

# SmartPark: IoT-Driven Automatic Parking Solution

DESIGN DOCUMENT

Team Number: sddec24-17

Client/Advisor: Md Maruf Ahamed

Team Members/Roles:

William Clemmons - Project and Software Lead

Mubassir Serneabat Sudipto - QA Lead and Software Engineer

Ethan Haberer - Software Engineer

Zachary Sears - Hardware Lead

Kennedey Reiling - Hardware Engineer

Brian Witherspoon - Hardware Engineer

Team Email: sddec24-17@iastate.edu

Team Website: <https://sddec24-17.sd.ece.iastate.edu/>

Revised: 12/11/2024

# Executive Summary

Parking on the campus of a University is often complicated and sometimes unbearable. Some areas are restricted to the general public, and the number of available parking spaces may be unpredictable or diminished during popular events. Furthermore, confirming whether there will be a convenient open parking spot for a driver is impossible. This problem could prevent students from attending classes on time or, likewise, a visitor from a game. To mitigate this issue, a system that allows users to visualize live data of a parking lot would benefit the livelihoods of Iowa State visitors, students, and professionals.

Our design had a list of criteria, such as the implementation of sensors, reservation capabilities via a mobile application, and live accurate data. Additionally, we had to consider the variety of users operating our system. We were motivated to create a user-friendly, intuitive, and reliable system. Our team split into hardware and software groups to manage their respective components.

After testing different detection methods, the hardware team selected [ultrasonic sensors](#) to detect if a vehicle is parked in a spot. Moreover, for communication with a [server](#) and [database](#), the hardware is controlled by an [Arduino](#) microcontroller. To show non-app users whether a space is reserved or open, the hardware module includes an [RGB LED](#) indicating the spaces' states. A single hardware module consists of four sensors, a microcontroller, and an LED, taking the necessary data from four spaces.

The application needed to allow users to reserve a space and pay conveniently. To develop the application the software group used React Native, allowing cross-platform coding. The server had to receive data from the hardware, store it in our SQL database, and send this information to the application.

Overall, from end to end, the system consists of our hardware module mounted on a post in the middle of four parking spots. The data taken will be sent to our database every second, becoming available to our application. If a user reserves a place, this [request](#) will be sent to the server and indicated by our hardware with a white light for the corresponding parking spot.

Our design follows our requirements but could be improved in future iterations. Some possible improvements involve increasing the reliability of the entire system, updating the user interface to enhance intuitiveness, and testing different sensors, such as cameras, to maximize accuracy.

This documentation will resolve most questions regarding research, design, implementation, etc.

# Learning Summary

## Development Standards & Practices Used

### Practices:

- Hardware
  - Circuit Design
  - Rapid Prototyping
  - Embedded Systems
- Software
  - Source Control
  - Testing
  - App Development
- Project
  - Agile
  - Kanban

### Engineering Standards:

All Standards are described in Section 2.2 of this Design Document.

- Hardware
  - IEEE 802.2a-1993
- Software
  - IEEE 3156-2023
- Project
  - Standard 14-5A-5 C

## Summary of Requirements

- Software
  - The user interface does not impede users from driving.
  - The app is intuitive for users.
- Hardware
  - Sensors should be able to detect when a car has pulled in or left a parking spot.
  - Withstand being outside for extended periods.
  - Reliably send data to the server and keep it up to date.
- Server

- MySQL database and Express.js server to send data from both the hardware and the application.
- An Advanced reservation system to minimize user interaction with the phone while driving, based on insights from user experience design.
- A robust and scalable infrastructure to handle the growing volume of users and environmental challenges, as informed by technical and quality assessments.

## Applicable Courses from Iowa State University Curriculum

- SE
  - SE 3190 - User Interfaces
  - COMS 3090 - Software Development Practices
  - CPR E 2880 - Embedded Systems
- CPRE
  - CPRE 3810 - Computer Organization and Assembly Level Programming
  - CPR E 2880 - Embedded Systems
- EE
  - EE 2850 - Problem Solving Methods
  - EE 2300 - Electronic Circuits and Systems
- CYBE
  - CYB E 2310: Cyber Security Concepts and Tools
  - CYB E 2340 - Legal, Professional, and Ethical Issues in Cyber Systems

## New Skills/Knowledge acquired that was not taught in courses

- Software
  - Server Design
- Hardware
  - Arduino programming
    - Connecting Arduino hardware to WiFi
    - Uploading sensor data to a server
  - Arduino/hardware simulation
  - Rapid prototyping and iteration
  - Soldering

## Table of Contents

<b>1 Introduction</b>	<b>7</b>
1.1 Problem Statement	7
1.2 Intended Users	8
<b>2 Requirements, Constraints, And Standards</b>	<b>8</b>
2.1 Requirements & Constraints	8
2.2 Engineering Standards	8
<b>3 Project Plan</b>	<b>9</b>
3.1 Project Management/Tracking Procedures	9
3.2 Task Decomposition	11
3.3 Project Proposed Milestones, Metrics, and Evaluation Criteria	11
3.4 Project Timeline/Schedule	12
3.5 Risks And Risk Management/Mitigation	13
3.6 Personnel Effort Requirements	14
3.7 Other Resource Requirements	14
<b>4 Design</b>	<b>14</b>
4.1 Design Context	14
4.1.1 Broader Context	14
4.1.2 Prior Work/Solutions	16
4.1.3 Technical Complexity	17
4.2 Design Exploration	17
4.2.1 Design Decisions	17
4.2.2 Ideation	17
4.2.3 Decision-Making and Trade-Off	17
4.3 Proposed Design	18
4.3.1 Overview	18
4.3.2 Detailed Design and Visual(s)	18
4.3.3 Functionality	26
4.3.4 Areas of Challenge	26
4.5 Design Analysis	27
<b>5 Testing</b>	<b>27</b>
5.1 Unit Testing	27
5.2 Interface Testing	28
5.3 Integration Testing	28
5.4 System Testing	28
5.5 Regression Testing	28
5.6 Acceptance Testing	28
5.7 Security Testing	28
5.8 Results	28
<b>6 Implementation</b>	<b>29</b>
<b>7 Professional Responsibility</b>	<b>29</b>
7.1 Areas of Responsibility	29
7.2 Project-Specific Professional Responsibility Areas	30

7.3 Most Applicable Professional Responsibility Area	31
<b>8 Closing Material</b>	<b>31</b>
8.1 Summary of Progress	31
8.2 Value Provided	32
8.3 Next Steps	32
<b>9 References</b>	<b>33</b>
<b>10 Appendices</b>	<b>34</b>
Appendix 1 -Operation Manual	34
Appendix 2 -Alternative/Initial Version	42
Appendix 3 -Other Considerations	42
User Profiles	42
Appendix 4 -Code	44
Appendix 5 -Team	44
Team Contract	44
Appendix 6 Glossary	48

## List of figures/tables/symbols/definitions

[Figure 3.2.1:](#) Hardware Gantt chart  
[Figure 3.2.2:](#) Decomposition of Software Team  
[Figure 3.4.1:](#) Projected Timeline of Project  
[Figure 3.4.2:](#) Team ClickUp board  
[Table 3.5.1:](#) Tasks, Risks, and Likelihood Probability  
[Table 3.6.1:](#) Task and Time Decomposition  
[Table 4.1.1:](#) Broader Context  
[Table 4.1.2:](#) Pros and Cons List of our solution  
[Figure 4.3.1:](#) Overall Design Flowchart  
[Figure 4.3.2:](#) Application User Interface  
[Figure 4.3.3:](#) Application User Interface Continued  
[Figure 4.3.4:](#) Application Flowchart  
[Figure 4.3.5:](#) Sketch of Physical Sensor Representation  
[Figure 4.3.6:](#) Schematic for Circuit  
[Figure 4.3.7:](#) Prototype Circuit  
[Figure 4.3.8:](#) Hardware Prototype Setup  
[Table 4.4.1:](#) Sensor Options  
[Table 7.1.1:](#) Areas of Responsibility  
[Table 7.2.1:](#) Project Specific Responsibilities  
[Figure 10.1.1:](#) App Home Screen  
[Figure 10.1.2:](#) Reserve Screen  
[Figure 10.1.3:](#) Payment Screen  
[Figure 10.1.4:](#) Pinout for an RGB LED  
[Figure 10.1.5:](#) LED Setup  
[Figure 10.1.6:](#) Ultrasonic Sensors Setup  
[Figure 10.1.7:](#) Arduino Nano Power Setup  
[Figure 10.1.8:](#) Arduino IDE Setup  
[Figure 10.3.1:](#) User Profiles

## 1 Introduction

### 1.1 PROBLEM STATEMENT

Currently, parking at Iowa State can be tricky. Finding parking is always arduous because you must consider multiple things. This includes, parking that is staff only, if the lot you want to park in is full, and how long it could take to find a spot. This project aims to eliminate these issues by streamlining the parking experience. Our team will create a detection-based system to monitor parking spots and update an app that students, teachers, or whoever may need to park on campus and allow them to view and reserve available parking to eliminate confusion campuswide.

## 1.2 INTENDED USERS

The intended users for our project are students, faculty, the parking division, and visitors on campus looking for parking spaces. Students often face the challenge of finding a parking spot before class starts. Although faculty have reserved lots, they still need help finding the most ideal space. Additionally, visitors to the campus who are attending events or meetings also need help finding parking spaces. Lastly, the parking division will need access to parking information to correctly identify when someone is inappropriately parked to issue tickets. By developing a detection-based system that monitors parking spots and updates an app, our project aims to address the parking issues these user groups face. This will allow them to view and reserve the most convenient and available parking spots, streamlining the campus parking experience and eliminating confusion.

# 2 Requirements, Constraints, And Standards

## 2.1 REQUIREMENTS & CONSTRAINTS

We have divided our requirements into two categories: Functional Requirements and Non-Functional Requirements. Functional Requirements include goals that have definitive results. Non-functional requirements describe conditions that do not list specific numerical values but are areas of focus.

### **Functional Requirements:**

Our functional requirements include live sensor data, a mobile app, a way to communicate with non-app users, accept payments, and a way for the parking division to view violations in real-time. Our sensors must send live, up-to-date information about parking spaces. Each sensor must be able to tell if there is a car in a parking space and the sensor information must be sent to our server to ensure the user gets reliable parking data. A mobile application must be user-friendly and allow users to reserve parking spots for a given lot. Another requirement we must meet is a way to communicate with users without the app by directing the user through a parking lot. Our final functional requirement is a unique way to accept payment from the user within the app and through an online version of our app.

### **Non-Functional Requirements:**

Our non-functional requirements include creating a system with low latency to keep the status of parking spots accurate. Additionally, the reliability of our application, server, and hardware are pertinent to ensure a user-friendly system. Furthermore, our server and application need to be online and working consistently. To hold the trust of our clients, our server and application must encrypt and protect user information while taking secure payments. Our application must be presentable and navigable to gain popularity and keep customers.

### **Constraints:**

The hardware for our system must have minimal downtime. However, uptime cannot be guaranteed since the sensor will be outdoors. Therefore, the sensor needs to be detected as offline as soon as possible so that maintenance can be done.

## 2.2 ENGINEERING STANDARDS

### **IEEE 3156-2023**



Standard Ruling: A standard for privacy-preserving computation integrated platforms is needed to meet the evolving requirements of multi-sourced data computing and sharing. Requirements of privacy computation integrated platforms, including the reference architecture, the functional requirements, the performance requirements, and the security requirements of privacy-preserving computation integrated platforms, are provided by this standard.

Justification: Our project will require us to take user payments through our app. We must implement security systems that protect those users from potential attempts to steal information.

### **IEEE 802.11g-2003**

Standard Ruling: IEEE 802.11g-2003 is an amendment to the IEEE 802.11 specification that operates in the 2.4 GHz microwave band. The standard has extended the link rate to up to 54 Mbit/s using the same 20 MHz bandwidth as 802.11b to achieve 11 Mbit/s. Under Wi-Fi's marketing name, this specification has been implemented worldwide. The 802.11g protocol is now Clause 19 of the published IEEE 802.11-2007 standard and Clause 19 of the published IEEE 802.11-2012 standard.

Justification: The Arduino [Nano 33 IoT](#) we are using connect over wifi using the 802.11 protocols, which we will be using to send and pull data from our server.

### **Standard 14-5A-5 C: Parking, Stacking Space Size, And Drive Dimensions**

1. The minimum size of a standard off-street parking space is nine feet by eighteen feet (9' x 18'), exclusive of aisle width.
2. The minimum size of a compact off-street parking space is eight feet by fifteen feet (8' x 15'), exclusive of aisle width.

Justification: While this standard is for Iowa City specifically, this applies more generally to parking lots nationwide. These two subsections of this standard apply to our project by giving us dimensions to work within. Therefore, any hardware we deploy to the lot should not infringe on these dimensions.

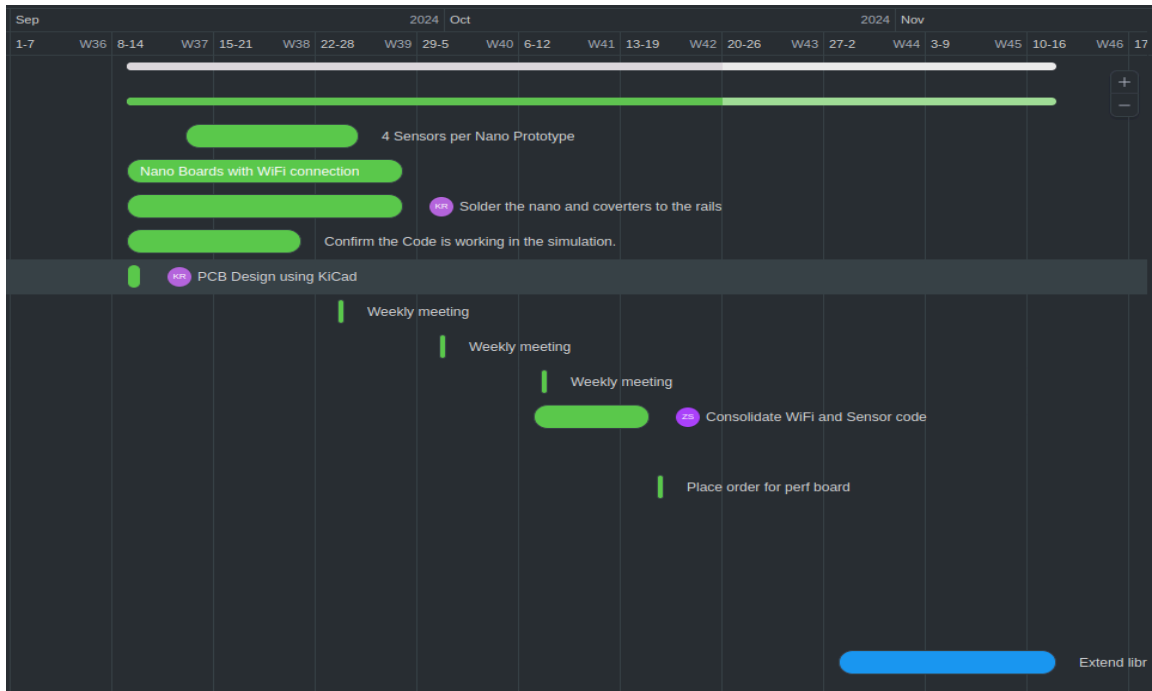
## **3 Project Plan**

### **3.1 PROJECT MANAGEMENT/TRACKING PROCEDURES**

Our team has decided to adopt the agile management style. As a group, we have decided it is best to meet twice a week to work on parts of our project while also meeting with the client/advisor once a week. While we meet often, we do not have a structured set of tasks each time we meet and just tackle whatever we feel is most pressing that day. This works best for us because, between the hardware and software, many unexpected issues could arise that, if we were on a stricter timeline, would cause significant shifts in focus that would delay the project completion time.

We are using a project management software called ClickUp to track our progress. This website allows us to organize our tasks into different categories based on the completion of the task. ClickUp allows specific team members to review and complete the assigned tasks as needed. This makes it easy for our team to see who should be doing what and who to contact if there is an issue. Additionally, we are organizing our files in Google Drive to stay orderly.

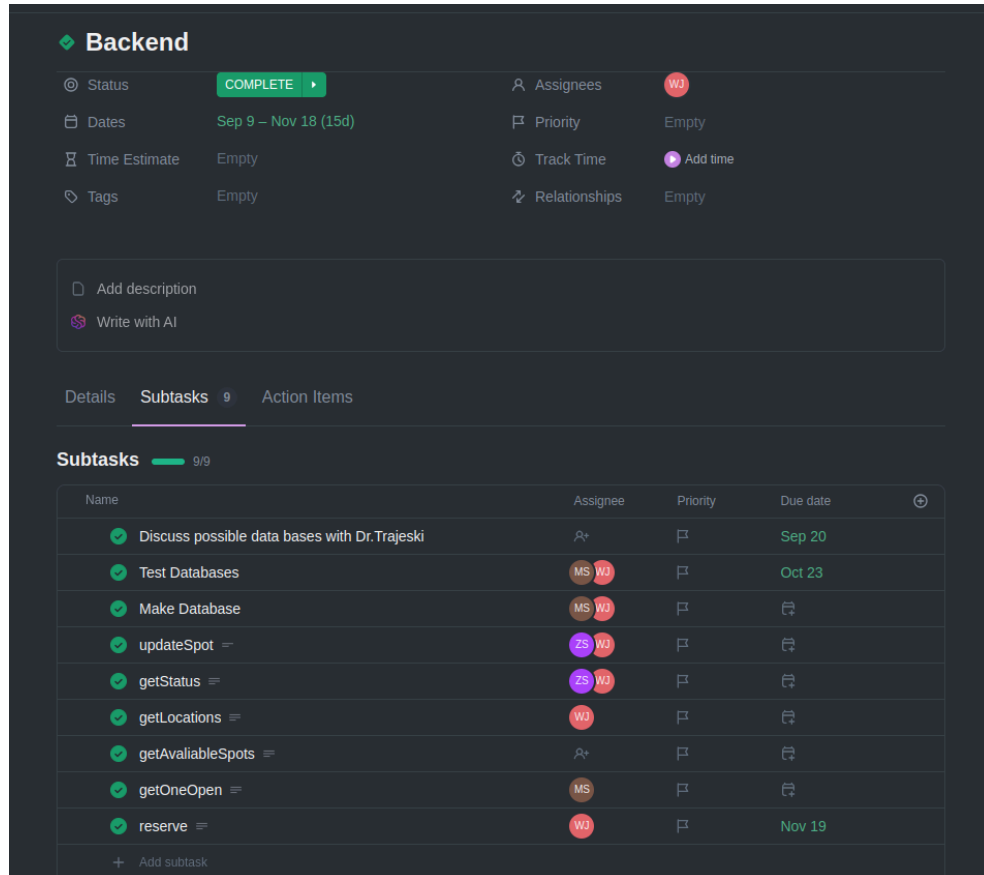
Previously, our team was using Trello. However, this became limited due to the inability to create Gantt charts, the lack of task categorization, and the increasingly complex relationship between our tasks. Switching to ClickUp addressed all these problems while giving us other valuable features.



**Figure 3.2.1:** Hardware Gantt Chart

### 3.2 TASK DECOMPOSITION

As an agile team, we decompose tasks as small as possible; for example, when considering the overall task of researching information about the hardware, we broke it down into researching boards, sensors, how to power the boards, etc.



*Figure 3.2.2: Decomposition of Software Team*

### 3.3 PROJECT PROPOSED MILESTONES, METRICS, AND EVALUATION CRITERIA

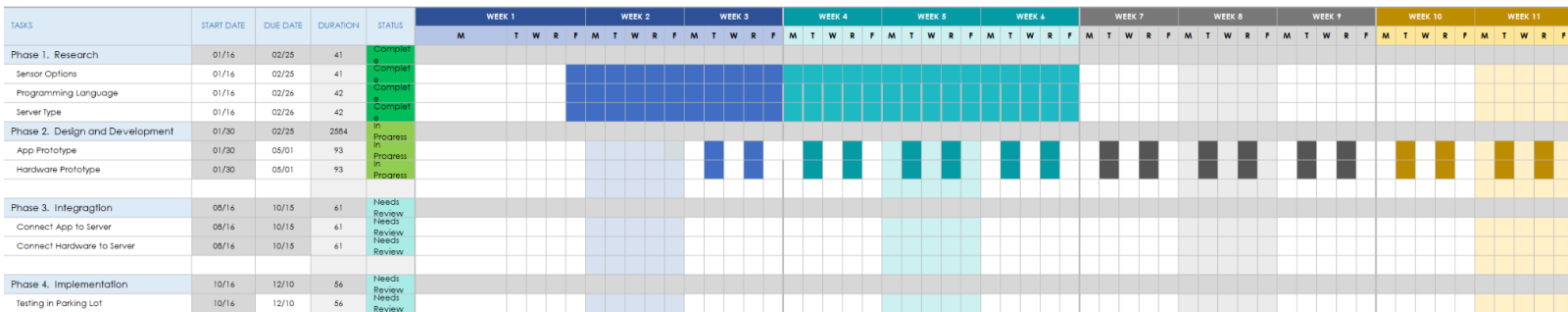
#### Hardware Team

One of the main upcoming objectives for the hardware team is to have a prototype by the end of the semester. We have broken this down into more achievable tasks to accomplish this goal. The first is to get the ultrasonic sensor working with Arduino. The success of this task will be measured by having LEDs connected to the Arduino that will light up if a vehicle is within 70cm of the ultrasonic sensors. The next task was to link multiple sensors to one board and get the system functioning for all the sensors. This is easily measured by getting the system to work with a standard test. Our goal at the end of this project is to know when a car has left a parking spot within 10 seconds of leaving. In other words, we want our program to have live data with a buffer of 30 seconds.

**App Development Team**

The app team has many milestones to accomplish to succeed in creating a usable application. For example, making the first functional prototype is the main upcoming objective for the app development team. This involves having a functional live application that displays our app requirements. We will first have to finalize our UI design by breaking this task into subtasks. This will be accomplished after discussing which design suits our needs. After choosing a UI design, we will program with React Native. Our prototype will have multiple pages, so the creation of each page could be broken down into a subtask. This extrapolates our main idea of prototyping, as each page will require its prototyping phase.

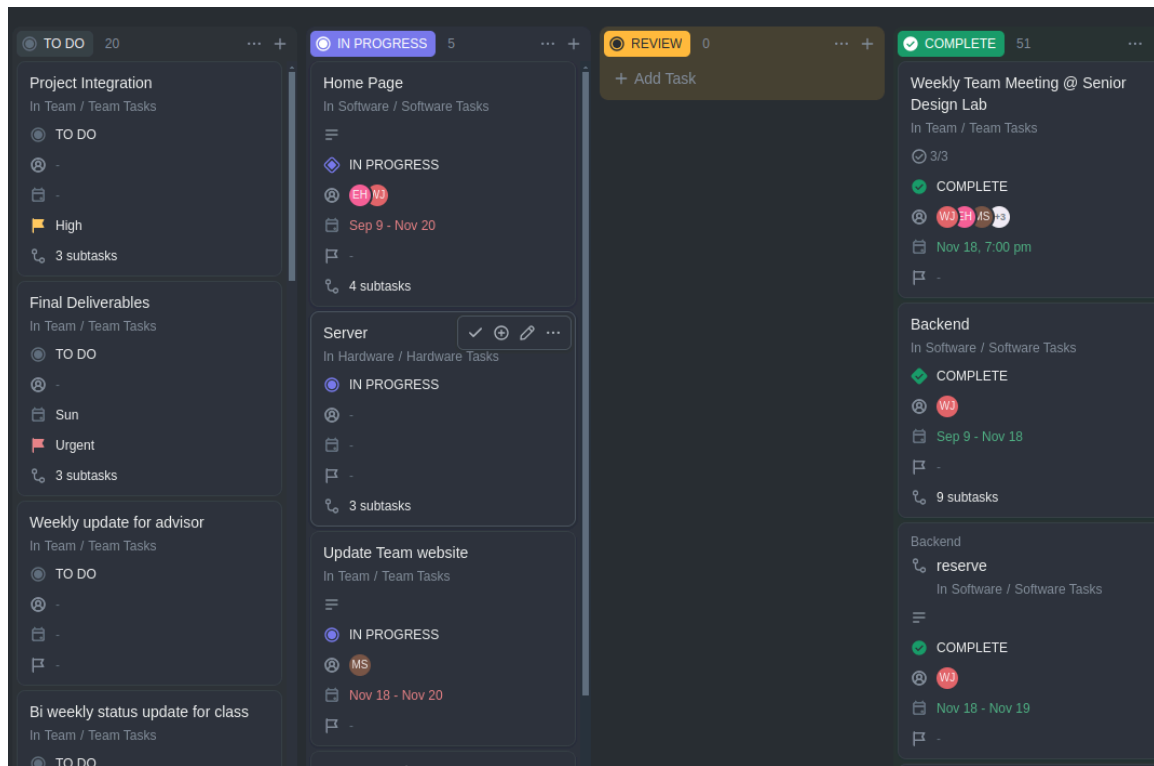
### 3.4 PROJECT TIMELINE/SCHEDULE



**Figure 3.4.1: Projected Timeline of Project**

The image above shows our team's Gantt chart. This is not one of our primary resources in scheduling. Instead, we use our team's Trello board, as mentioned earlier.

For each sprint, we focus on 1-2 tasks each group can work on throughout the week. Once the week is over, we discuss what task we got done, the issues we encountered, and how those issues will impact the following week. Once the problems have been discussed and potential solutions brought forth. We talk about what we should get done in the following week. Coming up with a weekly schedule allows us to address new problems quickly and continually improve our design as we become more knowledgeable about our project.



**Figure 3.4.2:** Team ClickUp board

### 3.5 RISKS AND RISK MANAGEMENT/MITIGATION

We have included a few scenarios in the table below to show examples of the risks attached to our tasks. Overall, our risks are insignificant but not negligible. To mitigate these risks, we will become aware of all possible risks by documenting them before each task is started. This will decrease the probability of these risks occurring, saving us many hours and potential money as it is less likely for equipment to become damaged. If we encounter a hazardous task, we will reevaluate the task to lower the risk factor. For our project, the main risk is losing time to ventures that do not end up contributing to the end goal of our project.

**Table 3.5.1:** Tasks, Risks and Likelihood Probability

Task	Risks	Probability
Learn the basics of React Native	No risks.	N/A
First app prototype	Spending many hours on an idea that does not satisfy our needs.	20%
Create hardware prototype	Similar to our other prototypes, we could lose many hours if the prototyping is unsuccessful.	35%

Testing hardware	We risk damaging the equipment to test the hardware in a real-world application.	10%
------------------	--	-----

### 3.6 PERSONNEL EFFORT REQUIREMENTS

**Table 3.6.1:** Task and Time Decomposition

Task	Hours Required
Design UI	10
First App Prototype	95
Fully Functional Server	30
Arduino Prototype	70
Test Bandwidth Capacity of Prototype	10
Hardware to Server	30
Documentation	20

### 3.7 OTHER RESOURCE REQUIREMENTS

The leading resource for completing our project is hours. With the volume of our school work, it is difficult to allot time for this project. Additionally, it will be necessary for us to acquire an Apple Developers subscription to publish our application to the App Store.

## 4 Design

### 4.1 DESIGN CONTEXT

#### 4.1.1 Broader Context

In a broader context, our design is meant for visitors, students, and staff alike. Many people who have been to the ISU campus have expressed that parking is an unsatisfactory experience as it is often a confusing and stressful task. With all of these people in mind, we must design this project with these users' safety and satisfaction. In [Table 4.1.1](#), our team has listed the areas we prioritize to protect the end users' well-being.

**Table 4.1.1:** Broader Context

Area	Description	Examples
Public health, safety, and welfare	This project affects the general well-being of the users of our application via the inherent risks of using an application in a vehicle, i.e., being distracted by looking at a phone while driving to navigate an area. This is especially pertinent in a pedestrian-heavy area such as a parking lot. However, with our planned navigation system, we intend to mitigate these risks by parking efficiency while the application is being used. New possible hazards are also introduced by deploying sensors to a parking lot. These sensors themselves are harmless to the public, however, the supporting hardware may increase the chances of single-vehicle collisions if not heeded adequately by drivers.	We have been aware of our ultrasonic sensors and have verified that they will be harmless to humans and animals. Having a navigation feature, our application will require a phone to be looked at while driving. We will have a disclaimer on the app to mitigate this.
Global, cultural, and social	Our project is aimed at employees of Iowa State, students, and visitors. It allows these groups to park more efficiently and more conveniently. If used correctly, all users will be able to park quicker.	Our reservation service will eliminate the need to drive in circles to find a parking spot.
Environmental	Our solution will increase power drawn from the relevant power plants in the area and could contribute to the detrimental environmental effects of electronics components mining and manufacturing.	While the power requirements of our hardware systems will be low individually, deploying this system to an entire parking lot may draw a significant amount of power. The software systems will require processing time on phones and a dedicated server, which must be kept cool, further increasing this solution's power draw. The resources needed for our hardware have to be mined and manufactured into the components necessary for deployment.
Economic	Our system's automation could lower the need for monitoring available parking lots. Additionally, if added to multiple parking lots, our system could offer a new job of maintenance and upkeep of the entire system. The university will be liable to perform any maintenance or pay for it if necessary.	The economic impact of our solution will require a payment from users per parking session. An installation and labor fee from the university or relevant parties to install and maintain the hardware we plan to deploy. The university's parking division will have to spend less on refueling their vehicles since our application will

		provide them with relevant overtime and no-payment parking data.
--	--	--

#### 4.1.2 Prior Work/Solutions

Throughout the planning process, we encountered three companies trying to solve a similar issue: parking can be stressful and overwhelming. The first company is ParkMobile, the second is Parkingapp.com, and the third is SpotHero.

ParkMobile makes it easy for users to pay for a parking spot; it also allows users to search for selective parking spots, whether that is to be close to a specific location or to be near an electric car charger (Lister, 2020). ParkMobile is unique in these ways. One advantage of ParkMobile is that they are supported by all platforms making it easy to use. Some things that differentiate our solution from ParkMobile are the ability to guide users to open parking spots and our live data on available spots (Zuckerman, 2019).

Parkingapp.com was very simple and basic. There were not any unique features of their app. Although they did not have any uniqueness to them, they did have some advantages. They are compatible with Parking.com and Parking Passport. Disadvantages for this company are their need for more information and the community. When researching this company, our team took this as a takeaway of what not to do and what we wanted to include for our users.

SpotHero allows users to see real-time parking availability and prices based on location. They are achieving something similar to our solution. One advantage of their product is their allowance of user reservations and payments. Although this sounds like a very sound and well-organized solution, it has disadvantages: the accuracy and reliability of their parking lot availability are only sometimes correct, and their service fees increase the overall cost of parking.

Based on the other solutions out there, we have decided that our solution must have advantages for each company to accomplish the best solution on the market by letting users reserve spots from the comfort of their homes, guiding the users to their reserved spots, and allowing users to search for available parking with their specific needs. Our final design will follow this pros and cons list in [Table 4.1.2](#).

**Table 4.1.2:** Pros and Cons List of our solution

Pros	Cons
Tension free operation	It relies on the accessibility of technology
Easy and secure way to pay	There is no complex prevention of inappropriate or illegal parking
Ability to reserve a parking spot	Reckless drivers could damage technology
Live data on parking lot availability	Handling app users and non app users
	Hardware maintenance



### 4.1.3 Technical Complexity

#### **Hardware**

On the hardware side, our team uses [WiFi NINA](#) signals to send data between Arduino boards and a server. This will be done via the NINA W102 chip on the Arduino Nano 33 [IoT](#) board. This chip allows the board to be used as an internet access point. This WiFi functionality is accessed through the Arduino library system. This connection between hardware and a server is essential for our project. It is how hardware and software are able to talk and interact. Hardware will send spot status to the server and receive if a spot is reserved. Based on this information and sensor data, it will update the system LEDs to the correct color, red occupied, green for open, and white for reserved. Finally, our hardware team will review the documentation and circuitry to minimize the system's power consumption while maximizing its sustainability and ease of maintenance.

#### **Software**

The software portion of our design involves multiple levels of software development. These include mobile application development, server communication, and managing a database. The application is the vessel for users to communicate with our server. The data needed by the server will be stored in our database. The server is essential as it is responsible for receiving the live data from our hardware. The user interface for the application will be simple, but this is a benefit of our design as it will ensure all users successfully operate our system. All three portions of the software team are essential to communicate with the user and hardware.

## 4.2 DESIGN EXPLORATION

### 4.2.1 Design Decisions

Several critical design decisions were pivotal in developing our innovative solution to shaping the final product. Selecting the appropriate sensor for vehicle detection was a foundational step, ensuring our system could accurately and reliably identify vehicles in various conditions. Recognizing the diversity of our user base, we also strategized how to assist users not utilizing our app, aiming for inclusivity and accessibility in our services. The reservation system presented another complex challenge, prompting us to devise a user-friendly, efficient method for managing bookings. Finally, determining the most convenient times and locations to make payments was crucial as we sought to streamline the user experience and enhance satisfaction. Each of these decisions was made carefully considering their impact on functionality, user experience, and the overall success of our project.

### 4.2.2 Ideation

When choosing sensors to detect vehicles, we went through a few iterations. First we started with [infrared sensors](#) because our client wanted to use those if possible. However, after some feasibility testing, we learned we would need an emitter and receiver for each spot. Not to mention that unskilled drivers would likely hit the emitter and receiver. Then, the team went through a brainstorming phase where we discussed possible ideas for a solution. Cameras, [LiDAR](#), and Ultrasonic sensors were brought up during the brainstorming. We even discussed a new technology allowing the parking spot to detect when weight is present using a pressure plate sensor. Ultimately, we went with Ultrasonic sensors because the team is most familiar with them, their relatively low cost, and their ability to function in various conditions that other options may fail.

### 4.2.3 Decision-Making and Trade-Off

Throughout our project, decision-making and looking at trade-offs were very important. We had to make multiple decisions throughout this project, but the two most important ones were what software to use for

our application and what microcontroller to use for our hardware. When considering what software to use for our project we went with React Native and Node JS. We came to this decision because react native allowed us to leverage our current Javascript experience.

- React Native:
  - Cross-platform
  - Easy integration with 3rd party services.
- Node JS:
  - included everything we needed without extra overhead

When looking at what microcontroller to use, we as a team decided to use Arduino, more specifically, Arduino Nano 33 IoT board. Our team has the most experience working with Arduino and thought that using one of their boards for our project would be most beneficial to us. Arduino has countless resources for help and troubleshooting. Arduino also makes it easy for hardware to interact with software. We decided to go with the Nano 33 IoT board for its size and specific capabilities.

- Arduino Nano 33 IoT board:
  - Small size with 14 input/output pins
  - Wifi Capability through the Wi-Fi Nina chip

These decisions were essential for making our project come together and end with a successful prototype.

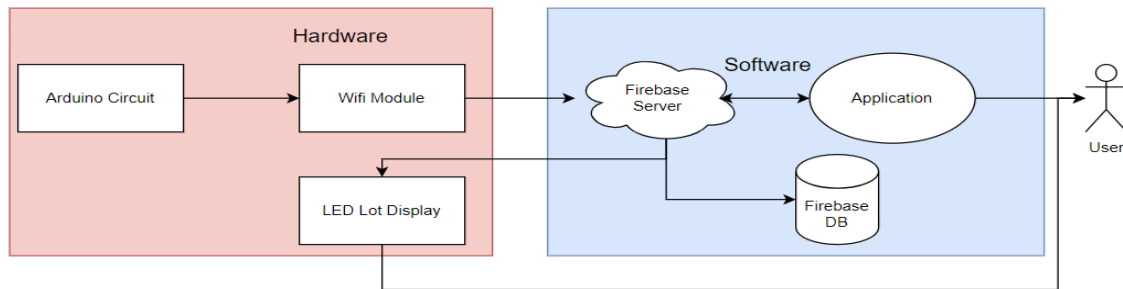
## 4.3 PROPOSED DESIGN

### 4.3.1 Overview

For our intelligent parking lot system to operate, ultrasonic sensors pointed at each parking space will detect the state of the parking space or whether a space is occupied. This information will be sent to our server utilizing an Arduino board connected to the internet through the NINA chip onboard. This information will then be updated in a server database. This information will be accessible by the mobile application. Our users will interact with our system via a mobile application. This app will allow users to search for a parking lot near their destination, reserve a space, and pay for parking. The user can simply pay for parking without participating in the reservation process. Once the user parks in their reserved parking spot, the application will transition into the payment page, where the user will securely enter their payment information. If the user bypasses the reservation system, they will park in the lot and be instructed to take note of the parking space number where they parked. They can enter this information into the app and pay for parking. If a driver does not have the application downloaded, they can scan a [QR code](#) located in the lot to access a “one-time use” version of the application where they can pay.

### 4.3.2 Detailed Design and Visual(s)

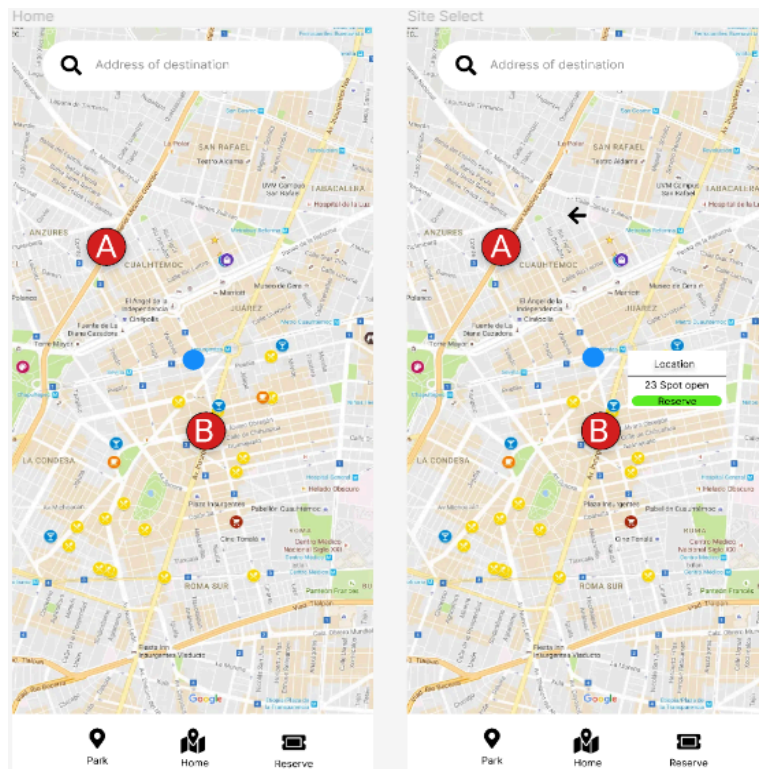
#### **Overall Design:**



**Figure 4.3.1:** Overall Design Flowchart

Our system is broken into two main components: hardware and software. The hardware component of the system takes the multiple Arduino circuits and uses the wifi module to output our hardware information to our server. The software component of our system uses the server loaded with the sensor information and sends it to the application. The application is a way for users to check and reserve parking spaces. The server is also used to send information back to the LED display. This display will represent the parking lot availability, which allows users without the application to use our system.

### Software:



**Figure 4.3.2:** Application User Interface

Payment

← Payment

Spot #

Pay

Full Name

Country

Address

Card

Card number

Expiration Date

Security Code

Submit Order

Park

Home

Payment (Reserve)

← Reserve

Spot #

25

Pay

Full Name

Country

Address

Card

Card number

Expiration Date

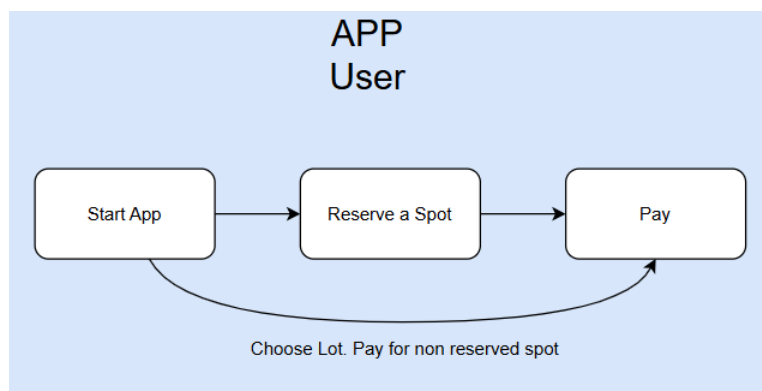
Security Code

Submit Order

Park

Home

**Figure 4.3.3:** Application User Interface Continued



**Figure 4.3.4:** Application Flowchart

Above is the flowchart for our application. The user interface of our application will consist of two different pathways: Reserve and pay.

### **Reserve:**

The user will open the application and choose the reserve tab on the navigation bar on the screen's bottom. The user will then enter their destination in a search box. Each parking lot will have a respective box showing the address, distance from the destination, and number of available parking spots. The user will choose their desired lot to continue to the next step. All of the map functionality will be completed utilizing Google Maps.

After choosing a lot, our application will signal the server to pull an available lot number. This number will be stored in a new variable to ensure actual reservations. Once parked, the application will continue to the payment function.

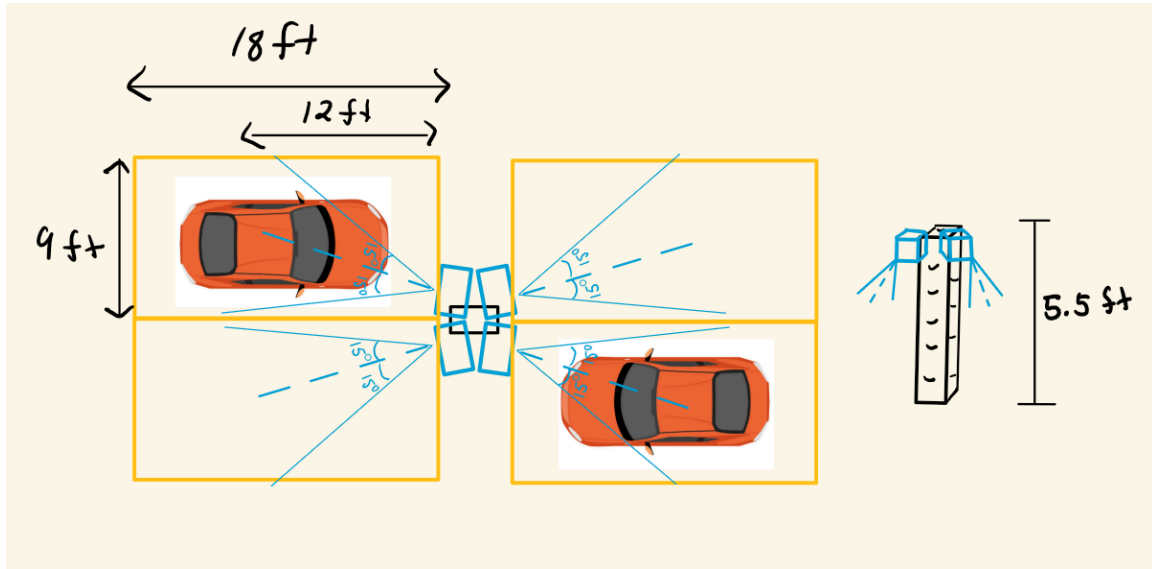
### **Pay:**

For the users who arrived at their reserved spot and those who did not participate in a reservation but are still using the lot, the payment process will begin by selecting the app's payment button or scanning a QR code in the parking lot. The payment page will have prompts for spot number, credit card information, and amount of parking time. The users who reserve a spot will have their spot number automatically entered into the spot number prompt. In contrast, the other users will have to take note of their space number, which will be located on a sign in front of the space, and enter it into the app manually. After the necessary information is entered, the app will create a receipt, which will be downloadable for the user. This is the final step of the payment process.

### **Hardware:**

On the hardware side, our team has set out to create an Arduino-based detection system that will be used to sense whether cars are parked or not. Hardware uses WiFi signals to send data between Arduino boards and a server. This will be done via the [NINA W102](#) chip on the Arduino Nano 33 IoT board. This chip allows the board to be used as an internet access point. This connection between hardware and a server is essential for our project. It is how hardware and software are able to talk and interact. Hardware will send spot status to the server and receive if a spot is reserved. Based on this information and sensor data, it will update the system's RGB LEDs to the correct color, red occupied, green for open, and white for reserved.

### Physical Parking Lot Setup:



**Figure 4.3.5:** Sketch of Physical Sensor Representation

1 Arduino Nano 33 IoT per post with four ultrasonic sensors connected to it. Each sensor will point downwards toward an individual parking spot from an elevated position on the post.

These sensors will be pointed down at an estimated angle of  $30^\circ$ , as represented by [Figure 4.3.5](#). This setup allows for minimal posts to be inserted into the parking lot.

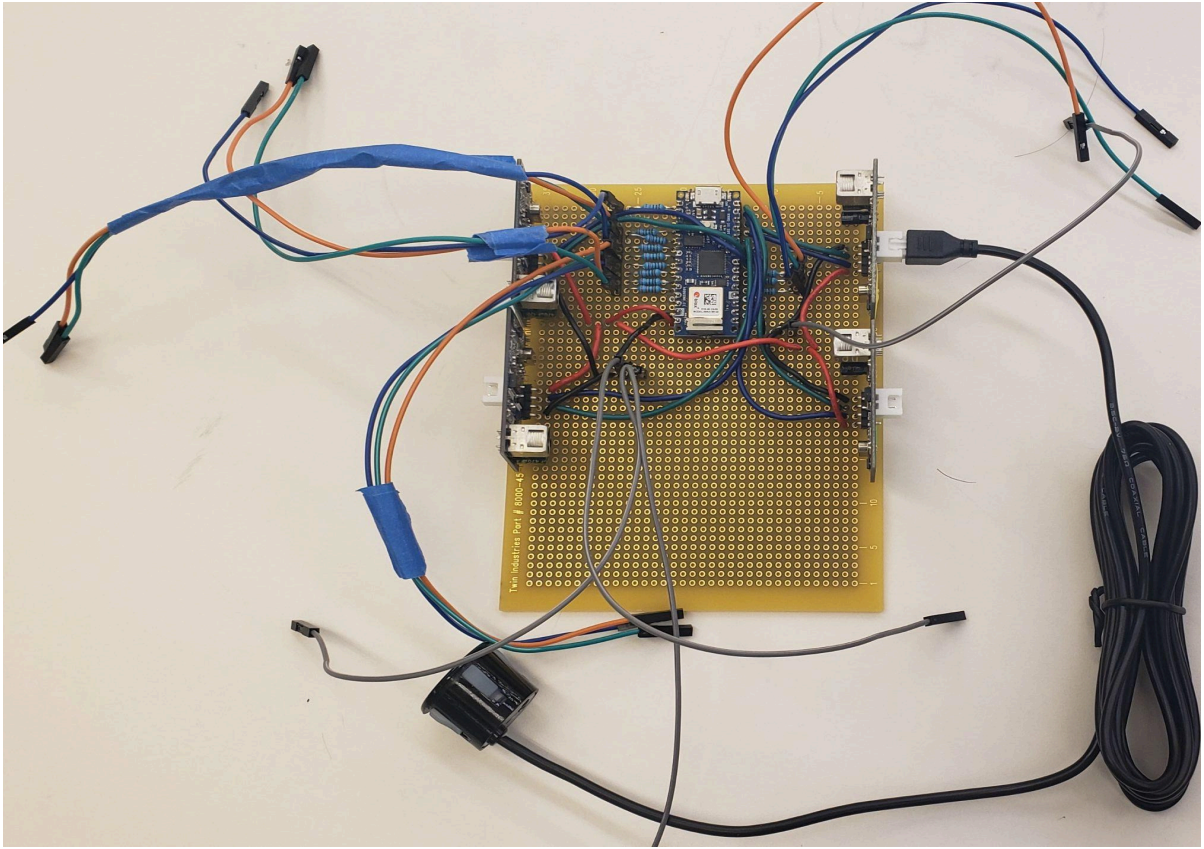
The diagram illustrates the hardware setup for the project. It features an **Arduino Nano v3.x** microcontroller. Power is supplied via a **+5V** rail and **GND**. Four **HC-SR04** ultrasonic sensors (U1, U2, U3, U4) are connected to the Arduino's digital pins. Each sensor's VCC is connected to +5V, GND to ground, TRIG to a digital pin (D13, D12, D11, D10), and ECHO to another digital pin (D9, D8, D7, D6). The Arduino's RESET pin is connected to a **3V3** supply. The Arduino's AREF pin is connected to a **3V3** supply. The Arduino's A2 pin is connected to a **GND** supply. Four **KPS-5130** piezo sensors (D1, D2, D3, D4) are connected to the Arduino's analog pins. Each piezo sensor's R (Resistor) is connected to a digital pin (D4, D3, D2, D13) through a **220** ohm resistor (R1, R2, R3, R12). The G (Ground) and B (Bias) pins of each piezo sensor are connected to a common ground line. The piezo sensors are also connected to a common ground line through a **220** ohm resistor (R4, R5, R6, R7, R8, R9, R10, R11).

Each Arduino Nano board will have four 5V (Connected to red wire represented in [Figure 4.3.6](#)) ultrasonic sensors connected to it. The sensors will trigger on their own pins (Teal wire represented in [Figure 4.3.6](#)).

Coming out of the Arduino Nano board, there are 4 RGB LEDs. Each color coming from the Nano goes through a 220-ohm resistor and then goes to the LED. These LEDs will be lit up either red, green, or white. Red represents an occupied spot, green represents an open spot, and white represents a reserved spot.



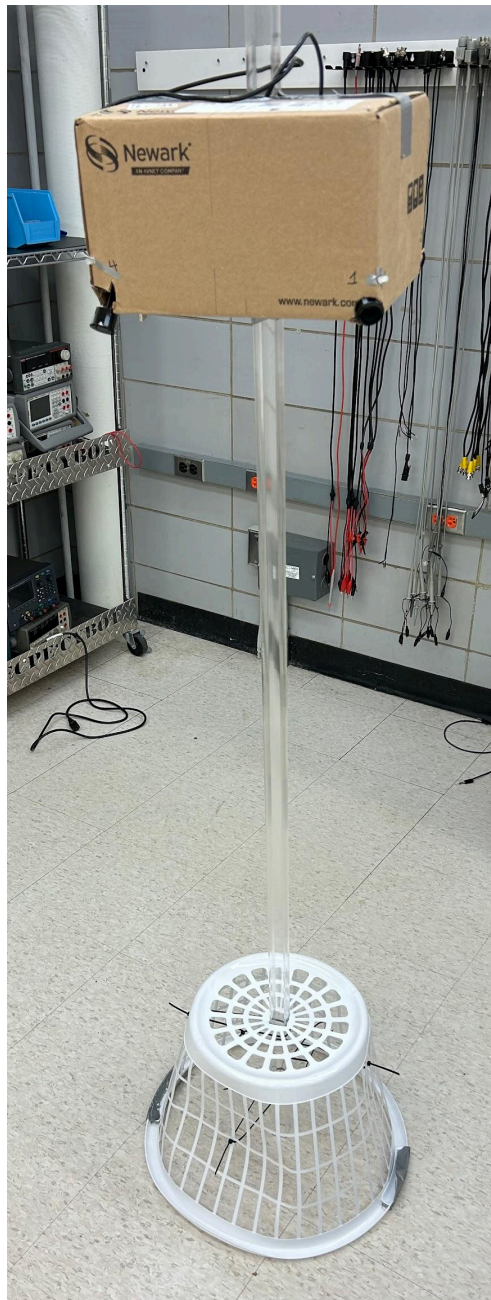
### **Final Prototype:**



***Figure 4.3.7:*** Prototype Circuit

This is an image of a prototype for our hardware circuitry. Everything is being controlled by an Arduino Nano 33. There are 4 waterproof sensors coming off of the Nano, each sensor has a power, trigger, echo, and ground pin, all connected to the Nano on different pins. This system also has 4 RGB LEDs coming from the Nano.





**Figure 4.3.8:** Hardware Prototype Setup

This is an image of our final design prototype for 4 parking spaces. There is a box mounted to a 5.5-foot pole. The box contains the 4 ultrasonic sensors, 4 RGB LEDs, and the Nano board circuitry. This setup has the sensors mounted in each corner of the box with an LED representing the spot status mounted right above the sensor. The rest of the circuit will be placed within the mounted box. This is only a prototype. If this design were to go further, we would mount a weatherproof box to protect our circuit and our components.

### 4.3.3 Functionality

Users can enter our lot regardless of whether they have the app or not. If the user has the app, they will be able to navigate to their desired location, see what parking lots are nearby, and determine if they are able to park there either as a student, a staff member, or a visitor. Once they determine the lot they want to go to, the user can check the availability of the parking lot. Based on that information, they can either drive directly to the lot or reserve a spot in the parking lot. Our application will direct the users who reserve a spot to their respective location and then will pay upon arrival. Users who do not reserve a place will pay after parking by using the app and entering their spot number, which will be posted near their parking space. All necessary data will be taken from ultrasonic sensors, sent to our server, and stored in a database.

### 4.3.4 Areas of Challenge

Our hardware team encountered a few challenges during the project. The first and most prominent issue was connecting to the WiFi with our boards. On program startup, the boards try to connect to the Wifi but almost always fail the first few attempts. We could not resolve this issue, but it maintains a stable connection once the board connects. The next major issue we encountered was the implementation of our interrupt handler. For our system, we wanted to make a post request every second, sending an updated spot status. To do this, we decided to use an interrupt handler to allow the one-second time interval to be as accurate and efficient as possible. While working on the functionality, we did not realize we needed to turn off the handler after one second, and the post request was made. After testing without turning off the handler, we ended up software-locking two of our boards. This was because the handler needed to be synchronized with our system clock, but when making our post requests, it caused it to desync. This issue set us back a few days as we were first unsure why the boards were locked, but upon further research, the issue was resolved.

Finally, for the software, we had to ensure that our application is intuitive and safe enough to be used while driving, even in a busy parking lot. We understand that using a phone while driving can be a safety risk. Therefore, we had to minimize the user input while driving and use an acceptable interface for moving applications. Our application does not take any input while the user is driving.

When developing our application, we encountered many challenges. Firstly, as this app was developed in a cross-platform software environment, some components appeared differently in testing, affecting some users' overall functionality. Our testing platform, Expo, had a software update that delayed our development slightly as well.

## 4.4 TECHNOLOGY CONSIDERATIONS

Describing the distinct technologies we used in our design by highlighting the strengths, weaknesses, and trade-offs made in technology. Also, discussing possible solutions and design alternatives below:

*Table 4.4.1: Sensor Options*

Sensor	Pros	Cons
Infrared (IR)	<ul style="list-style-type: none"><li>• Preferred by the client</li><li>• Cheap</li></ul>	<ul style="list-style-type: none"><li>• Requires two sensors per parking space</li><li>• Positioning sensors out of the way of users would be tricky</li><li>• Sensitive to the color of the car</li></ul>

<b>LiDAR</b>	<ul style="list-style-type: none"> <li>• Can detect everything in an area</li> </ul>	<ul style="list-style-type: none"> <li>• Expensive</li> <li>• Not entirely appropriate for our use case</li> </ul>
<b>Ultrasonic</b>	<ul style="list-style-type: none"> <li>• Cheap</li> <li>• Only need one sensor per parking space</li> <li>• Familiar to the team</li> </ul>	<ul style="list-style-type: none"> <li>• Little control over where the signal goes</li> </ul>
<b>Pavement</b>	<ul style="list-style-type: none"> <li>• Discrete</li> </ul>	<ul style="list-style-type: none"> <li>• Super expensive</li> <li>• Will have to renovate the whole lot to implement</li> </ul>
<b>Cameras</b>	<ul style="list-style-type: none"> <li>• Could work over multiple spaces</li> </ul>	<ul style="list-style-type: none"> <li>• Power intensive</li> <li>• Difficult to implement</li> <li>• Possibly expensive</li> </ul>

Each of these different sensors would ultimately lead to a similar outcome for our project. Our design would rely on these sensors to determine whether a car is parked in a spot. Based on this information, our final decision was to use ultrasonic sensors because the team is most familiar with them, and relatively low cost, and their ability to function in various conditions that other options may fail in.

#### 4.5 DESIGN ANALYSIS

On the hardware side, we have created a base prototype for the detection system. This system takes an Arduino Nano and connects four ultrasonic sensors and four RGB LEDs. We have tested this prototype, and it is working well. We then implemented Wifi functionality so the boards can send sensor information to our server and get reservation information to update our LED system. Based on this setup, we can successfully mimic the parking process within a parking lot. We have looked into weatherproofing options and have decided to encase the circuit in an enclosed metal case. We could not provide insulation for the board but looked into thermal bubble foil to keep our circuit at working temps while shielding it from fluctuating outer temperatures.

The software team coded an application that operates on IOS and Android devices for the application. The application allows a user to find a parking lot and reserve a spot if one is available. Then, the user can pay the parking fee promptly. Unfortunately, we could not accomplish a one-time-use function for non-application users, but implementing it could be completed quickly. Additionally, our application is currently out of stock to the public.

Our server and database implementation has been completed successfully. They operate as intended, allowing our system to communicate fully. Our server was made with Express.js, and our database is an SQL database. Together, these two components enable the storage of data and communication across hardware and software.

## 5 Testing

### 5.1 UNIT TESTING

We conduct hardware testing on a feature level to ensure consistent functionality in varying conditions. This includes testing the detection of objects with ultrasonic sensors and data transmission to the server. We do manual regression testing on the software side when new features are implemented. This approach was

taken because setting up and maintaining a testing library would have strained our human resources. It is resulting in a significant decrease in output.

## 5.2 INTERFACE TESTING

Our design incorporates two primary interfaces: a parking availability tracking hardware and a user-facing application that displays parking lot status. We will thoroughly test the hardware's sensors, communications, and reliability to meet our requirements. We will evaluate the software's user interface, performance, and safety. The user interface must be visually appealing and intuitive for most users. While developing, we checked that the application on both IOS and Android looked and performed similarly.

## 5.3 INTEGRATION TESTING

The two significant integrations are the connection from the hardware to the server and from the server to the application. We need events picked up by the hardware to flow through the pipeline in real time. By that, we mean that once a car has completely pulled out of a parking spot, it should appear in the app as open. Our best way of testing this is to simulate the experience in our senior design lab. We will connect the hardware and app to the server and simulate a car pulling into a spot. Our goal will be to upload the hardware detection information to the server, and then the server will send that information to the app, which will be updated within 20 seconds.

## 5.4 SYSTEM TESTING

The final step of testing would be field system testing. In this test, we would set everything up like a client's lot and test various interactions and edge cases. If any issue occurs, we need to troubleshoot it on-site, which could be better. We only want to do system testing once all our unit and integration tests pass.

## 5.5 REGRESSION TESTING

We explore ways to identify and prevent hardware malfunctions as we plan our hardware implementation. While our unit tests will cover much of our software regression testing, we are also working on a comprehensive test suite to detect potential regressions as we develop our code. As the hardware is responsible for capturing crucial data and the app is our users' main point of interaction, these components must remain stable and reliable.

## 5.6 ACCEPTANCE TESTING

Our acceptance test would mostly be handled through meetings with our client. Once we present our final demo, the client will have the final say on whether our design meets his vision. We have the requirements we came up with, but they can quickly become outdated if the client shifts thinking.

## 5.7 SECURITY TESTING

To avoid handling user payments, we use a third-party API from [Stripe](#). Stripe has a good history of making payments easy for users and developers. However, we must ensure that we store only the necessary user information. We should be able to detect these through our code reviews.

## 5.8 RESULTS

Our unit and integration test will allow us to catch issues early, verify that we are ready for the next step, and assist in identifying system regression. We will continually communicate with the client to meet their needs when meeting requirements.

## 6 Implementation

The implementation phase of our project was a complex process of combining various systems into one final prototype.

On the hardware side, we have various features that allow our boards to operate efficiently. We constructed functions that gave us the ability to connect to the school's WiFi system which will allow us to send spot status information to our server. Along with the Wifi functions, we added ways to verify the connection process for troubleshooting and maintenance purposes. We also designed functions for our ultrasonic sensors that allow us to take multiple measurements of the parking spots and take the average distance of any object detected for the most accurate information. We also implemented an interrupt handler that will help with updating the server consistently. With the handler, it allows us to make post requests every second using the Arduino's system clock. Lastly, we added RGB functionality so users can observe spot status'. This consists of an RGB LED handler that takes the measurement data combined with the information from our server and updates the color of the LEDs based on whether the spot is open, closed, or reserved.

On the software side, we developed both a server and a mobile application. For the server, we tested multiple databases to see what would fit our needs. We also implemented multiple requests that would be used by both the hardware and the mobile application. Then for the mobile application, we created two pages; a home page and a payment page. We broke the development down page by page. We simply created each component individually and tested its functionality. After functionality was ensured, we continued to add new components. After completing the front-end design, we added server requests to perform communications between the server and the application. These requests get information about each parking lot and assign the user to a parking space if they are reserving a spot.

## 7 Professional Responsibility

In this project, our team's approach to professional responsibility is embedded within a well-defined team contract that outlines individual roles, communication protocols, decision-making policies, and quality control measures, ensuring that all members adhere to high standards of ethical conduct and professional practice. Our team's commitment to professionalism is evident through structured strategies for collaboration, inclusive participation, and the resolution of any inclusion issues, with designated responsibilities such as client interaction and project leadership reinforcing the importance of maintaining professional integrity throughout the project's lifecycle. By agreeing to a clear set of consequences for any breaches of this contract, our team underscores each member's serious commitment to the project's success and upholding the professionalism expected in the engineering domain.

### 7.1 AREAS OF RESPONSIBILITY

We will discuss how the topics relate by applying the IEEE Code of Ethics to our broader context table ([Table 4.1.1](#)). The IEEE Code of Ethics stresses the importance of protecting the public by encouraging engineers to stand up for humanity against corruption. Global, cultural, and social topics concern being truthful when releasing information to the public and treating all individuals with respect. The environment portion

**Table 7.1.1: Areas of Responsibility**

Area	IEEE	Differences from NSPE
Work Competence	An engineer should not work in an area outside of their expertise	Very similar
Financial Responsibility	Avoidance of conflicts of interest	NSPE is more focused on shareholders and clients
Communication honesty	Communicate truthfully to protect the population	NSPE is more generally about the company
Health, Safety, and Well-being	Protecting the public by encouraging engineers to stand up for humanity against corruption	NSPE has similar terms for this category
Property Ownership	Keep private information secure	Protect the property of all clients and privacy
Sustainability	Make it known if the environment is in danger	NSPE is more detailed in this category
Social Responsibility	Encourage coworkers and seek to improve public knowledge	Similar guidelines

## 7.2 PROJECT-SPECIFIC PROFESSIONAL RESPONSIBILITY AREAS

**Table 7.2.1: Project Specific Responsibilities**

Area	Is it relevant	Performance
Work Competence	It is irrelevant to our project because we must learn and work outside our current knowledge. However, we still do our best to create an effective solution.	N/A
Financial Responsibility	It is essential to be responsible for the user's payment information and the money they send. We want to do our best to avoid fraud and embezzlement.	Medium

Communication honesty	When reporting to the client, we always ensure that we are telling him the truth regardless of how it may affect his opinion of us.	High
Health, Safety, and Well-being	Since we are creating an application meant to be used while driving, we need to ensure that we are not inhabiting the users driving when using our app.	High
Property Ownership	We are doing our best to make our client and everyone involved in this project feel included and considered	Medium
Sustainability	We are constantly considering the environmental impacts of our design	High
Social Responsibility	We do our best to keep each other in check. If someone is doing something questionable, we can always talk it out.	High

### 7.3 MOST APPLICABLE PROFESSIONAL RESPONSIBILITY AREA

Safety is our most important concern because we want to ensure our users are safe using the app.

## 8 Closing Material

### 8.1 SUMMARY OF PROGRESS

Our team has created an embedded real-time system that provides parking spot availability information to the students, faculty, staff, and visitors of Iowa State University. This system utilizes ultrasonic sensors to detect vehicles and send this information to a server and database. The server sends up-to-date information to a phone application. Finally, a user can reserve a parking spot through the application, this reservation is sent to the server, which updates the database and an LED light in the reserved parking spot.

#### Hardware Accomplishments

The hardware team has created a prototype that takes in measurements from ultrasonic distance sensors and uses the distance to determine if a spot is occupied. The hardware system is connected to the internet through a library that we designed and built.



## Software Accomplishments

The software team successfully created a mobile application that can reserve parking spots and take payments. We managed to have an appealing design that is user-friendly. Along with the application, we created a server and database to enable communications across domains. We performed great research on app development that will serve us well in the future.

### 8.2 VALUE PROVIDED

Our prototype gives a good glimpse into how a system like this could work. It successfully allows users to see which spots are available and allows users to reserve spots in advance. However, it needs some improvements before it can solve the initial problem. Currently, only users that drive into a lot can see the LEDs at each parking spot. It would be better if users could see not only how many spots are available (like our current implementation) but also which specific spots were available. That would allow users to drive directly to a subsection of the parking lot with the most open spaces. Another needed improvement is allowing users to choose which spot they reserve. If a user's lot preferences depend on how close they can get to a building, users need to be able to reserve spots directly instead of being assigned a spot by the system.

### 8.3 NEXT STEPS

There are a couple of next steps hardware would have wanted to take to improve our final product. The first step would've been implementing an interrupt timing system for when to send data to the server. This would have helped ensure the server does not get overloaded with requests. Another step the hardware team wishes they completed is printing the [PCB](#) design we created and coming up with a more finalized mounting for the sensors and the board. Doing this would have allowed for easier and more efficient mass production if our project were to be fully implemented.

A few things that the software team would like to have done are: Create a screen which visualizes all of the spots in a parking lot and allows users to select a spot to reserve. This would allow a more natural flow to the app and help it feel intuitive. Secondly, the application should help guide users to their parking space through some form of navigation API similar to Google Maps™ navigation.



## 9 References

- [1] IEEE SA, “IEEE Standards Association,” *IEEE Standards Association*.  
<https://standards.ieee.org/ieee/3156/10834/>
- [2] IEEE SA, “IEEE Standards Association,” *IEEE Standards Association*.  
[https://standards.ieee.org/ieee/White\\_Paper/10123/](https://standards.ieee.org/ieee/White_Paper/10123/)
- [3] “14-5A-5: CONSTRUCTION AND DESIGN STANDARDS:,” *American Legal Publishing*.  
[https://codelibrary.amlegal.com/codes/iowacityia/latest/iowacity\\_ia/0-0-0-23897](https://codelibrary.amlegal.com/codes/iowacityia/latest/iowacity_ia/0-0-0-23897)
- [4] M. Lister, “Get ready for takeoff!: 5 advantages of using the Parkmobile app for Airport Parking,” ParkMobile,  
<https://parkmobile.io/blog/get-ready-for-takeoff-5-advantages-of-using-the-parkmobile-app-for-airport-parking/> (accessed Apr. 16, 2024).
- [5] A. Zuckerman, “Parkmobile Review: Pricing, pros, cons & features,” CompareCamp.com,  
<https://comparecamp.com/parkmobile-review-pricing-pros-cons-features/> (accessed Apr. 16, 2024).
- [6] “Find ways to pay for parking,” ParkingApp.com | Find Ways to Pay for Parking,  
<https://www.parkingapp.com/> (accessed Apr. 16, 2024).
- [7] “Frequently asked questions,” SpotHero, <https://spothero.com/faq> (accessed Apr. 16, 2024).

## 10 Appendices

### APPENDIX 1 -OPERATION MANUAL

#### **Overall Operation**

Our system uses hardware and software elements. The users will only have to interact with our mobile application. The user is responsible for selecting a parking lot to park in and paying for it on the application. All other operations are automated. The hardware detects if a car is in a parking space. The data pertaining to each parking lot is stored in a database, and each parking space has an LED that indicates if a spot is taken, open, or reserved. Once the user wants to leave the parking lot, the hardware will automatically detect their absence.

#### **Application Operation**

Our users will fall into two main categories: Users who reserve parking spots and users who do not reserve parking spots. Each use case will require different operations.

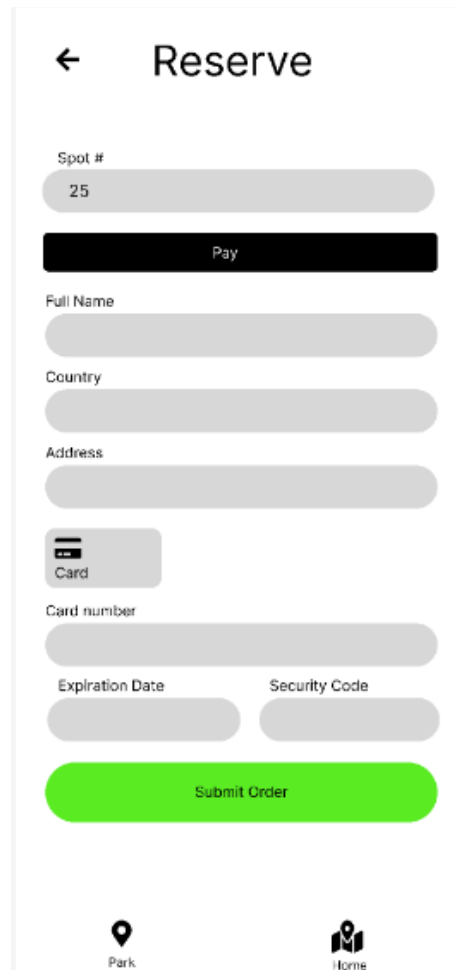
**i) Reservations** - Before a user plans to reserve a space, they must download our application from their smartphone's app store. When they open the app this is the homescreen they will see, [Figure 10.1.1](#). To reserve a parking spot, locate and navigate to a parking lot near your destination using the map,.



**Figure 10.1.1:** App Home Screen

Parking lots are shown with red pins. Tap on the pin corresponding to a lot. Once you have clicked on a spot, a pop-up will show the availability of the parking lot. Click “Reserve” to continue with the reservation process.

Next, you will be taken to the “Reserve” page, Figure 10.1.2.



The image shows a mobile application screen titled "Reserve". At the top left is a back arrow icon. Below the title, there is a "Spot #" label and a text input field containing the number "25". A black button labeled "Pay" is positioned below the spot number. The form continues with labels and input fields for "Full Name", "Country", and "Address". Below these is a "Card" label with a credit card icon, followed by a "Card number" label and a text input field. The bottom of the card section has two labels, "Expiration Date" and "Security Code", each with a corresponding text input field. A large green button labeled "Submit Order" is centered below the input fields. At the very bottom of the screen are two icons: a location pin labeled "Park" and a house labeled "Home".

**Figure 10.1.2:** Reserve Screen

Automatically, a parking spot number will be shown in the “Spot #” box. Enter your personal information to make a secure payment. You will have to enter the following information: License plate number, license plate state, full name, country, address, credit card number, credit card expiration date, and credit card security code. Finally, click “submit order” to complete your payment and reservation process. A receipt will show after the payment has been received.

**ii) Non-Reservations** - For users who are not reserving a parking space, there will be QR codes posted on poles in front of the parking spaces. The user must scan this code with a camera to be taken to a one-time-use version of the application. This page will be similar to the “reservation” page, but the “Spot #” box will be empty, refer to Figure 10.1.3.

The image shows a mobile application screen titled "Payment". At the top left is a back arrow icon. Below the title, there is a "Spot #" label followed by a grey input field. A black button labeled "Pay" is positioned below the input field. This is followed by labels and input fields for "Full Name", "Country", and "Address". A card icon is shown above a "Card" label, which is followed by a "Card number" label and a wide grey input field. Below the card number field are two separate grey input fields labeled "Expiration Date" and "Security Code". A large green button labeled "Submit Order" is at the bottom of the form. At the very bottom of the screen are two icons: a location pin labeled "Park" and a person icon labeled "Home".

**Figure 10.1.3:** Payment Screen

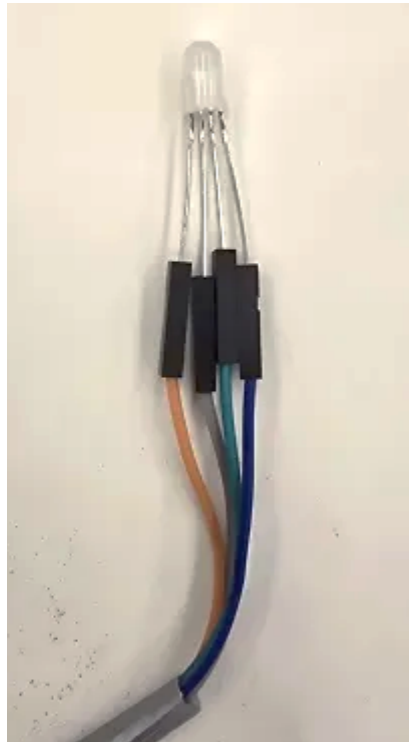
It will be up to the user to find their spot number, which will be on the pole in front of the spots as well. The user must enter the following information: License plate number, license plate state, full name, country, address, credit card number, credit card expiration date, and credit card security code. Finally, click “submit order” to complete your payment and reservation process. A receipt will show after the payment has been received.

## Hardware Operation

The physical setup of the hardware system will be fairly easy. Our system is soldered and set up on [perf board](#). There are only a couple of physical things required to do in order to have the hardware fully operational. The first step is to hook up the 4 RGB LEDs to their connectors. The grey connector will be hooked up to the longest lead on the LED. The orange connector goes to the small lead on the far left, directly next to the ground lead. The green connector will be hooked up to the second longest lead, which is also to the right of the ground lead. Finally, the blue connector will be hooked up to the other small lead on the far right of the LED (See [Figures 10.1.4](#) and [10.1.5](#) for a more visual representation).



**Figure 10.1.4:** Pinout for an RGB LED



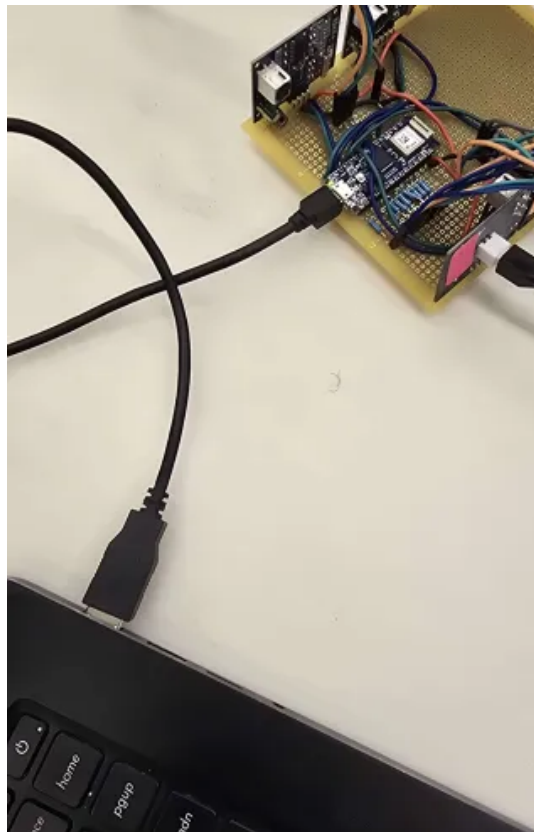
**Figure 10.1.5:** LED Setup

Once all 4 LEDs are set up, the next thing to set up is the 4 ultrasonic sensors. Each sensor will be plugged into a board with a connector. These sensors are set up this way in order to be waterproof. All of the sensors are built to withstand water, but the components that control and process the sensors are not built to withstand water. See [Figure 10.1.6](#) for a visual representation.



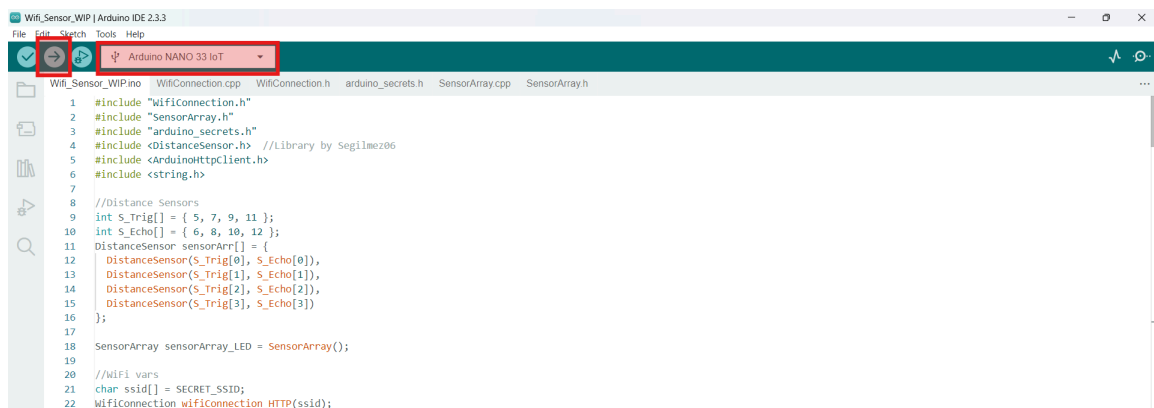
**Figure 10.1.6:** Ultrasonic Sensors Setup

The last physical step is to power the Arduino Nano 33. There will be a cord that is a USB connector to a micro connector. The USB side of the cord will be plugged into a computer, and the micro connector will be connected to the Arduino Nano. See [Figure 10.1.7](#) to see the cord setup.



**Figure 10.1.7:** Arduino Nano Power Setup

Once the physical setup is complete, we need to ensure that the code is uploaded to the board. To complete this, you will need to open Arduino's IDE to the code for our project and ensure the board is connected to our system by looking at the top left drop-down box and it should say Arduino Nano connected. If it is connected, click the arrow pointing to the right that is labeled upload. See [Figure 10.1.8](#) for screenshots of what to do.



**Figure 10.1.8:** Arduino IDE Setup



Once this all goes through, our system should be up and running. You should be able to see the LEDs changing color based on their status. If there is a car in the spot, the LED will be red. if a user reserves the spot, the LED will be white, and if the spot is open, the LED will be green.

If this is not the case, ensure everything is hooked up correctly. If everything is hooked up correctly, try unplugging the USB, replugging it back in, and reuploading the code to the Arduino Nano. This should fix the issue.

## APPENDIX 2 -ALTERNATIVE/INITIAL VERSION

### Hardware Designs

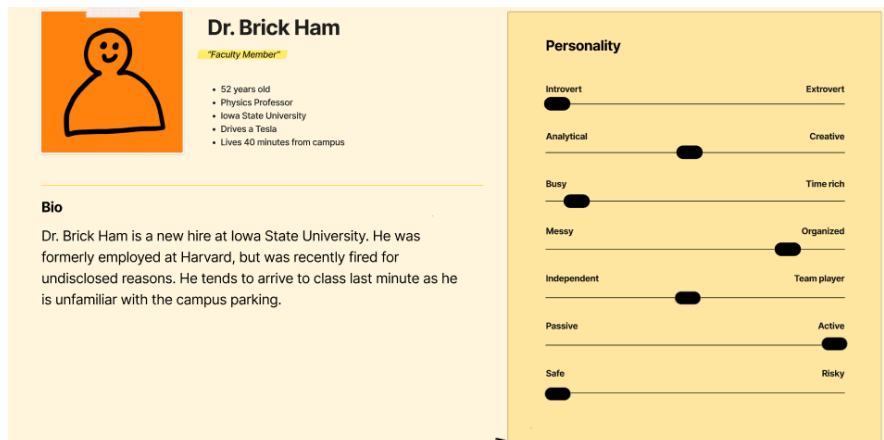
Initial designs for the hardware prototype included the use of the Arduino WiFi Rev2 board which has WiFi capabilities, but a much lower clock speed of sixteen MHz versus the Arduino Nano 33's clock speed of forty-eight MHz. The faster the system's clock is, the more operations per second possible. This is beneficial to us given that we need as many updates as possible to keep the database and thereby the application as accurate as we can. This initial design also had an infrared(IR) transmitter and receiver to be used as our sensors, set up as an IR curtain across parking spaces. When a vehicle would break the sensors' line of sight this would tell the system a spot is now occupied. However, this iteration was abandoned due to a variety of factors. The way we had designed the layout, a car could have been able to knock over a sensor, some vehicle colors could have absorbed the signal or bounced it in a way that could have been picked up by a receiver in another spot.

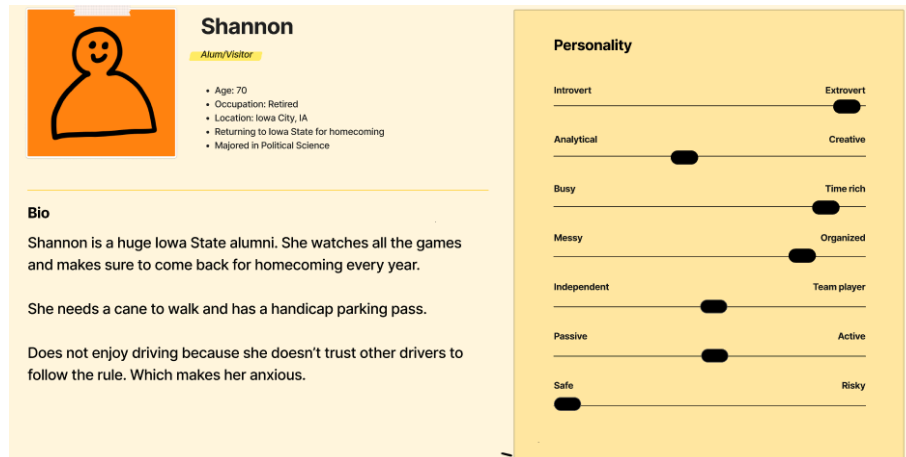
### Software Designs

At the beginning of the project we planned to use a Firebase cloud server. The reason for using Firebase was that we thought it would allow us to solve both how we were going to store data and where we would host the server in one go. However, due to Firebase being a NoSQL solution, it lacks ACID (Atomicity, Consistency, Isolation, Durable) properties. Therefore, we would have experienced unknown behavior when two users try to reserve the same spot. To fix this oversight, we moved to MySQL for our database solution as it uses basic SQL and therefore has transaction processing.

## APPENDIX 3 -OTHER CONSIDERATIONS

### User Profiles





*Figure 10.3.1: User Profiles*

## APPENDIX 4 -CODE

GitHub Repository Links:

Mobile App: <https://github.com/Smartpark-sddec24/Frontend>

Hardware: <https://github.com/Smartpark-sddec24/Hardware>

Server: <https://github.com/Smartpark-sddec24/Backend>

## APPENDIX 5 -TEAM

### Team Contract

**Team Name:** 17

#### **Team Members:**

- |                               |                    |
|-------------------------------|--------------------|
| 1) William Clemmons           | 2) Kennedy Reiling |
| 3) Brian Witherspoon          | 4) Zachary Sears   |
| 5) Mubassir Serneabat Sudipto | 6) Ethan Haberer   |

#### **Team Procedures**

1. Day, time, and location (face-to-face or virtual) for regular team meetings:
  - **Spring semester:**
    - **Project Team: Tuesday from 6:00 PM to 7:00 PM, Thursday from 6:00 PM to 7:00 PM.**
  - **Fall semester:**
    - **Hardware team: Monday, Tuesday, Friday from 1:00 PM to 3:00 PM**
    - **Software team: Friday from 10:30 AM to 12:30 PM**
    - **Project team: Monday from 7:00 PM to 9:00 PM**
2. Preferred method of communication updates, reminders, issues, and scheduling (e.g., e-mail, phone, app, face-to-face):
  - **Face-to-face meetings are preferred but online is acceptable.**
3. Decision-making policy (e.g., consensus, majority vote):
  - **Group conversation turning to majority vote.**
4. Procedures for record keeping (i.e., who will keep meeting minutes, how will minutes be shared/archived):
  - **Shared decision based on group majority decision.**

#### **Participation Expectations**

1. Expected individual attendance, punctuality, and participation at all team meetings:
  - **All group members must come to planned meetings. If unable to attend, the group member must inform the team.**
2. Expected level of responsibility for fulfilling team assignments, timelines, and deadlines:
  - **Each group member will be assigned a relevant task to their skill set as the project progresses. Each team member will be expected to complete the task by the timeline. The group will discuss and work around the issue if the deadline still needs to be met.**
3. Expected level of communication with other team members:

- Team members will be expected to communicate any adjustments made to the project to the team members that will be affected while also making sure other group members understand.
4. Expected level of commitment to team decisions and tasks:
- Everyone will be expected to participate and complete their tasks. There will be open and honest communication.

## Leadership

1. Leadership roles for each team member (e.g., team organization, client interaction, individual component design, testing, etc.):

Role	Description	Assignee(s)
Project Lead	Make sure that the team is organized and on the same page.	William Clemmons
Client Interaction	Responsible for reaching out to the client when needed.	Mubassir Serneabat Sudipto, Kennedy Reiling
Quality Control	Double-check designs and documents to make sure that they meet requirements.	Mubassir Serneabat Sudipto, Zachary Sears, Ethan Haberer
Hardware Design	Responsible for hardware aspects of the project.	Brian Witherspoon, Kennedy Reiling, Ethan Haberer, Zachary Sears
Software Design	Accountable for software aspects of the project.	William Clemmons, Mubassir Serneabat Sudipto

2. Strategies for supporting and guiding the work of all team members:
- a. Constant and honest communication is to be kept when working. Updates are to be given in the updates chat of our team discord.
  - b. Any help needed should be reported in the help channel of our discord.
  - c. Have a structured to-do list so each person knows what to do.
3. Strategies for recognizing the contributions of all team members:
- Have an updated page in our team chat to track what people have completed/contributed.

## Collaboration and Inclusion

1. Describe the skills, expertise, and unique perspectives each team member brings to the team.

Name	Major	Most Relevant Skills
William Clemmons	Software Engineering	Embedded Systems, Software, Presentations.
Kennedy Reiling	Electrical Engineering	Circuitry/Hardware design, Embedded Systems, Modeling, Client Relationship.

<b>Brian Witherspoon</b>	<b>Electrical Engineering</b>	<b>Circuitry, Hardware design, Modeling, Documentation.</b>
<b>Zachary Sears</b>	<b>Computer Engineering</b>	<b>Embedded Systems, Hardware design, Programming, Documentation.</b>
<b>Mubassir Serneabat Sudipto</b>	<b>Cyber Security Engineering</b>	<b>System Security Essentials, Scripting, Debugging, Penetration Testing, Technical Documentation, Professional Presentations.</b>
<b>Ethan Haberer</b>	<b>Electrical Engineering</b>	<b>Circuit Design, Modeling, Technical Documentation, Programming.</b>

2. Strategies for encouraging and supporting contributions and ideas from all team members:
  - **a. Having democratic and non-judgmental discussions about design decisions when necessary.**
  - b. Everyone will be encouraged to speak their opinions.**
  - c. Create a brain dump channel on Discord for when someone has an idea, they can put it in there for review later.**
3. Procedures for identifying and resolving collaboration or inclusion issues (e.g., how will a team member inform the team that the team environment obstructs their opportunity or ability to contribute?)
  - **a. Consult the team to gather group input on how to proceed.**
  - b. Adjust how the team operates to include anyone who needs to be included.**

## **Goal-Setting, Planning, and Execution**

1. Team goals for this semester:
  - **a. Completion of the overall design of our project.**
  - b. Gain a working-level knowledge of the components of the project.**
2. Strategies for planning and assigning individual and teamwork:
  - **a. Have a timeline with specific deadlines and tasks that must be completed.**
  - b. Delegate work based on skill set.**
  - c. Preplanning to understand what will be required and who will be responsible.**
3. Strategies for keeping on task:
  - **Have a specific plan from the beginning of our project and continuously update it to fit the timeline.**

## **Consequences for Not Adhering to Team Contract**

1. How will you handle infractions of any of the obligations of this team contract?
  - **One single offense: bring it up to the team either in the discord or in our next meeting.**
2. What will your team do if the infractions continue?
  - **After repeated offenses, the problem will be discussed with our advisor and professor if needed.**

\*\*\*\*\*

- a) *I participated in formulating the standards, roles, and procedures as stated in this contract.*
- b) *I understand that I am obligated to abide by these terms and conditions.*
- c) *I understand that if I do not abide by these terms and conditions, I will suffer the consequences as stated in this contract.*

- |                                      |                         |
|--------------------------------------|-------------------------|
| 1) <i>William Clemmons</i>           | DATE: <u>01/30/2024</u> |
| 2) <i>Kennedey Reiling</i>           | DATE: <u>01/30/2024</u> |
| 3) <i>Brian Witherspoon</i>          | DATE: <u>01/30/2024</u> |
| 4) <i>Zachary Sears</i>              | DATE: <u>01/30/2024</u> |
| 5) <i>Mubassir Serneabat Sudipto</i> | DATE: <u>01/30/2024</u> |
| 6) <i>Ethan Haberer</i>              | DATE: <u>01/30/2024</u> |

## APPENDIX 6 GLOSSARY

Term	Definition
Arduino	Brand of microcontroller to integrate hardware with software
Database	Manages and stores permanent data across the system
Infrared Sensor (IR)	Emits light in the infrared range and determine how much light is reflected back to the receiver
Internet of Things (IoT)	The interconnection via the internet of computing devices embedded in everyday objects, enabling them to send and receive data
LiDAR	Light Detecting and Ranging sensor sends out a laser and scans its surroundings
Nano 33 IoT	Specific Microcontroller from Arduino that we are using in our project
PCB	Printed Circuit Board
Perf Board	Perforated board used for prototyping electrical components by soldering
QR Code	Quick Response Code
Request	A message sent to the get information from the database
RGB LED	Red Blue Green Light Emitting Diode
Server	Software responsible for answering requests from both hardware and the mobile application. It retrieves and modifies the database
Stripe	A third-party software that gives users the ability to implement payment systems
Ultrasonic Sensors	Distance sensors that send out a sound wave which times how long it take for the wave to go to a object and then back to the sensor and based on this time it calculates the distance
WiFi NINA W102	Chip on the Arduino Nano that allows us to connect to WiFi networks