**Activity: Creating a Ride**

Drivers can create a ride that will be posted to the server to appear on the Find Ride Activity. This Activity can be accessed by tapping “Create Ride” on the Main Activity (or Ride List).

This Activity is fairly complex. It uses a Geocoder to return the address and coordinates of the selected campus. It also tracks the user’s location while the application is active, because that information has to be sent in the request. The activity also uses Android’s Date and Time Pickers, which are managed by internal classes.

When implementing this class, I had to send several requests at once, unaware that requests to the Internet run on separate threads, so I encounted a problem where I could never retrieve a campus’s address in time when the createRide request was sent. To fix this, the Activity makes use of a couple of internal classes that extend the AsyncTask class, and force the requests to run one after another, placing a 2.5 second delay on both. As a result, creating a ride will always take five seconds to complete all requests. The app currently hangs while the requests are sent. Ideally, I would like to implement some form of loading screen in between requests.

**Manifest Details**

This activity is implemented with CreateCar.java in the Android Manifest. Like most of the Activities, its screenOrientation is restricted to “portrait.” This Activity inherits the app’s theme.

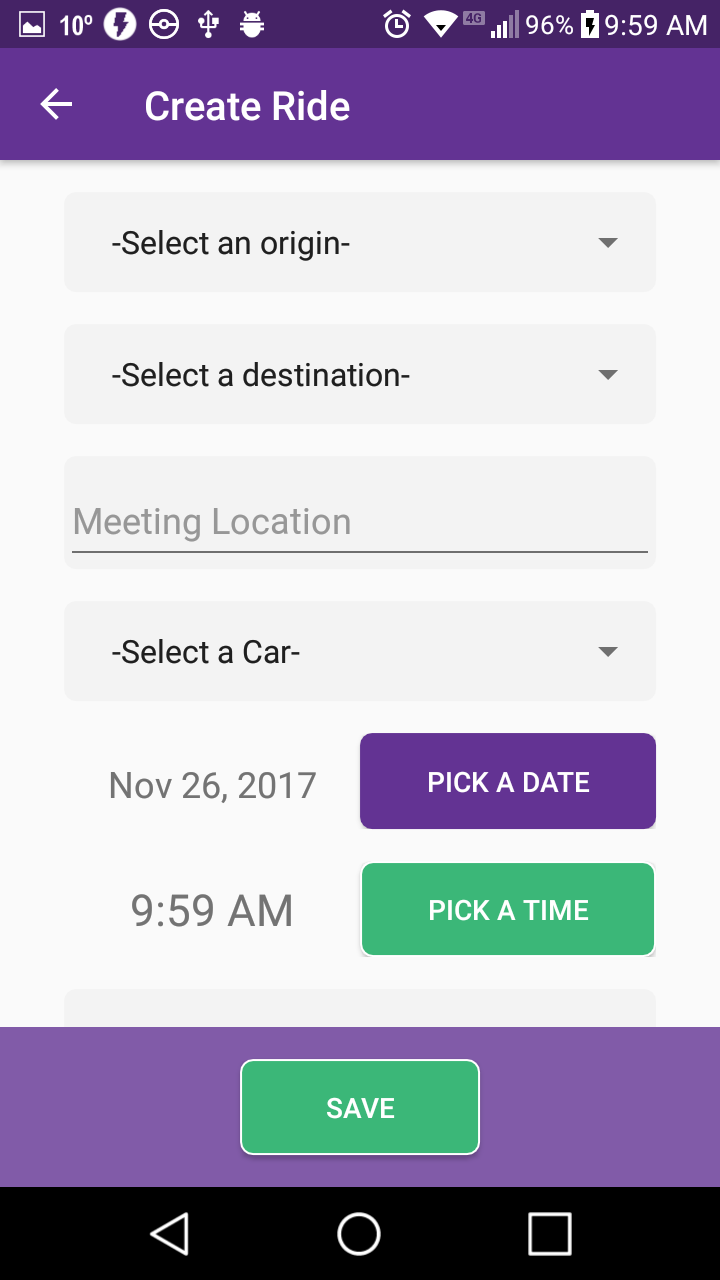
Android:windowSoftInputMode is used here, and is set to “adjustPan”. This prevents the createRideButton from being pushed up when the virtual keyboard appears, as it is anchored to the bottom of the screen.

The Intent filter provides standard access when accessed by name.

**Layout File**

Activity\_create\_ride.xml

* The sole layout file of the Create a Ride Activity, but there are a lot of interesting Views here.
* Currently, the TextInputLayouts (originLayout, destinationLayout, carLayout) are filled with Spinners, which are Views that display a drop-down list. If you look at the .xml’s text, you will see that next to each Spinner is a TextInputEditText that was commented out. At one point when the app was being developed for the Code of Atlanta competition, we needed the app to display addresses other than the campuses, which meant allowing the user to type in whatever address they want (“3333 road rd. City, State”, or “Walmart near Target Macon, GA,” etc.). The code to handle the Edit Texts still exists in the code behind, but is currently commented out.
* saveButton(Button)
  + calls the saveRide() method in the code behind, which begins the arduous process of creating a new ride…
* originInput(Spinner or TextInputEditText)
  + The InputEditText is commented out, but the Spinner retrieves a list of campuses from the campus\_coordinates.xml resource files, which the code behind will use to pull each campuses Address.
* destinationInput(Spinner or TextInputEditText)
  + The same as the originInput
* pickupInput(Spinner or TextInputEditText)
  + The same as the originInput
* carInput (Spinner)
  + When the Activity starts, a request is sent to retrieve the user’s cars, and they can be selected here.
* dateButton (Button)
  + onClick: showDatePickerDialog – Displays Android built-in date picker, that returns the selected date to the dateTextView.
* timeButton (Button)
  + onClick: showTimePickerDialog – Same as the dateButton, except it opens the time picker.
* ---repeatLayout(TextInputLayout)--- UNUSED
  + Commented in the .xml’s text. This was intended start a Service that would automatically create a recurring ride, but there’s currently no support for this.
* ---alertLayout(TextInputLayout)--- UNUSED
  + This would work with the recurring ride Service, sending an alert to the user when the ride is about to start again, but as with the repeat, there is no support for this.



**Class File**

CreateRide.java

* A cluster of code that probably tries to do too much at once. It uses a LOT of internal classes that might have been better off as their own java files, but I didn’t want to clutter the package anymore. I try to keep it fairly straightforward in organizing the classes and methods, but there’s probably a much better way this class could have been handled.
* Variables
  + –queue: RequestQueue – Sends JSON Requests to the server
  + –prefs: SharedPreferences – Retrieves the user’s stored ID from the phone
  + –userId: String
  + –originInput: TextInputEditText ---UNUSED---
  + –originSpinner: Spinner
  + –destinationInput: TextInputEditText ---UNUSED---
  + –destinationSpiner: Spinner
  + –pickupInput: TextInputEditText ---UNUSED---
  + –pickupSpinner: Spinner
  + –carSpinner: Spinner
  + –carIds: ArrayList<String> -- Associates car choices with the proper ID number.
  + –dateTextView: TextView – Displays the date from the date picker
  + –timeTextView: TextView – displays the time from the time picker
  + –seatsInput: TextInputEditText
  + –pickupArray: String[] – The names of the pickup locations
  + –coordinatesArray: String[] – The coordinates of the pickup locations
  + –originParams: HashMap<String, Object> -- Stores the origin’s address
  + –destinationParams: HashMap<String, Object> -- Store’s the destination’s address
  + –meetingParams:HashMap<String Object> -- UNUSED. Would store the meeting location’s address, as the meeting location is supposed to be picked from a dropdown menu, but there are currently no coordinates to use.
  + –userLat/userLng: double: - the user’s current coordinates as they create the ride
  + –TAG: String – the unique ID for the location tracker.
  + –mlocationManager: LocationManager – this will periodically track the user’s location
  + –LOCATION\_INTERVAL: int – The minimum time to pass before the app will update the user’s location
  + –LOCATION\_DISTANCE: float – the minimum distance that has to be traveled before the app will update the user’s location
  + mLocationListener: DriverLocationListener – The class actually records the user’s location
* Methods
  + \*onCreate(savedInstanceState): void
    - Displays the activity\_create\_ride layout, initializes the variables, start’s the locationManager to periodically get the user’s location, and creates a back button in the toolbar.
    - Sends method calls to fill in the Spinners, and to set the date and time TextViews’ texts.
  + +onOptionsItemSelected(item): Boolean
    - Returns the user to the previous Activity.
  + +saveRide(view): void:
    - Called from saveButton in the layout. It makes sure the user has actually selected a campus, selected a pickup location, and a car from the spinners, and then runs a validation check on the pickupInput and the seatsInput.
    - If the Activity passes validation, this method creates a GeocodeAddressTask and calls the execute method.
  + –setupSpinners(): void
    - This method is only used if the Spinners are used for selecting the origin, destination, and pickup location.
    - It creates two lists for the origin and destination, and uses a loop to copy the campus\_array values from the campus\_coordinates.xml file.
    - After the originSpinner’s values are set, an OnItemSelectedListener is created to display the possible pickup locations in the pickupArray, calling the setupPickupSpinner() method.
    - Next, sends a JSON Array GET Request to retrieve the user’s registered vehicles from the server, and stores them in a list.
    - All of the lists are then displayed on the appropriate spinners using an ArrayAdapter.
  + –setupPickupSpinner(): void
    - Creates an arraylist to hold the pickup locations, and retrieves them from the campus\_coordinates.xml file.
  + +showDatePickerDialog(view)
    - Called from dateButton in the layout. Displays Android’s built-in date picker.
  + +showTimePickerDialog(view)
    - Called from timeButton in the layout. Display’s Android’s built-in time picker.
  + –setDateTimeText(): void
    - This is run when the application is first created. It sets the date and time TextViews in the layout to display the current date and time when the Activity loads.
* Internal Classes (This gets its own category because there are **so many** in this file…)
  + –GeocodeAddressTask
    - Starts a new Thread to return the exact address and coordinates of the origin and destination.
    - Variables
      * -originStr: String – The origin address
      * -destinationStr: String – the destination’s address
      * -addressFound: Boolean – a flag to stop the process if an address is not found.
    - Methods
      * \*onPreExecute(): void
        + Get the address associated with the user’s selected campus,
        + Create a StringBuilder, and loop through the address to replace every empty character (‘ ‘) with a ‘+’ character. The address will be sent to Google Maps to pull the appropriate address and coordinates.
      * \*doInBackground(params): Void
        + Create the url using Google Map’s API, the origin’s address, and google maps key. Next, send a JSON Object GET request for the origin’s address. Store The Address, city, state, zip, latitude, and longitude within the originParams HashMap. If the address is not found, set addressFound to false.
        + Wait 2.5 seconds to allow the data to catch up.
        + Recreate the url using the API, the destination’s address, and Google Maps key. Send a JSON Object GET request for the destination’s address, and store the values within the destinationParams HashMap. If the address is not found, set addressFound to false.
        + Wait 2.5 seconds to allow the data to catch up.
      * \*onPostExecute(aVoid): void
        + If addressFound is false, the app displays an error message and ends the process.
        + Else, it creates an CreateRideTask and executes.
  + –CreateRideTask
    - Finally sends the actual request to create the new trip.
    - Variables
      * -df: DateFormat – Set Note
      * -dateTime: Date – See Note
      * -date: String – see Note
        + **Note:** These variables retrieve the date and time from the TextViews’ getText() methods. Neither the date nor time is sent to this class when the dialogs are closed.
      * -rideParams: HashMap<String, String>
        + All of the ride’s paramaters, even holding the values from originParams and destinationParams
    - Methods
      * \*onPreExecute(): void
        + Convert the date and time into an epoch and fill the rideParams HashMap will values from the originParams and destinationParams maps.
      * \*doInBackground(params): Void
        + Finally, sends rideParams in a JSON Object POST Request. If the request is successful, a Toast with a success message is made, and the Activity closes.
        + Otherwise, if the request fails, Display an error message and close the Activity
  + –DriverLocationListener
    - A class implementing the LocationListener interface to track the user’s location at LocationManager’s time interval. The interface demands that all the classes be implemented, but we only care about one of them.
    - +onLocationChanged(location): void
      * Set userLat and userLng to the user’s current latitude and longitude.

**Resource Files**

Campus\_coordinates.xml

* Holds string-arrays of each campus and their address, and also stores the pickup locations and their coordinates.

Google\_maps\_api.xml

* Contains the key to use Google Maps’ API calls.