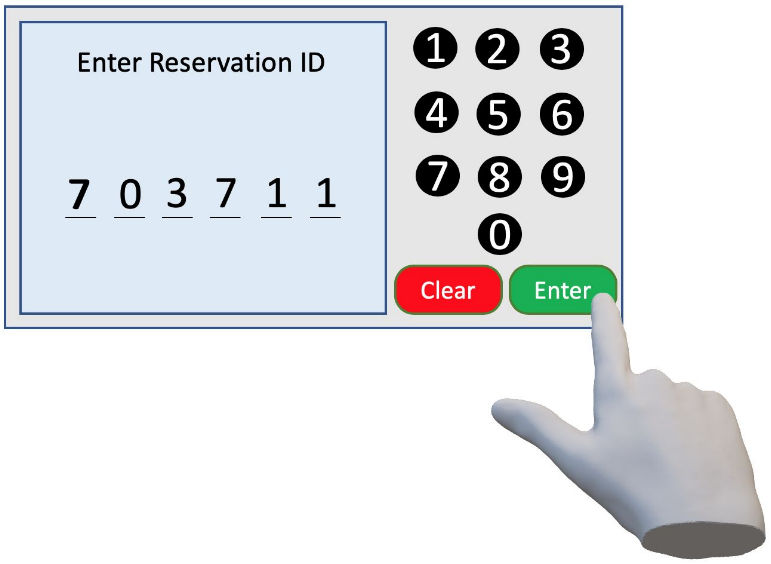

Figure 1: Student entering six digit reservation ID into master unit

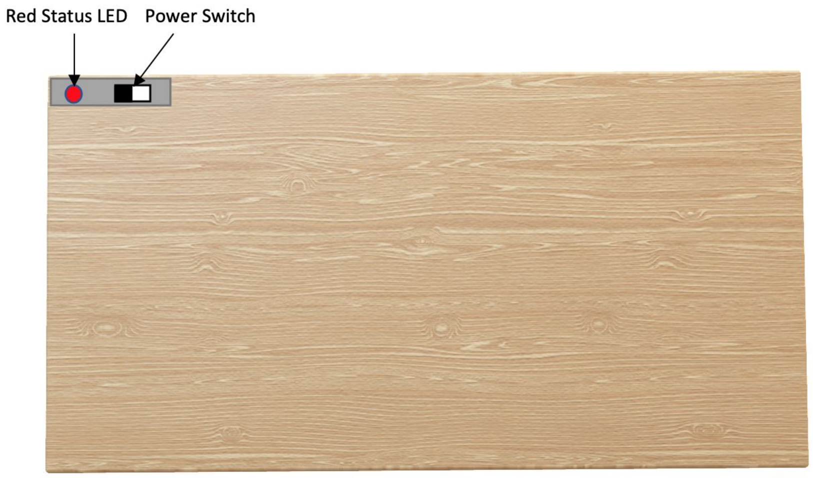

Figure 2: Top Down View of Desk with Slave Unit (consisting of red status LED and power switch) attached on the top left

1.4 High-Level Requirements

1. Within a span of 15 seconds, the student must be able to reserve a particular desk from a desk map corresponding to the chosen time slot and receive confirmation of the reservation with a unique 6 digit reservation ID. The student can also cancel this reservation until 15 minutes of the reservation time.
2. The student must be able to input the 6 digit reservation ID into the master unit via keypad, which should then map to the corresponding desk and turn on the status LED on that desk red within 5 seconds.
3. After 15 minutes, if the student does not confirm the reservation, the central system must add the UIN to a database which it keeps track of. The student must be penalized after the third non-confirmation by removing his/her right to reserve a desk for a period of a month.

1.5 Further Expansion

An interesting idea that can be applied for this project given the current context is that concept of social distancing when assigning desks to students trying to reserve them. Due to the novel coronavirus, social distancing has become a very important paradigm in combating the spread. Therefore something that we thought we could do to adapt our project as a further expansion even more to the current situation is designing algorithms that assign desks to students while abiding by the guidelines of social distancing.

2. Design

2.1 System Overview (Functional overview) and Flowchart

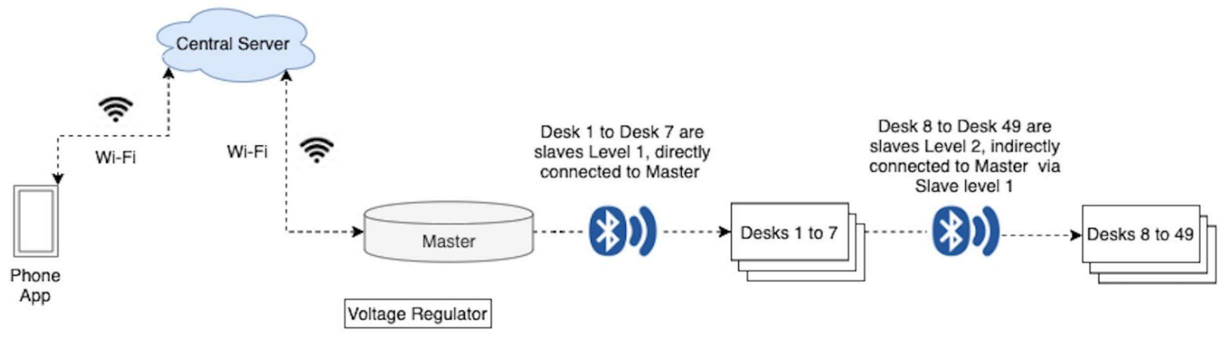
Central Server
Wi-Fi
Wi-Fi
Phone App
Master
Voltage Regulator
Desk 1 to Desk 7 are slaves Level 1, directly connected to Master
Desks 1 to 7
Desk 8 to Desk 49 are slaves Level 2, indirectly connected to Master via Slave level 1
Desks 8 to 49

Figure 3: Top-level System Design

The desk reservation system consists of three components: a phone app as the user interface, a master unit which controls reservations for 49 desks, and each individual desk. The master unit and its 49 slave desks together form a *bluetooth scatternet*, where the master unit along with 7 level-1 slave units form a piconet, which is also connected to piconets formed by these level-1 slave units with each of their 7 level-2 slave units. Whenever a user wants to reserve a desk, he/she must open the phone app and input the timings for which a desk should be reserved, in addition to his/her UIN. This information would then be communicated via WiFi to a central server, which would then return an accurate seat map of the library with desks available for reservation marked green. The user could now select any desk marked green, and upon receival of this information at the central server, the server would generate a unique reservation ID mapped to the user's UIN. This reservation ID along with the desk ID and the timings of the reservation would be sent to the master unit associated with the reserved desk via WiFi. Thus, the user will have the reservation ID and the master unit associated with the reserved desk stored in his/her app, and the master unit in charge of the desk will have the reservation ID, timings and desk ID of the reserved desk.

At the time of the reservation, the user must arrive at the library, walk upto the master unit associated with his/her reserved desk, and feed the reservation ID received earlier into the master unit's numeric keypad. The master unit would send the initiation information to the central server, so that if the reservation was initiated more than 15 minutes after the start time of the reservation, a penalty point would be added to the UIN associated with this reservation. This would allow us to discourage false bookings, as we would not allow any user with over 3 penalties to conduct any reservations during peak hours. The master unit would then display the desk ID associated with this reservation to the user via an LED screen, and communicate the desk ID to all the slave-I desks via Bluetooth. If the desk ID in the message matches the ID of one of these level-1 slave desks, a status LED placed on the corner of the desk would light up red. If not, each level-1 slave would pass on the message to all seven of its level-2 slave desks, and if the desk ID in the message matches any desks' ID, the status LED on the corner of that desk would turn red. The LED turning red would act as an indicator for anyone sitting on the reserved desk to leave, as the reservation has now been initiated. Then the user would walk upto the desk and use it for the allotted time.

The master unit would maintain a list of all the *passive* reservations communicated to it by the central server. Whenever a reservation gets initiated by a user, it would be shifted to an *active* reservations list. Thus, it would check for change of status in any of the active desk reservations by tallying the current time with the timings of the reservation every 15 minutes. This would allow it to track if any reservations that were active earlier have now ended. It would send the updated status of all reservations to its slave desks to ensure that the status LED for any desk that had turned red at the initiation of a reservation turns off within 15 minutes of that reservation ending. Additionally, all the active reservations that have now ended would be shifted back to the *passive* list. At the end of the day, the master unit would remove all the expired reservations from the passive list, thus eliminating any unrequired storage.

Below is the flowchart for the entire process split up into two parts:

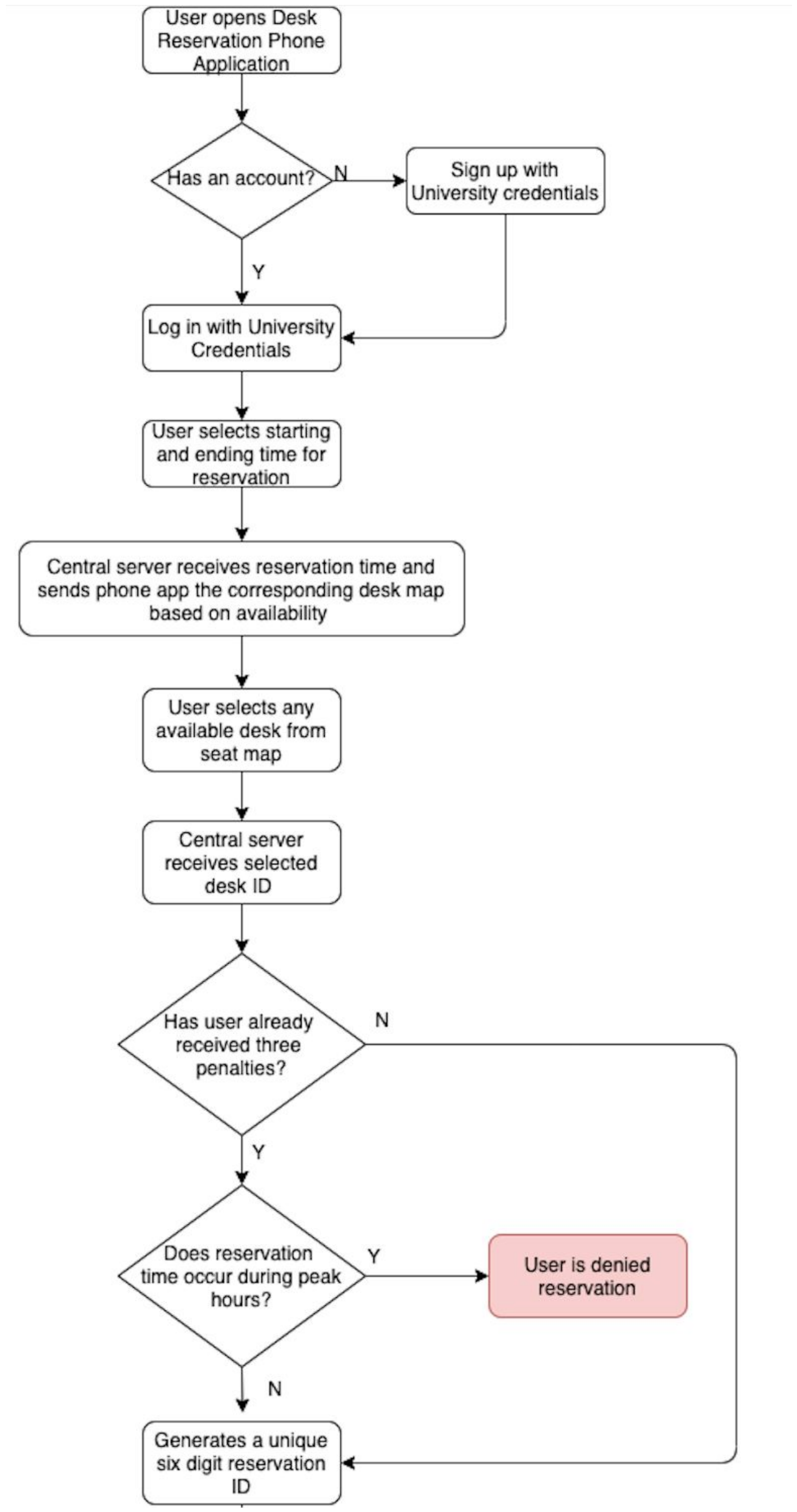
User opens Desk Reservation Phone Application
Has an account?
N
Sign up with University credentials
Y
Log in with University Credentials
User selects starting and ending time for reservation
Central server receives reservation time and sends phone app the corresponding desk map based on availability
User selects any available desk from seat map
Central server receives selected desk ID
Has user already received three penalties?
N
Y
Does reservation time occur during peak hours?
Y
User is denied reservation
N
Generates a unique six digit reservation ID

Figure 4: Flowchart until a six digit reservation ID is generated

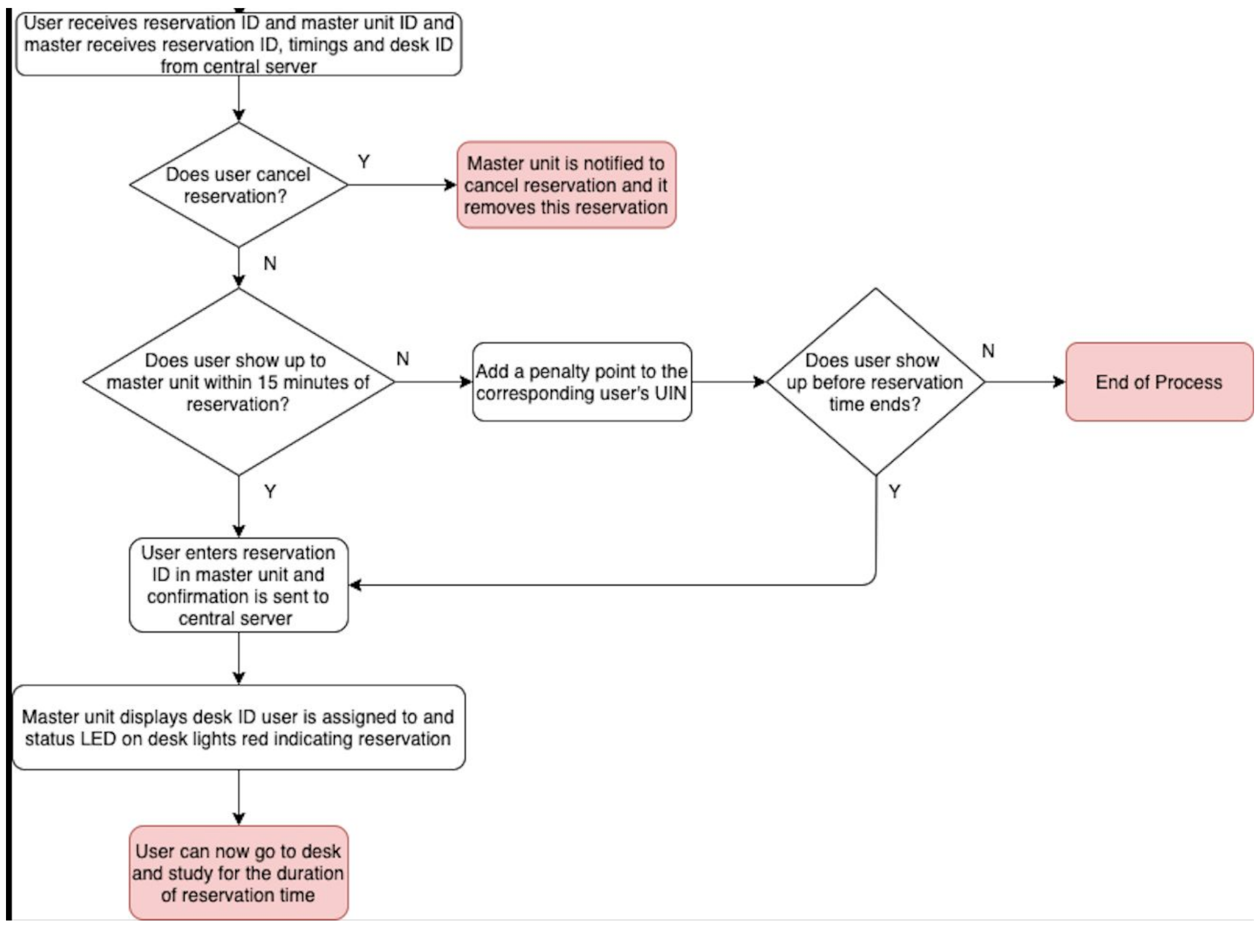
User receives reservation ID and master unit ID and master receives reservation ID, timings and desk ID from central server
Does user cancel reservation?
Y
Master unit is notified to cancel reservation and it removes this reservation
N
Does user show up to master unit within 15 minutes of reservation?
N
Add a penalty point to the corresponding user's UIN
Does user show up before reservation time ends?
N
End of Process
Y
Y
User enters reservation ID in master unit and confirmation is sent to central server
Master unit displays desk ID user is assigned to and status LED on desk lights red indicating reservation
User can now go to desk and study for the duration of reservation time

Figure 5: Flowchart for rest of process after generation of reservation ID

2.2 Block Diagrams

2.2.1 Master Unit

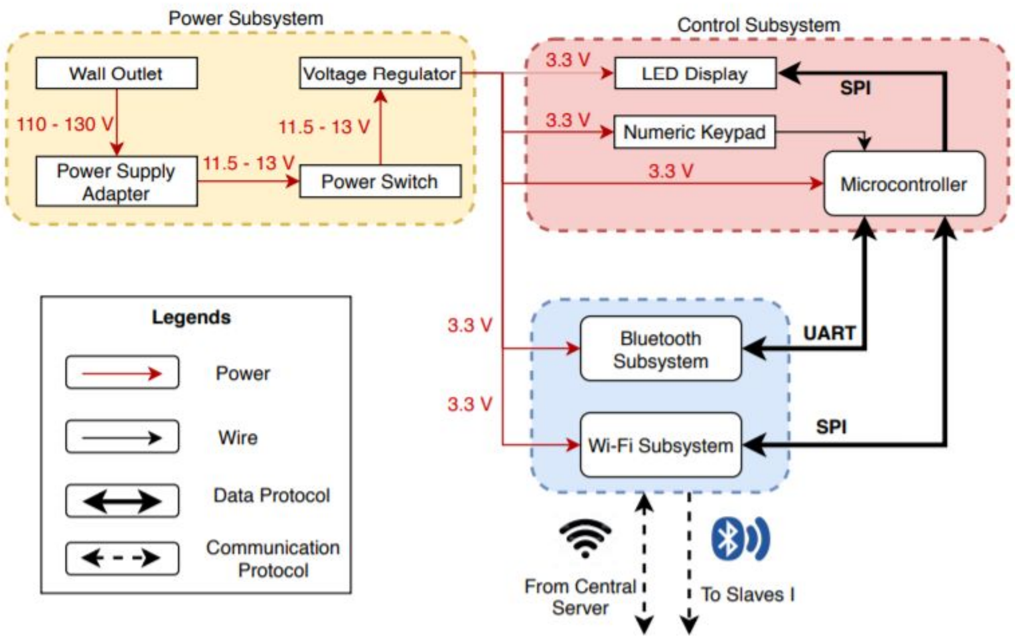

Figure 6: Master Unit Block Diagram

2.2.2 Slave Units - Level 1

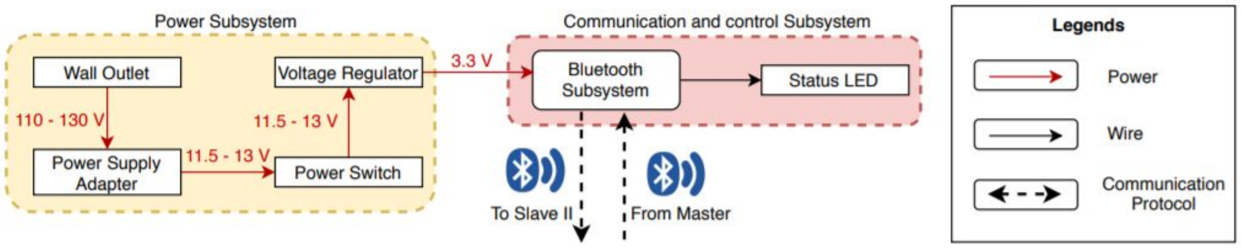

Figure 7: Slave Unit Level 1 Block Diagram

2.2.3 Slave Units - Level 2

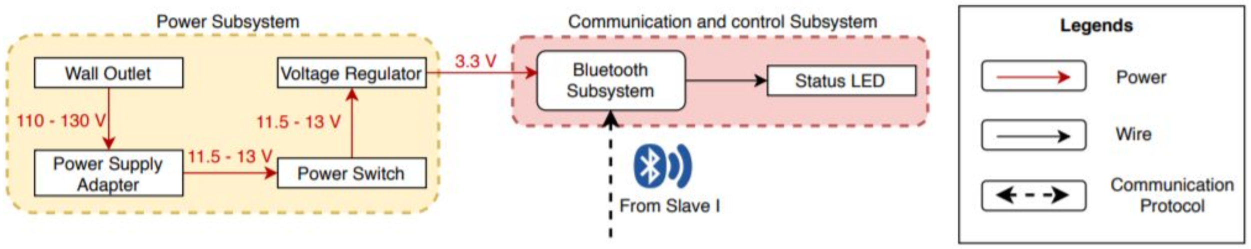

Figure 8: Slave Unit Level 2 Block Diagram

2.3 Physical Diagrams

2.3.1 Master Unit & Slave Units I, II

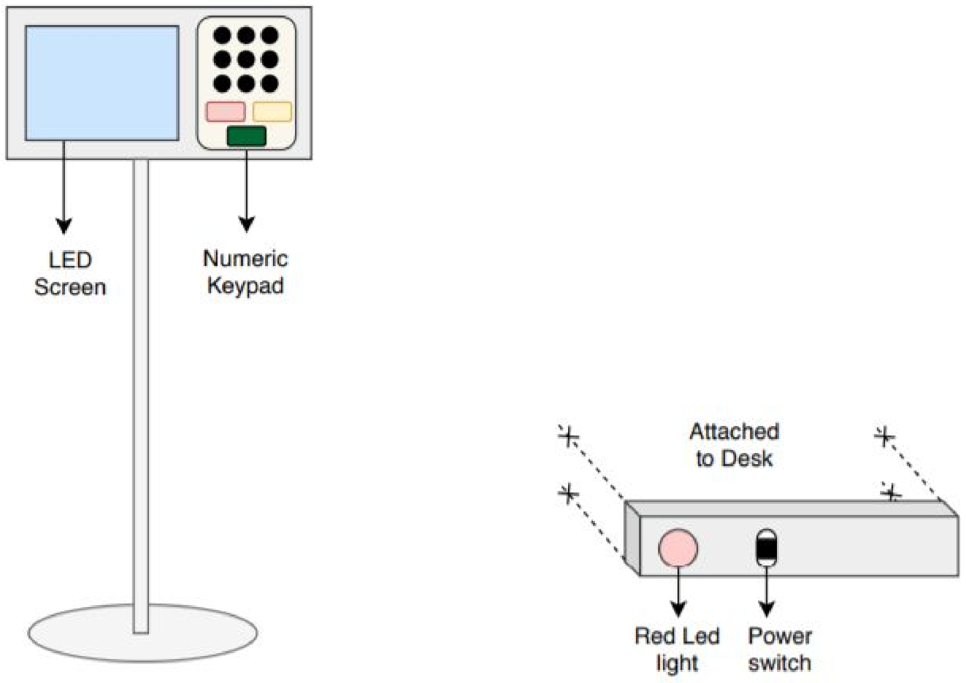

Figure 9: Physical Diagram of Master Unit (seen on left) and Slave Unit (seen on right)

2.4 Master Unit Subsystems

2.4.1 Power Subsystem

Power Supply adapter

As we plug in all the units into wall power, we will need a power adapter to step down voltage from 120V to 12V. The power adapter has to be cheap as this scales up with the desk and should have a cord, so we can fit the desk module at the edge of the desk for visibility and still be able to connect it to a wall plug.

Requirements	Verifications
***Requirement 1*:** Should output 11- 13V from an input of 110-130V.	a. Connect the power adapter to a wall plug and probe the output with a multimeter. b. The multimeter should read between 11 V and 13 V.
Requirement 2: Should have 2 ft long cord.	a. Take the power adapter and stretch it completely. b. Take a measuring scale and measure from its teeth to the endpoint which has a DC jack. c. The cord should at least measure 0.69 m or 24 inches.

Power switch

We don’t want the device to run indefinitely, and would like to switch it off if the library is on a break or not in use. Thus, we need a switch which disconnects the voltage regulator from the battery when switched off.

Requirements	Verifications
Requirement 1: A switch which disconnects the voltage regulator from battery when off and provides very less voltage (<=0.2 Volts) drop when switched on.	a. Connect the fully-charged li-ion battery with VDD and GND. b. Use a multimeter to check if the voltage output is in the specified range.

Voltage regulator

As we are using a power adapter, the voltage output would be around 12V and would fluctuate quite a bit between 11V-13V. The voltage regulator would need to reduce it and step it down. Additionally, the voltage regulator needs to be cheap, but it doesn't need to be energy efficient as we have input coming from the wall plug. Another requirement is that we have a lot of bluetooth networks and we would prefer it if the voltage regulator creates no interference. This all points out to using a linear regulator as compared to switching, as switching is more expensive and creates interference.

Requirements	Verifications
***Requirement 1*:** Can output stable 3.2-3.3V from an input supply of 11-13V.	a. Connect the fully-charged li-ion battery with VDD and GND. b. Use a multimeter to check if the voltage output is in the specified range.
Requirement 2: Maintains thermal stability below 50°C.	a. Connect a fully-charged lithium-ion battery with VDD and GND. b. Draw 250 mA current from the battery for 5-6 hrs. c. Monitor output voltage battery and ensure it does not drop below 3.5V earlier.

2.4.2 Control Subsystem

Microcontroller

The microcontroller should be compatible with the bluetooth module, WiFi module, LED Display and numeric keypad. It should communicate with the bluetooth module via UART and the LED Display and WiFi module via SPI.

1. It should be able to receive reservation ID along with the desk ID and the timings of the reservation via SPI from the WiFi module which would receive this information from the central server's broadcast.
2. It should take the reservation ID input from the numeric keypad and should send the desk ID associated with it via SPI protocol to the LED display.
3. It should send the desk ID whose reservation has just been initiated to the bluetooth module, which would communicate this to all the slave desks via bluetooth.
4. It should send the initiation information to the WiFi module, which would then forward it to the central server to keep track of penalty points if it is initiated more than 15 minutes after start time.

5. It must maintain *active* and *passive* reservation lists, such that every new reservation received via WiFi should be added to the *passive* list and whenever a reservation is initiated, it should be shifted from *passive* to *active*.
6. Every 15 minutes, it must tally the current time with the timings of all *active* reservations. Then it should send a bluetooth message to all of its slave desks containing information of the desks whose reservation has ended, and those desks should be removed from its list of reservations. Additionally, it must shift the reservation to *passive*, and all expired passive reservations must be deleted from storage at the end of each day.

Calculation 1: Required memory storage
The microcontroller needs to maintain two lists: *active* and *passive* reservations, which would store reservations associated with any of its slave desks. At any instant, assume that reservations for the next seven days for 49 slave desks are being stored, where each reservation must be for a minimum of 30 minutes. Each entry in the list is < 32 bytes as:

a) Reservation ID (6 digits) = 20 bits
b) Desk ID (2 digits) = 6 bits
c) Timings of reservation = 13*2 bytes to store start time and end time (UTC)

Thus for maintaining the lists, required storage:

$$49 \times 48 \times 7 \times 32 = 526{,}848 \text{ bytes} \approx 0.6 \text{ MB}$$

Calculation 2: Required communication speeds
In the worst case scenario, say that the master unit receives information regarding reservations associated with all its slave desks for a week via WiFi at the same instant. Thus, it would be receiving 49x48x7 reservations in a second, where each reservation is 32 bytes. In this scenario, the required speed of communication with the WiFi module would be
$49 \times 48 \times 7 \times 32 = 526{,}848$ bytes/second ≈ 0.6 MBps

Requirements	Verifications
Requirement 1: The microcontroller must be able to communicate over UART protocol with the bluetooth module.	a. Connect a microcontroller to a USB-UART bridge, like CP2102. b. Using a terminal, send data via the UART bus and verify whether this is the same as the data echoed back.
Requirement 2: The microcontroller must be able to communicate over SPI protocol with the LED display and the WiFi module.	a. Connect microcontroller to a USB-SPI bridge, like MCP 2210. b. Using a terminal, send data via the SPI bus and verify whether this is the same as the data echoed back.
Requirement 3: The microcontroller should be able to communicate over	a. Connect the microcontroller to a USB-SPI or USB-UART bridge. b. Send a 0.6 MB block of data from the USB bridge.

UART and SPI at speeds greater than 0.6 Mbps.	c. Echo data back over SPI or UART. d. Ensure that the data received matches the data sent, and time elapsed < 1s.

LED Display

The LED must display the desk ID associated with the reservation ID input by the user. It receives this desk ID from the microcontroller via SPI and displays it such that the user is notified the reservation has been initiated and is aware of the desk ID associated with his/her reservation. It must clearly display the 2-digit desk ID.

Requirements	Verifications
***Requirement 1*:** Must be able to receive input desk ID from microcontroller via SPI.	a. Connect the LED display to the microcontroller. b. Using the SPI communication procedure, seen above, send data and verify this data is correctly displayed on the LEDs.

Numeric Keypad

The numeric keypad must be a 10-digit keypad which takes a 6-digit reservation ID as input from the user, and passes the information to the microcontroller.

Requirements	Verifications
***Requirement 1*:** Must be able to receive input reservation ID and communicate it to the microcontroller.	a. Connect the keypad to the microcontroller. b. Enter a 6-digit number into the keypad and check if the microcontroller received the number correctly.

High level Schematic connection

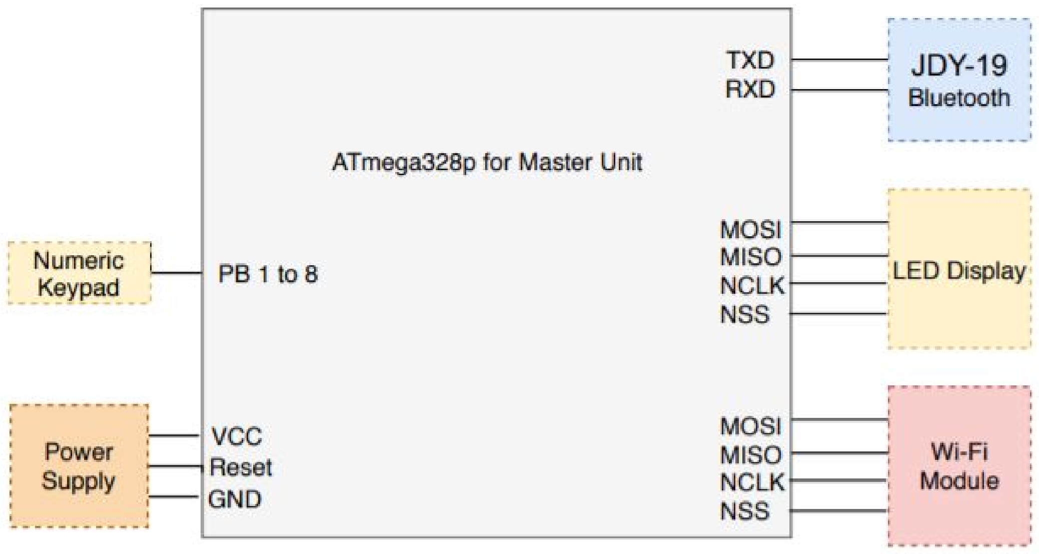

Figure 10: ATmega328P connection with other components

Pin layout

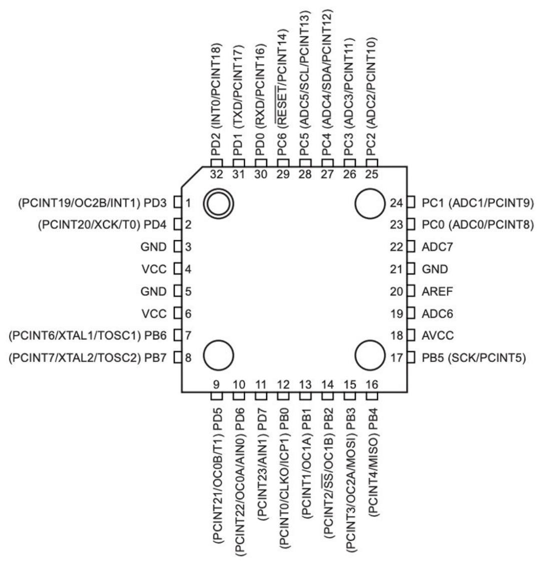

Figure 11: Pin layout for ATmega328P [3]

2.4.3 Communication Subsystem

Bluetooth Module

The master unit and its 49 slave desks together form a *bluetooth scatternet*, where the master unit along with 7 level-1 slave units form a piconet, which is also connected to piconets formed by these level-1 slave units with each of their 6 level-2 slave units. Bluetooth Module in the master unit would be assigned as master in the piconet, and this master would be required to maintain 7 simultaneous connections with the 7 level-1 slave units. Each bluetooth module has a unique address associated with it and this unique address is used to form the piconet. Additionally, in a piconet the slave bluetooth devices are typically within 10-15 metres of the master device. It doesn't need to be very cost effective as it will only be used once per 49 desks and has to be only moderately energy efficient as the master unit has a huge power supply.

1. Once a reservation gets initiated by the user, the bluetooth module must receive the desk ID associated with the reservation via UART from the microcontroller. Additionally, while checking every 15-minutes, all the reservations that the microcontroller finds to be passive must also be received at the module.
2. This received desk ID must then be communicated to all slaves in the piconet (7 level-1 slave units), such that if any of their desk ID matches with this communicated desk ID, their status LED turns red.
3. The inactive reservations received every 15 minutes must also be communicated to all slaves in the piconet, such that if any of their desk ID was associated with said reservation, they would be instructed to turn off the status LED.

Calculation: At the initiation or expiration of a reservation or expiration, the bluetooth module would receive the desk ID associated with the reservation from the microcontroller. It is necessary for this information to be passed along to the slave desk with this desk ID instantaneously (~1 ms). The information packet should contain the desk ID along with a status variable to indicate that the status LED should be turned on/off. This information would require 1 byte to be contained, and must be received by all level-1 slave desks in half the required time, so that the rest of the time can be used by them to forward the information to their level-2 slaves (if their desk ID does not match with the one contained in the information packet).

Thus, required speed for bluetooth communication = 1 byte/0.5 ms = 2kBps

Requirements	Verifications
***Requirement 1*:** It should be able to connect to seven bluetooth devices simultaneously within a 10m range.	a. Send an information packet to a bluetooth device stationed 10m away. b. Ensure that the packet is received by bluetooth module through connection to a microcontroller.

Requirement 2: It must be able to communicate with other bluetooth devices at ~2kBps.	c. Send an information packet of 2kB to a bluetooth device stationed 10m away. d. Ensure that the packet is received by the device in 1 s by connecting to a microcontroller.
Requirement 3: Should be able to communicate over UART protocol with a microcontroller.	a. Connect the bluetooth module to the UART input port of the microcontroller. b. Using a terminal, send data via the UART bus and verify whether data is received by bluetooth module.

Wi-Fi Module

The WiFi module in the master unit would need to perform basic data retrieval from the central server and be moderately energy efficient to ensure the master works over 2-3 days on the battery.

1. When a user reserves a desk at any given time, the central server would forward the reservation information via WiFi and the master unit of the piconet which the reserved desk is part of must successfully receive it.
2. When a user cancels his/her reservation via the phone app, this information must also be received by the master unit associated with the reservation via WiFi from the central server.
3. When a reservation is initiated by the user at the master unit, the central server must be informed of the initiation via WiFi to make sure that penalty points are added for the user if reservation is initiated more than 15 minutes after start time. The WiFi module would receive the initiation information from the microcontroller.

Calculation: Required bandwidth
The channel bandwidth of a wireless network connection determines the data's signal transfer rate. In the worst case scenario, say that all the desks associated with a certain master unit are reserved at the same instant. The central server must then send an information packet with information regarding all these reservations to the master unit of this scatternet via WiFi, and the information packet must be received by WiFi within a second. Thus the WiFi module would be receiving 49x48x7 reservations in a second, where each reservation is 32 bytes. In this scenario, the required speed of communication with the WiFi module would be

49x48x7x32 = 526,848 bytes/second ≈ 0.6 MBps

Requirements	Verifications
Requirement 1: It should be able to receive update requests from the	a. Connect the WiFi module to the SPI output port of the microcontroller. b. Send an information packet to the WiFi module

central server, and be able to send POST requests, in short basic API communication with the backend.	with a POST request. c. Check the central server's database to see if the information packet was received.
Requirement 2: Should be able to communicate over SPI protocol with the microcontroller to receive initiation information and send new reservations.	a. Connect the WiFi module to the SPI input port of the microcontroller. b. Using a terminal, send data via the SPI bus and verify whether data is received by WiFi module.
Requirement 2: The WiFi module should be able to connect to a network with bandwidth of > 0.6 Mbps.	a. Connect the WiFi module to the SPI input port of the microcontroller. b. Using a terminal, connect the WiFi module to a network with high bandwidth eg. IllinoisNet. c. Send a POST request to the central server with an information packet of 0.6Mb and verify whether data is received at the server within 1s.

2.5 Slave I & II Unit Subsystems

2.5.1 Power Subsystem

Power Supply adapter

As we plug in all the units into wall power, we will need a power adapter to step down voltage from 120V to 12V. The power adapter has to be cheap as this scales up with the desk and should have a cord, so we can fit the desk module at the edge of the desk for visibility and still be able to connect it to a wall plug.

Requirements	Verifications
***Requirement 1*:** Should output 11- 13V from an input of 110-130V.	a. Connect the power adapter to a wall plug and probe the output with a multimeter. b. The multimeter should read between 11 V and 13 V.
Requirement 2: Should have 2 ft long cord.	a. Take the power adapter and stretch it completely. b. Take a measuring scale and measure from its teeth to the endpoint which has a DC jack. c. The cord should at least measure 0.69 m or

	24 inches.

Power switch

We don't want the device to run indefinitely, and would like to switch it off if the library is on a break or not in use. Thus, we need a switch which disconnects the voltage regulator from the battery when switched off.

Requirements	Verifications
Requirement 1: A switch which disconnects the voltage regulator from battery when off and provides very less voltage (<=0.2 Volts) drop when switched on.	a. Connect the fully-charged li-ion battery with VDD and GND. b. Use a multimeter to check if the voltage output is in the specified range.

Voltage regulator

As we are using a power adapter, the voltage output would be around 12V and would fluctuate quite a bit between 11V-13V. The voltage regulator would need to reduce it and step it down. Additionally, the voltage regulator needs to be cheap, but it doesn't need to be energy efficient as we have input coming from the wall plug. Another requirement is that we have a lot of bluetooth networks and we would prefer it if the voltage regulator creates no interference. This all points out to using a linear regulator as compared to switching, as switching is more expensive and creates interference.

Requirements	Verifications
***Requirement 1*:** Can output stable 3.2-3.3V from an input supply of 11-13V.	a. Connect the fully-charged li-ion battery with VDD and GND. b. Use a multimeter to check if the voltage output is in the specified range.
Requirement 2: Maintains thermal stability below 50°C.	a. Connect a fully-charged lithium-ion battery with VDD and GND. b. Draw 250 mA current from the battery for 5-6 hrs. c. Monitor output voltage battery and ensure it does not drop below 3.5V earlier.

2.5.2 Control and Communication Subsystem

Bluetooth Module

The master unit and its 49 slave desks together form a *bluetooth scatternet*, where the master unit along with 7 level-1 slave units form a piconet, which is also connected to piconets formed by these level-1 slave units with each of their 6 level-2 slave units. Each bluetooth Module in the level-1 slave units would be assigned as master in a piconet, and would be required to maintain 6 simultaneous connections with the 6 level-2 slave units. The bluetooth modules in the level-2 slave units would be slaves in their corresponding piconets. The bluetooth module used in the slave unit and placed at the corner of each desk will be the main component of our slave system and would need to perform several tasks.

1. It should be really cheap as it would be used at every desk and hence a $1 increase in the cost would correspond to a $1000-1200 in the project cost for a library like grainger. For UIUC, a $1 increase in bluetooth module's cost would correspond to an increase of $40,000-50,000.
2. When a reservation is initiated at the master unit or an active reservation expires, the bluetooth module must receive the desk ID associated with the reservation from the master unit.
3. It must be able to check if the desk ID in the information packet matches its desk ID, if not it must forward the information packet to its level-2 slaves i.e the piconet in which it is the master. Thus it should be able to perform bare minimum control logic operations.
4. It must be able to turn an LED light on/off without the help of a microcontroller if its desk ID matches the ID in the information packet based on the status bit in the packet.

Calculation: At the initiation or expiration of a reservation or expiration, the bluetooth modules in the level-1 slave units would receive the desk ID associated with the reservation via bluetooth instantaneously (~1ms). It is necessary for this information to be received, and if the desk ID does not match, be passed along to level-2 slave units associated with the piconets that have these level-1 units as master. The information packet should contain the desk ID along with a status variable to indicate whether the status LED should be turned *on* or *off*. This information would require 1 byte to be contained, and must be received by all level-1 slave desks in 0.05 ms such that the rest of the time can be used for forwarding the information packet to level-2 slaves if required.

Thus, required speed for bluetooth communication = 1 byte/0.5 ms = 2kBps

Requirements	Verifications

Requirement 1**:** It should be able to connect to seven bluetooth devices simultaneously within a 10m range.	a. Send an information packet to a bluetooth device stationed 10m away. b. Ensure that the packet is received by bluetooth module through connection to a microcontroller.
Requirement 2: It must be able to communicate with other bluetooth devices at ~2kBps.	a. Send an information packet of 2kB to a bluetooth device stationed 10m away. b. Ensure that the packet is received by the device in 1 s by connecting to a microcontroller.
Requirement 3: Should be able to communicate over UART protocol with a microcontroller.	a. Connect the bluetooth module to the UART input port of the microcontroller. b. Using a terminal, send data via the UART bus and verify whether data is received by bluetooth module.

Status LED

The status LED on the corner of the reserved desk must light up when instructed by the bluetooth module, to indicate that the reservation for this desk has been initiated and anyone else using this desk must leave. Additionally, it must turn off when instructed by the bluetooth module as well, to indicate that the reservation for this desk is now inactive, and the desk is free for public use.

Requirements	Verifications
Requirement**:** Must take input from the bluetooth module and must be red in colour in order to be clearly visible from 1 m distance.	a. Connect the bluetooth module to status LED. b. Send a signal from the module to the status LED to turn on and verify that it flashes red.

2.6 Software

2.6.1 Phone Application

The phone application will be the primary means for the student to be able to manage his or her reservations. The student before anything will have to sign up using the university credentials (UIN and any password). Once the student signs up, he or she should be able to log in with no issues. After logging in, the student will be directed to a page where he or she can choose the reservation time and date. This will then direct the student to a page where the app will show the map of the desks for the student’s selected reservation time and will show available desks as green and unavailable desks as red. If a desk is free, the student can tap on it and will get prompted as to whether or not they want to confirm reservation. The figure below is what the UI might look like for the selecting of a desk on the phone application:

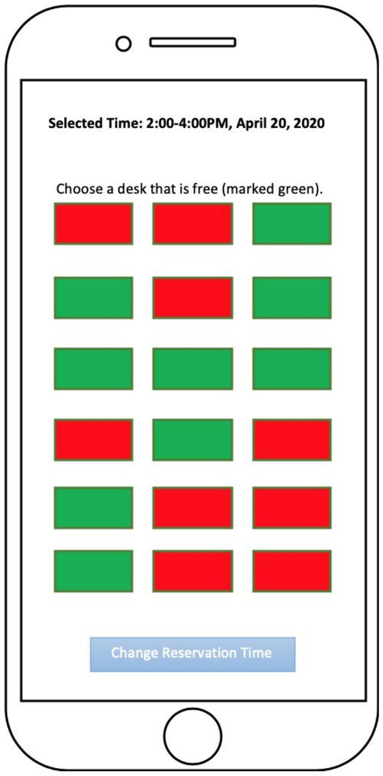
Selected Time: 2:00-4:00PM, April 20, 2020
Choose a desk that is free (marked green).
Change Reservation Time

Figure 12: UI for page where student can select an available desk from the map

After the student confirms the reservation for the desk, there will be a unique reservation ID that will be prompted from the app, which the student will then have to go enter at the master unit within 15 minutes. The student can also use the app to cancel the reservation within the first 15 minutes if something else comes up. There will be another page title 'Reservation History' on the app that the student can go to, to see all previous reservations, missed reservations, and total penalty points. The penalty points are important because with three penalty points, the student is limited to reserving desks only at non peak hours of reserving and so it is important that the student is careful when it comes to reserving desks if he or she does not have the intention to actually use the desk.

Requirements	Verifications
***Requirement 1*:** Phone application should allow the student to reserve a desk for upto 4 hours at a specific time and provide the student with a unique 6 digit reservation id.	a. Log in with a test account into the application. b. Select a reservation time and date and confirm that a corresponding map of available desks is shown. c. Verify that a six digit reservation is prompted on screen after an available desk on the map is selected.

***Requirement 2*:** Phone application should allow the student to cancel the reservation within 15 minutes of having reserved it.	a. Follow steps for requirement 1 to reserve a desk. b. Tap the ‘Cancel Reservation’ button on the app and verify that the reservation is removed by checking the reservation ID is absent from the corresponding database of the central server.
***Requirement 3*:** Phone application should show student’s history of missed reservations.	a. Program test account to have four total reservations and two penalty points. b. Log in with the test account into the application. c. Go to the page titled ‘Reservation History’ and verify that the application has kept track of account’s four reservations and two penalty points

2.6.2 Central Server

The central server acts as the main entity for controlling the desk reservation system. It performs a few critical tasks which allows the system to run and it is hosted at a fixed web address. This web address is used by the master unit and the phone application to send API requests.

1. It contains the desk map for the entire library and for each desk contains its future reservations schedule in a database. When the phone app sends a query for a reservation for a particular duration of time, the central server scans the database and based on the reservation schedule for each desk marks it as available or taken. Then it composes it into a message and sends it back to the phone app.
2. When the phone app sends a confirmation for reservation, the central server updates the database by adding this reservation to the corresponding desk entry and generates a random 6 digit code. This 6 digit code with the desk ID and time is composed as a message and is sent to the user and broadcasted to all the masters.
3. When a master unit restarts, it sends a post request containing its new IP address to the central server and the central server updates its directory of IP addresses of master units. The directory is used to flood reservation confirmations to the master unit.
4. Central server also maintains a penalized UIN list, to deny them reservation rights. It maintains this with a database where each UIN is a row ID, and the number of missed reservations is a column. It also maintains a timer for users who have been banned from

reserving and when the timer becomes zero, resets their missed count to zero. After every semester, the missed count becomes zero for everyone.

Requirements	Verifications
Requirement 1**:** Be able to receive POST and GET requests from phone app and master units.	a. Perform a GET request from the phone application to the master unit and verify that it is a valid request b. Perform a POST request from the phone application to the master unit and verify that it is a valid request
Requirement 2**:** Be able to maintain 4 separate databases.	a. Create four test databases and populate it with random data b. Verify that various operations like select, add, delete, etc. can be performed without issue on the databases.
Requirement 3**:** Be secure by not compromising student data and not completing API calls from non-master units.	a. Attempt an API call from a unit that is not a master unit. b. Verify that this is an invalid request.

2.7 Tolerance Analysis

The biggest obstacle and the trickiest part of our project involves the communication subsystem, particularly, bluetooth subsystem which involves formation of a *bluetooth scatternet*. Each master unit along with 7 level-1 slave units form a piconet, which is a type of Bluetooth connection formed between 2 to 8 Bluetooth-enabled devices. In a scatternet, a slave from a given piconet could be elected to become a master in another piconet, thus connecting multiple piconets. In our system, each of these level-1 slave units would be assigned the role of “master", where each of their piconets contain 6 level-2 slave units. Thus our scatternet would contain one master unit and 49 slave units ($7 + 7 \times 6 = 49$).

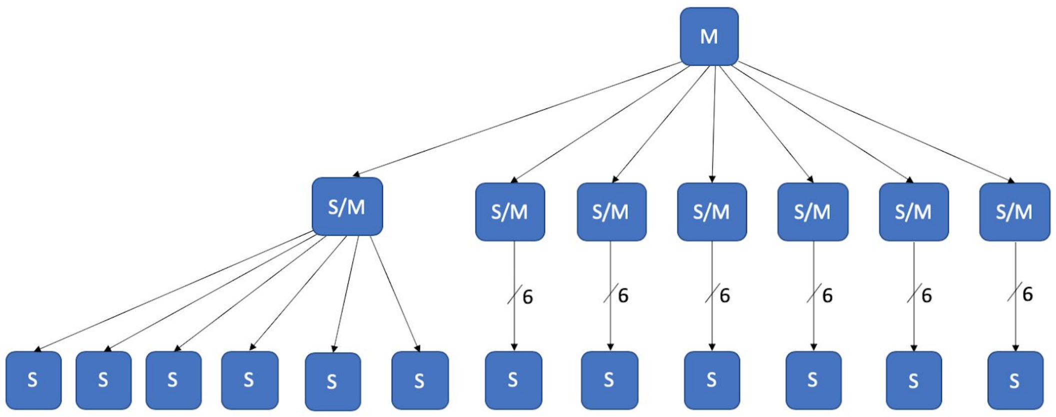

Figure 13: Piconet Network

A scatternet allows more efficient use of available bluetooth channel bandwidth, as compared to a piconet, in addition to supporting more than 8 nodes at once, owing to its ability to develop a multi-level network of bluetooth connections. We chose to implement a scatternet instead of a piconet due to the fact that we wanted to reduce the required number of master units (enabled with WiFi), as now each master unit is in charge of 49 desks, instead of the 8 traditionally allowed by piconets. Additionally, a piconet only allows a typical coverage area of 10 metres, implying that the distance between any master and slave unit cannot be more than 10 metres to ensure stable bluetooth connectivity. On the other hand, a scatternet can serve a larger coverage area, as only the 7 level-1 slave desks need to be within 10 metres of the master unit. The level-2 slave desks now need to be within 10 metres of their master unit i.e a level-1 slave, increasing our coverage area to a radius of 20 metres.

Table 1: Number of Interfering Piconets

Number of Interfering Piconets	Throughput in KVPS
2	0.950320386
4	0.903108836
6	0.858242737
8	0.815605569
10	0.775086599

12	0.736580595
14	0.699987556
16	0.665212444
18	0.632164946
20	0.600759236
22	0.570913748
24	0.542550974
26	0.515597251
28	0.489982578
30	0.465640433
32	0.442507595

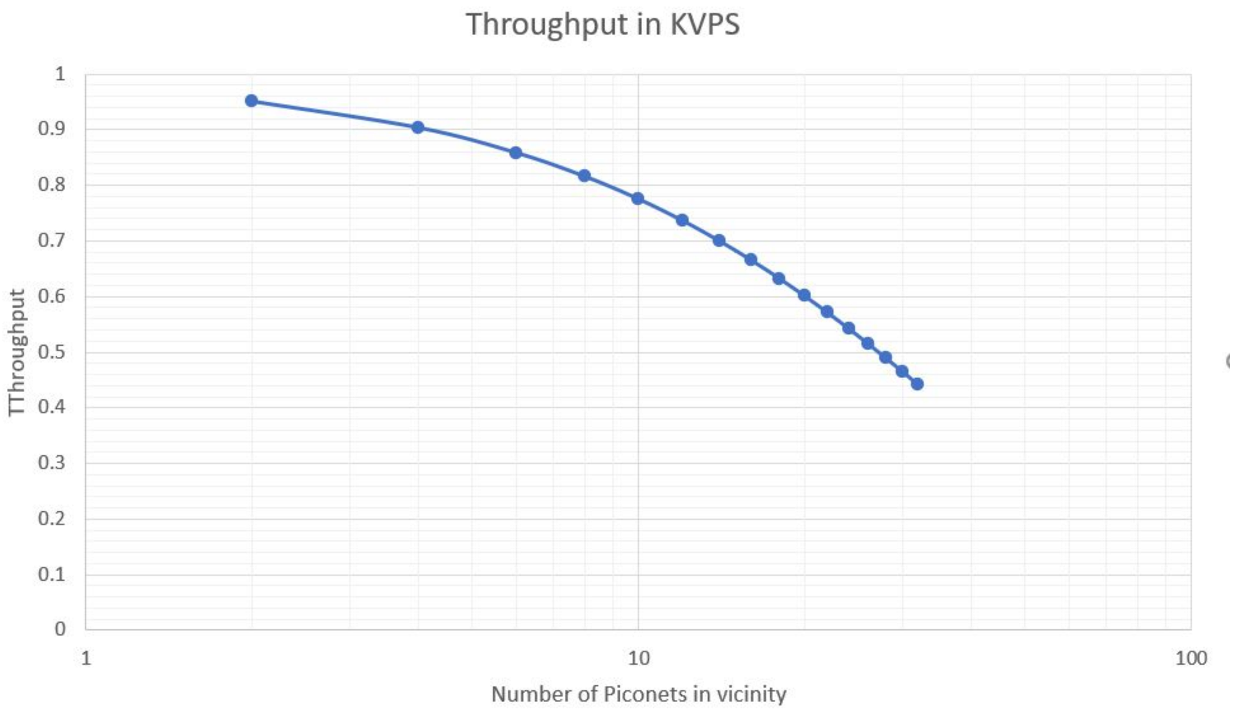

Figure 14: Throughput for Piconets

In a scatternet, or any bluetooth system with multiple devices in the 10m range, each piconet relies on *frequency hopping* to avoid interference from other piconets as well as non-Bluetooth devices in the 2.4 GHz frequency band. The Bluetooth standard uses a frequency hopping spread spectrum scheme of 79 non-overlapping frequency channels with 1 MHz channel spacing. In the frequency-hopping (FH) communication protocol in a piconet, the slaves are time- and hop-synchronized to the master and all data transmissions occur between master and slaves only. Interference between any two piconets occurs when the transmission frequency of any two piconets coincides or even frequencies are on adjacent bands. So inter-piconet interference could be due to either co-channel or adjacent channel interference. Since there are a limited number of frequency hopping bands available, this problem becomes more severe as the number of piconets in a given physical area gets large, especially a scatternet setup. This interference could potentially lead to data corruption, or even data loss, which is imperative to avoid in our scenario in order to make sure that the status LED on the reserved desk turns red to indicate that the reserved desk is in use.

Thus, for data packets to be successfully transmitted and received, the transmitted packet must not suffer from a collision. We know that data throughput is the rate at which the given data is processed or transferred. We can relate the probability of a 100% data throughput with the number of piconets as follows:
P1 = probability of hopping to any one of the 79 Bluetooth channels = 1/79
n = number of interfering piconets

$$P(\text{Throughput of a } 100\%) = (1 - P1)^{2n}$$

$$P(n = 8) = 0.815$$

This indicates that the rate at which given data can be processed drops down to 81% of the maximum possible value. On an average Bluetooth low energy, we have a maximum data throughput availability of 192 kbps.
Updated throughput availability = 0.815x192kbps = 156.48 kbps

As we calculated before, for our bluetooth module transferring in
Additionally, we know that data throughput = data per connection interval / connection interval

3. Differences

3.1 Details on previous project

The original project on the Desk reservation system was done during Fall 2019 semester and the underlying idea for design was "streamlining desk usage for students working at Grainger Library, allowing students to spend more time studying than searching for a desk". The project had 3 key components.

3.1.1 Mobile application and Central Database

This was the software component, through which the user would have interacted with the desk reservation system and chosen a particular seat to reserve. The central database also kept track of empty and occupied seats, ran algorithms for reserving seats and interacted with each desk on a 1:1 basis.

3.1.2 Desk module

This was the hardware component present at each deak and consisted of a microcontroller, a Wi-Fi module, an LCD screen, an RFID reader and Proximity sensor. The RFID reader was used to enforce the check-in and check-out process. Here users could use their Icards to tap in and out of reservations. The Wi-fi module communicated to the central server to receive updates about reservations and send information about occupancy. The LCD screen was used to aid the user, by updating desk status about occupancy, and the proximity sensor was used to detect whether any person was occupying the desk or not.

3.1.3 Enforcement at desk

Using the proximity sensor the system was also able to know whether the current user has left the desk or not. If somebody encroached the desk by overstaying, the proximity sensor via Wi-fi would have informed the central database about the breach and the central database would fine the student via student account, as it would have details about the student UIN.

3.2 Fundamental Differences in Design

Our project borrows some elements from the previous project such as the phone application and central database (we call it central server, as database by definition is just a storage unit and technically can't do any logical tasks). However, beyond the similarities on the software part our hardware implementation is drastically different and we believe has several advantages over the old system.

3.2.1 Limited Desk module & using Master Module

So, in our project the desk module is stripped down to only a bluetooth module and an LED light. The only purpose of the desk module is to indicate whether a seat is currently occupied or not by turning the LED on. This contrasts with the previous project where the desk module was the main hardware component and did the majority of work outside software. As in libraries, all the desks are always close-by having such a bulky desk unit at every desk didn’t make sense to us. Hence, we substituted the functionality of the desk module, such as checking in and LED display, with a single master unit (over 49 desks). This is the major design change we are using to bring down our cost.

3.2.2 Bluetooth Scatternet

In the old project, the central server could directly send information to each individual desk via Wi-Fi and the microcontroller there would have processed it and acted accordingly. As we don’t have a Wi-Fi unit on each desk, the central server communicates to the desk via the master unit, which in turn communicates to desk modules through Bluetooth scatternet. The scatternet works via flooding information down to slaves at level I and this further gets flooded down to slaves at level II. Here, the main engineering trade off is that we need to create scatternets out of piconets which is difficult to do and it would be very hard to re-establish this piconet if one device goes out. To do so, we need

3.2.3 Enforcement System and proximity sensor

In our project, we don’t enforce if a person overstays on a desk as either the desk is not reserved in which case it makes no sense to force the person out or it is reserved in which case the LED display will be red and the user will know that the other person who has reserved the desk has come. We don’t believe that any user will argue over whether to give a desk if the other person can show reservation confirmation, in which case we don’t need any proximity sensor.

3.2.4 Penalty System

An additional feature, which does not exist in the old project is penalizing users for missing reservations and enforcing they are on time for a reservation. In our project, via the central server we keep track of when the user started a reservation to ensure that the student isn’t late by more than 15 minutes. The central server also keeps track of how many times a user had no show. In the previous system, there is no way to ensure that people don’t misuse the system by overbooking or by no shows.

3.3 Analysis: Improvements in Cost of Project

The main area where our project improves upon the previous one is through cost. The main cost advantage comes from streamlining the project by removing chips and replacing modules with more optimized use. To better understand the project we simulate the implementation of both the projects in a 1000 seat library such as Grainger Engineering library. The development of mobile application and cost of hosting a central server (excluding network cost) will be relatively the same in both the projects. From the old design document we can find the total cost per desk module.

3.3.1 Similar cost for implementing for 1000 desks library

$$Cost\,for\,human\,resources\,(including\,software\,development) \;=\; \$37,500$$

$$Cost\,for\,hosting\,server\,and\,running\,software\,application \;=\; \$15\,(For\,domain\,name/yr)$$
$$+\;\$1200\,(For\,maintaining\,Mobile\,application)\;+\;\$600\,(\$50/month\,for\,Internet)\;=\;\$1815$$
$$Total\,Similar\,Cost\,for\,first\,5\,years\;=\;\$39,315$$

3.3.2 Differences in Cost for implementing for 1000 desks library

Old project

$$Cost\,of\,Individual\,Desk\,module\;=\;\$81.24$$
$$Cost\,1000\,Desk\,module\;=\;1000\cdot\,\$81.24\;=\;\$81,240$$

New project

$$Cost\,of\,Master\,unit\;=\;\$18.98$$
$$Cost\,of\,Slave\,unit\;=\;\$4.13$$
$$Total\,Master\,units\,needed\,over\,1000\,desks\;=\;\frac{1000}{49}\;\sim\;21\,units$$
$$Cost\,of\,Master\,unit\;=\;\$18.98$$

$$Total\,cost\,of\,hardware\,for\,implementing\;=\;21\cdot 18.98\;+\;4.13\cdot 1000\;=\;\$4528.58$$

So, just over the hardware units we have achieved a **94.425% cost reduction**.

3.3.3 Cost reduction achieved over different scales

We can model the cost of both projects via a line, where the offset is the fixed cost for software development.

Old project

$$TotalCostOld(\#\ of\ seats)\ =\ \$39,315\ \ +\ \ \#\ of\ seats\ *\ 81.24$$

New project

$$TotalCostNew(\#\ of\ seats)\ =\ \$39,315\ +\ (\frac{\#\ of\ seats}{49})\ \cdot\ 18.98\ +\ \#\ of\ seats \cdot 4.13$$
$$We\ can\ average\ out\ master\ cost\ over\ 49\ desks\ =\ 39.315\ +\ \ 4.52 \cdot\ \#\ of\ seats$$

Comparing the difference

Table 2: An exponential decay on cost reduction as we scale up

Scale of Implementation	Total cost of Project with old implementation	Total cost of Project with New implementation	Cost reduction
1000 Seats	$120,555	$43,835	63.64%
10,000 Seats	$851,715	$84,515	90.08%
100,000 Seats	$8,163,315	$491,315	93.98%
1,000,000 Seats	$81,279,315	$4,559,315	94.39%

We can also plot these numbers on a graph. Here, we notice that as we scale up, we get broder differences in our cost. This means a unique result for cost reduction which is the main function.

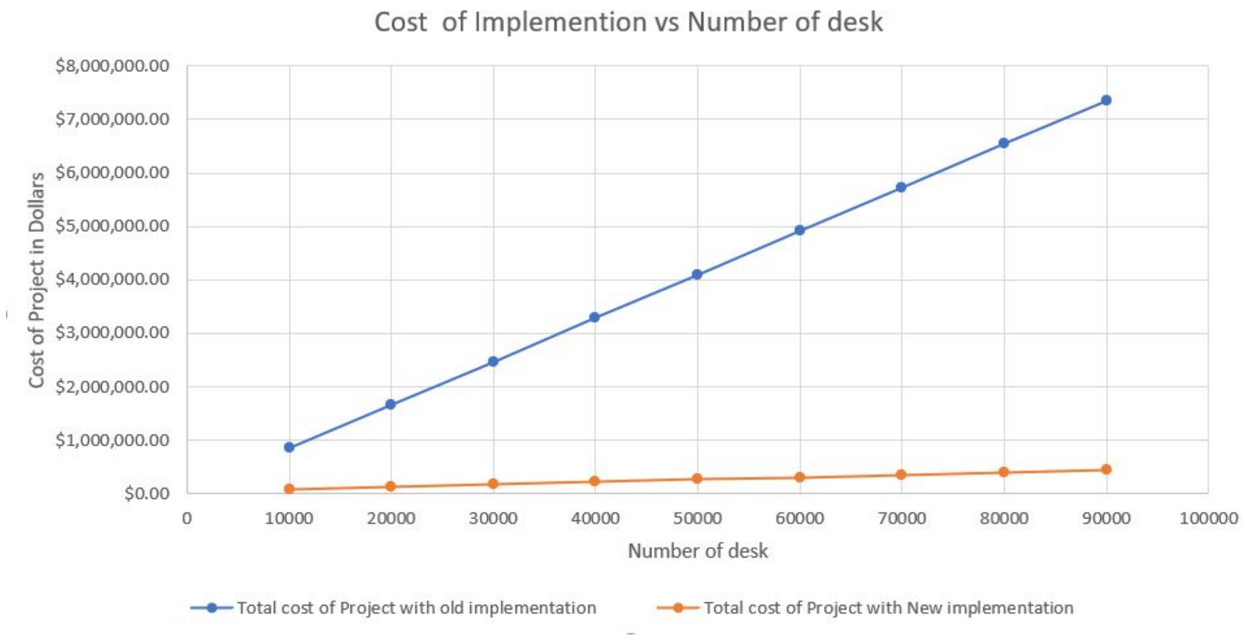

Figure 15: Graph showing cost of implementing our project vs old project.

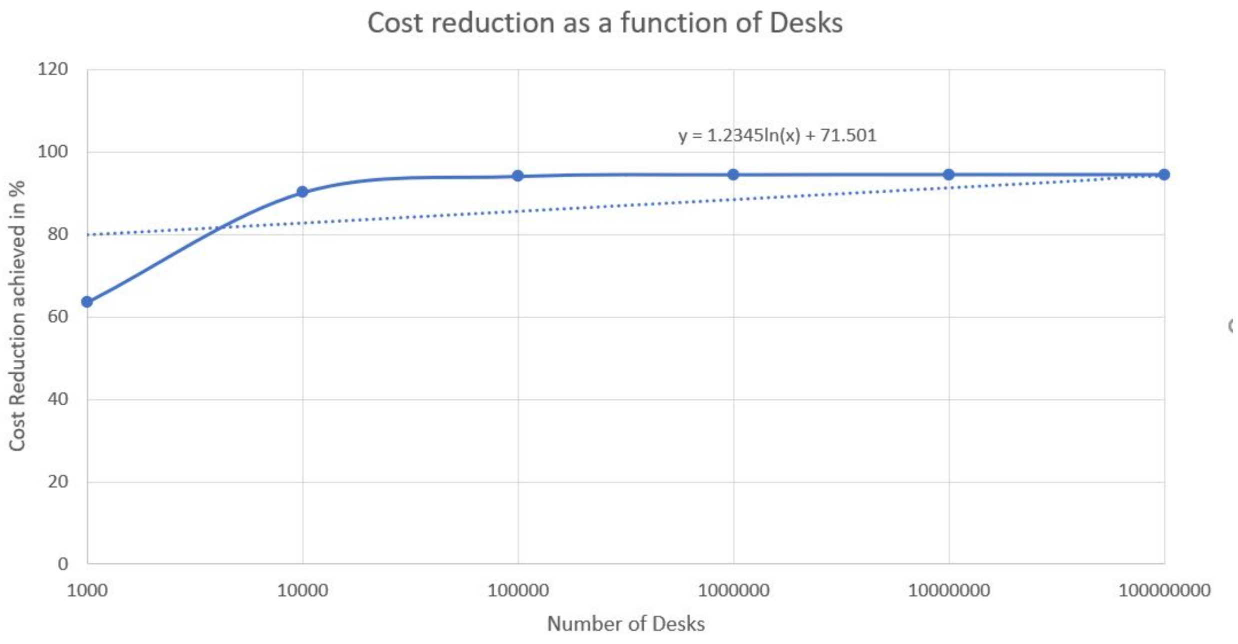

Figure 16: Graph for Cost reduction as a function of Number of desks.

From this graph, we can see that once our system scales up to almost 10,000 desks we achieve 90% cost reduction over the old project. However, this is a logarithmic decay graph and for a smaller number of desks our cost reduction is minimalistic compared to the old project. This means, if our project is used for only 100 desks, it might be better to use the old project due way less complexity

3.3.4 Trade offs

We also have to deal with few trades off due to changes in design to reduce cost. These trade offs come due to implementation cost, more complex reservation process and more technical depth.

1. The novelty cost of implementing the scatternet. Scatternet overall, is not a widely adopted technology and doesn't have much information available. If existing resources aren't available, we would need consultation from professors and outside experts. This can significantly increase the initial development cost of the project.
2. Increased time to access the reservation. Even after a desk has been reserved, a person could have to wait in line for upto 5 minutes, if all 49 desks have a new reservation starting at a given time. This is significantly higher than the old system, where the user could directly access the desk.

3.4 Analysis: Improvements in Network Congestion & Security

Our project improves network congestion and security by reduction in number of API endpoints, by stacking information over a single message, by increasing closed network communication (Bluetooth Scatternet) and by limiting the system to store only UINs. Each of these modifications provide improvements in the network and security area. We analyze these improvements within the context of Grainger engineering library which has 1000 desks approximately.

3.4.1 Reduction in number of API endpoints

In the previous project, each desk with a unique Wi-Fi chip would result in 1000 API endpoints which would all be communicating with the Central database. Now, assume due to a system wide failure, all the API endpoints get disconnected and would need to reform the connection. To reform the connection, we would need to do a TCP handshake, which is a process which is used in a TCP/IP network to make a connection between the server and client. It is a three-step process that requires both the client and server to exchange synchronization and acknowledgment packets before the real data communication process starts [4]. The process begins with sending a SYN packet, receiving a SYN-ACK and then ending with an ACK message. Now, within 1 second we will have 1000 TCP handshakes and each TCP handshake costs upto 128 bytes. Now, failures can also happen in a system constantly.

Total data exchanged for 1000 TCP handshakes = 1000 * 128 bytes = 125 Kilobytes

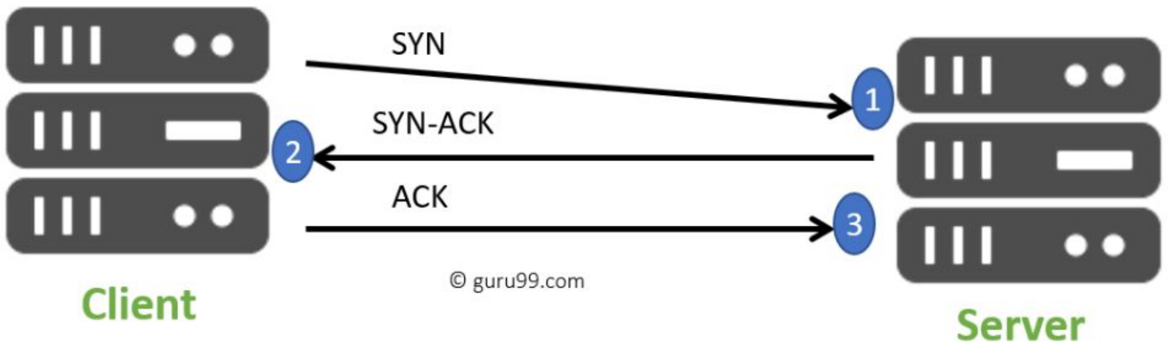

Figure 17: A TCP handshake [5]

Assuming that we have 2.5% failure over every hour due to several factors like network issue, electronic failures and power issue.

Total failure every hour = 1000 * 2.5% = 25 failures/hr
Total failure within a year = 25 * 365 * 24 = 219000 failures

Each failure would result in a new TCP handshake

Total data used over a year due to failures = 219000*128 = 27375 Kilobytes

In our system, we have 1 master over 49 desks, which results in approximately 21 Master units for 1000 desks. Using 21 Wi-fi connections instead of 1000, gives us approximately 50 times less network congestion.

Table 3: Project comparison with different API Endpoints

	Old Project with 1000 API endpoints	Our project with 21 API endpoints
Complete failure	125 Kilobytes	2.625 Kilobytes
Over a year	27375 Kilobytes	574 Kilobytes

3.4.2 Stacking information over a single message

Another network improvement comes from stacking information over a single message. Assume we have new reservation information which needs to be sent to desk units. And we want to send new information to 49 desks. In the old system they would be sent over 49 different TCP connections as each unit has a separate Wi-Fi connection. Whereas, we would need to send one 1 message in our new system. A normal TCP package has an overhead of 16/40 bytes of IPv4/6 header and 16 bytes of TCP header [6].

$$Cost\ of\ TCP\ overhead\ for\ a\ single\ message = (40 + 16)\ bytes = 56\ bytes$$

$$Cost\ of\ TCP\ overhead\ for\ 49\ message\ =\ (56\ bytes) * 49\ =\ 2744\ bytes$$
$$Improvement\ over\ old\ system\ by\ =\ \frac{(2744-56)}{2744} \cdot 100\ =\ 97.95\%$$

So, compared to the old system, our system would have a **97.95% reduction** in data overhead.

3.4.3 Increasing closed network communication (Bluetooth Scatternet) and limiting the system to store only UINs

Security is improved as an attacker would need to be physically present to compromise a bluetooth scatternet and even if he/she was able to compromise we only store UINs, which can't do any damage beyond identity theft. This compared to the old system, where they wanted to charge the student a fine, they would need to store somewhere financial information of the person. If we can launch a man in the middle attack over the network we can exploit this security vulnerability as follows:

1. Using the APIs used to charge a fine to the person, post arbitrary charges on the account.
2. Using the UIN, get an email address and send an email for a refund.
3. In the email, ask for details of the bank account to send money back.

3.4.3 Trade offs

We also have to deal with few trades off due to changes in communication protocol. These trade offs come due to several reasons such as more complex implementations to hierarchical structure of the project.

1. Although we have improved the network bandwidth consumption we have replaced it with Bluetooth Scatternet. We will have 50 devices within a scatternet and it is possible that there are several such scatternets in close proximity to each other.
    a. There is 1.5% chance of interference in a piconet [7].
    b. There are 7 piconets in one scatter net, so the chance of interference is: $1 - 0.985^6 = 10.039\%$ for one flood operation
    c. If there are 3 piconets in co-existent due to congested nature of library, the chance of interference = $1 - 0.985^6 = 27.19\%$
    d. Each time we flood, we can have one fourth of the messages to interfere. This error doesn't cost our system as the bluetooth system is merely to turn the LEDs red and even without that the user can go to the desk and use it.

2. Another trade off comes from the number of failure points and impact of each failure on the system. Before this, the failure of one module would have rendered one desk useless, but now if one master unit fails 49 desk units will be out of the reservation system. This added to the fact that, we have 5 points of failure (Phone app, Central

server, Master unit, Slave level I, Slave II) compared to only 3 (Phone app, Central database, Desk unit) in the old system.

4. Cost Analysis

4.1 Labor

For our project, we will consider a time span of 10 weeks, where we work 10 hours per week for an hourly rate of 50$. The labor costs can be calculated by the formula seen below using the values mentioned above:

$$Labor\ Costs\ =\ 2.5\ \times\ (group\ size)\ \times\ (hourly\ cost)\ \times (hours\ per\ weeks) \times\ \ (\#\ of\ weeks)$$

Inputting the corresponding values:

$$Labor\ Costs\ =\ 2.5\ \times\ 3\ \times\ 50\ \times\ 12\ \times\ 10$$

Therefore we have:

$$Labor\ Costs\ =\ \$37,500$$

4.2 Parts

Table 4: Cost of Parts

Name	Manufacturer	Product ID	Quantity	Price ($)	Total Price ($)
Bluetooth Module	Abra Electronics	JDY-19	2	1.00	2.00
Wifi Module	Espressif Systems	ESP8266	1	6.95	6.95
Power Switch	TE Connectivity	1825255-1	2	1.53	3.06
Power Adaptor	N/A	N/A	2	0.99	1.98
Voltage Regulator	Texas Instruments	TPS7A24	2	0.26	0.52
LED Display	Jameco	225963	1	1.75	1.75
Numeric Keypad	Sparkfun	COM-14662	1	4.50	4.50
Microcontroller	Microchip Technology	ATMEGA328P-AN	1	2	2.11

Arduino Uno R3	Arduino	A000066	1	18	18
USB to TTL Serial Connector	Adafruit Industries LLC	1528-2128-ND	1	9.95	9.95
Red LED	Sparkfun	COM-09590	1	0.35	0.35

4.3 Total Costs

Table 5: Total Costs

Section	**Cost ($)**
Labor	37,500
Parts	51.17
Total	37,551.17

5. Schedule

Table 6: Schedule for Project

Week	**Pragya Aneja**	**Ninad Godbole**	**Varad Khandelwal**
Week 1	Design Review and Incorporating Feedback and Critique	Design Review and Incorporating Feedback and Critique	Design Review and Incorporating Feedback and Critique
Week 2	Ordering Parts, Begin Drafting PCB	Ordering Parts	Ordering Parts, Begin Drafting PCB
Week 3	Testing and Programming Bluetooth module	Testing and Programming Wifi module	Design PCB for first round of PCBWay order
Week 4	Implement Bluetooth and Wifi module for Communication Subsystem of Master Unit	Construct Power Subsystem Circuit for Master Unit	Construct Power Subsystem Circuit for Slave Levels 1 and 2
Week 5	Work on designing Phone	Implement Bluetooth	Construct

	Application and integrating with other subsystems	and Wifi module for Communication Subsystem for Slave Levels 1 and 2	Microcontroller Circuit for Master Unit and Slave Levels 1 and 2, Work on designing Phone Application
Week 6	Finish leftover work, Integrate all components and Perform Final Testing	Finish leftover work, Integrate all components and Perform Final Testing	Finish leftover work, Integrate all components and Perform Final Testing
Week 7	Mock Demo and Final Corrections	Mock Demo and Final Corrections	Mock Demo and Final Corrections
Week 8	Final Demo	Final Demo	Final Demo
Week 9	Final Presentation, Final Paper	Final Presentation, Final Paper	Final Presentation, Final Paper

6. Safety and Ethics

The IEEE code of ethics which are made up of 10 guidelines as well as the ACM code of ethics provided a good framework for us when we consider the ethical aspects with regards to our project. From this IEEE and ACM ethics framework, there are a couple of ethical issues that are worth discussing.

A few ethical issues for our project correspond to one of the ACM code of ethics which says “Be fair and take action not to discriminate” [8]. The ethical issues here correspond to some of the parts or subsystems which could potentially be faulty, leading some students to not be able to use the reservation system properly which would be unfair to them. One example of this is if the status LED on the desk does not light up red as it has stopped working. This could confuse the student as to the status of the desk and worsen the experience. Another example is if the numeric keypad stops working properly. This would become very problematic as this is what the student uses to confirm the reservation ID and so if it doesn’t work the student won’t get access to the desk. There are other parts that if they stop working will create an unfair experience for some students who went through the correct process of reserving a desk through this system. We will therefore strive to overcome these issues by making sure that the parts are high quality and tested thoroughly before placed in use.

Another ethical issue that should be brought up is students potentially taking advantage of the system which relates to one of the ACM code of ethics which says “Be honest and trustworthy” [8]. An example of this is if a student decides to reserve a desk without really having the intention of using the desk but just as a backup and then does not show up to the reservation, causing someone else to miss out. An additional ethical issue which is extremely important involves the data security of the students who are using the phone application as well as their UIN when reserving the desks, which corresponds to one of the ACM code of ethics which says “Respect privacy” [8]. We will therefore make it a priority to make sure all of our systems keep the data of the students secure.

Safety is the other aspect of this project that we have to be aware of which also relates to the first IEEE code of ethics which says “to hold paramount the safety, health, and welfare of the public” and one of the ACM code of ethics which says “Avoid harm” [9][10]. The biggest safety issue with regards to this project involves our usage of the lithium ion battery. As usage of these batteries increases, they begin to overheat and in a low likelihood scenario, can catch on fire putting a student in a lot of harm if he or she is near the battery [10]. We therefore have to be aware of the safety hazard of the lithium ion battery in our design.

7. References

[1] “How do hotel reservation systems increase efficiency and profits?,” SiteMinder, 27-Aug-2018. [Online]. Available: https://www.siteminder.com/r/hotel-distribution/hotel-direct-bookings/hotel-reservation-system-increase-profits/. [Accessed: 03-Apr-2020].

[2] Square, “How to Optimize Salon Scheduling,” Square. [Online]. Available: https://squareup.com/us/en/townsquare/effective-salon-scheduling-strategy. [Accessed: 04-Apr-2020].

[3] “ATmega328P,” Microchip. [Online]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf?fbclid=IwAR1NmfLAtSnuxQ2ExBtGuWSFB522n5f5T6iS3rcBEIZ4TgZybzSZvs6PSNU. [Accessed: 27-Feb-2020].

[4] A. S. Bhargava, “Using Piconet Avoidance Techniques to Reduce Interference in Bluetooth Networks,” Kansas State ECE, 2004. [Online]. Available: https://www.ece.k-state.edu/crl/publications/usingpiconetavoidancetechniquestoreduceinterferenceinbluetoothnetworks.pdf. [Accessed: 17-Apr-2020].

[5] “TCP 3-Way Handshake (SYN, SYN-ACK,ACK) - Guru99,” Guru99. [Online]. Available: https://www.guru99.com/tcp-3-way-handshake.html. [Accessed: 18-Apr-2020].

[6] “What is better? Lots of small TCP packets, or one long one?,” Game Development Stack Exchange, 01-Mar-2015. [Online]. Available:

https://gamedev.stackexchange.com/questions/96945/what-is-better-lots-of-small-tcp-packets-or-one-long-one. [Accessed: 18-Apr-2020].

[7] “Informit,” InformIT. [Online]. Available: https://www.informit.com/articles/article.aspx?p=21324. [Accessed: 18-Apr-2020].

[8] “The Code affirms an obligation of computing professionals to use their skills for the benefit of society.,” Code of Ethics. [Online]. Available: https://www.acm.org/code-of-ethics. [Accessed: 03-Apr-2020].

[9] “IEEE Code of Ethics,” IEEE. [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html. [Accessed: 14-Feb-2020].

[10] BatteryGuy, “Home,” The BatteryGuy.com Knowledge Base. [Online]. Available: https://batteryguy.com/kb/knowledge-base/safety-issues-with-lithium-batteries/. [Accessed: 04-Apr-2020].