Parkour:

A Parking Sharing System

Design Document

Team Number: 20
Client & Advisor: Dr. Ahmed Kamal

Team Members:
Jeremy Galang
Gabrielle Johnston
Jorden Lee
Ethan McGill
Jason Neville
Lorenzo Zenitsky

Team Email: sddec20-20@iastate.edu
Team Website: http://sddec20-20.sd.ece.iastate.edu/

Executive Summary

Development Standards & Practices Used

- Agile project development
- IEEE 802.11 Wireless Internet Standard
- IEEE 802.21 Cellular Data Standard
- Open Source License(s)
    - The software that we use must be allowed for commercial use, as we are making a closed source project that will be released to the public.
- Mozilla Development Network Javascript Standards

Summary of Requirements

- Our application will require a dedicated frontend and backend that supports both iOS and Android applications.
- A platform will be needed that can host our server 24/7 once the application is up and running.
- The application will need to be published on the Google Play and Apple App Store so that anyone can use our app.

Applicable Courses from Iowa State University Curriculum

- COMS 227 + 228 (Object-Oriented Programming and Data Structures)
    - Knowledge of object-oriented programming and data structures are essential and will be utilized.
- COMS 252 (Linux Operating Systems)
    - Useful for setting up the VMs and hypervisor on our server.
- COMS 309 (Software Development)
    - Knowing how to research unfamiliar technologies would be very applicable to this project.
    - Utilizing Agile development practices.
- COMS 311 (Algorithms)
    - Knowledge of algorithms would be beneficial for our application's performance.
- SE 319 (Construction of User Interfaces)
    - Teaches the basics of how to design an attractive and usable user interface through a variety of languages and tools.
- COMS 362 (Object-Oriented Analysis and Design)
    - Some project management techniques would be useful such as CRC cards and knowledge of UML. This would be helpful for planning.
- COMS 363 (Introduction to Database Management Systems)

    - Gained experience with SQL and other databases. It will be useful as we develop our database and backend system.

New Skills/Knowledge acquired that was not taught in courses

- How to build an application using React Native
- Exposure to web development for mobile applications
- How to create a backend server with API endpoints using Node.js
- Utilizing VMWare Fusion to create virtual machines to host our server
- Connect and use MongoDB as a database

Table of Contents

List of figures

List of Tables

1 Introduction

1.1 Acknowledgment

Outside of our own team members, Dr. Ahmed Kamal, the client who proposed this project, has been and will continue to be, a significant contributor to the overall lifecycle of the project.

1.2 Problem and Project Statement

The problem that our project hopes to solve is finding parking. In large events, such as football games or the Iowa State Fair, designated parking can be limited, expensive, and overall a hassle.

The Parking Sharing System aims to provide an alternative for finding parking at busy events or locations. This application will enable two kinds of users: to either find parking spaces, or host parking spaces. The host will be able to post a parking spot to the application. They will provide an address, size of space, time frame the space is available, and a price for that time. The guest will be able to search through the existing posts to find a spot close to the event they want to go through. All transactions will be processed through the application. Similar to ridesharing apps such as Uber and Lyft, this application is crowdsourced and will heavily rely on individuals willing to use the application to host parking spots, as well as individuals willing to use the application to find and reserve the listed parking spots. We plan to create an application that makes event parking easier, cheaper, and an overall better client experience. Because of the parking sharing idea, this application will be environmentally friendly, saving on fuel consumption.

1.3 Operational Environment

The backend server application will live on a publicly hosted server, so the frontend application can connect. The frontend application will be available through the Apple App Store, and the GooglePlay Store, so it will reside on both Android and iOS devices.

Physically, the application will function at its best in large cities or local events such as concerts, football games, etc. It is not limited to specific events and large cities, but it is available wherever users can host and need to find additional parking. The application will need a large user base to make it viable. Urban and suburban areas are suitable areas because there are ample people to use the application, list their properties, and to reserve the listed spots.

1.4 Requirements

The application will depend on two types of users:

- Individuals or groups who are willing to host parking spaces
- Individuals looking for parking spaces

1.4.1 Functional Requirements

- Host User:
    - Have access to a list of reservations that contains the following information:
        - Guest Information
        - Reservation Time slot(s)
        - Vehicle license plate numbers and models
    - Securely receive payment from the guests
    - Provide a listing of their property with location and descriptions
    - Manage reservations, such as having the ability to cancel or extend reservations
- Guest User:
    - Create an account that allows them to reserve parking spots
    - Securely send a payment to the host
    - Negotiate parking prices
    - Be able to navigate easily to the host's location
- Both Users:
    - Download the application from the respective mobile device app store
    - Log into their account on the application
    - Access a list of available parking spots
    - View the available listings on a map with their respective locations

1.4.2 Nonfunctional Requirements

- Intuitive and simple user interface
- Application feels responsive
- Allow users to update their public profile page easily
- Users should be able to upload images of the property
- Web browser application port
- Optional/Advanced settings
- Each payment should be documented with a corresponding log and receipt
- The application makes API requests quickly

1.5 Intended Users and Uses

The intended users include, but are not limited to individuals who are:

- Guests seeking parking spaces
- Hosts willing to rent out parking spaces on their property
- Administrators who will authenticate hosts and moderate posted parking spots

The intended use for the application is to provide additional parking spots during events such as visiting a city, going to work, attending a concert, or cheering at a football game. The guest wanting to attend the said event could easily pull up the app and find a nearby host so they could park nearby. The host would be able to make money by hosting their property at said event.

1.6 Assumptions and Limitations

1.6.1 Assumptions

- User (both hosts and guests) has the application downloaded on either an iOS or Android device
- Users might want to access the application on the web
- Strong network/internet connection
    - Users will have a working network and/or internet connection so they may properly use the features of the application.
- Strong user-base of drivers and hosts
    - There will be a plethora of drivers and hosts in areas of high traffic so that a user may get the most out of the application. This app will be most useful in areas with many people and where parking is an issue, like sporting events, concerts, etc. Individuals in more rural areas with less traffic are assumed to not have as many parking difficulties as those in cities and other highly populated areas.
- Drivers have a sustainable payment system set up with the app to pay hosts for parking on their property
- The host is legally entitled to use, lease or sublease the parking spot(s)
- Both drivers and hosts have read and agreed to the various terms and conditions of the application and all they entail

1.6.2 Limitations

- Regional support
    - The more popular parking apps tend to serve only a specific state or region of the US whereas a majority of the parking apps akin to what we are trying to build tend to try and support all users from any location. More often than not, a user will download a residential parking app only to find out that it features little to no

support in their area because they may not live in a busy and well-populated city like Los Angeles, New York City, or Boston.

- Available users in any given area
    - Some areas may have more hosts than guests and vice versa. This ratio is expected to vary, and there is not much that can be done about it. However, regardless of the ratio, our app will still require hosts and guests. After all, it is assumed that there will be a fair amount of hosts and guests available in most areas.
- User compliance
    - In a perfect world, the individuals who will use our app will do so after reading and agreeing to the terms and conditions. However, most users don't read a software product's terms and conditions, they simply accept without fully understanding what is being asked of them from the software. For example, drivers are supposed to arrive and leave at a predetermined time agreed upon by both parties. In the worst-case scenario, they leave their car for several days and the host has to call a towing service to remove the car. On the other side, the host may interact with the driver's car in an undesired fashion when the driver is away.

1.7 Expected End Product and Deliverables

The expected end product is a mobile application that is available on both Android and iOS, with the possibility of a web application. The application should have the following deliverables:

1.7.1 Frontend Deliverables

- Users will be able to sign up, log in, and interact with the application
- Hosts will be able to create and view their listing, along with information about users that are reserving their spaces and receive payments through the app
- Guests will be able to view parking spot listings, reserve a parking spot, and send payments to the host

1.7.2 Backend Deliverables

- A database that contains:
    - User credentials
    - Transaction history between users
    - Parking history
    - Locations of available parking spaces with associated fees

2 Specifications and Analysis

2.1 Proposed Approach

As stated in the problem statement, when attending events, it can be very difficult to find parking and at a reasonable rate. Thus, Parkour aims to provide a parking alternative where drivers can pay to park their ride within the residential borders of a host's living space, such as a driveway or front yard. The app will be released as both an iOS and Android application and will feature an intuitive, easy-to-use, simple UI so the users can worry more about a parking spot than a cluttered home page.

After meeting with the client, the team received the research paper on which the application is based. The only stipulation was that this project would be 100% software. A full list of requirements is available in section 1.4. Additionally, see section 3.2 for more details on the proposed technologies that will be utilized.

2.2 Design Analysis

2.2.1 Frontend

- **React Native**: The team has chosen to develop the frontend using React Native, an open-source mobile application framework. It is used to develop mobile applications for Web, Android, iOS, and Microsoft's Windows. It provides developers to access native platform capabilities using React, a JavaScript library created and maintained by Facebook.

2.2.2 Backend

- **Node w/ Express**: The team has chosen to develop the backend using Node.js because it is a very popular backend language to run on servers. There are plenty of resources for team members who are unfamiliar with Node and the Express library, a widely-used web application framework for Node.
- **Database**: The team has chosen to use MongoDB as the database because it is a popular choice among Node and Express.js developers and there exists many resources and community support. Since it is also a cross-platform document-oriented database program, it doesn't rely at all on the manipulation of relational data such as MySQL, which oftentimes presents a steep learning curve for developers.

2.2.3 Other technologies

- **VMWare Fusion**: This is the hypervisor that's used for creating virtual machines on the backend server. (See 3.2 for more details)

2.3 Development Process

Our group has chosen to follow the agile methodology as per the agile manifesto established in 2001. Routine meetings and frequent testing will be implemented for the application. This also includes frequent collaboration with our client to ensure that his needs as a client are being met and the timeline that he has set forth is being followed.

With the help of the agile methodology, we will cycle through a process of planning, executing, and evaluating that includes continuous team and client collaboration to ensure that the final version of every deliverable is as satisfactory as possible. As a team, we will most likely rely on the agile process framework of Scrum which features frequent daily stand-ups and bi-weekly sprints. Since the development of this application will not be done in the environment of a standard paying 8-hour/day software company, stand-ups will be held a minimum of once per week so everyone can get caught up and on the same page as everyone else. Sprints will be roughly around one week to two weeks depending on the difficulty and size of the task at hand so that there is ample opportunity to discuss our recent advancements with the client for any immediate feedback.

2.4 Conceptual Sketch

Below are conceptual sketches for several different screens inside of the application. Since these are conceptual these screens could change by the end of the project.

Figure 1. In app map screen

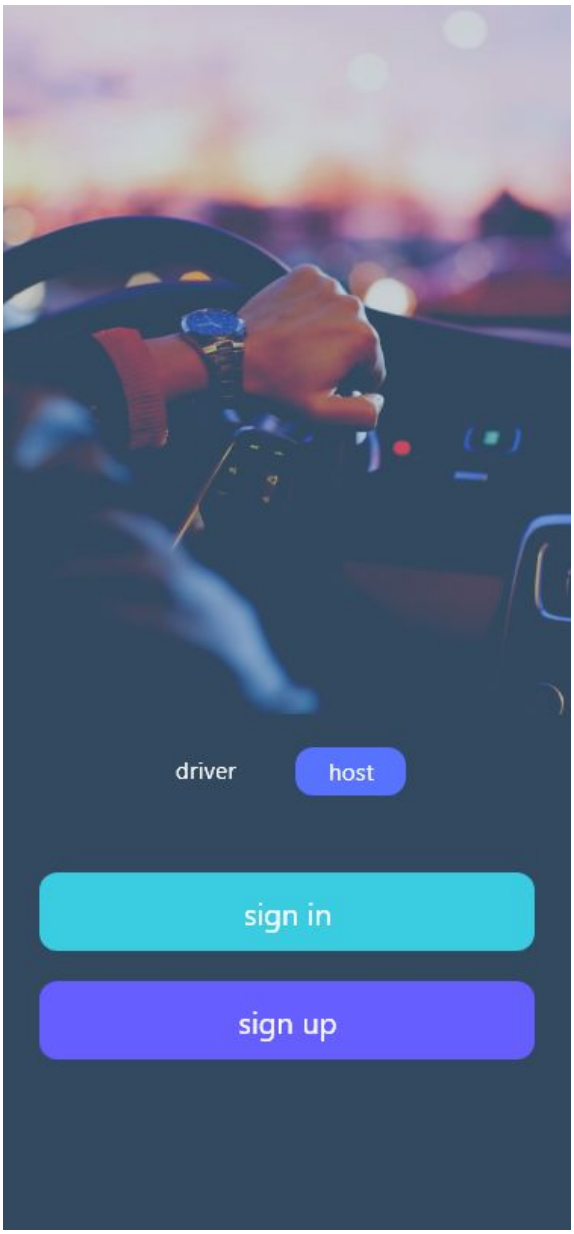

Figure 2. In app splash screen

On the left is another proposed idea for the view of the map. The sketch on the right is a toggle between driver and host along with the ability to sign in or sign up as the highlighted individual.

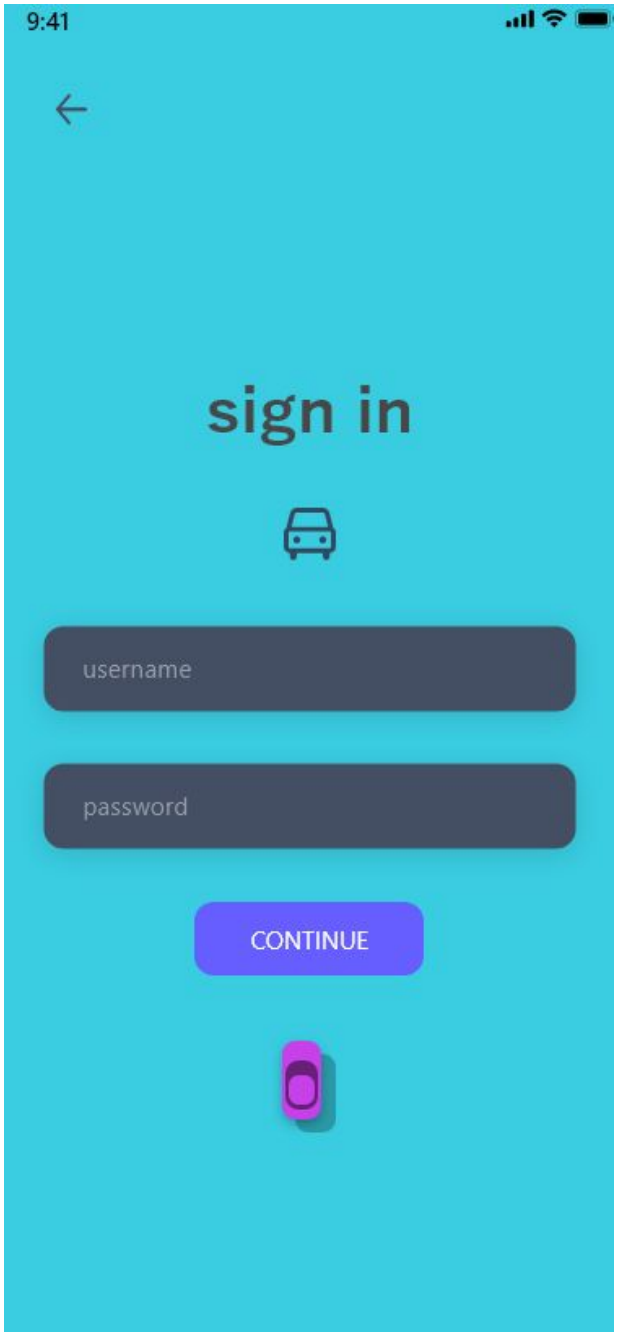

Figure 3. In app sign in page

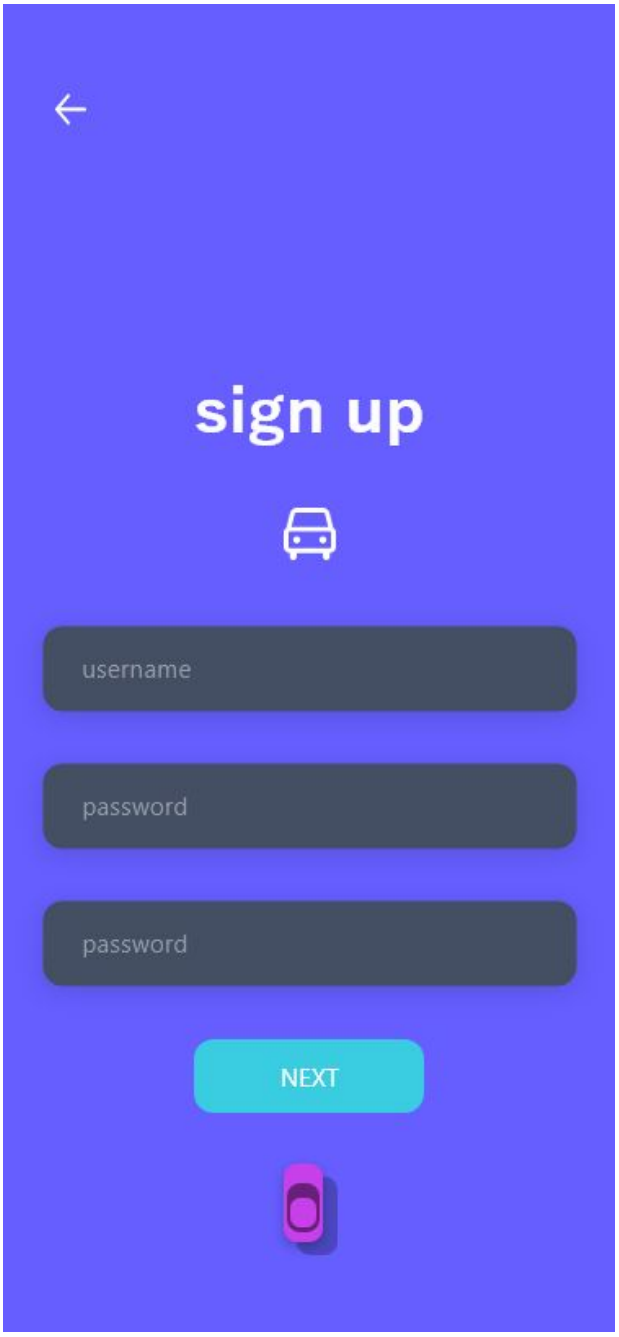

Figure 4. In app sign up page

These two sketches are the basic sign in and sign up pages.

3 Statement of Work

3.1 Previous Work and Literature

One similar iOS application in comparison to this project is Prked. This app was very similar to our idea, but it was very barebones in terms of functionality. It was posted at the very beginning of the spring semester. The application, despite frequent updates, continues to be very buggy and is unusable at this point in time.

We are basing our project on the paper “Collaborative-Aware Intelligent Parking System for Green and Smart Cities based IoT” that is written by our advisor. It has not been published yet but describes the parking system as it relates to game theory and optimization.

3.2 Technology Considerations

Since the project will be made up of two separate mobile applications (iOS and Android), our team wanted to find a technology solution that would enable us to create one application that could be exported to be used on Android and iOS. Therefore, we will be using React Native to develop the frontend. Ionic was also considered but React was ultimately chosen because of the better performance, its popularity in corporate environments, and community-supported codebase which features plenty of development resources and third-party libraries/frameworks.

The front end application will be connected to a single backend hosted on a web server. The team has not committed to one language, but plans on using some variant of Typescript that is good for a large number of concurrent users. The backend will also be connected to a database to securely store information needed for our application. Like the backend technology, the team has not committed to a single database language, although technologies like MySQL, MongoDB, Postgresql are all being considered.

VMWare Fusion will be used as the hypervisor for creating and managing our virtual machines. These virtual machines will be configured to automatically host our servers on startup. Once these VMs are configured, we will export them, deploy them on a hypervisor and ensure they are communicating with our application correctly. We will modify and maintain these VMs throughout the course of our project.

3.3 Task Decomposition

Our project is divided into a mobile (frontend) application, paired with a backend server that we will make requests to get user information and other stored data that might need to be stored in

a database. Since we will have two separate projects going, the team will split up and simultaneously develop the front and backend.

3.3.1 Frontend Tasks

- Initialize the skeleton React-Native Project
- Sign up page
    - Establish a connection to the backend server
    - Make post request to the server to create user
    - Receive a response from the server
    - Create the user interface
- Login page
    - Make a post request to the backend server
    - Receive a response from the server
    - Create the user interface
- Home Page
    - Create the user interface for the screen the users first see after logging in
- Navigation Bar/Menu
    - Create the component that allows users to navigate the application quickly
    - Create placeholder links for all of the future pages that will be added to the application
- Profile Page
    - Make a get request to the server to retrieve the user’s information
    - Receive a response from the server with the user info
    - Create the user interface for the profile page
    - Fill out the user page with the information
    - Add an edit profile link in the profile pageAdd a profile picture and the user's name on the top of the navigation bar/menu?
- Payment Page
    - Connect the application to the chosen payment APIs
    - Create the user interface
    - Send post to the server to log the payment occurred
- Post Parking Space
    - Send post request to the server to store the new parking space
    - Receive a response from the server
    - Create the user interface to post a parking space
- List Available Parking Spaces
    - Send a get request to the server
    - Receive a response from the server
    - Create the user interface for the page
    - Display the spaces retrieved from the database
- Reserve Parking Space

    - Make a post request to the server to modify the number of spaces on the backend
    - Receive a response from the server
    - Add a payment element to pay immediately
    - Create a user interface to reserve a parking space
- Show the Parking Spaces on a Map
    - Make a get request to the server to get available spots in the loaded area
    - Use a map API (Google/Apple/Maps) to show the map
    - Plot the locations on the map
    - Provide directions to the selected parking location
- Create optional settings

3.3.2 Backend Tasks

- Initialize a skeleton Node.js project
- Setup a database to be used with the project
- Connect the backend server to the database
- Create a user model to be used in the database
    - Include the obvious items such as username, email, password, first name, password, phone number, jwt token, isVerified
    - Also include profile information
- Create a User Signup API endpoint in the backend
    - Create a POST new user API Route that will get triggered when a user signs up on the frontend
    - Create JWT Token and send them to the frontend to store
    - Encrypt user information
- Create a User Sign in API endpoint in the backed
    - Create the needed GET, POST, PUT, DELETE requests
    - Decrypt and verify user login
    - Send back if the user successfully logged in
- Create a Receipt API
    - Create a model to store Receipt objects
    - Create the needed GET, POST, PUT, DELETE requests
    - Store the receipts in the database, link the receipts to users through their ids
- Create a model for the spot locations
    - Include address, and maybe coordinates
- Create a model for a listing
    - Include user that posted the listing
    - Include pricing and stipulation information
    - Include the location object
- Create the Parking API
    - Create the needed GET, POST, PUT, DELETE requests
- Create additional query options

3.3.3 Administrative/Misc Tasks

- 491
    - Biweekly and weekly reports (as requested by the advisor/client)
    - Publish Design Document Version 1
    - Publish Design Document Version 2
    - Publish Design Document Version 3
- 492
    - Biweekly and weekly reports
    - Prepare project poster
    - Present the project

3.4 Possible Risks and Risk Management

There are many risks associated with this application. An early risk was how to securely handle payment information. We mitigated by deciding to use a third party API to set up payment through a well-established company. Users will be given the option of using Google Pay, Apple Pay, PayPal, Venmo, or Amazon Pay. This helps our consumers to use a familiar payment method and it helps our team by not having to worry about huge security risks.

The second big risk is the liability that is associated with involving vehicles. There will have to be very strict language in our terms and agreements for users of the application to absolve us from any fault if there is an accident or theft of service. Currently, instructions for owners are being devised if, for whatever reason, a vehicle is left at their parking spot longer than the agreed-upon time.

The next risk that our team has thought through is how to verify users. This would be to keep all users accountable and safe while using our product. The plan is to ask for personal information such as a driver's license number for the owners. We will also ask for the license plate number and other identifying information about the car from the renters, so an owner can verify that the correct vehicle is parked in the agreed-upon spot.

The last and most recent risk associated with this project is COVID-19. There is a lot of uncertainty about how we will be able to communicate with the team as well as work on documentation, development, and testing online. Currently, we all have access to the internet for communication, but we will need to continue to work together to overcome any and all issues that crop up during this project. Our plan was to test our prototype in the fall, but if the situation persists then we may not be able to use our app alongside some events such as football games.

3.5 Project Proposed Milestones and Evaluation Criteria

Here is a list of milestones for our project:

Proposed Milestone	Checkpoint Date(s)	Evaluation Criteria
Establish team roles and expectations	January 16th	Team has a set of clear expectations and roles.
Requirements Gathering	February 16th	Full understanding of what our application may need or potentially need is fully understood. Minimize the need to spend extra time researching further technologies.
Present Project Plan via presentation and design document	March 8th	Powerpoint slide that entails all required details of our project. Design document has a start on how everything will be planned out.
Transition from research to learning respective technologies	April 2nd	Everybody has gone through some tutorials regarding the project and did further research if needed. Having basic familiarity with the technology/language they intend to use for development.
Finish User API development	August-September	User has a functioning login system and attributes are logged in the database.
Delegate multiple tasks and begin implementing features	September 15th	User API development is fixed, now can begin implementing everything else.
Work on implementing Maps API	October 9th	Maps are integrated into the home screen, and users can see locations on the map. Users can publicly post their location on a global map.
Property Posting and verification APIs implemented	November 15th	Properties are posted and APIs can verify whether they are legitimate properties or not.
Payment APIs working	November 30th	Hosts and guests should be able to securely transfer payments between each other and accurately keep track of how much is being paid.
Project Debugging	December 9th	Application will go through automated

		and manual testing. Application should not have any bugs or issues, and be dealt with during this phase.
Prototype	December 10th	Working prototype of the application and all of existing features work as expected. Host should be able to post their properties and guests should be allowed to make a reservation to the host.
Final Presentation	TBA	Application will be demoed and ready for presenting. Other presentation visuals such as a powerpoint will be available.

Table 1: Proposed milestones

We may change or modify some of the dates or the order on how these tasks will be completed as we see fit within the scope of our project.

3.6 Project Tracking Procedures

Our team will work in an agile environment to track our procedures. We meet weekly to discuss what goals have been met and how the team is keeping up with the timetable that we have built to finish all tasks in time. We divide work up into manageable weekly sprints to ensure high morale and effective communication. Our team will be using GitLab tracking to manage our progress throughout this project. We will be working with GitLab over other project tracking systems because all team members are already familiar with how to use it.

3.7 Expected Results and Validation

Our application must be able to:

- Successfully allow a guest to download the app to their mobile phone.
- Allow said guests to either host or find nearby parking spaces via the app.
- Provide a reservation to the host saying the guest wishes to park at their location
- Allow the host to verify the guest and their identity and set certain guidelines for said guest.
- Securely exchange and validate payment between host and guest
- (More to be added as we see fit)

4 Project Timeline, Estimated Resources, and Challenges

4.1 Project Timeline

First, our team will do research on what technologies and options that are available to us. We had the choice of either doing an API or building a new app from the ground up.

First Stage (January 2020 - March 2020)

- Familiarize team members with one another
- Work on Design Document
- Gather requirements
- Research technologies for frontend/backend

Second Stage (August 2020 - October 2020)

- Work on implementing application
- Create APIs on backend for app to interface to
- Create iOS and Android applications
- Connect MongoDB DBMS with backend server
- Establish a working connection between server and apps
- Implement all important features for applications

Third Stage (October 2020 - End of Project)

- Fix bugs and refactor code design
- Create a working prototype
- Recruit users for testing purposes
- Test application for football games in the fall
- Implement security features for server and applications
- Implement other miscellaneous features in applications
- Continue to improve reliability and efficiency of applications

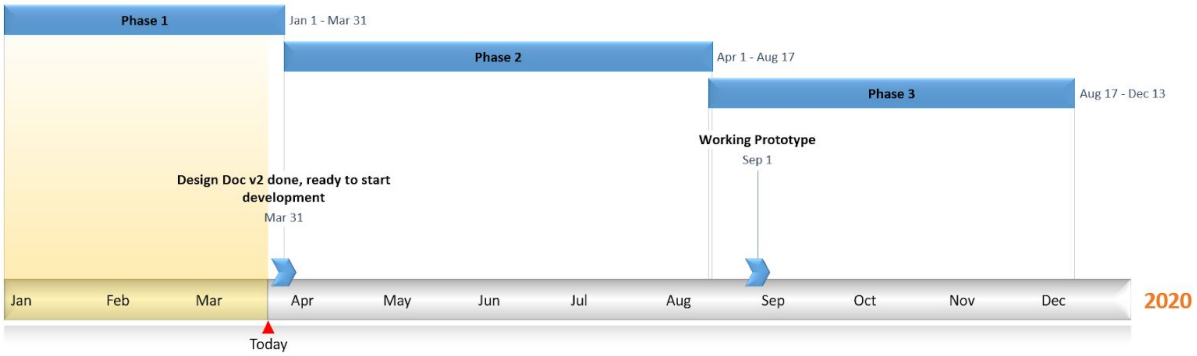

Figure 5: Gantt chart for schedule

4.2 Feasibility Assessment

By December 2020, this project should be a full stack mobile application anyone can download and use off of the Apple or GooglePlay store. The applications frontend should be refined, easy to use, and professional. The backend should process API calls quickly, and securely.

Realistically the team believes that all of the functional goals are attainable by the above date. The team expects there could be issues with verifying homeowner's identities and proving they own the property they are listing, but large scale applications like Airbnb do not do this type of check so the team will find a solution as more important function requirements are finished.

4.3 Personnel Effort Requirements

Task	**Description**	**Time (hours)**
Research frontend and backend technologies	The frontend technology would need to allow for a fast, high performing UI for a mobile application. Our backend will need to store user and application data.	20+
Research payment options	Payment processing and handling options for users and	10
Come up with solutions on handling user interactions	Handling interactions that might pose as a liability, such as parking for longer durations than allocated.	30
Learn respective frontend/backend's syntax,	This will depend on individual experience with the chosen	20

framework or language.	languages/frameworks we are working with. Most of us may not be familiar with these languages, so we need to familiarize ourselves with it first.	
Construct frontend UI	Develop an easy to use UI for the mobile app.	70
Construct backend server	Develop a backend server that would communicate with the frontend and store user/app data.	50
Frontend + Backend testing	Ensure that frontend and backend are both communicating properly	20
Frontend and backend debugging	Check for bugs within the application.	40
Creating Virtual Machines and configuring them with our backend.	Setting up a virtual machine to host our server. Configuring and ensuring the backend server works within the VM with our frontend.	30
Deploying the VMs to a hypervisor or managing VMs	The virtual machines will need to be deployed to a hypervisor.	5
Configuring the server(s) to work with the virtual machines	Once we finish finding the servers and setting up the virtual machines.	10+
Beta testing	Testing of the application in our local environment, ensuring the behavior works.	30
Deploying application to the Google Play/App Store	Publishing the app could take time, depending on how each store processes their apps.	7-14 days
Live testing	Once all the servers and applications are deployed, we can test it with a live group of users if possible.	20

Table 2: Personnel Effort Requirements

4.4 Other Resource Requirements

No additional resources.

4.5 Financial Requirements

A few financial requirements include:

- Server costs
- Cost of publishing our app to the Google Play and App Store
- Verification API costs

From an economic standpoint, the project will have to include the functionality to carry out monetary transactions for the users of the two mobile applications. To facilitate these transactions, all of which cost developers a small fee, integrations with popular payment options like PayPal will be needed. Also, the integration of software like Google Maps will eventually cost us a small fee to use because more often than not, companies charge based on the number of calls to their APIs. Second, the cost of publishing our app to the Google Play Store for Android devices includes a one-time fee of $25. On the other hand, Apple charges a yearly fee of $299 for a group of developers to publish their app to their App Store. Lastly, we will be using servers for hosting purposes. For now, we are only concerned with hosting our product in Iowa, so our server costs will be next to nothing. However, if we ever choose to expand our scope to accompany other regions of the country and/or the world, we will be subjected to much higher server costs.

5 Testing and Implementation

5.1 Interface Specifications

Since our project is software alone, we will have an iOS and Android application with a backend server. We will have no hardware aspect to this project. Because of this we do not have to worry about hardware interfacing.

5.2 Hardware and software

Our project is only software based, because of this there are no required hardware specifications. The team will be testing the frontend on xcode and android studio, and the backend will be tested using visual studio code with simple test cases.

5.3 Functional Testing

5.3.1 Frontend

We will be using Espresso for functional UI testing because it can mock user keystrokes which are very useful for obtaining information. We will use Jenkins automated testing to automatically test functions that we know to be working every time a merge is made. This ensures that code that is being merged to production is not breaking other code. We will be using Jest for additional testing to carry out similar tests that we would be doing using Mockito if we were using Java. Because we are not using Java, we will use Jest for React framework testing.

5.3.2 Backend

On the backend we will be using four different javascript libraries that were made for testing server side applications. We will be using Mocha, Chai, Chai-http and Sinon. We will be using Mocha and Chai/Chai-http for the backend unit testing because Mocha is a very flexible framework that works well with other assertion test frameworks like Chai. The use of Mocha coupled with Chai allows us to write JUnit like tests for server side javascript code. For integration testing we will be using Mocha with the addition of Sinon. Sinon is a popular testing javascript framework for mocking objects, or methods. We can use Sinon to mock API calls to automate testing involving communication between the frontend and the backend.

5.4 Non-Functional Testing

5.4.1 Frontend

On the frontend, we will be using simple, manual user testing to test accessibility, how user-friendly the app is, and it's response time. We will use Espresso to test simple keystrokes a user might execute. We use Espresso because it is much faster than a manual test.

5.4.2 Backend

The backend will again use Mocha, Chai, Chai-http, and Sinon to test the nonfunctional requirements. The main requirement the backend will be testing is the speed in which the backend communicates with the frontend, and how long the backend takes to communicate with the database. To test the speed of communication between the server and the frontend application, as well as the speed of querying the database, we will set up automated tests similar to the integration tests in section 5.3.2. In each of these tests we will simply start a timer at the beginning of each test, then mock an api call, or query the database. After the api call, or query is complete the test will output the total time taken for the test, which we can analyse and optimize if needed.

5.5 Process

Below is a simple flowchart that demonstrates our process on our testing plan.

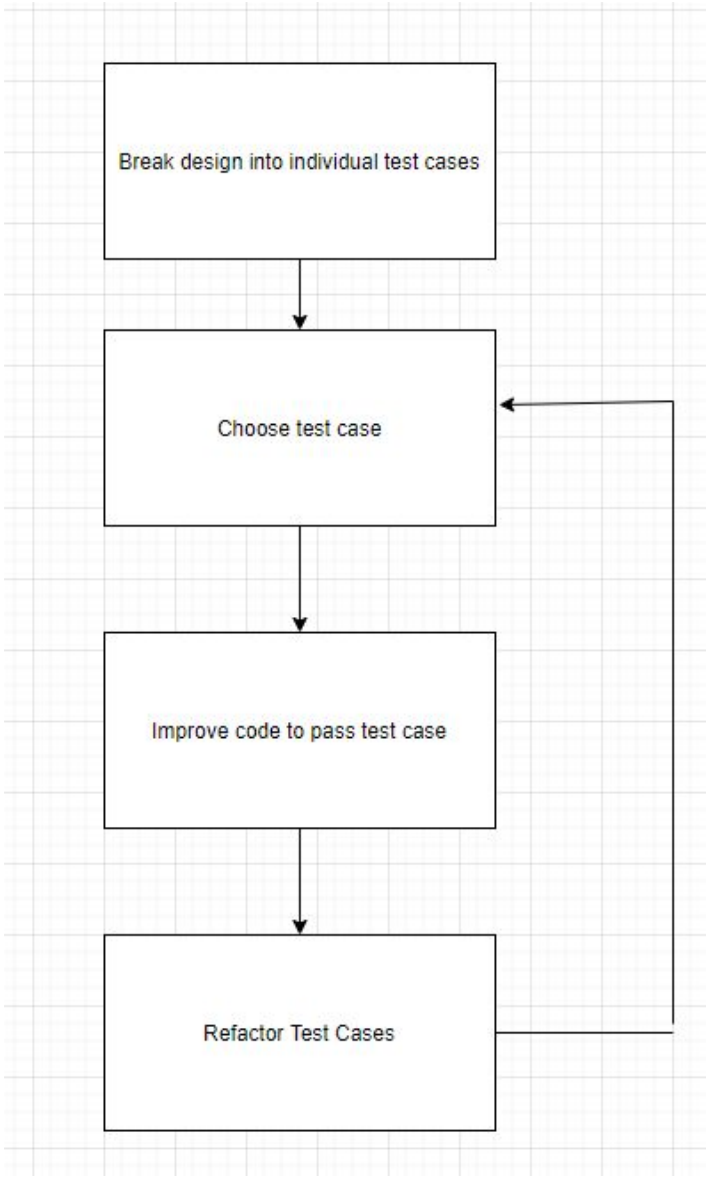

Figure 6: Testing plan

Our testing plan follows the Test Driven Development software development process. In this process, requirements for the project are broken down into different individual test cases. This is so that everything that is created so far can be measurable. After this, code is redesigned for each test case so that the test case passes. Once the developer does this for a number of requirements, the tests are refactored and the process is completed again. This is so the tests get more detailed and the code gets much more secure.

5.6 Results

Since we're using React Native, we expect to have a high performing application that performs on Android and iOS. The UI navigation should map to the proper pages and navigate to the correct locations. The backend needs to properly store user data securely and efficiently within a time period. Since we are using an API to deal with transactions and payments, we don't expect to store sensitive data (i.e. credit card info) on our databases.

At a user level, the application should have a simple functioning login system where they can register as a host or guest. Both users need to be able to switch between roles and have different levels of permissions based on their roles. Some the results we should see include:

The host should be able to:

- List their property and location on the map
- Manage and see the guests parked at their property
- See guest information, such as plate numbers, car models, names, etc
- Securely receive payment from guest users

The guest should be able to:

- View a list of **verified** hosts on a map
- Create a reservation for their parking spot
- Securely provide payment to the host

6 Closing Material

6.1 Conclusion

Our project aims to reduce the stress of parking at big events such as fairs and concerts by providing an alternative to both popular commercial parking apps by giving users the ability to park on a host's residential property instead of hectic chaos of more populous areas such as cities. To accomplish this goal, we are planning on developing a React Native mobile application and deploy it to both the Google Play Store for Android and Apple's App Store for iOS. For next semester, we plan on developing new functional versions of the project through one-to-two week sprints adhering to the Agile methodology framework of Scrum. Throughout these one-to-two week sprints, we will keep up-to-date with one another through weekly online stand-up meetings to ensure that every team member is on the same page with each other and confident on the tasks that they must accomplish in any given sprint iteration. The project will prove to be a success only through rigorous and ongoing integration between testing and development to ensure a properly working deliverable after each sprint. Our ultimate end goal is to develop an application that will provide to its users a different approach to modern parking apps that will not only help reduce stress and save both money and time, but will also help strengthen the community bond between users.

6.2 References

[1] K. Beck, M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R. C. Martin, S. Mellor, K. Schwaber, J. Sutherland, and D. Thomas, "Manifesto for Agile Software Development," *Manifesto for Agile Software Development*, 2001. [Online]. Available: https://agilemanifesto.org/. [Accessed: 21-Apr-2020].

[2] "Collaborative-Aware Intelligent Parking System for Green and Smart Cities based IoT". Adel Mounir Said, Ahmed E. Kamal, and Hossam Afifi [Accessed: 31-Jan-2020]

[3] Jasra, "MERN Stack", *GeeksforGeeks*. [Online]. Available: https://www.geeksforgeeks.org/mern-stack/. [Accessed Apr. 2020]

[4] MDN contributors, "JavaScript guidelines," *MDN Web Docs*, 08-Nov-2019. [Online]. Available: https://developer.mozilla.org/en-US/docs/MDN/Contribute/Guidelines/Code_guidelines/JavaScript. [Accessed: 24-Apr-2020].

[5] “802.11-2016 - IEEE Standard for Information technology--Telecommunications and information exchange between systems Local and metropolitan area networks--Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications,” *IEEE*. [Online]. Available: https://standards.ieee.org/standard/802_11-2016.html. [Accessed: 24-Apr-2020].

[6] “802.21-2017 - IEEE Standard for Local and metropolitan area networks--Part 21: Media Independent Services Framework,” *IEEE*. [Online]. Available: https://standards.ieee.org/standard/802_21-2017.html. [Accessed: 24-Apr-2020].

6.3 Appendices

<To be added and updated when necessary>