# Ridesharing App Lesson Titles/Descriptions

01.) Intro to Course / App Walkthrough

In this lesson, Caleb introduces the course and gives a brief overview of the content, then gives a demonstration of the completed build of the ridesharing app.

02.) Xcode Project Creation / Setup of CocoaPods

In this lesson, you will create an Xcode project and a Podfile then configure it to install the CocoaPods required to complete this course.

03.) Building HomeVC’s User Interface

In this lesson, you will build the user interface for HomeVC – the most important ViewController in our app – the one that displays the MapView, allows you to search for locations, request trips, and more.

04.) Creating Custom View Subclasses for HomeVC

In this lesson, you will create several view subclasses that are required to customize the appearance of HomeVC. You will also write some functions within the subclasses that allow for engaging animations and user experiences.

05.) Creating a Sliding Tray Menu with ContainerVC

In this lesson, you will build a sliding tray-style menu which can be animated open and closed nicely upon the press of a button. This is a commonly used and very handy feature to have in our app.

06.) Creating a UIView Extension

In this lesson, you will create an extension of UIView and add a method which can perform a frequently used animation in our app. Later on in the course, you will add more functions to add greater functionality to our app.

07.) Building LoginVC’s User Interface

In this lesson, you will build the user interface for LoginVC, the screen which allows you to create and authenticate users – both passengers and drivers.

08.) Creating Custom View Subclasses for LoginVC / Adding to UIView Extension

In this lesson, you will create view subclasses for LoginVC to customize its appearance, then you will add functions to our UIView extension to add functionality.

09.) Adding RevealingSplashView

In this lesson, you will import the RevealingSplashView library, set up LaunchScreen.storyboard, and create/display an animated instance of RevealingSplashView.

10.) Creating Project in Firebase / Adding Firebase to Xcode Project

 In this lesson, you will login to the Firebase console using your Google account credentials, create a new project, download/import the configuration file into your Xcode project, and set up the project in the AppDelegate.

11.) Creating a DataService Singleton

In this lesson, you will create a singleton class called DataService which will allow for the accessing and modification of our Firebase database. We will use it to create and store driver and passenger accounts and eventually check if a driver or passenger are on a trip.

12.) Enabling Sign In and Authentication

In this lesson, you will enable email authentication in Firebase and set up the code necessary to log users in and create accounts.

13.) Configuring LeftSidePanelVC to Change Depending on User Account Type

In this lesson, you will display account type data for a user of a particular type. You will add several labels to help provide context and valuable account information to a user. If the user is a driver, they will have access to a toggle switch that allows them to turn on/off their “pickup mode” capabilities.

14.) Configuring Location Services and Permissions / Centering the MapView on User Location

In this lesson, you will enable location services and write a function which will enable you to center the map on the user location whenever the CenterMapButton is pressed.

15.) Creating an UpdateService Singleton

In this lesson, you will create another singleton class, but this one will be used to update data of different kinds throughout our app. We will use it to save the driver and passenger’s current GPS location in real-time to Firebase every time they move.

16.) Creating the DriverAnnotation Subclass / Configuring viewForAnnotation(\_:) from MKMapViewDelegate

In this lesson, you will create a subclass of MKAnnotation called DriverAnnotation and write the function necessary to visually display custom annotations for drivers on the MapView.

17.) Loading Instances of DriverAnnotation from Firebase for all Drivers

In this lesson, you will write a function used to display drivers on the map from GPS coordinates saved in Firebase. You will also enable the drivers to be independently animated as well as hidden on the MapView.

18.) UX Improvement 1: CenterMapButton Improvements

In this mini-lesson, you will improve the user experience of your app. You will enable the CenterMapButton to be faded out and hidden when tapped, then faded in and shown when the MapView has been moved by a user sliding on the screen.

19.) Setting up UITextFieldDelegate Delegate Methods

In this video, you will set up the UITextFieldDelegate protocols for using the UITextFields in our app. You will enable the return button to function when we want to search. You will then set up a nice animation to occur when selecting the destination text field. Finally, will then set up various actions to occur when we tap the cancel button in the UITextField.

20.) Searching for Nearby Locations using MKLocalSearch

In this video, you will write a function to harness the power of MKLocalSearch, allowing you to search for destinations nearby. We will set up our UITableView to show these results and perform actions if selected.

21.) UX Improvement 2: Animating and Hiding UITableView with a Downward Scroll

In this video, you will improve the user experience in HTCHHKR by allowing a user to dismiss the UITableView with a swipe down. You will write some conditional code so that the UITableView can only be swiped down when there are no results showing for the best possible experience for the user.

22.) Selecting A Search Result from the UITableView

In this video, you will write code which will control what happens when a search result is chosen from the UITableView. You will set up an instance of PassengerAnnotation to display on the current passenger’s location in MKMapView as well as push their destination coordinate up to Firebase.

23.) Dropping A Pin For The User’s Desired Destination

In this video, you will set up an instance of DestinationAnnotation, which will be displayed on the map at the GPS location of the destination the user selected from the UITableView search results.

24.) Adding an MKPolyline to MKMapView to Show Trip Route

In this video, you will write a function which will enable you to find a route that connects the PassengerAnnotation and DestinationAnnotation. It will be displayed on the map with custom styling (color, width, etc.) that you choose.

25.) UX Improvement 3: UIViewController Extension Loading View

In this video, you will create an extension of UIViewController and write a function that will allow you to create and fade in a semi-transparent black view with a spinner at it’s center. You will use this to indicate loading throughout our app so that a user is not unsure of what to do when data is loading.

26.) Removing Destination Coordinate from Firebase and Annotations/Overlays from MKMapView

In this video, you will add code within the textFieldshouldClear(:\_) delegate method to remove the ‘tripCoordinate’ child from our passenger’s Firebase account as well as remove all annotations and overlays shown on the map. We will simultaneously center the MKMapView on the passenger’s current location. This will allow for a selected destination to be changed without getting in the way of the user.

27.) Zooming in on MKMapView

In this video, you will write a function which will be used when multiple annotations are shown on the screen as well as a route line. It will center the MKMapView on the annotations for the passenger pickup point and the desired destination point. You will set up the Center Map button to work whether there is a destination showing or not.

28.) Showing Alerts on UIViewController with Alertable Protocol

In this video, you will write a protocol extension called Alertable which can be called on any instance of UIViewController. It will present a UIAlertViewController with a message you pass in. We will use this for error handling and as a way to interface with the user should there be errors in our app.

29.) Building PickupVC in Interface Builder / Setting up Custom View Subclasses

In this video, you will build PickupVC in Interface Builder – an instance of UIViewController which will be presented on driver accounts when a passenger requests a ride. After building the interface, you will write a custom subclass which will customize the appearance of an MKMapView.

30.) Allowing Passengers To Request Trips

In this video, you will write a function which will allow for a passenger to request a ride after pressing the action button. You will create an observer which for driver accounts which will monitor to see if a trip has been added in Firebase. If it has, the PickupVC will be presented on the driver account.

31.) Allowing Drivers to Accept Trips

In this video, you will enable drivers to accept trips once PickupVC is presented. Once accepted, the driver’s unique ID will be passed into the trip child for the requesting passenger. You will modify the driver’s status to be unavailable and also configure the trip so that it can be started once the passenger is picked up.

32.) Showing Route and Annotation to Pickup Passenger for Trip

In this video, you will set up the action for selecting the ‘Accept Trip’ button on PickupVC. You will allow for a new route and annotation to show up on the driver’s MKMapView which will route them to the passenger.

33.) Allowing a Passenger or Driver to Cancel Trip

In this video, you will allow for a passenger or driver to cancel the trip that they are on whether it is accepted or not. Once canceled, the MKMapView will zoom and center on the user’s location whether or not they are a passenger or driver.

34.) Modifying searchMapKitForRouteWithPolyline(\_:) Function to Show and Zoom Routes and Annotations for Trip

In this video, you will modify the searchMapKitForRouteWithPolyline(\_:) function so that it can zoom properly for both driver and passenger once a trip has been accepted. Since the route and annotations shown are removed and a new route is updated, you will allow the MKMapView to zoom to accomodate this.

35.) Fixing and Removing Duplicate Overlays on MKMapView

In this video, you will fix an issue where the searchMapKitForRouteWithPolyline(\_:) function is called twice – once before the route is removed to show a new pickup route, and once after it is removed – thus showing a leftover route when it should be removed. We will add a conditional check so that only a single route can be added at a time.

36.) Setting up CLRegion for Passenger and Destination

In this video, you will utilize the power of CLRegion so that when the driver gets close enough to the passenger, the button’s title will change to say ‘START TRIP’. Likewise, when the driver gets close enough to the destination while on an accepted trip, the button will say ‘END TRIP’.

37.) Making the Action Button ‘Smart’

In this video, you will set up a function which can perform various actions – like request ride, start trip, get directions, end trip, etc. – so that our button can have multiple functionalities in our app. We will set up the code to get directions to the passenger as well.

38.) Starting a Trip

In this video, you will enable a driver to start a trip as soon as they are within the pickup passenger’s region. Once the ‘START TRIP’ button has been pressed by a driver, the annotations and route are removed and a new one is added showing the route and annotation to the destination point.

39.) Finishing the App: Getting Directions with Apple Maps to the Destination / Ending a Trip

In this video, we will set up the action button to be able to get directions to the destination point and end the trip by removing the trip child from Firebase, removing annotations and overlays, and centering the map on the user location.

40.) Code Cleanup and Creation of Constants File

In this video, you will clean up code and remove unnecessary print statements you may have left in your code as you followed along. You will also create a Constants file which we will use instead of the many Strings utilized throughout this app in order to make our codebase less error prone.