Rubric Item	Description	Example Location - Code	Status
A	The Android mobile application is a version 4.0 or higher, loads properly, and all Android project files, APK, and signed apk files are in one zipped.zip file.	Version: (API 26). App Working.	In-Progress
A1	The mobile application allows the user to enter all term titles and all start and end dates for each term.	Used a Floating Action Button to add a new Term (TermListActivity.java Line ~70). Each term can then be edited via the Floating Action Button in the TermDetailsActivity and subsequent EditTermDetailsActivity. (TermDetailsActivity.java Line ~93, and EditTermDetailsActivity.java)	Done
A2	The features included in the mobile application are coded to allow the user to add an unlimited number of terms.	The FAB add term button will add an unlimited(?) number of terms to the ListView. (TermListActivity.java Line ~70)	Done
A3	The mobile application includes validation to prevent the user from deleting a term when courses have been assigned to that term.	The Delete Term button in the EditTermActivity will create a toast if any courses still exists (EditTermActivity.java Line ~66)	Done
A4A	The mobile application is coded so that the user can add an unlimited number of courses to each term.	The Floating Action Button in the TermDetailsActivity can add an unlimited number of courses to the ListView (TermDetailsActivity.java Line ~71)	Done
A4B	The mobile application allows the user to display a list of courses associated with each term.	The TermDetailsActivity hosts the ListView containing the full list of courses within each Term (TermDetailsActivity.java Line ~112).	Done
A4C	The mobile application allows the user to display a detailed view of each term and the view includes all the given information.	The TermDetailsActivity displays the Term Name, Start Date, End Date, and list of courses (TermDetailsActivity.java Line ~45)	Done
A5	The mobile application allows the user to enter all given details for each course.	The CourseDetailsActivity holds an Edit Floating Action Button. This button leads to the EditCourseActivity where course name, start, end, and status can be modified (CourseDetailsActivity.java Line ~113, EditCourseActivity.java Line ~134)	Done
A6A	The mobile application is coded so that the user can add as many as 5 assessments to each course.	The CourseDetailsActivity holds an Add Floating Action Button that adds an unlimited number of assessments to the ListView (CourseDetailsActivity.java Line ~86).	Done
A6B	The mobile application allows the user to add optional notes within each course.	The CourseDetailsActivity contains the Course Notes button. This button leads to CourseNotesActivity where data can be modified (CourseDetailsActivity.java Line ~139, CourseNotesActivity.java Line ~46)	Done

A6C	The mobile application allows the user to enter, edit and delete any and all information for each course.	The EditCourseActivity allows for the entry and modification of data, as well as a Delete Course button (EditCourseActivity.java Line ~86 and ~134).	Done
A6D	The mobile application allows the user to display optional notes within each course.	The CourseDetailsActivity holds the Course Notes button. This button leads to CourseNotesActivity where the notes for each course are contained (CourseDetailsActivity.java Line ~139, CourseNotesActivity.java Line ~72)	Done
A6E	The mobile application allows the user to display a detailed view of the course information for each course, including the expected completion date for each.	The CourseDetailsActivity displays course name, start, end, and status, as well as the list of associated assessments (CourseDetailsActivity. java Line ~58)	Done
A6F	The mobile application allows the user to set alerts for both the start and end date of each course.	The EditCourseActivity has the "Apply Start/End Alarms" button which will set an alarm for both the start and end dates of the course, as long as they are in the future. Toast notifications are provided. (EditCourseActivity.java Line ~60)	Done
A6G	The mobile application allows the user to share messages or email which automatically populate with the notes.	CourseNotesActivity holds the "Send via SMS to:" button that sends the course notes to the entered phone number with the system's sms application (CourseNotesActivity.java Line ~59)	Done
A7A	The mobile application allows the user to add performance and objective assessments for each course, including the title of the assessment and the expected completion date for each assessment.	The CourseDetailsActivity holds an add Floating Action Button that adds assessments to the ListView. The ListView click leads to EditAssessmentsActivity where all data can be modified (CourseDetailsActivity. java Line ~86, EditAssessmentActivity. java Line ~95).	Done
A7B	The mobile application allows the user to enter, edit, and delete all assessment information.	EditAssessmentActivity allows for modifying all assessment data (EditAssessmentActivity.java Line ~95)	Done
A7C	The mobile application allows the user to set alerts for all start and expected completion dates for each assessment.	The EditAssessmentActivity allows for the creation of an alarm at any time (EditAssessmentActivity.java Line ~64)	Done
B	The screen designs include the layout for each given screen, as described, and includes appropriate GUI elements for each layout.		In Progress

C	The mobile application includes both scheduling and progress tracking elements.	The app can schedule alarms (EditAssessmentActivity.java Line ~64) Terms, Courses, and Assessments database objects contain progress data. (Term.java, Course.java, Assessment.java via ROOM) This progress data is tracked on the home page (HomePageActivity.java)	Done
C1	The mobile application includes all 11 of the given implementation requirements with a minimum API level of 14.	**Below**	
1	An ArrayList	Throughout - used for each ListView. (TermListActivity.java Line ~56)	Done
2	An Intent	Throughout - used for navigation and communication between activities. (TermListActivity.java Line ~61)	Done
3	navigation between multiple screens/activities	Same as Above	Done
4	3 activities	10 Activities currently.	Done
5	Events	Throughout. ListView on click listeners (TermListActivity.java Line ~50)	Done
6	Portrait/Landscape Operability	In-Progress	Done
7	Interactive Capability	Throughout. Every Activity is interactive. (TermListActivity.java Line ~50)	Done
8	Database	A Room/SQLite database was used for data persistence. (FullDatabase.java)	Done
9	App Title and Icon	A customer icon was created for this app.	Done
10	Notifications or Alerts	Toasts throughout (EditAssessmentActivity.java Line ~138) Alarms for Courses and Assessments (EditAssessmentActivity. java Line ~135)	Done
11	Declarative/Programmatic Buttons	Declarative Buttons are throughout (activity_home_page.xml Line ~10) Two buttons on the home page are Programmatic (HomePageActivity.java Line ~51).	Done
C2	The mobile application includes all 5 of the interface requirements listed.	**Below**	
1	Ability to Scroll Vertically	All of the ListViews scroll vertically to show all items. (TermListActivity.java Line ~50)	Done
2	Action Bar	Title and Progress in TitleBar. (TermDetailsActivity.java Line ~166)	Done
3	Two Layouts	Throughout	Done
4	Input Controls	Can't delete term with courses present. (EditTermActivity.java Line ~63)	Done
5	Buttons	Throughout	Done

D	The storyboard demonstrates the flow of the mobile application and includes all of the menus and screens from part B.		In-Progress
E	The screenshots provided demonstrate the creation of a deployment package.		In-Progress
F1	The explanation of mobile application development through the context of the architecture involved, includes hardware and software capabilities and limitations.		In-Progress
F1A	The response identifies both the minimum and the target SDK.	Android 8.0 and newer (API 26)	Done
F2	The response describes challenges faced, specifically during the development of the application.	Finding references relevant to the API selected. Android is obviously a very rapidly evolving platform, and the large amount of outdated materials on the internet show this clearly. Also, because some elements are very new, it isn't easy to find sufficient learning material. Another challenge is dealing with the plethora of variables involved with both the emulated environment and non-uniform hardware devices.	Done
F3	The response describes how each challenge, listed in F2, was overcome.	In order to get my head around some of the newer concepts and elements involved in the production of this application, I had to look through lots of tutorials and forum posts. I also found the need to read through Kotlin based tutorials for Room related questions, and try to read through this language I was not familiar with. Similar to searching for relevant learning materials, finding emulation and hardware settings that seemed to work well took extra time.	Done
F4	The response describes the changes that would be made to at least one aspect of the project, were it to be done again.	If I were to do this project again, I would spend a little more time on planning and diagramming before setting off on implementing anything. The planning I did for this project helped out tremendously. I would also use a recyclerView and custom adapter to significantly improve the aesthetics of the app. I would also do some user testing and work out a nicer interface. It may have been a good idea to merge the editing functionality into the details page instead of using a separate activity.	Done

F5	The response provides a definitive description of the use of emulators and the pros and cons of using an emulator vs. using a development device.	Emulators are used in close connection to the Integrated Development Environment to provide a convenient and cost effective way to develop apps for a wide variety of devices. Emulators are a great tool because they can take the form of virtually any phone or device. However, emulators are slower than their hardware counterparts. This lack of performance makes testing more difficult and time consuming.The requirement to test the application on an external physical device adds a significant extra step on top of traditional software only development. More time is needed for all interaction with the device. Also, it may not be practical to test the application on a large array of devices, leaving blind spots in testing and QA before deployment.	Done