	Misc Notes:	- to give full screen feel - getsupportactionbar.hide() - in Check login, remove parse.getAutomaticUser() - use @string to be able to rename variables - resume at 21:38, 18:21, 31:50	change lastKnownLocation using Location Manager to mention "GPS_PROVIDER"	infoTextView shows "TextView" when opening app for first time. - How I got this: Brand new emulator	
		Change login tab to login screen w Username and Pwd. - It is easier to follow and debug this way. - Using the 'back' button and 'logout' button cause problems otherwise.	Added username to the ListView in line 92 in ViewRequestsActivity.java	Problem occurs because of same ID (name) when loggin in as user and student - Logging out logs out the user, but we cannot log back in as that user to see in the app	
		Excessive logging problem caused by AndroidX migration. - Refactor this -> Migrate to AndroidX - Clicking that shows error that 'you need to have compile SDK set to atleast 28 in module build.gradle to migrate to AndroidX - Migration successful. Did not get rid of the Lsun/misc/unsafe loggings	Another error. When logging in as user, and then logging in as student, the 'user' tuple written in the database gets overwritten with a 'student' tag	Implemented a login page for username and password - added feature for keyboard to go away when clicking anywhere else -- added exception for when getFocus() is null, i.e. nothing is focused	
		Change 'user' to 'customer' or 'consumer' everywhere - Ie_customer, Ie_student?			

File modified/created	Task notes	Main task goal	Steps for the task	Test cases pertinent to the task.	Errors encountered, severity.
Main Activity Page		**Create activity** page for the main front page	default creation by Android Studio		Errors for .xml files. Incomplete xml. Just close the project, delete the project. Empty the "caches" folder in "C:\Users\<Username>\.AndroidStudio3.5\system" And create the project again. It should build fully without any errors. The colors.xml file should have the <resources/> tag and not the <project/> tag
Main Activity Page		Add Logo	Get a vector, png		File name should only have lower case letters or numbers, or an underscore. (a-Z), (0-9), '_'
			Add to 'drawable' folder in 'res' (resources folder in AS)		
			Add in the corresponding Activity layout		
			Modify layout constraints as needed		
Main Activity Page		Add a switch	Add a switch widget from list Remove text for switch Add unique id for switch		
			Add TextViews to left and right to show options		
			Equalize margins for TextView using XML text view instead of designer		
Main Activity Page		Add start button	Link button with its onClick(): - Add onclick function in corresponding Activity page - create a boolean for type of user -- getCurrentUser.put() to add the type of user to Parse -- getCurrentUser.saveInBackground() with a new SaveCallBack --- call the redirect() function inside the callback.	test_1: start button name matches correct function in main activity test_2: log switch value to console to check binary switch status test_3: Check if correct userType is added to Parse data tuple based on switch status	
Main Activity Page	Check login For anonymous login we do not send any details. Login is similar to signup Still save details of user in app, such as rider or driver, last booking done	Login user	Must be done during creation, i.e. app start: Get current user to check if user is already logged in. 1. If logged in: + test_4 -- redirect as rider or driver. First use only **Log** to print to console that you are redirecting 2. If not logged: + test_3 -- use .logIn() in ParseAnonymousUtils to add a new LogInCallBack	test_3: check if user is not logged in for 'null' test_4: check if the value of the boolean that designates the type of user is not null	
				test_5: check test 1-4 Clicking the 'start' button will display log on console as 'redirecting as customer' or 'redirecting as student' based on the status of the switch	
Rider Activity Page		**Create activity** for Rider	- Will be a google maps activity - Setup google maps API -- Go to URL > Select/Create Project > Create API Key > Paste it in google_maps_api.xml		

Main Activity Page		Create the redirect function	Will redirect to rider or driver activity page: **rider** - Create a new Intent and use it to redirect to the Rider activity page - Call the redirect() in the onClick() for "Add start button" - Call the redirect() in "Login user" when user is already logged in.	test_6: check if the user type is "rider" or "driver" using '.equals("rider)' test_7: make sure to type 'startActivity(intent_name) after creating intent	
Rider Activity Page		Create an update map function	- get latitudes and longitudes for current user location and store it as a LatLng variable - clear existing markers - move the camera to current user location using CameraUpdateFactory - add marker on the map on users current location		
Rider Activity Page		Setup Google Maps to show user's location	Declare LocationManager and LocationListener		
		"	- Initialize locationManager & locationListener in the onMapReady() - locationManager will get LOCATION_SERVICE using getSystemService() - locationListener will be initialized to a new LocationListener() -- In the onLocationChanged(): --- call the updateMap() - use requestLocationUpdates on locationManager		
		"	Check for permissions: - test_8a: -- if permission is not granted: --- request for permissions using ActivityCompat.requestPermissions. -- if permission is already granted: --- requestLocationUpdates using locationManager --- get lastKnownLocation using locationManager --- test_9 ---- updateMap with the lastKnownLocation	test_8a: check if permission is **NOT** granted using contextCompat.checkSelfPermission with PackageManager.PERMISSION_GRANTED test_8b: check if permission **IS** granted for this Activity for ACCESS_FINE_LOCATION using contextCompat.checkSelfPermission with PackageManager.PERMISSION_GRANTED test_9: check if lastKnownLocation is not null	
Rider Activity Page		Add method when user has given us permission	- autocreate onRequestPermissionResult - test_10 -- test_11 --- test_8b ---- requestLocationUpdates using locationManager ---- get lastKnownLocation using locationManager ---- updateMAp with lastKnownLocation	test_10: check if requestCode is 1 test_11: check if grantResults.length > 0 and that permission has been granted. test_12: check in the AndroidManifest.xml file if we ask for permission for ACCESS_FINE_LOCATION	
				test_13: test 6-12 When a user is already logged in and clicks the start button, it will open a google maps image with the user's location in the center, with a marker depicting user location	
				test_14: test using a Log.i. If logged in, the app should automatically show map marker	
Rider Layout Page (.xml)		Add "Call Cab" button	Add button to RiderActivity page - Give unique id and create onClick()		
Rider Activity Page	This is before checking if there is a request already active. Related to C64	Add callCab()	This will basically be the creation of a new ParseObject (i.e. request tuple) for the new request - test_8b -- requestLocationUpdates using locationManager -- get lastKnownLocation using locationManager -- test_16 --- create a new ParseObject for the "Request" --- put() currentUserName on the created object --- create a ParseGeoPoint and store the last known location data in it --- save the ParseGeoPoint in the "Request" object created earlier and save it to Parse ---- After Add "Call Cab" button (Activity) ---- Change the text in the call student button from "Call Cab" to "Cancel Cab" -- test_17 --- display toast saying location could not be found	test_15: test the "call cab' button by using **Log**. test_16: check if lastKnownLocation is not null test_18: check if "Request" object is created in Parse test_19: check if the created "Request" object has a location field (as a ParseGeoPoint) and a username	
Rider Activity Page	This should be before above entry	Add "Call Cab" button (Activity)	- Declare the button in Rider Activity Page - Initialize in onCreate()	test_17: check if the name of the button matches it's id in the .xml	
Rider Activity Page	This is in addition to the code in C60	Create state variable for active request	- Declare Boolean in Rider Activity - check if active "Request" is in place in the onCreate() -- Create a ParseQuery with exact name as the "Request" object created in Parse -- Use the whereEqualTo() to check for the same "username" -- Using the findInBackground() set the status of the active request Boolean and the text of the "Call Cab" button --- In the new findCallBack: ---- test ---- test ----- set the state variable for active request to true, stating there is already a request in progress ----- set the text of the "Call Cab" button to "Cancel Booking"	test_20: check if there are no errors test_21: check if there are more than 0 objects from the whereEqualTo query	
Rider Activity Page		" C64(Create state variable for active request), C62	- In the "Call Cab" button set the active request state variable to true - before creating the request, check if an active request already exists: -- If it does not exist, **C57** will be followed -- If it does exist, then upon clicking the button, the "Cancel Cab" will change back to the "Call Cab" --- follows same procedure as **C62**. ---- query if a "Request" object exists on Parse with the current username. ---- after performing test_20 and test_21: ----- loop through all the objects and delete objects in the background ----- set the state variable for active request to false ----- set the Text of the "Cancel Cab" button back to "Call Cab"	test_22: check if objects were deleted in Parse test_23: check if state variable was changed back to false test_24: check if the "Cancel Cab" button was changed back to "Call Cab"	Using findInBackground() from a fragment can cause an error and crash the app. Use query.find() to solve the problem.

				test_25: test 13-24. When a user is already logged in, it should redirect acoordingly. If a request doesnt exist, it should be created and stored in Parse, and the button in Customer Activity should reflect the change. If a request already exists, the user should be able to cancel it. The corresponding request object in Parse should then be deleted and the UI for the button in the Customer Activity should reflect the change.	
ViewRequestsActivity Page		**Create activity** to view nearby Drivers for booking	Empty Activity Create a list of requests that constitute nearby Drivers available. Show distance of each Driver from the customer in the list		
Rider Activity Page (.xml and .java)		Add a logout button	- Add a logout button - Add the logout() that logs the current user out and redirects to the Main Activity page -- ParseUser.logout() -- Create Intent for Main Activity and redirect	test_26: check if logout button onclick matches in the activity .java page test_27: check if the user was actually logged out test_28: check if Intent redirects correctly to main activity	Two problems occur with the logout() button: - It logs out the currentUser, and since we are using ParseAutomaticUser, we cannot get that user back. - It redirects to MainActivity, but the main activity immediately redirects back to the Rider Activity since it checks for the redirection in its onCreate. - To solve the first issue, only implement the logout() once an actual login is put in place. - To solve the second issue, refer to C75(Add a home page button)
Rider Activity Page (.xml and .java)		Add a home page button	- Add a "Home" button - Add the home() that takes user back to main activity page - **Refer to F75** to implement temporary fix for immediate redirection		To solve the immediate redirection from Main Activity() remove checking if the current user already has a "riderOrDriver" field in the onCreate(). Instead move it to the startButton() method, so the redirection happens only when the button is clicked.
Main Activity Page		Add redirect functionality to View Requests Activity Page	- Same block as **C42** - Will redirect to rider or driver activity page: **driver** - Create new Intent to redirect to View Activity page.	test_29: check if intent redirects to correct activity page test_30: check if intent was passed to startActivity();	
build.gradle (Module)		Add RecyclerView dependency	- implementation 'androidx.recyclerview:recyclerview:1.0.0'		
View Requests Activity Layout page (.xml)		Add a ConstraintLayout and RecyclerView	- Add a RecyclerView inside a ConstraintLayout - Fix constraints - Give unique id, i.e. requestRecyclerView, viewRequestsLayout		
Requet List Layout file (.xml) [Singular layout file] [Named requests_list.xml]		Layout file for a single RecyclerView item, i.e. each row in the RecyclerView	- Add a LinearLayout and give unique id. - Add a TextView inside the LinearLayout and give it a unique id, e.g. ***requestTextView*** -- Make sure the 'layout_height' parameter of the LinearLayout is set to "wrap_content" and not match_parent -- Keep the 'layout_height' parameter of TextView as "wrap_content" as well		ERROR HERE
View Requests Activity page		Create a custom data class, 'RequestDataClass'	- Declare it before ViewRequestsActivity class. - will be a data class, and will have one String parameter, e.g.val ***requestTitle***: String		
View Requests Activity Page		GPS related code will be similar to **C50-51**(Setup Google Maps to show user's location)	Declare LocationManager and LocationListener		
		" [updateListView() will be defined ahead]	- Initialize locationManager & locationListener in the **onCreate**() - locationManager will get LOCATION_SERVICE using getSystemService() - locationListener will be initialized to a new LocationListener() -- In the onLocationChanged(): --- call the **updateListView()** (This function will be defined ahead) - use requestLocationUpdates on locationManager		
		" [updateListView() will be defined ahead]	Check for permissions: - test_35a: -- if permission is not granted: --- request for permissions using ActivityCompat.requestPermissions - test_35b -- if permission is already granted: --- requestLocationUpdates using locationManager --- get lastKnownLocation using locationManager --- test_37 ---- **updateListView()** with the lastKnownLocation	test_35a: check if permission is **NOT** granted using contextCompat.checkSelfPermission with PackageManager.PERMISSION_GRANTED test_36b: check if permission **IS** granted for this Activity for ACCESS_FINE_LOCATION using contextCompat.checkSelfPermission with PackageManager.PERMISSION_GRANTED test_37: check if lastKnownLocation is not null	
		" onRequestPermissions() similar to **C53**(Add method when user has given us permission) [***updateListView***() will be defined ahead]	- autocreate onRequestPermissionResult - test_38 -- test_39 --- test_35b ---- requestLocationUpdates using locationManager ---- get lastKnownLocation using locationManager ---- ***updateListView*** with lastKnownLocation	test_38: check if requestCode is 1 test_39: check if grantResults.length > 0 and that permission has been granted. test_40: check in the AndroidManifest.xml file if we ask for permission for ACCESS_FINE_LOCATION	
RequestAdapter.kt		Create a RequestAdapter kotlin class This will also have the custom ViewHolder	- create a class ***RequestAdapter*** with one private parameter that will be a MutableList of the custom data class created earlier -- e.g. private val items: MutableList<***RequestDataClass***> - This ***RequestAdapter*** class will extend RecyclerView.Adapter<***MyViewHolder***> -- ***MyViewHolder*** is the name of the custom ViewHolder class that will be created ahead		

RequestAdapter.kt [2nd class in this file] [Can be refactored to own file if needed]		Create the custom ViewHolder class, ***MyViewHolder***	- Create another class, ***MyViewHolder*** with two parameters, a LayoutInflater and a ViewGroup -- inflater: LayoutInflater, parent: ViewGroup - This class will extend RecyclerView.ViewHolder with one parameter that basically calls inflater.inflate on the layout file created for RecyclerView item [***requests_list***.xml] -- inflater.inflate(R.layout.***request_list***, parent, false - create a private var of type TextView; var ***requestItemTextView***: TextView - in the initialize function of the class (init{}), assign the ***requestTextView*** from D83, i.e. the name of the TextView from the layout file for each row of RecyclerView to this class variable; ***requestItemTextView*** = itemView.***requestTextView*** - ***requestTextView*** has to be accessed from the default variable '**itemView**' in theRecyclerView class . We get access to this since we extend RecyclerView.ViewHolder		
RequestAdapter.kt [2nd class in this file] [Can be refactored to own file if needed]		contd...	- create a function 'bind()' that takes one parameter of type ***RequestDataClass***, e.g. requestVar -- assign the ***requestTitle***, which was a variable from the data class to the text field of ***requestVar***. --- requestVar?.text= = requestVar.requestTitle		
RequestAdapter.kt		Continue C87, the RequestAdapter class	Three functions need to be overriden. - function onCreateViewHolder() that takes two parameters of type ViewGroup and Int and returns an object of the custom ViewHolder class (***MyViewHolder***) -- e.g. (parent: ViewHolder, viewType: Int) -- This is where we inflate the layout conatining the RecyclerView --- use LayoutInflater to get context from the parent ---> e.g. val rootView = LayoutInflater.from(parent.context) --- then return MyViewHolder(rootView, parent) - function getItemCount() that takes nothing and returns an int -- return ***items***.size, where 'items' is the name of the MutableList<***RequestDataClass***>		
RequestAdapter.kt		contd..	- function onBindViewHolder() that takes two parameters of type MyViewHolder, and int -- e.g. (holder: MyViewHolder, position: Int) -- create a variable requestVar of type RequestDataClass that stores the element in 'items' at index 'position' --- var requestVar: RequestDataClass = items[position] -- bind that variable to the holder --- holder.bind(requestVar)		
View Requests Activity Page		Declare and initialize the ***RequestDataClass*** object Assign an adapter and layout to the RecyclerView in the xml	- Create a global MutableList<***RequestDataClass***> object, ***requestDataObject***. In the onCreate() - hide the action bar by using supportAction?.hide() - Assign a LinearLayoutManager with 'this' as the context to ***requestsRecyclerView.layout***, which is the unique id of the RecyclerView in the xml - Assign an object of type ***RequestAdapter*** with ***requestDataObject*** as its parameter to ***requestsRecyclerView.adapter***, which is the unique id of the RecyclerView in the xml -- requestsRecyclerView.layout = LinearLayoutManager(this) -- requestsRecyclerView.adapter = RequestAdapter(requestDataObject)		
View Requests Activity Page	This is replaced	Create the updateRequestListView(Location)	Function to update the RecyclerView with active requests - test_41 --Create a ParseQuery to check nearby requests arranged according to distance and update the ListView --- create a ParseQuery that .getQuery() on the "Request" ParseObject in Parse, e.g. nearbyObjectsQuery --- create a ParseGeoPoint() variable that gets it latitude and longitude from the passed-in '***location***' parameter to updateRequestListView(***location***) --- use a .whereNear() on ***nearbyObjectsQuery*** between "location" and the created ParseGeoPoint variable. This shows requests closest to the current location --- add another constraint using .whereDoesNotExist() using "driverUsername". This basically only lists requests that do not have a driver already assigned to them --- set nearbyObjectsQuery.limit = 5 to only show 5 requests closest to the ParseGeoPoint variable created earlier	test_41: check if the passed location parameter is not null	
		"	contd. - use findInBackground() on the ParseQuery (Modify for Kotlin) This is at the same level as earlier line. i.e. 3 dashes -- override the default function ***done()*** with a List<ParseObject>, and ParseException? as parameters. i.e ***objects***, and ***e***. -- test_42 --- test_43 ---- clear any earlier requests (i.e. the ArrayList<String>) ---- for each object returned add it to the list ----- store the "location" for the current object in the for loop as a ParseGeoPoint? ------ test_44 ------- get the distance for the current object in the desired quantity; e.g. miles, kilometers. ------- store the distance as a double/float ------- use ***requestDataObject***.add() to add an object of ***RequestDataClass*** with a String as its parameter. The String in this case would be the distance obtained earlier, maybe along with the username. This is basically the String that will be displayed in the **TextView** in the **request_list**.xml	test_42: check if there are no errors test_43: check if there are more than 0 objects for the returned using **isNotEmpty()** whereNear() ParseQuery test_44: check if the stored current object is not null	- Clear the RequestDataClass object (requestsDataObject) after checking if there are any objects. Clearing it before causes the list shown in the UI to duplicate temporarily (for fraction of a second). Just a bug. -- The UI probably adds the new updated list before the old requests list is cleared. - The first item in RecyclerView does not appear
		"	--- test_45 ---- using ***requestDataObject***.add(), add an object of type ***RequestDataClass*** with input -- notifyDataSetChanged() on the arrayAdapter to update the it	test_45: there are no objects to list	
				test_-:Choosing driver option in the home page switch should redirect to ViewRequestsActivity page. This page will show the list of requests arranged by nearby requests.	Error. The first request element in the list will be slightly obscured by top of the page.
app Module build.gradle		Add dependencies for CoordinatorLayout and AppBarLayout	- add dependency androidx.coordinatorlayout:coordinatorlayout:1.1.0 - add dependency com.google.android.material:material:1.1.0 for AppBarLayout		ConstraintLayout w RecyclerView causes first element to be partially obscured. Using a coordinatorlayout solves that problem

View Requests Activity layout page (.xml)		Add the coordinatorlayout and appbarlayout to the xml	Change the ConstraintLayout to a CoordinatorLayout in the xml. - The RecyclerView will be a child of this CoordinatorLayout		
View Requests Activity Page		Add an in progress status while getting the list of requests	In the onCreate() after initializing the requests ListView: -- clear() the ***requestDataObject***. -- add a ***RequestDataClass*** with String "Getting Drivers nearby..." to the ***requestDataObject***		
Parse Dashboard at the server address		Add additional "Requests" with latitude and longitude manually	Make sure the Amazon AWS EC2 Instance (or the cloud service being used) is running: - Check if the server address of the instance matches the one in the Android Project. Log onto the parse dashboard: - Go to the "Request" objects. - Add addtional requests with latitude and longitudes		
				test_46: test_26-45 When logged in as a Driver, on we should be able to see the list of active requests with distances.	
Driver Location Activity Page		**Create activity** for Driver	- Will be a google maps activity The following will be already setup from creating the Customer Activity (**C39**) - Setup google maps API -- Go to URL > Select/Create Project > Create API Key > Paste it in google_maps_api.xml - Will be redirected here from the ViewRequests Activity page, when one of the items in the requests list is clicked -- For that we will pass the Customer's location and the Driver's location to the DriverLocation Activity Page		
FILL VIEWHOLDER ONCLICK DETAILS HERE	ViewHolder click working implementation files	FILL VIEWHOLDER ONCLICK DETAILS HERE	ONCLICK using ViewHolder	FILL VIEWHOLDER ONCLICK DETAILS HERE	
FILL DATABINDING DETAILS HERE	FILL DATABINDING DETAILS HERE	FILL DATABINDING DETAILS HERE	Databinding layout page changes (.xml) onCreate() changes (.kt)	FILL DATABINDING DETAILS HERE	
FILL ONCLICK DETAILS HERE	FILL ONCLICK DETAILS HERE	FILL ONCLICK DETAILS HERE	OnCLICK using Binding Adapter, can be called from activity	FILL ONCLICK DETAILS HERE	
View Requests Activity Page	CHANGE THIS	Create ArrayList<Double> to store and then pass the latitude and longitude of the request	In the class: - Declare two ArrayList<Double> for storing the request Latitudes and request Longitudes - In **C89/D89** after adding the storing the distance as a double: -- add the latitude from the current request object to the just created requestLatitude ArrayList<Double> -- do the same for longitude.	test_47: Make sure the usage of Double or Float is consistent throughout the program.	
View Requests Activity Page	CHANGE THIS	Pass the Student and Customer Location through Intent	In the onCreate(): - create a setOnItemClickListener on the requests ListView with a AdapterView.onItemClickListener as a parameter -- Let it autocomplete and create a onItemClick(..., int i, ...); where **i** is the number of the row that was clicked. So if the first row was pressed, then i would be a 0. --- in the onItemClick(..., int i, ...): --- test_48 ---- get lastKnownLocation of the current Student using locationManager --- test_49 && test_50 && test_51 ---- create a new Intent that redirects to the StudentLocation Activity Page ---- add the following to the created intent using intent.putExtra() ----- the request Latitude, request Longitude, student Latitude, student Longitude ------ to add the request Latitude and Longitude, use .get(i) on the created request Latitude ArrayLIst<Double>. where i refers to the row of the item (i.e. the request) that was clicked. ------ to add the student latitude and longitude, do getLatitude() and getLongitude() on the lastKnownLocation	test_48: check if permission IS granted for this Activity for ACCESS_FINE_LOCATION using contextCompat.checkSelfPermission with PackageManager.PERMISSION_GRANTED test_49: check if the .size() of the requests Latitude is greater than **i**. Lowest value for i is 0 for the first row being clicked, so the size of requests must be atleast 1. test_50: same as test_48 but for the request Longitude test_51: check if lastKnownLocation is not null	
FILL ONCLICK DETAILS HERE	FILL ONCLICK DETAILS HERE	FILL ONCLICK DETAILS HERE	ONCLICK BINDING ALSO VIEWHOLDER ONCLICK	FILL ONCLICK DETAILS HERE	
Student Location Acitivity Page		Get extras from the View Requests Activity and display marker	- Declare an intent globally for the class In the onMapReady(): - Initialize the created intent for the Class with getIntent() - Create a new LatLng variable and initialize it with the Students Latitude and Longitude. -- This is done using intent.getDoubleExtraa("<Name of variable>"). Here the name to input is the name given while adding extras using the.putExtra() in **C101** -- Create and display a marker using this LatLng.	test_52: check if the name passed in putExtra and getDoubleExtra matches	

				test_53: test_47-test_52 Clicking on a reuqest from the requests ListView should open the Student Location Activity Page and show a **singular** marker of the Student (as done in **C103**)	
Student Location Activity Page		Create and display markers for Customer and Student	In the onMapReady(): - Remove the LatLng variables created for Student Latitude and Longitude in **C103** - Create a LatLng for customerLocation using intent.getDoubleExtra("<Name of Latitude/Longitude in ViewRequestsPage>") - Create a LatLng for studentLocation using intent.getDoubleExtra("<Name of Latitude/Longitude in ViewRequestsPage>") - Create a marker ArrayList<Marker> - Add the marker for the customerLocation in the created ArrayList<Marker> - Add the marker for the studentLocation in the created ArrayList<Marker> - Create a LatLngBounds.Builder object. - Loop through all Markers: -- Include the position of each marker in the LatLngBounds.Builder object: builder. include(marker.getPosition());	test_54: check if name of the customer Latitude is same as the one in passed from the View Requests Page test_55: check if name of the customer Longitude is same as the one in passed from the View Requests Page test_56: check tif the marker ArrayList<Marker> has been initialized test_57: check if the LatLngBounds.Builder has been initialized	
		contd.	- Create a LatLngBounds and initialize it using the earlier LatLngBounds.Builder object: builder.build(); - Get the width of current display using getDisplayMetrics() and store it as an int - Get the height of current display using getDisplayMetrics() and store it as an int - Create an int, 'padding', and initialize it as 20% of the height after casting it as an int - Create a CameraUpdate object, cu, using CameraUpdateFactory using the newLatLngBounds(bounds, width, height, padding) - Call animateCamera(cu) on the GoogleMap object		
Student Location Layout Page (.xml)		Create UI for a 'Confirm Booking' button	- Add a 'Confirm Booking' button -- Add the constraints for it and give it a onClick() name		It is possible the MapFragment in the layout will not be under any parent Layout (such as Relative Layout, or Constraint Layout) - To fix it simply copy the Relative/Constraint Layout code from another xml and paste it in the text view of the required xml - **MAKE SURE** to change the 'tools:context' field in the xml to the new Layout name and not the old one
View Requests Activity Page		Also pass in "username" using putExtra() like in **C99** & **C101**	- Declare and initialize an ArrayList<String> for usernames. - Similar to **C99**: -- add the "username" field from the current 'object' in the nearbyObjectsQuery to the created ArrayList<String> for usernames. - Similar to **C101**: -- use intent.putExtra() to add the "i"th value from the created usernames ArrayList<String>. -- Also add in test_58	test_58: check if usernames.size() is is greater than *i*; where the "i(+ 1)"th row is clicked by the user. Add this test to test_49-50-51	
Student Location Activity Page		Create functionality for ConfirmBooking. We must make sure Parse knows the Request has been accepted so no one else accepts the Request	In the confirmBooking() - Create a ParseQuery on the "Request" object in Parse - Set a whereEqualTo constraint on "username" and compare it with the intent. getStringExtra("username") - use findInBackground on the query using a FindCallBack<ParseObject> -- test_59 --- test_60 ---- Loop over all the objects: for each object: ----- use object.put() to create and put a new field "studentUsername" to link that Customer with the appropriate student. ------ Get the current username of the student using getCurrentUser().getUsername() ----- saveInBackground using a SaveCallBack: ------ test_61 ------- create and locally store the student Latitude, student Longitude, request Latitude (Customer), request Longitude (Customer) using intent.getDoubleExtra() ------- Create a New Intent called directionsIntent to open up the directions page -------- initialize it with: use android.content.Intent.ACTION_VIEW as the activity. -------- Uri.parse("http://maps.google.com/maps?saddr = studentLatitude, studentLongitude &daddr=requestLatitude, requestLongitude")	test_59: if there are no errors, i.e e == null test_60: if the size() of objects is greater than zero test_61: if there are no errors, i.e. e == null	Error will occur saying the map has not been laid out yet, and we are trying to access it before then. So we do **C116** using the OnGlobalLayoutListener()
Student Location Activity Page		Fix **F114**	In the onMapReady(): - We will be moving entire **C107** & **C108** into a new function. -- give a unique id to the parent Layout of Student Location Activity Layout page (The xml file) -- create an object, 'mapLayout', of type corresponding to the parent layout of the Student Location Activity Page (for e.g. ConstraintLayout) -- e.g. ConstraintLayout mapLayout = (ConstraintLayout)findViewbyId(R.id. **uniqueIDOfStudentLocationLayout** -- Add a 'addonGlobalLayoutListener with a CallBack for the same --- mapLayout .getViewTreeObserver() .addOnGlobalLayoutListener(new ViewTreeObserver .OnGlobalLayoutListener()) ---- Move **entire C107** & **C108** into this function. - the intent should be initialized outside of this function	test_62: check to make sure the intent is initialized before the mapLayout creation (and by extension before the onGlobalLayoutListener()	
Student Location Activity		Change marker colours for the Student and Customer marker	Use BitmapDescriptorFactory to choose a colour (Hue)		
View Requests Activity Page		Remove the accepted request from the list of requests displayed from View Requests Activity	In updateListView() - After the whereNearBy query in **C88** -- Add a whereDoesNotExist("<Name from **C114** = studentUsername>") after it. This will filter out to only display Request objects that do not have a Student already assigned to them, in the requests ListView	test_63: check that the string name in the whereDoesNotExist() matches the string id for the Student Username assigned in **C114**	
				test_64: test_54-63 Clicking on a request should open up a google maps activity showing the two markers (Student and Customer) on the screen. Clicking 'Confirm Booking' will ask if we want to open the Google Maps app (or any similar app, if it exists). The directions between the Student and Customer will be shown in the just opened Google Maps app.	

Google Maps Activity suddenly shows blank screen.	No errors anywhere	Added the following to AndroidManifest.xml: <meta-data android:name="com.google.android.gms.version" android:value=" @integer/google_play_services_version" /> - This causes a E/Google Maps android API: Authorization Failure	It is an authorization/authentication failure for the API key.		
build.gradle file(Module)			add this to dependencies - compile 'com.parse:parse-android:1.15.7' - compile is deprecated. Use implementation and update this file to the new way it is done - sync gradle		
ParseSetup.kt		Setup Parse	- Created Amazon AWS instance - Get serverID, clientKey, appKey - Create a new Kotlin class (in this case ParseSetup) - import android.app.Application and com.parse.Parse - create the class and its default onCreate - Add Parse.initialize(Parse.Configuration.Builder(this) - using the dot operator '.' add applicationID, clientKey, server, and build() - create a sample parse object and then test to see if it reflects in the Parse Backend Dashboard		Error Parse.ParseRequest failed.
AndroidManifest.xml		Add ParseSetup.kt	Add .ParseSetup to its <application> tag in the Android Manifest Also add AppTheme to the already existing MainActivity class inside its <activity> tag		- Error about AppTheme. After adding ParseSetup to <application> we also have to add the right "AppTheme" to the MainActivity inside its <activity> tag