



University of Arkansas – CSCE Department

Capstone II – Final Project – Fall 2020

### Study Spots

Alexander Davidson, Rashi Ghosh, Oliver Hubbard, Hunter Lowery

#### Abstract

Finding a place to study is important and often difficult for college students. With Study Spots, students can now find an appropriate place to complete their tasks without having to waste time looking for a location that is not overly crowded and that meets their specific needs. This application integrates the Google Places API and utilize Firebase to retrieve location data. The frontend is displayed to users using React Native, and Redux ensures all data is synced between the frontend and backend. Each group member has contributed their skillset towards the delivery of the final product.

#### 1.0 Problem

Living in a college town has many perks, from the amazing football games to the diverse campus events. However, one downside is the inability to find a consistently available location for studying, whether it be for finals or a class project. A lot of students are unable to focus on their tasks in their own apartments or dorms and seek other places to complete their assignments or prepare for exams. There are several times during each semester when the on-campus libraries and study rooms, and local coffee shops fill up to capacity. These weeks often correspond to midterms and finals. Unfortunately, it is during these crucial periods that the study areas are needed the most. College students often drive around town looking for a place to study, only to discover that the areas of choice, mainly coffee shops, are overcrowded and too loud. This leaves students with limited options for completing their work efficiently.

#### 2.0 Objective

The objective of this project was to create an application that can aid students in finding a place to study, based on how crowded the location is at certain times. This has addressed some of the shortcomings of existing venue locating services offered by Yelp and Google, and may help in optimizing student performance while promoting local businesses.

#### 3.0 Background

##### 3.1 Key Concepts

The application used the Google Places API to access information on locations and Google Firebase to store that information and provide it to a user.

Google Places is an API developed by Google that allows clients to access the data used by Google Maps relating to locations that Maps has information for. The API is limited in requests featuring only 5 unique request: Place Search, Place Details, Place Photos, Place Autocomplete, Query Autocomplete. Through these calls all information on a location can be accessed and digested by a client to display relevant to what a user is currently searching. The request is made using HTTP and requires an API key for verification. Google returns the data to the client in a JSON format where the client can dissect the information and distribute it accordingly. The API does have a cost to use which is charged per 1000 calls with varying pricing, but Google doesn't charge below a certain threshold of calls for development purposes so this wasn't an issue we had.

Google Firebase is a service platform provided by Google that works as a cloud-based NoSQL server database. It is able to be integrated with a variety of languages and uses the REST API to manipulate data on the server. The data is stored in JSON format on the server and persists even in an offline environment so changes will be updated once the database is online again. Firebase handles all user and location data that we store for the operation of the app. It also comes with a pre-built user creation feature that we utilized to save all user information. This pre-built feature cut down on development time and simplified the operation of user status/creation.

### **3.2 Related Work**

A YouTube channel called Framework Television has created a video series where they code a coffee shop locator app using the Google Places API. This coffee shop locator lacks depth however and focuses on only displaying coffee shop, no other information is provided. We intend to expand upon this and provide a user with various rating and reviews of a location to give a better understanding of that location.[6]

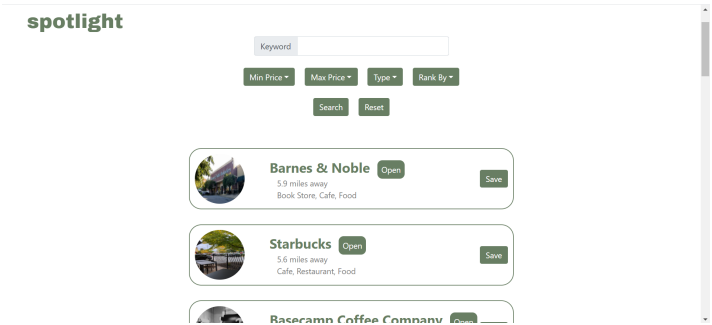
Yelp is a well-known review and search website that at the core functionality accomplishes what we intend to do; locate coffee shops, but it lacks an in-depth rating system for niche criteria relating to what a student would desire from a location. [7]

Google Maps needs no explanation, we will be using it for the base of our application but develop a review system on top of that to provide users with information we consider important to selecting a study location that a user prefers as well as simplifying data that Maps provides and providing it in a more accessible and user friendly format. [8]

### **4.0 User Experience**

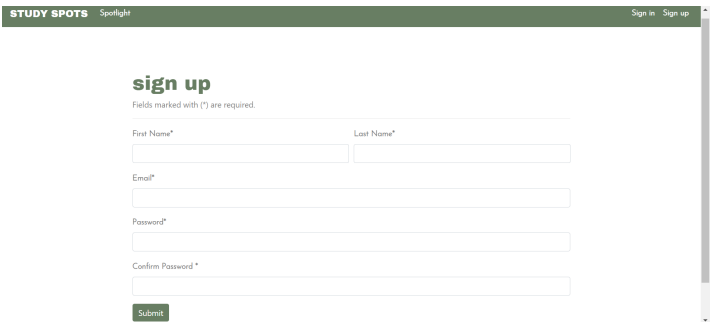
There are several pages to the application, each with different features. When the application is first loaded, the user is on the Spotlight page, which lists nearby locations based on the user's zip code. The Spotlight page contains a filter with several attributes that the user can

select to refine their search. These attributes include price, type and rank. The places are each listed with a picture, title, relative distance, and whether they are open or closed. There is also a save button. The save button is only useable if the user is logged in to their account. Users may also click on a place and will be redirected to that spot's individual page.



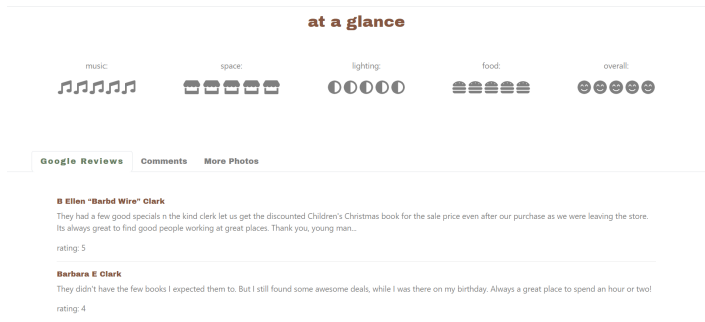
The screenshot shows a web interface titled "spotlight". At the top, there is a search bar labeled "Keyword". Below it are four filter buttons: "Min Price", "Max Price", "Type", and "Rank By". There are also "Search" and "Reset" buttons. Below the filters, there is a list of search results. The first result is "Barnes & Noble", which is 5.9 miles away and is a "Book Store, Cafe, Food". It has a small image of the store and a "Save" button. The second result is "Starbucks", which is 5.6 miles away and is a "Cafe, Restaurant, Food". It also has a small image and a "Save" button. The third result is "Raceramp Coffee Company", which is 5.4 miles away and is a "Cafe, Restaurant, Food". It has a small image and a "Save" button.

The Sign In and Sign Up pages are located at the top right of the navigation bar. The Sign Up page contains a list of fields, including name, email, and password. Once an account is created, the user is directed to their My Spots page, which now appears in the navigation bar. The user can now use the save button on the Spotlight page to save their preferred spots to their My Spots page. The Sign In page is as simple as it sounds. A user can log in using their email and password and will be redirected upon a successful login.



The screenshot shows a web interface titled "sign up". At the top, there is a navigation bar with "STUDY SPOTS", "Spotlight", "Sign in", and "Sign up". Below the navigation bar, there is a form with the following fields: "First Name\*", "Last Name\*", "Email\*", "Password\*", and "Confirm Password \*". There is a "Submit" button at the bottom. A note above the form states "Fields marked with (\*) are required."





## 5.0 Design

### 5.1 Requirements, Use Cases and Design Goals

There were several requirements and use cases to be satisfied in order to achieve the goals of the app. The following were implemented:

1. *Application should be cross-platform.*
2. *Users should be able to create accounts.*
3. *A guest view should be supported.*
4. *Only coffee shops/libraries/relevant locations should be listed.*
5. *Users should be able to sort the list of study spots based on a variety of attributes.*
6. *Users should be able to filter study spots.*
7. *Users should be able to read and write comments or reviews about a study spot.*

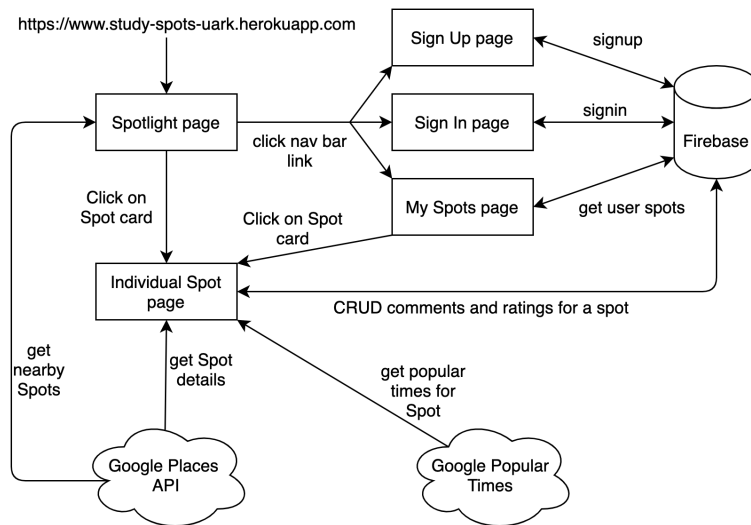
An additional requirement was added during development for the Spotlight page:

8. *Button to quickly save spot to a signed in user's My Spots*

### 5.2 High Level Architecture

First, a description of the general structure/layout of the app will be discussed. The following diagram provides an overview user workflows in the context of of pages (rectangles), database functionality, and other cloud services:

**Commented [RG1]:** include only title & whether it was implemented, add additional requirements from trello



The app defaults to opening in the “Spotlight” (a list of nearby study spots retrieved from Google Places API that can be filtered and sorted). From here, the user can log in/sign up. Then, they are in user view, and have access to a tab called “My Spots” (which is their saved/favorited study spots). Both “Spotlight” and “My Spots” have the Popular Times/Google Places API integrated in order to pull data about the busiest spots and their locations, hours, etc. When a particular spot is clicked, the user is redirected to a page dedicated to that spot, which displays information on location, open/busy hours, Google user reviews, Study Spots user comments, and photos.

Now that a general idea/structure is in place, the implementation details/decisions will be discussed (regarding how the frontend and backend are implemented). There are three major software frameworks/services for this project. The frontend application (hosted by Heroku) will be developed using React, Redux [5] (with Redux acting as a hybrid backend), and Google Firebase [1], using JavaScript. In addition, the React app is served to the user from an ExpressJS server, also hosted on Heroku.

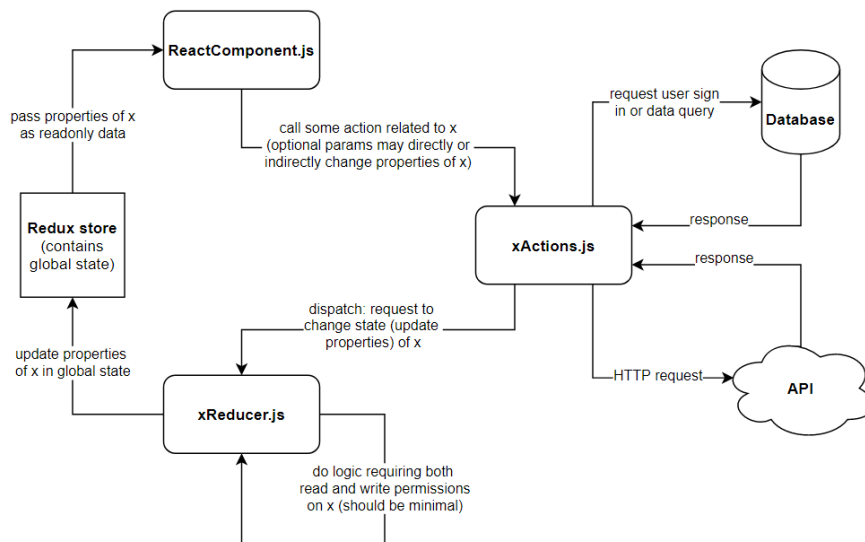
Because this app is essentially full-stack (with a UI and front end, as well as a server and database backend in order to store user and study spots data), React, Redux, and Google Firebase were chosen for the implementation/software architecture. These frameworks work incredibly well together, and can be easily integrated in order to make the development process smoother and more unified. React is a JavaScript framework for building web UIs using reusable, modular components. Redux is a state container that lies in between the frontend and a true backend, ensuring the entire app is synced and there are no inconsistencies. Since the scope of this project was sufficiently small, Redux is used as a pseudo-backend, since Firebase as a database server

and Google Places API can handle the simple requests for data used on the front end. Google Firebase was chosen due to its easy authentication and simple, document-based database service, Firestore.

The following diagram was used to design the React/Redux interface.

### Passing/Updating Data with Redux

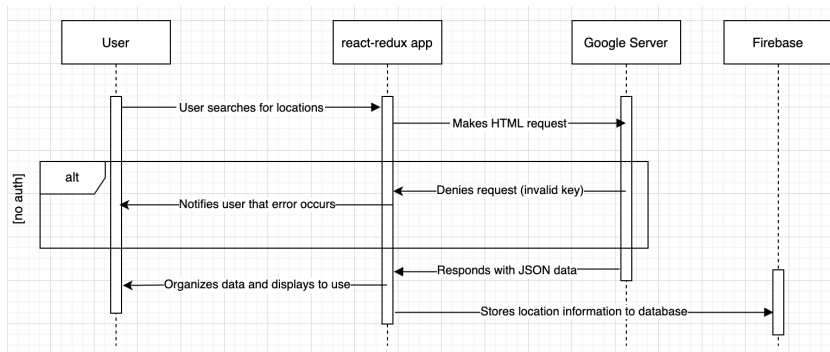
Example: a React component that signs a user into their account. x could represent a user's account with properties such as username, password, email



React components display data to the end user. React is a good choice for the front-end framework due to its dynamic nature. It uses local component state in order to dynamically update what the user sees as they interact with different aspects of the page.

In order to ensure states are synced across all pages, for all users, for all guests, etc, a state container, Redux, is used as a central store for a single instance or session of the application. Data retrieval from various cloud services can be accessed by React components using Redux "actions". An action is a function that gets a specific data object related to some entity the user is interacting with on the front end. These data objects include user data, Spot details (from Google Places API), Spot comments/ratings (from Firestore), etc. The actions then dispatch the data (or error messages) to Redux "reducers", which directly update the central application state.

Altering data requires a few more steps in order to ensure that not only are the components synced with each other, but that the front end is synced up with the database. Sequence flow diagrams such as the following were used to aid in implementing this type of functionality.



A React component will respond to user inputs (for example, a new user registration). Then, the component will dispatch an Action which creates/updates/deletes data instead of retrieving data. Actions will make asynchronous calls to Firestore to modify data in the database, and then allow the alteration to proceed through to the Reducers in the Redux Store, just as it does with data retrieval. The Reducers then alter the overall state of the app, and notify all relevant React components of the state change for dynamic updates. In this way, when a component changes something in the backend, it is ensured that first the database is updated, and then the Redux Store updates the state of the entire app based on the updated database.

As for the API calls, the Popular Times [3]/Google Places API [2] are integrated inside the individual app components where needed.

### 5.3 Risks

Risk	Risk Reduction	Results
User account privacy/security	Google Firebase has security rules that can be implemented in order to protect user's data.	Google Firebase was sufficient
Places API costs	Minimize calls to API to reduce total cost by storing data from previous calls for a timespan where data shouldn't vary for a location.	The application is small, with few users, so costs are minimal



Using a public “CORS anywhere” server as a proxy for API requests that refuse same-origin responses	Eliminated the need for the proxy server by integrating the same functionality in the Heroku server that returns the React app to the user’s browser.	System is faster and more reliable with dedicated proxy server.
---	---	---

**5.4 Major Tasks** – Below is an overview of the original major tasks for this project and how they were completed during development

*1. Research Background and Propose Solution*

The team researched the setting (real or virtual) in which students encounter the problem as well as any existing solutions. Once the problem was defined, the team proposed an efficient, accessible solution in the form of a mobile or web application. The team defined high level requirements and software architecture in this phase (see section 4.1-4.3). This phase completed with no delays.

*2. Design Architecture*

The team decided how the chosen systems and frameworks were to be used to implement the architecture of the solution. The frontend application was written using the React/Redux framework in Javascript, and Google Firebase was used for data management and user authentication. The team originally planned to define a rigid database schema to use with Firebase but decided this was not necessary, due to the flexible nature of Firebase and the non-complex data objects used by the application.

*3. Implement Architecture*

With the Scrum lifecycle methodology, the team wrote React components and Redux actions to implement the models designed in phase 2. This phase consisted of splitting up and assigning tasks on a Kan-Ban style development board on Trello. Using GitHub for source control, the team worked asynchronously on jobs and met on Microsoft Teams video chat twice a week to stay organized and on-schedule. This phase was originally supposed to span over 4 sprints of 2 weeks each, but during development the first sprints were shorter, and the final sprints were longer. Application development did run over schedule by about 2 weeks, but this was not a debilitating roadblock since some cushioning was built into the schedule

*4. Testing*

While working on jobs from the board, team members performed unit testing in their local environments before pushing code changes to development branches. All code was reviewed by at least one team member before merging to the development branch, and additional tests were performed on the development branch before merging to the master project branch. The team originally planned to dedicate at least 2 weeks for testing after all features had been implemented, but we had to cut this short due to time constraints.

## 5. Documentation

This phase consisted of compiling all documentation (test reports, use cases, architecture diagrams, etc.) and writing the final report.

**5.5 Schedule** – Project development occurred mainly in the fall 2020 semester over 15 weeks. Task 1, the research phase, was completed during the spring 2020 semester in the form of a quad chart and this proposal. Below is a breakdown of how the schedule changed during the project.

Task Breakdown	Original Time Constraint
1. <i>Research Background and Propose Solution</i> Completed on schedule during the spring 2020 semester.	3/9 - 5/1 (2 weeks)
2. <i>Design Architecture</i> Completed on schedule during the first weeks of the fall 2020 semester.	9/7 - 9/13 (1 week)
3. <i>Implement Architecture</i>	
3.1. <i>Sprint 1 - Application Foundation</i> Completed ahead of schedule by 1 week.	9/14 - 9/27 (2 weeks)
3.2. <i>Sprint 2 - Home, Login, Account</i> The team decided to move development of the home page to sprint 3, and we decided to combine the home and Spotlight pages into one. Because of this, they were able to complete the sprint about a week ahead of schedule.	9/28 - 10/11 (2 weeks)
3.3. <i>Sprint 3 - "Spotlight" Page</i> This sprint was the longest and caused the most disturbance to the original schedule. Many challenges arose during this sprint due to the growing complexity of the project. The team members also spent much time learning to use new libraries and APIs during this sprint which they did not foresee during project planning. Most of the work for this sprint was completed over 4 weeks with some small tasks spilling over into sprint 4	10/12 - 10/25 (2 weeks)
3.4. <i>Sprint 4 - "Spot" Pages, Comments</i>	10/26 - 11/8 (2 weeks)

This sprint also lasted longer than expected, with development spilling into the beginning of December. The team focused on implementing the final features and polishing the look and feel of the application during this sprint.

#### 4. *Testing* 11/9 - 11/15 (2 weeks)

Since phase 3 pushed back the schedule by several weeks, most of the testing planned originally had to be cut from the schedule. As a result, the testing documents supplied are comprised of basic workflows which a user might act on while using the application.

*(Schedule cushion week)* 11/16 - 11/22 (1 week)

*(Thanksgiving break, no items scheduled)* 11/23 - 11/29 (1 week)

#### 5. *Documentation* 11/30 - 12/13 (2 weeks)

Due to schedule delays, the team spent about a week writing documentation.

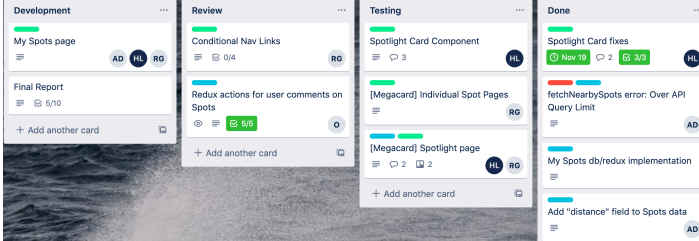
## 6.0 Testing

Testing was carried out primarily in two separate ways: testing during development by the coder/s, and unit testing by a separate member of the group to verify proper functionality. Testing would also un-intentionally occur during the creation and connection of front-end and back-end elements where for example data handling issues occurred.

Testing during development was naturally less rigorous than unit testing, typically just being a side-product of getting a component to function properly. Most testing in this phase was checking if the various states of the application were updating correctly as well as writing data to Firebase properly.

Proper testing was carried out in a more structured manner, with separate units having their own tests developed for them. Testing would typically begin at the start of a new sprint when most or all functionality of a unit was implemented. A card in Trello and a document on Teams were created specifically for testing a unit, and a tester would begin developing test cases for expected results of a unit. The document would contain multiple tests comprised of a case, its result, and if necessary then the reason for failure. If any cases failed, then the member/s responsible for development of that unit would look into the issue and resolve or escalate it.

Scenario/Action	Expected Results	Pass/Fail	Comments
Select a rating	Rating displays in icons and "your rating" updates	Fail	<p>Tested in Google Chrome on Onyx Coffee Lab spot. ChIJa00m55kayYcRnz5 WcvjDiMI User signed in.</p> <p>Icon changes to proper rating, but "your rating" remains as unrated. For some reason, rating isn't saved for the user to the spot in <u>Firestore</u>, but there are previous entries on the spot from other users. (FIXED in <u>Indiv</u> ratings #55)</p>
Select Google Reviews tab	Shows reviews of spot from Google	Pass	
Select Comments tab	Show reviews of spot from Study Spots users	Fail	<p>Just says comments, feature is still being implemented.</p> <p>(FIXED)</p>



The rest of the testing occurred accidentally when errors would occur during further development. In this case whoever had the issue would bring it up at the next sprint meeting or in a message board on teams. Then the developer closest to that unit would take up investigating the flaw and correcting it or escalating the issue. Once the testing for a unit was concluded, the card in Trello would be moved to the "Done" category and a unit is hopefully finished with development.

## 7.0 Conclusion

This section will list a few final reflections on the project.

- **Architecture:** The separation between React and Redux was very effective for team development. This allowed two team members to develop the user interface while two other team members worked on API and database connectivity, asynchronously.

However, the implementation of Redux became more complex over the course of the project, and a dedicated backend server may have been a better choice for this project.

- **Schedule:** Development of the Spotlight page caused significant delays in the project schedule. In total, the schedule fell behind by about 3 weeks. There were up 2 weeks built into the schedule as a cushion for delays such as this, so the team was still able to satisfy the main requirements of the application. However, the team decided to cancel the development of a mobile application, and not as much testing could be performed on the final product as a result of this setback.
- **Future Improvements:** If the team had had more time to work on the project, they would have implemented a few more features that are commonly seen in web applications such as Study Spots:
  - The app is lacking an ability for users to change their account credentials, which is crucial in case a user forgets their email or password.
  - The team would like to add a user preferences system which would create a more tailored experience when searching for desired locations in the Spotlight.
  - The team could utilize the Google Maps API to create imbedded maps in the individual Spot pages, which would give users a tangible idea of where a location is without needing to use another navigation service in some cases

Overall, the project was a success. The final product is the Study Spots web application, which can be accessed from <https://study-spots-uark.herokuapp.com>. The source code can be found at <https://github.com/rghosh96/StudySpotsFrontEnd>.

## 8.0 Key Personnel

**Alexander Davidson** – Davidson is a senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed Database Management Systems, Software Engineering, and Programming Paradigms. Davidson has interned at Cerner Corporation in Kansas City, MO on the RxStation development team. He has also developed a React/ASP .Net Core web dashboard during an internship with SVI in Fayetteville, AR. Davidson is the project leader and will be responsible for project oversight, scheduling, and various coding or database management tasks.

**Rashi Ghosh** – Ghosh is a senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas. She has completed Programming Paradigms, Software Engineering, and Database Management Systems. She has had an internship at Cerner in Kansas City, where she worked with JavaScript to help develop/maintain a web application for health care managers to monitor their patients. Additionally, she has worked on two different full-stack personal projects utilizing React-Redux-Firebase to create web apps. Specific tasks will be allocated once the design is finalized,

however, she will most likely be very involved in the underlying structure of the web app – the front-end with React, the middleware with Redux, and the backend with Firebase/Firestore.

**Oliver Hubbard** – Hubbard is a senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed Programming Paradigms, Database Management Systems, Software Engineering, and AI. Specific tasks will be API integration and once the foundations for the project are created more focuses will be specified.

**Hunter Lowery** – Lowery is a senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas. She has completed relevant courses such as Programming Paradigms, Software Engineering, and Database Management. During her sophomore year, she had an internship with web applications at Newport News Shipbuilding in Virginia. This semester, she has completed a software engineering internship with Marshalltown and focused on updating an old web application. She will be responsible for working with the frontend and assisting with anything else necessary.

## 9.0 References

- [1] Google Firebase, <https://firebase.google.com/>
- [2] Google Places API, <https://cloud.google.com/maps-platform/places>
- [3] Popular Times API, <https://github.com/m-wrzt/populartimes>
- [4] React Native, <https://reactnative.dev/>
- [5] Redux, <https://redux.js.org/>
- [6] Framework Television, <https://www.youtube.com/watch?v=eLGtNm4dSxc>
- [7] Yelp, <https://www.yelp.com/>
- [8] Google Maps, <https://www.google.com/maps>