Code Exercise: Writing Espresso Tests					Mobile Development Frameworks III
For this assignment, you will be writing tests for a simple CRUD app.					
The storage component of the app should be unit tested to ensure functionality.					
The UI and intents portions of the app should be tested using Espresso and the UiAutomator.					
Code Structure and Efficiency	**10%**	**Excellent (100%)**	**Good (80%)**	**Fair (45%)**	**Poor (0%)**
This is a measure of how well your code is structured and how efficiently it runs. It is not expected that you will get a 100% in this category at all times. If you read your feedback thoroughly and apply it after each submission, your grade in this category will improve over time.		App meets all "Good" requirements.	Meets all positive "Fair" requirements.	Does not suffer from any negatives listed in the "Poor" column.	Application is missing required elements.
		Application contains proper separation of fragment and activity code and uses interfaces to communicate between the fragment and activity.	Does not suffer from any negatives listed in the "Fair" or "Poor" columns.	Application does not use fragments for all screens.	Application is non-functional.
		Application contains no efficiency issues.	Application uses fragments for all screens.	Application contains more than minor efficiency issues.	
Storage Tests	**20%**	**Excellent (100%)**	**Good (80%)**	**Fair (45%)**	**Poor (0%)**
The storage utility class in the starter project should be unit/integration tested to ensure storage works properly. Be sure to mock all relevant code components as to not alter production storage in anyway. Separate test methods should be written for saving, loading, and deleting data.		App meets all "Good" requirements.	App meets all positive "Fair" requirements.	Does not suffer from any negatives listed in the "Poor" column.	Code is not unit tested.
		Unit tests are contained in their own class that is properly named for the class that is being tested.	App does not suffer from any negatives listed in the Fair or Poor columns.	Unit tests are present in the proper project used for instrumented tests.	
		Unit tests are named to be specific as to what they're testing.	All classes are properly mocked so that no changes are made to the deployed application.	One or more unit tests fail or do not account for the given test cases.	
			All unit tests pass when run.	Objects are not properly mocked resulting in changes to the deployed application.	
Espresso Tests	**35%**	**Excellent (100%)**	**Good (80%)**	**Fair (45%)**	**Poor (0%)**
Espresso tests should be written to test each activity individually as well as testing the entire application in a single run through. You will need a separate test file to handle different rules for each of the activities being tested. Be sure to verify all UI elements are filled out properly as well as all intents match the desired elements. All UI interactions should be performed using Espresso, not UiAutomator.		App meets all "Good" requirements.	App meets all positive "Fair" requirements.	Does not suffer from any negatives listed in the "Poor" column.	Espresso tests are not present.
		MainActivity is tested on its own to ensure the proper intent is used to open the DetailsActivity when a list item is clicked.	App does not suffer from any negatives listed in the Fair or Poor columns.	Espresso tests are present in the proper project used for instrumented tests.	
		Full application flow is tested to ensure proper functionality. Start on the MainActivity, open the form, fill out the form, save the data, select the item in the list, view the details, delete the data, validate the data is no longer in the list.	MainActivity is tested on its own to ensure the proper intent is used to open the FormActivity when clicking on the add button.	FormActivity is tested on its own to ensure that filling out the UI and clicking save results the activity closing. Activity should only close when clicking save if all fields are filled out.	
			DetailsActivity is tested on its own to ensure the UI fills out properly based on the passed in data.	One or more tests fail assertion.	
			DetailsActivity is tested on its own to ensure the activity closes when the delete button is clicked.		
Automation Tests	**35%**	**Excellent (100%)**	**Good (80%)**	**Fair (45%)**	**Poor (0%)**
Automation tests should be written to test each activity individually as well as testing the entire application in a single run through. You will need a separate test file to handle different rules for each of the activities being tested. Be sure to verify all UI elements are filled out properly as well as all intents match the desired elements. All UI interactions should be performed using the UiAutomator, not Espresso.		App meets all "Good" requirements.	App meets all positive "Fair" requirements.	Does not suffer from any negatives listed in the "Poor" column.	Espresso tests are not present.
		MainActivity is tested on its own to ensure the proper intent is used to open the DetailsActivity when a list item is clicked.	App does not suffer from any negatives listed in the Fair or Poor columns.	UiAutomator tests are present in the proper project used for instrumented tests.	UiAutomator is not used for performing interactions. (i.e. click events, list item selection, action bar buttons)
		Full application flow is tested to ensure proper functionality. Start on the home screen of the device, select the app icon to open the MainActivity, open the form, fill out the form, save the data, select the item in the list, view the details, delete the data, validate the data is no longer in the list.	MainActivity is tested on its own to ensure the proper intent is used to open the FormActivity when clicking on the add button.	FormActivity is tested on its own to ensure that filling out the UI and clicking save results the activity closing. Activity should only close when clicking save if all fields are filled out.	
			DetailsActivity is tested on its own to ensure the UI fills out properly based on the passed in data.	One or more tests fail assertion.	

			DetailsActivity is tested on its own to ensure the activity closes when the delete button is clicked.		
	100%				