GUJARAT TECHNOLOGICAL UNIVERSITY

COMPUTER ENGINEERING/INFORMATION TECHNOLOGY
ANDROID PROGRAMMING
SUBJECT CODE:2180715
B.E. 8th SEMESTER

Type of course: Bachelor of Engineering

Prerequisite: Java programming and Object-oriented programming, Knowledge of RDBMS and OLTP

Rationale:

Teaching and Examination Scheme:

Teaching Scheme			Credits	Examination Marks						Total Marks
				Theory Marks			Practical Marks			
L	T	P	C	ESE (E)	PA (M)		PA (V)		PA (I)	
					PA	ALA	ESE	OEP		
3	0	2	5	70	20	10	20	10	20	150

Content:

Sr. No.	Content	Total Hrs	% Weightage
1	**The Basics:** ● **Hello World:** Intro to Android, Why develop apps for Android?, Flavors of Android operating systems, Challenges of developing for Android (multiple OS, need backwards compatibility, need to consider performance and offline capability) ● **Concept: Create Your First Android App:** Overview of the development process - Java, Android Studio , Project layout in Android Studio, Target and minimum SDKs, Android Virtual Device (AVD) Monitor, Viewing logs in logcat and AVD, Android manifest file , App Architecture: An app consists of one or more activities. For an activity, write Java code and layout xml, and hook them together, and register the activity in the manifest file. ● **Concept: Layouts, Views and Resources:** Layout elements can be viewed and edited in Layout Editor and XML, Introduction to the range of UI elements, Resources (layouts, strings, styles, themes), Identifying	**10**	

	resources with IDs, Programmatically referencing resources using resource IDs, on Click attribute, Getting user input from a view, Programmatically changing UI elements, Layout Managers, Defining layouts for activities, inflating the layout. ● **Concept: Scrolling Views:** How to make activities scrollable: compare ScrollView, ListView, RecyclerView , Getting the resource ID for a UI element by inflating a layout (needed for RecyclerView) , How to implement RecyclerView (requires layout managers and ViewHolders) , Performance impications of different kinds of scrolling UI elements ● **Concept: Resources to Help You Learn:** Resources to help you learn: Samples that ship with the SDK, Templates for projects, developer.android.com, Android developer blog , Android developer YouTube channel, Source code and samples in github, Stack overflow, Google search! ● **Activities and Intents :**About activities, Defining Activities , Activity Lifecycle , Activity navigation , About intents ,Explicit vs Implicit intents ,Passing info to new activity ,Returning data from activity ● **The Activity Lifecycle and Managing State:** Activity lifecycle , Activity lifecycle callback methods , Activity instance state ● **Starting Activities with Implicit Intents:** Starting activities by sending implicit intents, Intent filters and enabling your activities to receive intents, ShareCompat ● **Testing and Debugging, and Backwards Compatibility:** Debugging your apps, Testing your app, Support libraries		
2	**User Interface:** ● **User Input Controls:** Getting user input , Changing keyboards , Buttons , Dialogs and pickers , Spinners, checkboxes, and radio buttons , Gestures , Speech recognition (not done), Sensors (not done) ● **Menus:** Options menu, contextual menus (floating and action bar), and popup menu, Adding menu items. Handling on Clicks from menus. ● **Screen Navigation:** Terminology, Different ways a user can navigate through an app, Action bar, Settings menu, Navigation drawer, Directed workflow (funnels), Best practices for navigation ● **Themes and Styles:** Best practices for themes and styles, Performance benefits for themes, When and how to use	**10**	

	drawables, best practices for drawable, When and how to use nine-patches, best practices for nine-patches, Tools for creating drawables ● **Material Design:** What is material design? Material design best practices. Material Design guidelines, Implementing Material Design look and feel, with compatibility with previous versions, Support library for Material Design design, Transitions and Animations ● **Adapt layouts for multiple devices and orientations:** Why we need to consider different screen sizes and orientations , Screen density (dip or dp), How to create adaptive layouts using resources folders , Different ways to create images that scale nicely, Images and image formats and how they affect performance (download speeds). ● **Accessibility:** Why accessibility matters, Accessibility considerations: Color blindness, poor vision, poor hearing, physical limitations, Accessibility guidelines , Testing for accessibility , Screen readers, Making your app more accessible: Color and Contrast, button size --> Material Design guidelines, considerate layouts and navigation ● **Localization:** How to prep your app for localization, LTR and RTL (eg Arabic) text. ● **Testing the User Interface:** Automated testing of UIs, User testing your UI with real users, Using the Espresso and UI Automator frameworks for testing UIs		
3	**Background Tasks:** ● **Connect to the Internet: Background Tasks,** Synchronous versus async tasks, What is the UI thread and when should you use it? , Example of a background task -- retrieving data over the internet, Creating background tasks. (schedule, send data, etc.) , Implementing AsyncTask (doInBackground(), callbacks) , Limitations of AsyncTask , Passing info to background tasks, Initiating background tasks, Scheduling background tasks (intro only, more later). ● **Connecting to the Internet:** Permissions, Building URIs, Opening and closing Internet connections, Parsing JSON in Android. (Because it's common.) , Sending requests and parsing response. ● **AsyncTaskLoade:** Intro to AsyncTaskLoader , loadInBackground() , AsyncTaskLoader callbacks , Benefits of loaders ● **Broadcast Receivers:** What is a Broadcast Receiver and a Broadcast Intent? , Broadcast Receiver Security and Lifecycle ● **Services:** What is a service? Long running task without a UI, Difference between Activity and Service , Start and	**10**	

	stop services, Lifecycle methods, Foreground services, IntentService class, App priority (critical, high, low), How to create a new Service. ● **Notifications:** What is a Notification? , Notification Design Guidelines. ● **Triggering, Scheduling, and Optimizing Background** AlarmManager ● **Transferring Data Efficiently:** Less data, less often! Cell radio life cycle, Job Scheduler. Why to use Job Scheduler instead of SyncManager/SyncAdapter, Difference between alarms and job schedulers.		
4	● **Data -- Saving, Retrieving, Loading** ● **Storing Data in your app:** Internal versus external storage, Privacy, sharing, security, encryption of your data , Shared Preferences: Store private primitive data in key-value pairs , SQLite Databases: Store structured data in a private database , Store data on the web with your own network server, Firebase for storing and sharing data in the cloud, Concept: Preferences , What are Settings and Preferences? , Settings best practices (harder to take away settings than to add, for usability reasons, Storing and retrieving preferences as key/value pairs using SharedPreference, Different Settings types, Settings menu, Using Activity and PreferenceFragments to allow users to set preferences ● **Store data using SQLite database:** Overview of SQLite, OpenHelper Android class , Querying (dev) Searching (user) databases , Best practices for using databases in Android , Best practices for testing your database ● **Using Content Resolvers to access data:** Content Providers and Content Resolvers work together, what is a content provider? , What is a content resolver? , How do they work together? , How to implement and use Content Resolvers ● **Content Providers:** When to implement content providers , How to implement content providers (overview), Content URIs , UriMatcher, Content Provider authorities , Required methods on ContentProvider (query, insert, delete, update) , MIME types , Contracts , Making content provider data accessible to other apps by modifying manifest, and protecting data with permissions. ● **Using Loaders to Load and Display Data:** Using loaders to asynchronously load data into an activity or fragment, Benefits of Loaders -- why use them? , Loader states (started, stopped, reset) , LoaderManager , Methods & callbacks to implement in Loaders: loadInBackground(), deliverResult() onStart/StopLoading(),	**10**	

	onReset/Cancelled()),Registering listeners , Using CursorLoader with ContentProviders		
5	**Polish and Publish** ● **Permissions:** The permissions model ● **Libraries:** Using libraries ● **Widgets:** What are widgets? When to use them and how to implement them. ● **Publishing your App:** Different ways to monetize your app (overview only) ● **Making and publishing APKs:** Guidelines for publishing in Google Play , Make and sign the APK, Beta test your app , Publish your app to Google Play	**5**	

Suggested Specification table with Marks (Theory):

Distribution of Theory Marks				
R Level	**U Level**	**A Level**	**N Level**	**E Level**

Legends: R : Remembrance ; U = Understanding; A = Application; N = Analyze; E = Evaluation and above Levels (Revised Bloom's Taxonomy)

Reference Books:

Course Outcome:

This course teaches final-year Computer Science students how to develop Android apps. To be able to understand the process of developing software for the mobile. To be able to create mobile applications on the Android Platform. To be able to create mobile applications involving data storage in SQLite database

List of Experiments:

1. Install Android Studio, Hello World, Logging

- Install Android Studio.
- Create a virtual device.
- Create and Run Hello World on emulator and device.
- Explore project layout.
- Generate and view log statements.
- Explore manifest file.

2. Practical: Make Your First Interactive UI

- Add Views and UI elements in Layout Editor to the app's home screen.
- Edit layout XML.
- Add click behavior to a button (show a toast).

- Change the UI through a button click.
- Write a method to use string resource to define a message to appear in the UI.
- Experiment with using different layouts.
- Explore other UI Elements in the Layout Manager.

3. Practical: Working with TextView Elements

- Use a scroll view for text with minor HTML formatting

4. Practical: Learning Resources

- Get answers from android.developer.com.
- Create new projects with different templates.
- Create a new project based on a sample in the SDK.
- Find out how to add a launcher icon for your app.
- Find out the most popular Android OS in India.

5. Practical: Create and Start Activities

- Create a new activity and layout
- Start the new activity from an existing activity with an explicit intent
- Pass user-entered information from one activity to the other
- Pass information back to the main activity

6. Practical: Lifecycle and State Callbacks

- Add Lifecycle callbacks
- Save and restore instance state

7. Practical: Start Activities with Implicit Intents

- Send an implicit intent to start an activity (open web site)
- Send an implicit intent to start an activity (open location)
- Use an intent filter to allow other apps to start an activity in your app
- Use ShareCompat.IntentBuilder

8. Practical: Using the Debugger
9. Practical: Testing your code
10. Practical: Use support library

11. Practical: Use Keyboards, Input Controls, Alerts, and Pickers

- Experiment in your app with different keyboards for user input, spelling suggestions, and auto-capitalization.
- Add a spinner input control for selecting one value out of a set of values.

Lecture hours:

- Create new app to show an alert, and record the user's selection (OK or Cancel). MOVE TO CONCEPT.
- Update app to show date and time pickers and record the user's selections.

12. Practical: Use an Options Menu and Radio Buttons

- Set up an options menu and overflow menu
- Add items to the option (overflow) menu.
- Add radio buttons for user selection.
- Add Up navigation to the app bar.

13. Practical: Create a Recycler View

- Create an activity that displays data in a RecyclerView.
- Make the items in the list clickable
- Add a floating action button to add items to the list

14. Practical: Theme, Custom Styles, Drawables

- Define and use a theme
- Define and use a custom style that uses a drawable

15. Practical: Add a FAB and Cards

- Create an app that uses a Floating Action Button (FAB)
- Add an activity that uses cards. Optionally, style the cards.
- Customize your app's theme and styles to use Material Design styles and colors.

16. Practical: Put yourself in the Users shoes

- Test your app for accessibility, using Talkback and Explore by Touch. Switch to monochrome color space
- Put in earplugs, can you still use your app?
- Wear the darkest glasses you can find, can you still use your gloves?
- Put on gloves, can you still use your app?
- How would you make one of the apps you have written so far more accessible?

17. Practical: Implement Localized Strings

- Create localized strings in your app
- Test by changing default language

18. Practical: Use Espresso to test your UI

- Use Espresso to Test Your UI

19. Practical: Create an AsyncTask

- Create a simple AsyncTask to do work in the background

20. Practical: Google APIs Explorer, JSON, Books API

- Use the Books API in the Google APIs Explorer to investigate request format and JSON response format
- Create a new app that uses the Books API and AsyncTask to search for the author of a book..
- Write the code to parse the response and extract and display the relevant information
- Debug errors when the Internet permission is missing
- Add the missing permission to the Android Manifest.
- Verify your fix by running and testing your app.

21. Practical: Use AsyncTaskLoader

- Use AsyncTaskLoader instead of AsyncTask to show book search results in a RecyclerView

22. Practical: BroadcastReceiver

- Create an app with a BroadcastReceiver

23. Practical: Notifications

- Trigger a Notification
- Add Actions to your Notification

24. Practical: Alarm Manager

- Implement an alarm manager

25. Practical: Job Scheduler

- Use JobScheduler to do background updates

26. Practical: Firebase Job Dispatcher

27. Practical: Get and Save User Preferences

- Implement Settings menu to allow users to enter preferences.
- Implement code to retrieve and user user preferences

28. Practical: Save user data in a database

- Create an app that allows users to enter notes
- Save the notes in a SQLite database

- Create an app that stores data in an SQL database.
- Display the data in a RecyclerView.
- Allow users to add, delete, and edit data items.

29. Practical: Querying and Searching a Database

30. Practical: Implement a Content Provider

- Add a content provider for your SQLite database

31. Practical: Use a ContentResolver to query your data

- Use a content resolver to query the database
- Display the results of the query
- Use the content resolver to add data to the database

32. Practical: Implement a Loader

- Implement a loader
- Register a Listener for the Loader
- Test the loader by checking that the Items in the UI update when the data generated by the loader changes
- Use an AsyncTaskLoader to update a scrolling list of notes titles as the user adds more notes
- Register a Listener for the Loader
- Test the loader by checking that the Items in the UI update when the underlying data changes

33. Practical: Beta testing your app

- Running a beta test on Google Play