



**University of Arkansas – CSCE Department  
Capstone II – Preliminary Report – Spring 2022**

**MAGIC: A Transportation Toolkit and Application for Older Adults**

**Emily Lea, Patrick Page, Patrick Karangwa, Alan Torres, Sailesh Sirigineedi**

**Abstract**

Among the growing population of older adults there is a lack of reliable, familiar transportation services to get them to the grocery store, medical appointments, etc. in order to meet their primary needs and even for non-essential tasks such as recreational activities or meeting with friends and family. This could lead these older adults to seek unfamiliar and confusing transportation services. The main objective is to build a mobile application with a toolkit that will assist the elderly in locating appropriate, safe, and reliable transportation services in the area, as well as scheduling a ride if a personal driver is available. After this, the next-most important aspect will be designing the application to be as user-friendly as possible to accommodate the elderly. To build this easy-to-use mobile application, we will be using React Native, Python/Django, and PostgreSQL for most of this project's needs.

**1.0 Problem**

In the US, the population of older adults ages 65 and over is expected to quickly rise within the next decade. By 2030, roughly 20% of the population in the US will consist of older adults [1]. Arkansas is currently within the top 50<sup>th</sup> percentile in the nation with regard to the growth rate of the aging population [2]. Rural areas - many of which have poor, sparse transportation resources - will see a greater increase in their older adult population than other regions will [1]. The lack of access to transportation resources may not be limited to rural areas, however. In more densely populated urban areas with fewer numbers of older adults, public transportation may be one of the first resources to be cut in times of budget issues.

Transportation by means of an older adult driving themselves is not possible in many cases due to poor health or physical mobility issues. Transportation by means of family members may be infeasible if the older adult does not live near relatives or if a relative is unable to transport them on specific dates and times. Older adults in areas that do have public transportation may be unfamiliar with using it, may be concerned for their health and safety in using it, or may be unaware of the transportation resources available to them.

In all of these situations, the core issue is a rapidly growing demographic's lack of information about safe, reliable transportation resources in their region. Even if transportation resources are present in many regions, older adults may not be aware of the options available to them. Essentially, not knowing about transportation resources translates to lack of transportation. Lack

of transportation hinders the ability of the older adult population to care for their basic needs, such as getting to the grocery store or to medical appointments. This can also interfere with their ability to meet secondary needs such as going to recreational activities or socializing with friends and family. If the growing population of older adults is left with little resources regarding access to transportation resources, they will be unable to meet their basic needs or may be forced to seek out unsafe, unfamiliar transportation methods.

## **2.0 Objective**

The objective of this project is to create a user-friendly mobile application to assist the elderly in finding safe, reliable transportation to medical appointments, grocery stores, recreational activities, and a variety of other necessary destinations. The mobile application will include a toolkit, very easy-to-use for the elderly, which allows them to locate the appropriate transportation service or personal driver in their area, and schedule a ride to whichever appointment they need to get to. The MAGIC toolkit app is designed with the elderly in mind because the primary purpose of the app is to help the elderly, many of whom are not very technologically inclined. With this in mind, the MAGIC Toolkit application will also attempt to keep the elderly clients as comfortable and care-free as possible during this process, placing an emphasis on drivers they can trust, mostly friends and family.

When the common word “user-friendly” is used in this context, it means that we have to ensure the objective of our application will be met in its entirety. We will be taking into account that if an elderly client attempting to use our app cannot figure out how to use it, then the application is pointless to them and does not achieve its ultimate purpose for that particular client. Therefore, ease-of-use is one of our primary objectives, only surpassed by the element which assists the clients in getting the proper transportation at the correct time. This objective involves designing our application so that even the least technologically inclined clients will be able to use the application to achieve their goal of getting reliable transportation when they need it most.

To be more specific, we will use React Native, Python/Django, and a PostgreSQL database to implement a mobile application with a variety of different location-based transportation resources to assist the elderly in getting to and from the places they need to go. We intend to make the application easy for anyone to use, regardless of their background or previous experiences with different technologies. With this application, the elderly will be able to receive assistance in an area that might have been previously lacking in their lives. We strive to place an emphasis on facilitating friends and family volunteers to give their elderly friend or family member a ride as opposed to complete strangers. We emphasize this because studies have shown that the elderly are wary of getting transportation assistance from complete strangers. Our objective will need to be met with the general idea that transportation services from friends and family come first, and then other transportation resources will be made available if that is not available. We want to provide a safe, reliable transportation assistance application which leaves the client feeling care-free and as comfortable as possible every step of the way.

## **3.0 Background**

### **3.1 Key Concepts**

For this project, there will be users on both iOS and Android devices, because users will have to have a smart phone to work this application and somewhere around 99.3% of all smart phones are run by either Android or iOS operating systems. This is important because this leads us to our design decision of what programming languages and frameworks to use when developing this application.

We decided to use React Native, a very versatile JavaScript framework, which has gained its popularity due to the high reusability of code. In other words, this is a great framework for cross-platform development, as most of the code will work on either iOS or Android devices. It is approximated that the reusability of code between Android and iOS devices is 90-95%. This is why we decided to implement the front-end of our application with React Native.

Our team also decided to use Django, a web framework based on the Python programming language. Django was designed to help developers get from their initial design concepts to project completion as quick as possible. It follows the typical model-template-views architectural pattern. Django is fast and easily scalable. These are just some of the reasons we decided to use Django to implement the backend of our application.

For our database purposes, we will be using PostgreSQL, an open-source database management system. This is where we will be hosting all of the usernames, passwords, driver information, client information, home addresses, and any other important information needed to help our application successfully run.

For hosting the Django backend application and the database, we decided to use Heroku and Heroku Postgres. Heroku is a cloud-based platform that allows users to provision and deploy applications and to manage the database interacting with those applications.

### 3.2 Related Work

Here are the services that are currently available and some of the shortcomings that our application will improve upon:

- **Uber:** The company Uber has launched several programs to assist senior citizens get to their destinations safely:
  - Uber launched a program in Gainesville, FL that aims at teaching seniors how to use their Uber app [3].
  - *UberWAV* was launched in Toronto, Canada to give people with wheelchairs an option to choose a wheelchair-accessible vehicle [3].
  - *UberASSIST* was designed for people with special accommodations [3].
    - This service provides trained drivers to help elderly adults.
    - Vehicles are wheelchair-accessible.

The shortcomings that will be addressed by MAGIC:

- Senior citizens have reported that the user interface of the *Uber* app is confusing and makes the app hard to use. MAGIC has a simple user interface that is more friendly to older adults.
- MAGIC focuses on family, friends, and neighbors being the ones offering rides instead of complete strangers, an issue that has raised concerns among older adults.
- **Go Go Grandparent:** this program is specifically oriented towards senior citizens and it accomplishes the following:

- The user purchases a membership for \$9.99/month [4].
- The subscription allows them to call a number which prompts them to make choices on the dial pad, and then an Uber or Lyft is called on their behalf, without having to create an account for any of those platforms [4].
- The user can dial back the number and select to be dropped off home once they are ready to go back home.

The shortcomings of this service that will be solved by MAGIC include:

- Price: this service is very expensive because on top of the \$9.99/month subscription fee, the user has to pay \$0.27/minute fee + the base fare for the Uber or Lyft. MAGIC will be free-of-charge because it will be based on volunteer drivers [4].
- This option is not convenient for seniors who suffer from hearing loss since it requires them to listen to prompts on the phone. MAGIC will encompass a very simple and straight-forward user interface.
- **Lyft:** The company Lyft has recently added a feature that would allow senior citizens to order a ride by phone call. Here is how they accomplish that:
  - The user calls a phone number and then they get to choose various prompts to connect them to a driver, without the necessity of having the mobile app [5].
  - Updates are communicated via text message [5].

The shortcomings that will be addressed by MAGIC are:

- More simplicity offered by MAGIC compared to having to dial and hear different prompts, which can be very confusing.
- The application would cost no money.
- MAGIC focusses on family, friends, and neighbor's volunteer services, eliminating the issues that can come with riding with strangers.

## 4.0 Design

### 4.1 Requirements, Use Cases, and Design Goals

The basic features and requirements include providing information about available transportation resources as part of a toolkit, providing trip planning guidance based on origin and destination, and trip coordination for older adults with families and friends. We would also like to have trip coordination with volunteers. We want account creation and logging in for all users – both older adults and drivers – as well as a system for older adults to rate the drivers. The following requirements and use cases will be included in our design process:

#### Requirements:

- Provide reliable trip planning guidance and facilitation for older adults, i.e., passengers.
- Must be user-friendly to target the elderly population with little to no technical skills/experience.
- Must have both a different view of the application for drivers and passengers
- Must allow drivers to set and update the times they are available to provide transportation.
- Should allow drivers to send a friend request to a passenger by looking up their name in the database.

- Passengers should be able to accept or decline requests from drivers to be included among the trusted drivers.
- Provide the ability for the passenger to view transportation options from only friends, family, or other trusted individuals in their social circle.
- Provide handicapped accessible transportation resources for passengers in need of these resources.
- Provide a rating system for passengers to rate their drivers, an important part of many transportation-related applications.
- Must provide the ability for passengers to schedule a date and time for a transportation request.
- Provide a list of available drivers from which the passenger can choose who they would prefer to receive transportation from.
- Must have push notifications for drivers to receive with transportation requests in their area during their available time slot.
- Must have push notifications for passengers to receive, with their request being accepted/denied by the driver they requested.
- Provide the ability for passengers and drivers to create an account to log into application.
- Must have a navigation menu to accompany any buttons on the home screen and other views within the application.

#### Use Cases:

- Elderly adult schedules a trip some amount of time in advance, producing a list of drivers available at that date and time from which the passenger, i.e., the client, can pick whichever they would like. The requested driver received the request and accepts it, then gives the client the requested transportation at the given time.
- Elderly adult schedules date and time of requested ride, selects preferred driver, but driver declines request for whatever reason. The elderly adult is notified of the driver's denial of their request and is able to send a request to one of the other available drivers.
- Driver accepts request to give transportation to a client at a scheduled date and time, but has to cancel the ride at the last minute due to emergency or for whatever reason. Must at least consider some way to quickly fix this to ensure the client still gets a ride if at all possible.
- Client requests a ride right when they need it, without giving any advanced notice. This would require some consideration into a prompt response system or even a category of drivers which are available for quick rides, or some other design and implementation to handle this use case. If this use case is not considered during design and implementation, the client would quite likely have to wait a good amount of time for a driver to accept the transportation request.
- Client requests a ride from a specific place, but then leaves the location where they requested the ride from. Need to consider real-time geo-location services (when permissions are enabled) to allow the driver to find the client and/or ability for driver to call the client on the phone. A button for the driver to call the client can be easily implemented on the "Accepted Requests" page which can be accessed by the driver in their side navigation menu (accessed from the navbar menu button). The location of the passenger can be accessed anytime the passenger changes location using a React Native useEffect hook which can change anytime the location of the passenger changes. This

can be further implemented by in-app navigation or even navigation using other transportation app such as Google Maps or Waze.

- Driver arrives to pick client up at scheduled time but the client is not there. Driver might need a way to call client to let them know that they are ready to give them transportation.

## 4.2 Architecture

### 1. Architecture

The MAGIC app we have developed is well on its way to being a production level mobile application which could be used by the general public. The app, when any necessary future work is completed, should enable the elderly to request rides from drivers to get to wherever they need to go. The React Native front-end that we have is very user friendly, especially for the passengers, as they may not be as technologically inclined as the younger generations. In this section we will get into specific technologies used to create the application, the architecture of the application, how we implemented the different parts of the application, potential impact, and the any future work that might be needed to make this application ready to go into production.

To begin our design and implementation, we designed a Postgre SQL database. After the database was set up and operational, we created the back-end code using Django/Python. While we were still working on setting up the Django/Python back-end and Database API's, we started front-end development using React Native and the Expo mobile development framework. Up to this point, we have the vast majority of the UI/UX needed for a production level mobile MAGIC application. The following discussion will be more in depth on what we have already implemented to date, and then there will be some high-level discussion of any future work that might be needed to move this application into a production environment and deploy it to the apple/google app stores.

We have an initial login page with the ability to login as an existing user or select from two different types of users to create an account, either a driver or a passenger account. Creating a driver account requires more information to be entered than a passenger account. The MAGIC app will require drivers to enter their first name, last name, email address, phone number, password, license plate number, the make/model of their vehicle, and whether it is wheelchair accessible or not. The passenger account creation page asks for quite a bit less information than the driver account creation page, simply their first and last name, home address, email address, phone number and password. We have also developed a way to save and host images for each driver (and passenger) to create a more comfortable experience for the passengers in which they see a picture of those from whom they will be receiving transportation. We have the image uploader working and functional but we have run into a few problems with showing the images on the account info/settings page for that specific driver/passenger. Our development team did not have time to figure out how to remotely host/retrieve images on a remote server or database, so we simply had to access them using the URI to the direct access storage location for the image on that specific device.

Once either the driver or the passenger correctly fills out their information and presses submit, their account will be created and stored in the corresponding passenger/driver database table using a backend API call. Once this happens, they will be taken to their respective home page, the driver home page for drivers or the passenger home page for passengers. The UI for the passenger and driver home pages have very similar User Interfaces, but the functionality and User Experience for them are very different, for obvious reasons. For both the driver and passenger home pages, there are only four buttons (not including the navbar which will be

discussed in a couple of paragraphs). However, these buttons are different depending on the type of account that is logged into the application.

For the driver home page there is a button labeled “Available to Drive” that a user can click to set themselves as available to drive immediately. There is another button on the driver home page labeled “Set Availability” which allows the driver to set a schedule for their availability for that week, and then a final button labeled “Send Friend Request” for them to send a friend request to a passenger. The friend request functionality is to enable drivers to send a friend request to their friend/family member because studies show that the elderly are less trusting towards complete strangers, so their family and friends can send friend requests so that they know they are requesting a ride from someone who they can trust.

For the passenger home page there is a button labeled “Go Home”, which enables passengers to request a ride home from wherever they are located at that time to their home address. The second button on the passenger home page is labeled “Schedule a Trip” and this button will enable passengers to schedule a ride immediately or in the near future to wherever they would like to go. The final button on the passenger home page is labeled “Resources” and navigates the passenger to a Resources page where they can call local transportation resources in their area, simply by pressing a phone icon next to the resource in which they would like to call. There is one more button that is on both the passenger and driver home pages and that is the help button. This button is mainly for the user to contact customer support or a software developer to get help with something or report a bug/problem with the application. The help page allows the user to call a number for the MAGIC development/management team to leave a voicemail, or to enter and send an e-mail directly from the help page modal. The user should be able to report any bugs/errors or complaints they have about the application to someone who has the ability to either fix it themselves or assign someone else to fix it (in the future when there are staff maintaining the application).

To implement the ability of passengers to schedule a trip, we chose to implement a date/time selector page, a destination selector page, and a driver selector page. We chose for the User Experience to take the passenger to the destination selection page first. The destination selection page utilizes the Google Places API and contains a simple text input bar to allow the user to search for the address which they need transportation to. The Google Places API gives dynamic suggestions based on what the passenger has typed, which again helps the app remain more user-friendly. Once the user picks a destination, they are taken to a date/time selection page, which simply allows the passenger to select the date and time in which they would like to receive the transportation.

Once the passenger selects a destination, they will be taken to the driver selection page, which will have a list of drivers from whom they can choose to request a ride. With the date/time of the ride request already in the system, we now load drivers available at that date/time into a React Native FlatList component for the passenger to view them. We also have a filter process for the driver to filter by friend/family drivers, non-friend/family drivers, and all drivers based on what they choose and are comfortable with. When the passenger clicks these buttons, the React Native FlatList component displaying the local drivers should filter based on their selection. The passenger can then click on the driver they would like to request a ride from and the list item of that driver should be highlighted in a different color from the rest. From there, all they need to do is click “submit request” and the request should be sent to the correct driver.

Once the request is sent to the driver, the driver then should be able to accept/deny the request. At this point the planned trip should be added to the “Rides” section of the corresponding passengers’ sidebar menu and the “Accepted Requests” section of the corresponding drivers’ sidebar menu. Once the driver navigates to the “Accepted Requests” page they should be able to add the scheduled ride to their Google or Apple calendar to assist them in remembering the scheduled ride. We also used Expo push notifications, which are explained better below, to notify when requests are received, accepted or denied. Knowing the geolocation of the passenger allows the driver to know where to pick them up at the time of the trip, and once they arrive at the location, they can Notify the passenger that they have arrived. The passenger will then receive a notification on their phone that their driver has arrived.

The next thing we had to decide how to implement is the “Set Availability” function on the Driver home page, which in-turn enables the “Driver Selection” page for the passenger to only show the drivers that are available to give rides at that date/time. The schedule availability page allows the driver to select their availability for that week, allowing them to select for each day of the week, starting with Sunday. This allows us to put this information in the database and later retrieve this information for that specific driver. This, in turn, allows the passenger to view only the filtered list of available drivers (and in the future filter by availability and location) before they are ever filtered further by the passenger.

As mentioned above, there is the capability for the drivers to send friend requests to passengers they may know and this functionality is implemented as a modal on the driver home page screen. Once the send friend request button is clicked, the modal pops up over the home page, and a list is populated with passengers to whom they can send friend requests. At this point we have only a finite number of passengers available to send requests to (for testing purposes), but this should scale well with any number of passengers in the area. The more passengers you have the more time it may possibly take to load them, but because we are using a FlatList for this it shouldn’t load them until the user scrolls down the list. We have also implemented a search filter bar which allows the driver to type a name or a phone number and as soon as the driver begins typing, the list is filtered based on what he has typed. The driver can filter the passenger list by searching for a specific phone number or a name and the filter should narrow down the options as the user types either. Once they find the passenger, they can click (highlight) that passengers name and select the send friend request button below the list.

We have implemented the sending/receiving of friend request notifications/push notifications using Expo push tokens which are unique to every device (not to accounts). When a person logs in or creates an account, an Expo push token is created and stored in the database. As states above Expo push tokens are unique to a physical device, not to a user’s account. To send/receive a friend request, the user is asked to enable notification permissions, or their permissions are checked, and then the request is sent to the Expo servers. The request contains the recipient’s Expo push token and any other information that we want to include. At this point, Expo’s servers handle the sending of the notification to the recipients Google or Apple device. Once a friend request is accepted, the user ID of the Driver is stored in the database next to the corresponding Passenger user ID. This enables back-end calls to retrieve a user’s friend list with relative ease. This is helpful for things like filtering drivers based on those who are friends/family and those who are not, or simply allowing the user to view their friends list.

The MAGIC application necessitates the use of Geolocation for multiple purposes. It requires the driver to be notified where the passenger is so that the driver knows where to go to pick them up, and then they need to know where to go when the passenger needs a ride home.



For future work we plan to implement a way for the driver to access GPS Navigation apps such as Google maps or Waze outside of this app but for now we have implemented a way to ask the passenger user for their permission to access their location and then if they accept this automatically gives their location information to the application, which can then be stored in variables or in the database (for saved locations such as their home address but this is future work).

As stated above, there is also a Navbar that was custom designed for this application. It was designed in React Native just like the rest of the app, but it was separated into its own component for reusability across the entire application. This navbar contains a menu icon, which will enable a side menu to pop out for user selection/navigation simplification. The navbar also contains the title of the current screen in the middle, and an account page icon in the top right which acts as a shortcut to the account info/settings page for that user. The account button works for both passenger accounts and driver accounts.

The navbar side menu has different buttons depending on whether it is a passenger account or driver account accessing it. The passenger side navigation menu contains a "Home Page" button to return them to the passenger home page, a "Profile" button to take them to their profile information page (account info/settings page), a "Rides" button to view their rides, and a "Friends" button to view their friends list. The driver side navigation menu contains some of the same buttons but also a slightly different button. It contains the same "Home" and "Profile" button as the passenger menu, but it contains an "Accepted Requests" button to show their Friends List which contains the passengers who have accepted their friend requests. Both the passenger and driver side navigation menus contain two buttons at the bottom, an "About Us" button to navigate them to the About Us page which tells the user about the MAGIC app, and a "Sign Out" button which allows them to sign out of the application if they would like.

As stated above there is still future work that needs to be done to get this application ready for production and use by the general public. First things first I will mention that the Passenger should be able to add their accepted ride requests to their Apple/Google calendar. While it should not be required for ease of use, it should be an option for those elderly who do know how to use technology well enough to know how to utilize calendars on their phone. Another important feature for future work is to allow for the application to be translated to other languages. This will widen the demographic that can successfully and easily use the application.

Another item for future work is add a settings tab to the account information page, one of which would be to allow the application to be put in "Dark Mode" for those who need this to keep from hurting their eyes. The option for drivers to disable notifications could also be implemented even though this could possibly increase the chance that they would forget a scheduled ride. Another additional setting on the settings page could be for the passenger or driver to increase the font size of the application to make it more easily readable for those who are visually impaired. Along the same lines, screen reading could be offered to those who are legally blind, so that they can still use the application. This would be a more in-depth task as it would have to be implemented on all of the different pages in the application, some of which would be much more difficult than others (such as the destination selection page, date/time selection page and driver selection pages).

In the future, the Local Resources page would need to be more fully implemented with resources local to the passengers' location. At this point, this is just a mock page because we did not have time to implement a way to gather local transportation resources in the area and display them to the screen, but this could definitely be achieved with the correct API's and programming. Resources such as free public transportation in the area, reliable taxi services, and handicapped accessible transportation are all examples of transportation resources that could be displayed (if they are local to the passengers' current location).

Future work also needs to be done on the navigation aspect of the MAGIC app. For instance, the MAGIC team could implement in-app navigation to help the driver get to their passenger. This would be ideal as it would keep the application up on the user's phone and they would not have to switch between this app and other navigation applications. However, this is a very large task, and could also cost quite a bit of money to implement, so another alternative would be for 2<sup>nd</sup> party navigation apps such as Google Maps or Waze to be used. The development team could implement a way for the driver to click an accepted transportation request on their "Accepted Requests" page and be taken to either Google Maps or some other third-party navigation app for navigation to the passengers' current location for pickup.

Future work also includes an easier form of logging in (or staying logged in) to make the application more user-friendly for those elderly who have trouble remembering email addresses, phone numbers, and/or passwords. Another important feature for future work, one that could potentially help save a life, is the ability to call emergency services at the touch of a button and give them the current location of the passenger/driver who called. There is much more future work that can be done with the MAGIC app, and the remainder of the future work will be discussed in the Future Work section of this report (section 4.2 subsection 7).

- **Components:**

- Passenger/Older Adult side
  - After downloading the application, the user will be prompted to choose whether they are a driver or passenger. For new users, they will have to first choose which side of the application they will be using and then create the corresponding account type. After the initial account creation, they will have to log in with their credentials and they will be redirected to their appropriate side of the application.
  - When signing up, they will be required to provide their first and last name, email address, home address, and phone number. Then, they will create an account username and password for authentication purposes.
  - *Home Page:* this will be comprised of an easy and straight-forward user interface. It will have three main buttons, one for the passenger to request a ride home, another one to plan a trip to a desired destination, and another one to get access to alternative transportation resources. The home page will also have a navigation pane that allows the user to access various functionalities like managing notifications, reporting an issue, going to a settings page, etc.
  - *Go Home:* when signing up, the passenger will be asked to provide their home address. This address will be stored in the database and used whenever the passenger wants to go home so they don't have to enter it all the time. Once they click on this button, they will be prompted to choose



- Drivers will download the same application as passengers do and choose the option for drivers. Just like passengers, they will be asked to login if they already have an account or sign up if they don't.
- The driver's sign-up page will be a bit different from that of a passenger. On top of providing their credentials and address information, they will be asked to enter information about the vehicle they will be using. In addition to that, all drivers will be required to provide a profile picture, that way their passengers can identify them and avoid confusion.
- By default, all drivers will appear in an unfiltered list of available drivers to the passenger. The driver can send a request to the passenger to be added to a list of trusted people, or "friends".
- *Home Page*: after logging in, the user interface of the driver's side of the application will be similar to the one for the passenger. The home page will have three buttons, one that allows the driver to set their availability to "now", one that allows them to provide a schedule for when they will be available in the future, and one that allows them to send a request to the passenger.
- *Available To Drive*: this button, when clicked, will allow the driver to indicate that they are readily available to offer rides. They will select how long they will be available. That will automatically show them as available when a passenger associated with them requests a ride. When available, the driver can receive requests from passengers and be able to accept or decline the requests. The request will come in form of a notification.
- *Set Availability*: this button, when clicked will take the driver to a page where they can provide a weekly schedule for when they are available to drive. This page has the seven days of the week where the driver can choose their start time and end time for each day. Once they choose their availability, it will show up on their phone calendar. The driver can always come back to this page and update their start and end times for each day whenever they want. The calendar on their phone will send them a reminder when the trip is approaching.
- *Send Friend Request*: when clicked, this button will allow the driver to search a passenger by their name and/or phone number in the database so they can be added to a list of trusted drivers. When the request is accepted, the driver will be added to passenger's list of trusted people.
- *Navigation Pane*: drivers will have a navigation pane with app settings and a way to report an issue. They should also be able to manage notifications.
- *Check in to location*: drivers will have the ability to check in on the app once they arrive at the older adult's location. This will send the older adult a notification that their ride has arrived.
- *Account Management*: just like on the passenger side, drivers will be able to access their account information through clicking the user account icon. When clicked, the user will be able to change their profile picture, vehicle information, as well as their address. They should also be able to log out from their account as well.

## 2. Technologies Used

- This application will use PostgreSQL as a database management system to store and manage data that is collected at sign-up or during app usage.
- The frontend of the application will be implemented using HTML, CSS, JavaScript, and React Native as a framework. React Native is supported on both iOS and Android operating systems, and therefore won't require making two applications for both systems.
- Python will be used on the backend side of the application, using Django as a framework.
- We will use Git to make our repositories as well as to coordinate and integrate each of our portions of work with one another.
- Trello will be used to manage the project as well as to document and track the progress on our tasks.
- Heroku will be used to host the PostgreSQL database and to deploy the Django backend application.
- Testing will be done on a combination of Android and iOS physical devices and emulators.

### 3. Interface Design with Screenshots

The user interface designs displayed below show an intuitive, user-friendly interface for our older generation target audience. Many of the buttons have an icon or text indicating what they are for. The text is large and easy-to-read, and a good amount of contrast exists between the text and the background. The color palette chosen helps distinguish different parts of the application without being too harsh or difficult to decipher. The initial page in the application (Fig. 1) allows the user to either create an account or to log into an already existing account. In the Account Creation pages (Fig. 2-4) for both the drivers and passengers, an alert will be displayed and the user will not be permitted to proceed if the required information has not been entered. The Home pages (Fig. 5-6) for both the drivers and passengers display three main buttons and a help button, each with icons illustrating what they represent. In the trip planning process, each stage has an explanation of how to use the components. For example, the Driver Selection (Fig. 7) page shows a searchable list that filters in real time according to the user's preference. The passenger will have the option to view a list of one of the following: all available drivers, only available drivers from their list of friends, or only drivers that are not on their list of friends. The Destination Selection (Fig. 8) page, that is used for trip planning, allows the passenger to begin entering either the name or address of their desired destination, and a list of those destinations nearest to them will be displayed. The Local Resources (Fig. 9) contains a selectable button for each resource listed on the page, as well as a phone icon indicating that the resource will be able to be called from the application. The Send Friend Requests page on the driver accounts (Fig. 10) has a list of selectable items that can be filtered by entering a passenger's name or phone number. From the driver home page, clicking on 'Available to Drive' will let the driver know they are set as currently available, and once this is set, they can turn this off (Fig. 11). The Set Availability page (Fig. 12) gives drivers a clear way to set or update a weekly schedule of days, with daily start and end times that they would like to use to transport passengers. Both the drivers' and passengers' Settings pages (Fig. 13, Fig. 14) contain text boxes and an image uploader box in which they can change their user information. The text boxes populate with the users' current information. The navigation top bar includes a top-left icon which brings out the side menus, which differ between drivers and passengers. From the driver's side menu (Fig. 15), the driver can go to the home page (Fig. 4), the settings ("Profile") page (Fig. 13), the accepted requests page (Fig. 22), the about us page (Fig. 23), and the sign out button. The accepted

requests page (Fig. 22) shows the list of transportation requests that the driver has accepted. This page is split into ‘Today’s Rides’ and ‘Future Rides’. On this page, once the driver has arrived at the pickup location, the driver can click on the ride (purple box with the time and date of the ride, and the name and phone number of passenger) and this pulls up the check in modal (Fig. 20). When the driver clicks on ‘Check In’, this sends a notification to the passenger that lets them know the driver has arrived. The about us page (Fig. 23) gives a brief background of the main objective for this project. The sign out button takes the user back to the login screen. From the passenger’s side menu (Fig. 16), the passenger can go to the home page (Fig. 5), the settings (“Profile”) page (Fig. 14), the friends page (Fig. 24), the about us page (Fig. 23), and the sign out button. The friends page (Fig. 24) shows the list of driver friends that the passenger has accepted. On this page, the passenger can also choose to delete any of their current friends.

Figure 1: Login

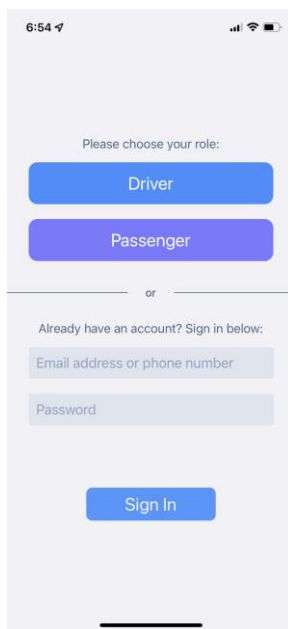
The image shows a mobile application login screen. At the top, the status bar displays the time 6:54, signal strength, Wi-Fi, and battery icons. Below the status bar, the text "Please choose your role:" is centered. There are two large, rounded rectangular buttons: a blue one labeled "Driver" and a purple one labeled "Passenger". Below these buttons, the word "or" is centered. Underneath, the text "Already have an account? Sign in below:" is displayed. This is followed by two input fields: "Email address or phone number" and "Password". At the bottom of the form is a blue "Sign In" button. The entire screen has a light gray background.

Figure 2: Driver account creation

6:33

Personal Information:

First Name

Last Name

Email Address

Phone

Password

License Plate Number

Vehicle Make

Vehicle Model

Wheelchair Accessible:

Figure 3: Passenger account creation

6:33

Click the circle below to upload an image of yourself.

Personal Information:

First Name

Last Name

Home Address

Email Address

Phone

Password

Figure 4: Driver home page

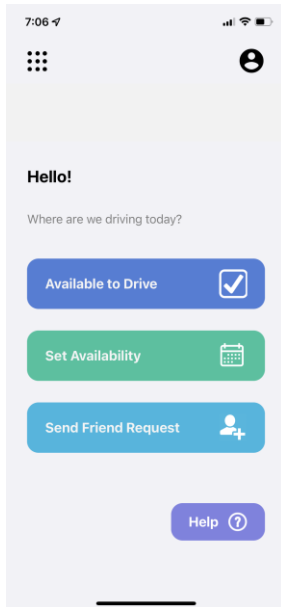


Figure 5: Passenger Home page, includes geolocation

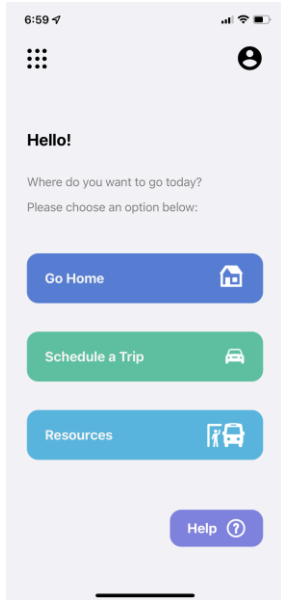


Figure 6: Trip planning, passenger accounts: selecting starting and ending locations



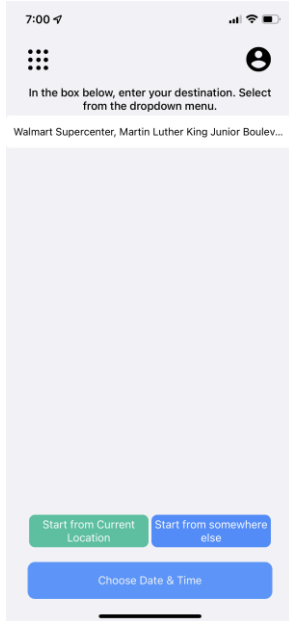


Figure 7: Trip planning, passenger accounts: selecting a date and time

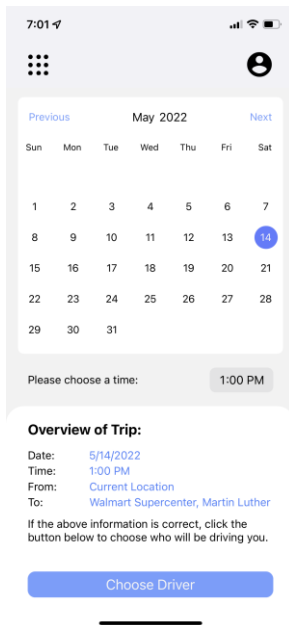


Figure 8: Trip planning, passenger accounts: selecting a driver

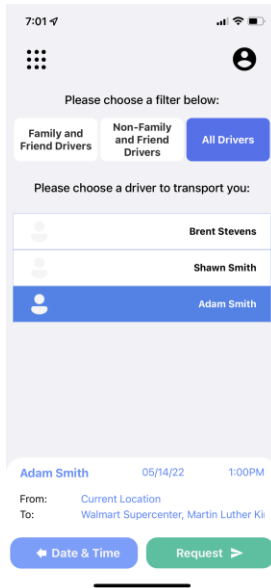


Figure 9: Passenger accounts: Local Resources page



Figure 10: Driver accounts, Friend Requests page

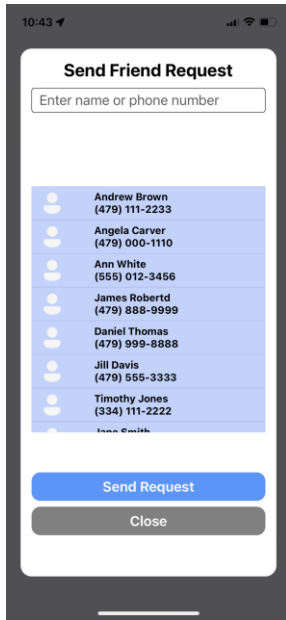
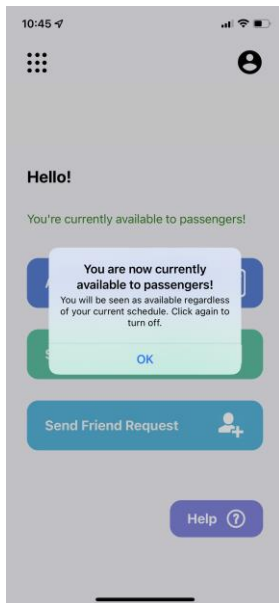


Figure 11: Driver accounts, Set Immediate Availability



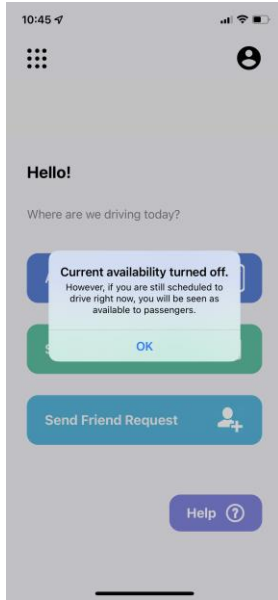
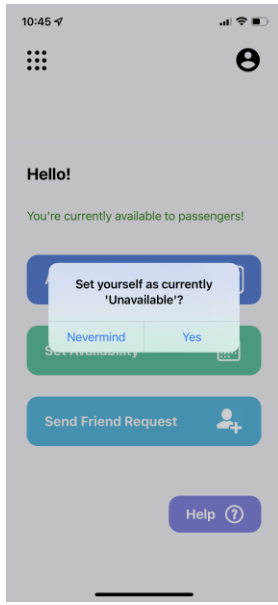


Figure 12: Driver accounts, Schedule Availability

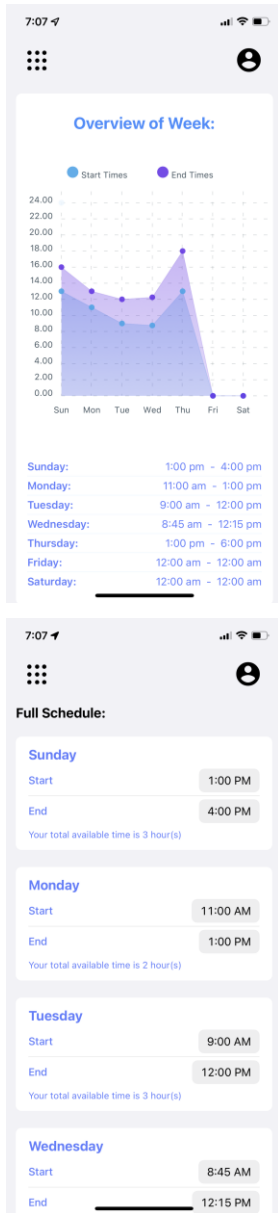


Figure 13: Driver accounts, Settings page

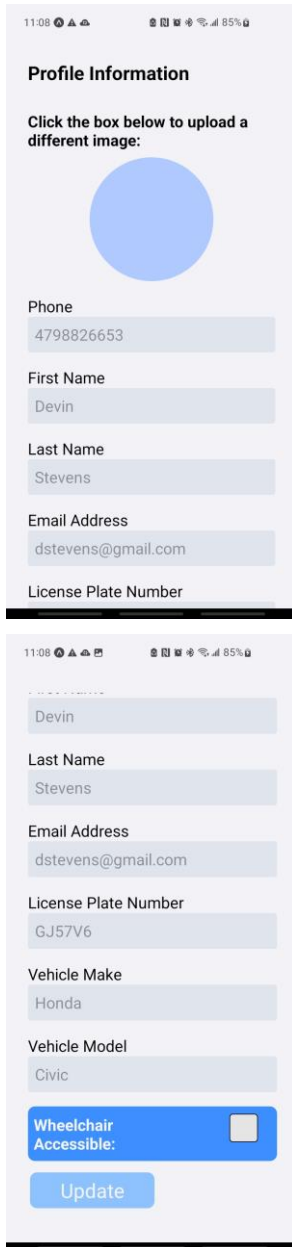


Figure 14: Passenger accounts, Settings page

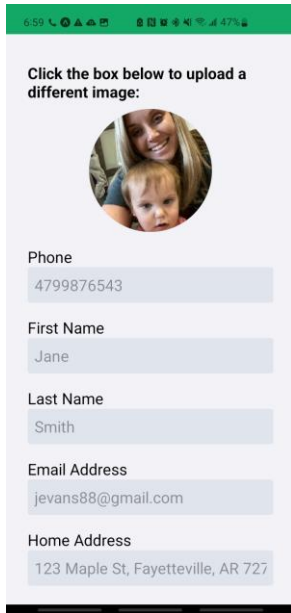


Figure 15: Driver accounts, Navbar

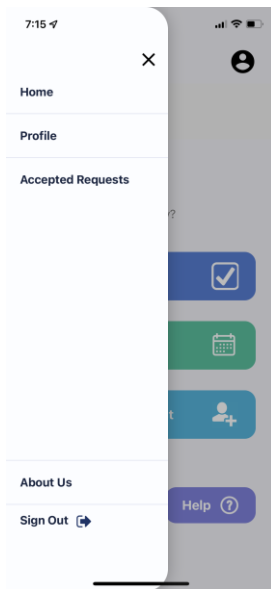


Figure 16: Passenger accounts, Navbar

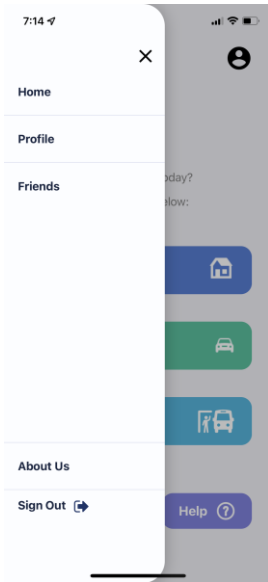


Figure 17: Driver accounts, Driver check-in upon arriving to location

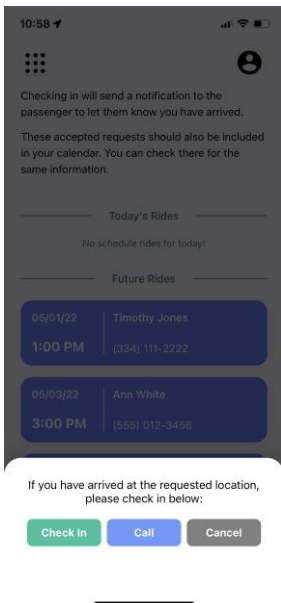


Figure 18: Passenger accounts, rating system



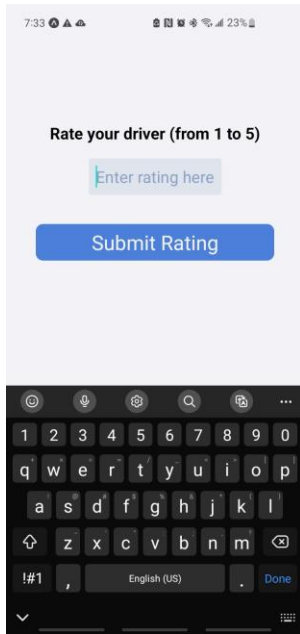


Figure 19: Driver accounts, list of accepted transportation requests

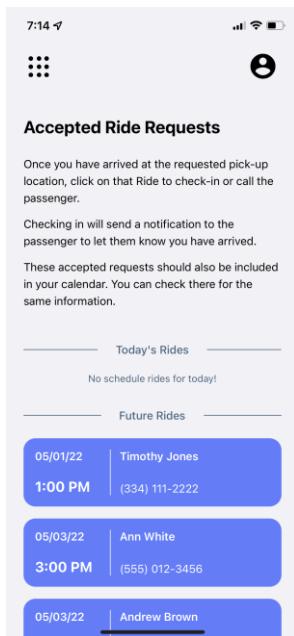


Figure 20: About Us page

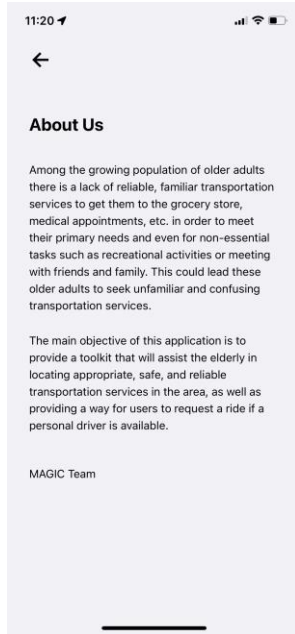
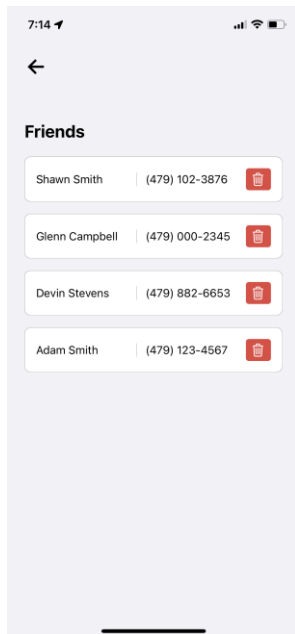


Figure 21: Passenger accounts, list of friends page



#### 4. Implementation

The general flow of the application development can be classified into the following stages: database, backend creation, account creation and login, home page and shared layout elements, friend requests, trip planning, resources, driver rating system. Each of us have divided the tasks up to be worked on in parallel during sprints that generally last from one to two weeks.

After setting up our development environments and repositories, the database was designed and created locally using PostgreSQL, then it was migrated to Heroku once the backend application was set up. It consists of the tables users, user\_credentials, drivers, passengers, driver\_schedule,

friends, images, push\_tokens, ratings, resources, and transportation. Users contains all users' phone numbers and their status as driver or passenger. The table user\_credentials contains the users' login credentials and their status as driver or passenger. These tables contain the users' status as a passenger or driver so that, when their information is returned, we can use their status as a parameter for React components. The drivers table contains the drivers' information entered during account creation, and the passengers table contains the passengers' information entered during account creation. The driver\_schedule table is used to store the drivers' availability schedules for one time range each day of the week. The friends table contains a record for each accepted friend request from a driver to a passenger. The table images contains the phone and email of the user, as well as the uri of the photo uploaded. The push\_tokens table contains the user's phone number, email address, and push token generated by Expo. The ratings table contains each driver's phone number, number of ratings received, and their overall rating. The resources table contains a resource entity's phone number, zip code, name, and a description of the resource. The transportation table contains the details of a confirmed transportation request as follows: the driver's phone number, passenger's phone number, scheduled date, scheduled time, passenger's first name, and passenger's last name.

The Django backend application was created and run locally before deploying it to Heroku. APIs were implemented to retrieve, create, or update data for each table in the database. These were created to be called primarily using POST requests. While the backend application and database APIs were undergoing construction, the UI/UX for several pages were created in React Native. At the beginning of the project, we started contributing to a shared components and shared styling file in the app in order for the sake of stylistic consistency and code reusability. Generally, our approach has been to construct the UI/UX for a page, work on the backend separately, and add the backend functionality at a later date. This has allowed us to work in parallel on a variety of tasks. Larger tasks, specifically those including functionality, have been assigned to multiple people at times.

Our approach to the frontend work of the application has generally been to follow the flow of the application. We started by creating the home pages, login page, and account creation pages. Two home pages were created: one for the driver account and one for the passenger account. Due to how similar the pages were stylistically, we decided to assign the UI/UX work to one person. Another person worked on the frontend of the login page while the backend work for account credential verification and page navigation was being done. The frontend work for the account creation and settings pages were done by the same person due to the fact that they both work with the same information: both pages contain text boxes that allow a user to either add or update their information.

After this set of tasks was completed, a navbar component was added to each page. The specific elements and drawer navigation will be implemented once the necessary pages have been implemented. The destination selector portion of the passenger accounts' trip planning pages and the driver accounts' availability scheduling was implemented. The location finding in trip planning was implemented using the Google Places API, while the driver availability scheduling was implemented by creating a shared component using React Native's date-time picker. The remainder of the database APIs and methods that would generally be needed going forward were completed during this time, as well.

Our next set of tasks included using the credential verification API call to determine whether credentials were valid and to determine whether the user should be taken to the driver or passenger home page. The functionality of the account creation and settings pages was

implemented during this time by calling an API that would store a new user's information in the database or update it in the database. The driver selector page of trip planning was implemented, as well, by calling an API that returns a list of drivers. The first name, last name, and phone number of the driver were retrieved and stored into a selectable item of a filterable list. The phone number was returned from backend so that when a driver is selected for a transportation request, their phone number will be present for a request to be sent to. Our final task in this set was a local resources page consisting of a list of hard-coded resources with a phone icon indicating that a person can call the resources from the application. The calling functionality will be implemented later in the semester.

After these were completed, we began implementing our next set of tasks. Geolocation for the passenger accounts was implemented using Expo's Location package to retrieve the passenger's location coordinates when they enter their home page. The friend requests page for the driver accounts was implemented, as well. Further filtering of available vs unavailable drivers was added in a later task. An API call was made to retrieve a list of passengers, each of which was stored in a searchable, filterable list. An image uploader was implemented using Expo's ImagePicker package.

The functionality of the help form was implemented once the previous set of tasks were completed. The ability for a user to email a developer account was added, as was the ability to make a phone call to a test phone number. During this time, styling was improved and made consistent across the application. The date and time selector page was implemented using the CalendarPicker and DateTimePicker components from ReactNative. The date and time selected on this page were passed to the next page in the trip planning process. The driver selector page in trip planning was given additional ways to filter drivers. First, only drivers available during the desired time of transportation were displayed. The user is also given options to sort drivers by whether or not they are on the passenger's friends list. During this set of tasks, tables were added to the database and APIs to backend to filter drivers based on availability as well as to add, update, and delete information regarding local resources and ratings.

Once these were completed, a button was implemented on the driver home page to allow a driver to set their availability to "immediate" and to remove immediate availability. The functionality of the Go Home button was added. Clicking this button retrieves the passenger's home address from the database and passes it to the remaining pages in trip planning. Navbar functionality was added, as well. The navbar for drivers includes Home, Profile, Accepted Requests, About Us, and Sign Out. The navbar for passengers includes Home, Profile, Friends, About Us, and Sign Out. The final task in this set was to add the ability for a person to make a phone call instead of having to send an email from the help form.

The final group of tasks included implementing friend requests. This was done by first generating and storing an Expo push token unique to a person's device upon account creation or login. When a request is to be sent, the recipient's token is retrieved from backend, added to a request with other data about the person sending the request, and is sent to Expo's servers. The notification is then sent from Expo's servers to the recipient. An alert with confirmation and denial options was added to the recipient's notification to allow them to add the friendship and store it in the database. A friend requests page and a friends list page were also added during this set of tasks. A rating page was added in which a passenger can enter a rating from 1-5 for their driver and then can send the rating to the database. A means for drivers to check-in was added to allow a driver to send a push notification to a passenger to let the passenger know the driver has arrived to their location. The final task, was implemented using push notifications sent from the

passenger to the driver. The ability to add details of a request to a calendar exists as a button component.

## **5. Lessons Learned**

The lessons learned in the construction of the application so far includes technical skills and project planning. We have all been learning what technically is needed in the construction of a React Native application with a Django backend, specifically when it comes to the two applications interacting with one another. Over the course of the project, we have each been able to become more acquainted with estimating the effort, resources, and time a task may take to complete. We have also become more acquainted with the UI/UX design process as opposed to implementing something that was already designed ahead of time.

## **6. Potential Impact**

The potential impact of this project is to improve the quality of life of the elderly clients by assisting them with transportation, making their lives less stressful and more fulfilled. This will have functionality on the front end for both halves of the user base, the elderly people who are needing rides, and the drivers who are able to provide the rides. We want the app to be user friendly for both of the user bases. This app will specifically be designed to be easy to use for old people as it will include a toolkit for the most common errands.

## **7. Future Work**

Possible future work for MAGIC includes adding the following: background checks and driving history for driver accounts, a web component to the application, alternate login methods, in-app navigation, the use of languages other than English, the ability to adjust text size, the ability to turn on “dark mode”, the ability to contact emergency services, the ability to add local transportation resources, and the suggestion of recreational activities in the area. Implementing background checks and driving history checks is a process that occur during the creation of driver accounts and would help increase the safety of the older adults using the application. Safety could also be increased by adding the ability in both driver and passenger accounts to contact local emergency services from the application. Adding the ability to access the application on the web as opposed to only as a mobile application would allow older adults to access the application if they are unable to use a smart phone. Alternate login methods such as the ability to sign in to the application with Facebook or Google could be used to allow both drivers and passengers more flexibility in how they would like to log in. This can increase the user-friendliness of the application for any elderly who have trouble remembering email addresses, phone numbers, and/or passwords.

Implementing in-app navigation with Google or Apple would be most beneficial to drivers in traveling to pick up the passenger and in taking the passenger to their desired destination. Adding the use of languages other than English to MAGIC will allow older adults who may not be very familiar with English to still have access to the transportation resources they need to meet their needs. Implementing the ability for a user to have a degree of control over elements of the user interface could make a user’s experience in the app better. For example, adding the ability for a visually impaired older adult to easily adjust the text size in the app would make their experience go much more smoothly. Also, the ability for a user to turn on “dark mode” can help those elderly (or drivers) who can see well but the brightness of the app may put strain on their eyes. For the passengers, we could also look at providing a ‘Favorites’ list of locations. This way, the passenger could set and remove their most frequent locations and it would make the transportation request process a little bit easier. A final consideration for

future work is to allow local transportation resources and local recreational resources to add their information to the application after going through a verification process. The verification process would ensure the organizations are legitimate, and older adults would be able to find out more about resources for transportation and recreation in their area. The last thing that should be considered for future work is to implement the ability for passengers to add their future transportation appointments to their Apple/Google calendars to help them with reminders so they don't forget about a ride they have scheduled. Some elderly may not know how to do this very well but others will so to add this option will likely increase the effectiveness of the app for those people.

### 4.3 Risks

Risk Summary	Mitigation Measures
An older adult may not know their exact current location when wanting to go home.	The older adult's location is saved through a geolocator that will be implemented with React Native.
The older adult may be unfamiliar with mobile applications, specifically with ones regarding transportation	The user interface will be made simple and intuitive, and instructions will be present in trip planning to resolve any confusion that may arise.
The older adult may not trust just anyone to give them a ride to a given location.	The older adult will have the ability to build a list of friends and family by accepting a friend-or-family request from someone. They will have the option in trip planning to view only the drivers listed as friends or family.
A login process needs to be present to mitigate security risks, but needs to be simple enough for its components to be easily remembered.	All users will log in to the app with their phone number and a password chosen in the signup process. The password has no requirements for special characters or uppercase/lowercase characters.
How will the older adult be able to find their driver if they are getting transportation from someone they do not know?	The drivers are required to enter a photo of themselves, a vehicle description (make, model), and license plate number. This will prevent an older adult from being unable to find their transportation. This can also mitigate physical safety concerns due to the fact that an older adult can easily verify if a driver is who they state they are in the application.
What if no drivers are available at the older adult's time of desired transportation?	Contact information for alternate transportation resources will be displayed on the driver selection page of trip planning.
How will an older adult know that their transportation has arrived or that their transportation request has been accepted?	Notifications will be sent between drivers and passengers in transportation requests, request acceptance and denial, and a driver checking in to state that the older adult's ride has arrived.

## 4.4 Tasks

1. Create UI/UX design drafts
2. Set up development environments
3. Design and implement the database
4. Add a help button that can later be used to ask questions or send feedback regarding the application
5. Implement the UI/UX for the driver and passenger Home pages
6. Implement the UI/UX for the Login page
7. Implement the UI/UX for the Account Creation and Settings pages
8. Implement the UI/UX for the Local Resources page
9. Trip planning: destination selector page (frontend)
10. Implement the destination selector page, both UI/UX and functionality
11. Implement the UI/UX for a navbar
12. Implement and test database APIs; deploy the Django backend app to Heroku
13. Implement login credential verification
14. Account creation and Settings pages
  - a. Account creation page: store a user's account information in the database
  - b. Settings page: retrieve and display user's information, and update changes in the database
15. Implement the UI/UX for the driver selector page.
  - a. Retrieve a list of drivers from the database; display them in a filterable, searchable list
16. Implement geolocation for passenger accounts
17. Implement the UI/UX for the friend request page
  - a. Retrieve a list of passengers from the database; display them in a filterable, searchable list
18. Implement the UI/UX needed for a driver to set an upcoming availability schedule
19. Implement an image uploader component on the Account Creation and Settings pages
20. Implement the functionality of the Help button
  - a. Allow users to send feedback or questions about the application to an email account
21. Trip planning: filter drivers by availability and friend/family status; combine and store trip planning request details
22. Implement further filtering for the driver selector page
  - a. Implement a backend method(s) that returns a list of drivers whose scheduled availability corresponds with the passenger's desired time of transportation
  - b. Implement the ability for passengers to choose to view only available drivers on their friends and family list
23. Implement the ability for drivers to set and remove their availability to "now"
24. Improve the styling across the application; make styling consistent.
25. Implement the ability for users to send and receive transportation requests.
  - a. Implement the ability for passengers to send transportation requests to drivers.
  - b. Ensure that drivers are receiving the transportation requests.
26. Implement the ability for users to send and receive friend requests.
  - a. Implement the ability for drivers to send passengers a friend request.
  - b. Ensure that passengers are receiving the friend requests.
27. Implement the ability for users to respond to requests.
  - a. Implement the ability for passengers to confirm or deny friend requests.

- b. Ensure that drivers are aware of the passenger's response to the friend request.
  - c. Implement the ability for drivers to confirm or deny transportation requests.
  - d. Ensure that passengers are aware of the driver's response to the transportation request.
28. Add or edit database tables and APIs for local resources, the rating system, and driver availability.
  29. Sync details of confirmed transportation requests to a user's Google or Apple calendar
  30. Implement the functionality of the navbar
  31. Implement the ability for drivers to check in on the application once the driver has arrived to the passenger's location
  32. Implement the functionality of the Go Home button.
  33. Implement the ability for passengers to call local transportation resources from the application
  34. Implement the ability for users to call a help number on the help form instead of send a message.
  35. Implement a Friends List page in which a passenger can view all of the driver whose friends request they have accepted.

## 4.5 Schedule

	<b>11/8-12/10</b>
Emily	UI/UX Design Drafts
Patrick P	UI/UX Design Drafts
Patrick K	UI/UX Design Drafts
Alan	UI/UX Design Drafts
Sailesh	UI/UX Design Drafts
	<b>12/10-1/17</b>
Emily	Environment Setup
	Database Construction
Patrick P	Environment Setup
Patrick K	Environment Setup
Alan	Environment Setup
Sailesh	Environment Setup
	<b>1/18-1/30</b>
Emily	Account Creation, Settings pages (frontend)
Patrick P	Driver, Passenger Home pages (frontend)
Patrick K	Login page (frontend)
Alan	Help button (frontend)
Sailesh	Date/Time selector page



MAGIC Toolkit and Application

	<b>1/31-2/9</b>
Emily	Create and deploy the Django app to Heroku; create and test some database APIs
	Login credential verification
Patrick P	Nav bar (UI/UX)
Patrick K	Trip planning: destination selector page (frontend)
Alan	Set driver availability (frontend)
Sailesh	Date/Time selector page
	<b>2/10-2/20</b>
Emily	Account Creation, Settings pages (functionality)
Patrick P	Trip planning: driver selector page (frontend)
Patrick K	Local Resources page (frontend)
Alan	Set driver availability (frontend)
Sailesh	Date/Time selector page
	<b>2/20-3/9</b>
Emily	Image selector, uploader
Patrick P	Friend requests page (frontend)
Patrick K	Passenger geolocation
Alan	Passenger geolocation
Sailesh	Date/Time selector page
	<b>3/10-3/26</b>
Emily	Preliminary report and presentation
Patrick P	Preliminary report and presentation
Patrick K	Preliminary report and presentation
Alan	Help button functionality (due 3/16); improved styling, made styling consistent across the app; Preliminary report and presentation
Sailesh	Help button functionality (due 3/16); Preliminary report and presentation
	<b>3/27-4/9</b>
Emily	Implement a way to add details of confirmed requests to Google, Apple calendars; Backend work for driver availability; Date/time selector page

Patrick P	Driver Selector page: filter drivers by availability and presence on a passenger's friends list; Date/time selector page
Patrick K	Sending/receiving friend requests
Alan	Add a way for drivers to set/remove immediate availability; Date/time selector page
Sailesh	Implement a rating system
	<b>4/9-4/17</b>
Emily	Sending/Receiving friend requests; Implement a way to add details of confirmed requests to Google, Apple calendars; Backend work for local resources, rating system
Patrick P	Implement navbar functionality; add the ability for a person to make a phone call from the Help form
Patrick K	Sending/receiving friend requests
Alan	Implement a way for drivers to check in upon arrival to a location; Implement navbar functionality; add the ability for a person to make a phone call from the Help form
Sailesh	Implement a rating system; Send/receive transportation requests
	<b>4/18-4/25</b>
Emily	Sending/Receiving friend requests; Implement a way for passengers to rate drivers; Implement a way to add details of confirmed requests to Google, Apple calendars
Patrick P	Implement a way for passengers to rate drivers
Patrick K	Sending/Receiving friend requests; Send/Receive transportation requests
Alan	Implement a Friends List page; Send/Receive transportation requests
Sailesh	Implement a rating system; Send/receive transportation requests

## 4.6 Deliverables

- User interface design drafts
- Database design and schema
- Text file of the test data entered into the database at the beginning of the project
- A Python Django application containing the following:
  - o Models mapping to each database table
  - o Serializers and viewsets for each model
  - o APIs to do the following for the driver\_schedule table: schedule availability, get and set immediate availability, and return a list of available drivers

- APIs to do the following for the drivers table: return a list of drivers with their information, find a driver by phone or email
- APIs to do the following for the friends table: add a driver to the passenger's friends list, delete a driver from the passenger's friends list, return a person's friends list, return available drivers from a passenger's friends list, return a list of available drivers not on a passenger's friends list
- APIs to do the following for the passengers table: find a passenger by phone or email, get a passenger's home address
- APIs to do the following for the user\_credentials table: retrieve a user's credentials by phone, retrieve a user's credentials by email, verify credentials
- APIs to do the following for the users table: add a new user to applicable database tables, update a user's information in all applicable tables in the database, retrieve a user by phone
- APIs to do the following for the images table: retrieve an image
- APIs to do the following for the push\_tokens table: retrieve a person's Expo push token
- APIs to do the following for the ratings table: calculate a rating based on a new rating, retrieve a driver's rating
- APIs to do the following for the resources table: add a resource, delete a resource
- APIs to do the following for the transportation table: add a confirmed transportation request, return a list of confirmed transportation requests
- A React Native application containing the following:
  - Files containing all modules, packages, and dependencies
  - A common file of styles used across the application
  - A folder containing files for each page in the application, as follows:
    - About us (driver and passenger)
    - Account creation (driver and passenger)
    - Trip planning: date and time selector (passenger)
    - Trip planning: destination selector (passenger)
    - Trip planning: driver selector (passenger)
    - Home (drivers)
    - Home (passengers)
    - Settings (drivers)
    - Settings (passengers)
    - Friends list (passengers)
    - Local resources (passengers)
    - Login (drivers and passengers)
    - Scheduling Availability (drivers)
  - A folder of reusable components containing the following:
    - a checkbox
    - the help form
    - a styled button
    - a button that, when clicked, adds details of a transportation request to a user's calendar
    - an image picker and uploader
    - a day-of-the-week and time selector used in scheduling a driver's availability
    - the navbars for driver and passenger accounts

- A zip file of all final code submissions
- Final presentation submissions

## 5.0 Key Personnel

**Emily Lea** - Emily Lea is a senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas. She has completed the following courses that may be relevant to this project: Software Engineering, Database Management Systems, Computer Networks, and Operating Systems. She has six months of experience working as a Junior Consultant in software development at Rural Sourcing, Inc. She will be responsible for the initial database design and construction and will help out on backend or frontend after.

**Patrick Page** – Patrick Page is a Senior Computer Science/Computer Engineering major at the University of Arkansas. He has completed courses relevant to this project such as Programming Paradigms, Software Engineering, Database Management Systems, Information Retrieval, Mobile Programming, Computer Networks, and Operating Systems. He will be responsible for front-end mobile app development, UI/UX design, and connecting the front-end to the back-end when the time comes.

**Sailesh Sirigineedi** - Sailesh Sirigineedi is a senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed Programming Paradigms, Software Engineering, Database Management Systems, and Operating Systems. He also has experience from summer internships of both front end and back end as well as some database knowledge.

**Alan Torres** – Torres is a senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed project-relevant courses such as Programming Paradigms, Software Engineering, Database Management Systems, Computer Networks and Operating Systems. He has four months working with website UI/UX design through an internship from Spring 2021. He also has a few months working with database design/implementation and REST APIs through another internship from Summer 2021. He will be responsible for helping out with any database design aspects, UI/UX design for the mobile app, and working on any back-end and front-end implementation when needed.

**Patrick Karangwa** – Patrick Karangwa is a senior majoring in Computer Science at the University of Arkansas. He has taken classes like Programming Paradigms, Software Engineering, Operating Systems, and is currently enrolled in Database Management Systems Management, Algorithms, and Data Mining, which will all serve as background knowledge for this project. He also has background knowledge in static web application development through various different personal projects he created. For this project, he will primarily be working on the front-end side of the development process, although he will also jump back and forth to the back-end as well.

**Dr. Suman Mitra** – Dr Suman Mitra, our project sponsor, is an Assistant Professor in the Department of Civil Engineering at the University of Arkansas. His research focuses are the travel behaviors of disadvantaged populations, travel demand modeling, and sustainable transportation, shared mobility services and autonomous vehicles, transportation and health, and land use-transportation interaction.

**Dr. Matthew Patitz** - Dr Matthew Patitz, our project advisor, is an Associate Professor in the Computer Science and Computer Engineering Department at the University of Arkansas. His

research focuses on DNA computing and algorithmic self-assembly. He received a CAREER award in 2016 and has published over 60 peer-reviewed journal and conference papers.

## 6.0 Facilities and Equipment

For the project we will be using our individual computers, Git, GitHub, React Native, Python Django, PostgreSQL, and Heroku in the construction of the application and toolkit. We anticipate using packages from these technologies as well. To test the application, we have used a combination of iOS and Android emulators and physical devices.

## 7.0 References

- [1] “Demographic Changes and Aging Population,” <https://www.ruralhealthinfo.org/toolkits/aging/1/demographics#:~:text=Between%202020%20and%202030%20alone,65%20years%20old%20and%20over.>
- [2] “Which US States Have the Oldest Populations?”, <https://www.prb.org/resources/which-us-states-are-the-oldest/>
- [example] Authors, “Article in Title Case,” Conference or Journal, Publisher, Year
- [3] “8 Ridesharing Services For Seniors”, DailyCaring, <https://dailycaring.com/8-ridesharing-services-for-seniors/>
- [4] GoGoGrandparent, <https://gogograndparent.com>
- [5] O’Brien, Sara, “Lyft Focusses On Seniors With New Option To Book Rides By Phone Call”, CNN Business, 2021, <https://www.cnn.com/2021/02/24/tech/lyft-call-a-ride/index.html>