Instructions

Add a Ride History List

Background

Our ride sharing app needs a way for users to check their ride history. This will allow them to view their previous ride along with the cost of the trip. You are tasked with adding this list to the app.

Here is your task

1. Open the `RideSharer.xcodeproj` file in the `RideSharer_History_Start` folder.

2. Add a table view to the `RideHistoryViewController` view controller and display the list of items in the `rideHistory` array in the table view. The data array is a collection of tuples, which each contain two values.

3. When a user taps on a row, display an alert box with an "OK" button that dismisses the alert. The alert title should say "Price", and the alert message should say the price for that ride.

   To detect when a user taps on a row, you will need to first make the `RideHistoryViewController` conform to the `UITableViewDelegate` protocol, like you did with the `UITableViewDataSource` protocol. Add `tableView.delegate = self` to the `viewDidLoad` method.

   Use the `didSelectRowAt` delegate method to display the alert.

4. Test the app by running it and going to the Ride History view. Make sure the list is populated and the alert displays when you tap on a row.

5. When you are finished, compress the entire project directory (including the xcodeproj file) to a zip file and submit it.

Set Up

Before you tackle this task, you will need to download and install Xcode from the Mac App store.

Resources

Swift tuples: https://docs.swift.org/swift-book/ReferenceManual/Types.html#ID448
Accessing values in a tuple: https://abhimuralidharan.medium.com/tuple-in-swift-a9ddeb314c79
Building a table view: https://programmingwithswift.com/create-a-uitableview-with-swift/
Detecting a row selection in a table view: https://developer.apple.com/documentation/uikit/uitableviewdelegate/1614877-tableview
Implementing an alert: https://stackoverflow.com/a/33340757/3060347.

Add a Map with User Location

Background

Our ride sharing app needs a view added where users can request a ride. The first step is to add a map which shows the user's location. Your task is to add a map to the app which shows the user's current location.

1. Open the `RideSharer.xcodeproj` file in the `RideSharer_Map_Start` folder.

2. Add a map to the view. Use constraints to make it cover the entire view, like you did with the ride history table view, and similarly create an IBOutlet for it.

3. Get the user's current location and display it on the map.

4. If the user denies location access, show an alert telling the user that they have not provided permission.

5. Run the app and ensure the location request is displaying. Allow the request and see that your location is showing on the map. Reset the location permissions by removing the app, then re-running it. This time, deny permission to track your location. Run the app again and see that the alert is displayed.

6. Create a git repository and upload your project to Github.

   If you haven't already, create an account at https://github.com. Download and open Github Desktop, and log in with your account.

   In Github Desktop, select File > Add Local Repository. Choose the directory of the project you've been building. You'll see a warning, since a git repository hasn't been created yet for the project. Click the "create a repository" text.

On the next screen enter a name for the repository, or keep the default. The other options you can ignore for now. Click Create Repository.

An initial commit should have been created automatically. You can now click Publish repository at the top of the window. This will push your code to Github.

In the Github Desktop top menu, select Repository > View on Github.

6. Submit the URL to your repository.

Set Up

Before you tackle this task, you will need to download and install Xcode from the Mac App store.

Resources

Adding a map with user's location: https://iosapptemplates.com/blog/swift-programming/mapkit-tutorial/
Swift switch statements: https://www.hackingwithswift.com/sixty/3/8/switch-statements

Glossary

UIKit: The Apple framework that provides elements the developer needs to display and work with iOS user interface elements.

View Controller: The "brain" of the current screen that handles displaying information and reacting to user actions, such as tapping on a button. The view controller contains views and control elements (such as buttons or sliders). All view controllers are subclassed from the UIViewController class contained in UIKit.

Delegate: An object that has functionality delegated to it. For example, with a table view in iOS, the functionality for what happens when a user taps on a row can be delegated to the view controller to handle.

Protocol: A set of requirements that a data type (such as a class) must conform to. The requirements are methods and/or properties that the type must have.