Competency Based Learning Materials (CBLM)

Android Mobile Application Development

Level-4

Module: Performing Project Work with Android

Code: CBLM-OU-ICT-AMAD-07-L4-V1

National Skills Development Authority
Prime Minister's Office
Government of the People's Republic of Bangladesh

Copyright

National Skills Development Authority
Prime Minister's Office
Level: 10-11, Biniyog Bhaban,
E-6 / B, Agargaon, Sher-E-Bangla Nagar Dhaka-1207, Bangladesh.
Email: ec@nsda.gov.bd
Website: www.nsda.gov.bd.
National Skills Portal: http:\\skillsportal.gov.bd

This Competency Based Learning Materials (CBLM) on "Performing Project Work with Android" under the Android Mobile Application Development, Level-4 qualification is developed based on the national competency standard approved by National Skills Development Authority (NSDA)

This document is to be used as a key reference point by the competency-based learning materials developers, teachers/trainers/assessors as a base on which to build instructional activities.

This Competency Based Learning Materials is a document for the development of curricula, teaching and learning materials, and assessment tools. It also serves as the document for providing training consistent with the requirements of industry in order to meet the qualification of individuals who graduated through the established standard via competency-based assessment for a relevant job.

This document has been developed by NSDA in association with industry representatives, academia, related specialist, trainer, and related employee.

Public and private institutions may use the information contained in this CBLM for activities benefitting Bangladesh.

Approved by __ th Authority Meeting of NSDA Held on ----------

How to use this Competency Based Learning Materials (CBLMs)

The module, Performing Project Work with Android contains training materials and activities for you to complete. These activities may be completed as part of structured classroom activities or you may be required you to work at your own pace. These activities will ask you to complete associated learning and practice activities in order to gain knowledge and skills you need to achieve the learning outcomes.

1. Review the **Learning Activity** page to understand the sequence of learning activities you will undergo. This page will serve as your road map towards the achievement of competence.

2. Read the **Information Sheets.** This will give you an understanding of the jobs or tasks you are going to learn how to do. Once you have finished reading the **Information Sheets** complete the questions in the **Self-Check.**

3. **Self-**Checks are found after each **Information Sheet**. **Self-Checks** are designed to help you know how you are progressing. If you are unable to answer the questions in the **Self-Check** you will need to re-read the relevant **Information Sheet**. Once you have completed all the questions check your answers by reading the relevant **Answer Keys** found at the end of this module.

4. Next move on to the **Job Sheets. Job Sheets** provide detailed information about *how to do the job* you are being trained in. Some **Job Sheets** will also have a series of **Activity Sheets**. These sheets have been designed to introduce you to the job step by step. This is where you will apply the new knowledge you gained by reading the Information Sheets. This is your opportunity to practice the job. You may need to practice the job or activity several times before you become competent.

5. Specification **sheets**, specifying the details of the job to be performed will be provided where appropriate.

6. A review of competency is provided on the last page to help remind if all the required assessment criteria have been met. This record is for your own information and guidance and is not an official record of competency

When working though this Module always be aware of your safety and the safety of others in the training room. Should you require assistance or clarification please consult your trainer or facilitator.

When you have satisfactorily completed all the Jobs and/or Activities outlined in this module, an assessment event will be scheduled to assess if you have achieved competency in the specified learning outcomes. You will then be ready to move onto the next Unit of Competency or Module

Table of Content

Module Content

Unit of Competency: Perform Project Work with Android

Module Title: Performing Project Work with Android

Module Description: This module discusses the aspects that must be given attention when Performing Project Work with Android. It shows the knowledge and skills requirements for interpreting project management basics, developing an application in Android, developing user story, performing unit testing, creating project presentation and developing awareness about rights

Nominal Duration: 30 Hours

Learning Outcomes:

Upon completion of this module the trainees must be able to:

1. Interpret project management basics
2. Develop an application in Android
3. Develop user story
4. Perform unit testing
5. Create project presentation
6. Develop awareness about rights

Assessment Criteria:

1.1 Concepts of project management is interpreted
1.2 Resource management is interpreted
1.3 Process management is interpreted
1.4 Technology management is interpreted
1.5 Team communication and reporting are acknowledged

2.1 Android Architecture Components are identified
2.2 Project model is selected as per project requirement.
2.3 Key principles of selected model are implemented.
2.4 User interface of a mobile application is designed.
2.5 Git as a source control system is used.
2.6 Android application is developed.

3.1. User story is explained.
3.2. Story estimated.
3.3. User stories of a project work is defined.
3.4. Project management tool is used.
3.5. Project stories are made.

4.1 Test cases are identified

4.2 Testing tools are used
5.1 Project document is created.
5.2 Final project presentation in a group and/or individual is created.

6.1 The policies, rules and regulations that govern the work and workplace are upheld.
6.2 Illegal conduct or illegitimate action is reported to appropriate management.
6.3 Propriety or confidential information is protected.

Learning Outcome 1: Interpret Project Management Basics

Assessment Criteria:

1. Concepts of project management is interpreted
2. Resource management is interpreted
3. Process management is interpreted
4. Technology management is interpreted
5. Team communication and reporting are acknowledged

Content:

1. Concepts of project management
2. Resource management
3. Process management
4. Technology management
5. Team communication and reporting

Resources Required/ Conditions:

The trainees must be provided with the following:

- Handouts or reference materials/books/ CBLMs on the above stated contents
- PCs/printers or laptop/printer with internet access
- Digital projector and Screen
- Bond paper
- Ball pens/pencils and other office supplies and materials
- Relevant learning materials
- Workplace or simulated environment

Methodologies

- Lecture/discussion
- Demonstration/application
- Presentation
- Blended delivery methods

Assessment Methods

- Written test
- Demonstration
- Observation with checklist
- Oral questioning
- Portfolio

Learning Experience 1: Interpret Project Management Basics

In order to achieve the objectives stated in this learning guide, you must perform the learning steps below. Beside each step are the resources or special instructions you will use to accomplish the corresponding activity.

Learning Steps	Resources specific instructions
1. Trainee will ask the instructor about Interpret project management basics	1. Instructor will provide the learning materials "**Performing Project Work with Android"**
2. Read the Information sheet/s	2. Information Sheet No: 1 Interpret project management basics
3. Complete the Self Checks & Check answer sheets.	3. Self-Check/s Self-Check No: 1 Interpret project management basics Answer key No. 1 Interpret project management basics
4. Read the Job Sheet and Specification Sheet and perform job	4. Job- Sheet No: 1- Interpret project management basics Specification Sheet 1 – Interpret project management basics

Information Sheet 1: Interpret Project Management Basics

Learning Objectives:

After completion of this information sheet, the learners will be able to:

1.1 Interpret Concepts of project management
1.2 Interpret Resource management
1.3 Interpret Process management
1.4 Interpret Technology management
1.5 Acknowledge Team communication and reporting

1.1 Concepts of project management

Project management in the context of Android app development encompasses a set of practices and methodologies that guide the planning, execution, and delivery of an app.

A. Key concepts tailored to Android app development:

a. Agile Methodology: Agile is a popular project management approach in Android app development. It emphasizes iterative development, where requirements and solutions evolve through collaboration between self-organizing cross-functional teams.

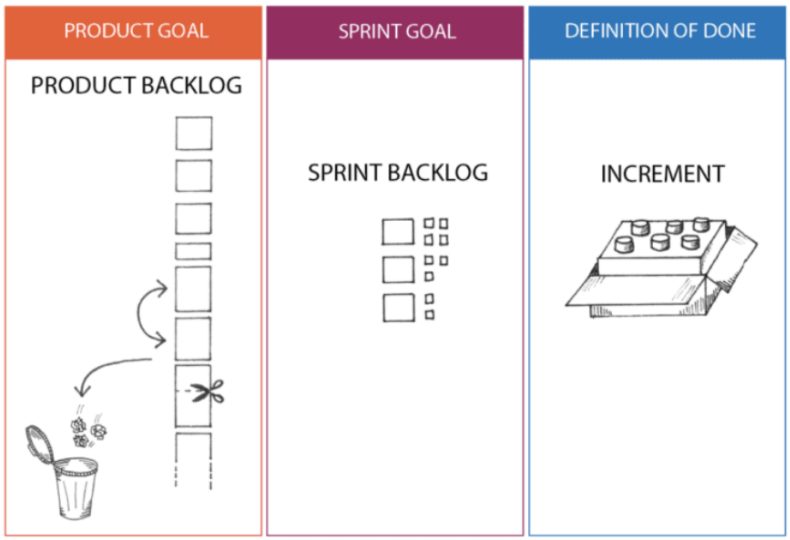
The agile project management methodology prioritizes maximum value against the business goals within an allowable time and budget. This method empowers team members and supports the constant delivery of value to the project.

As an iterative approach to completing a project, agile management promotes velocity and adaptability.

In short, the agile methodology is flexible. Rather than following a linear path from start to finish, different incremental steps and iterations are delivered in shorter bursts. This allows for maximum flexibility, as things can change throughout the project life cycle.

b. Scrum Framework: Often used within the Agile methodology, Scrum is a framework that helps teams work together. It encourages teams to learn through experiences, self-organize while working on a problem, and reflect on their wins and losses to continuously improve.

Scrum is a project management methodology that follows the same core values and principles as agile. So you'll want to keep those agile concepts in mind if you're using this method.



All work gets completed in short cycles, called sprints. Scrum teams meet on a daily basis to quickly discuss the current tasks they're working on and any obstacles they're facing.

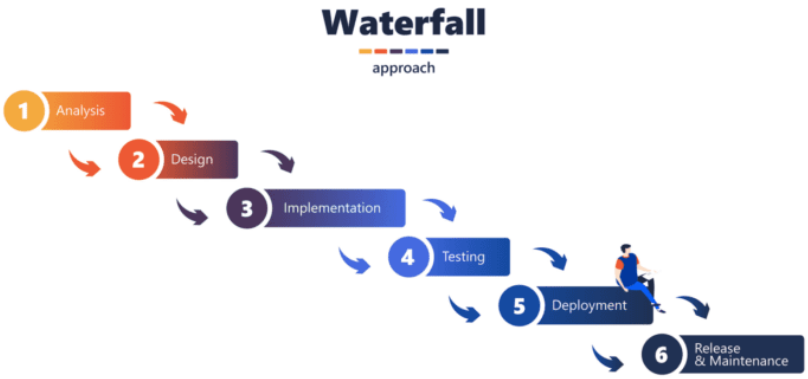
Daily scrums are also known as stand-up meetings. The idea here is that the meeting is so short that nobody has to sit down. Even if your software team is working remotely, it's important that you stick to the daily scrum.

c. **Waterfall Model:** This is a sequential design process, often used in software development processes, where progress is seen as flowing steadily downwards (like a waterfall) through several phases such as conception, initiation, analysis, design, construction, testing, deployment, and maintenance.

Waterfall Project Management

The waterfall methodology is the simplest way to plan any project. It's not necessarily the most common methodology for app development, but it can work well for basic apps.

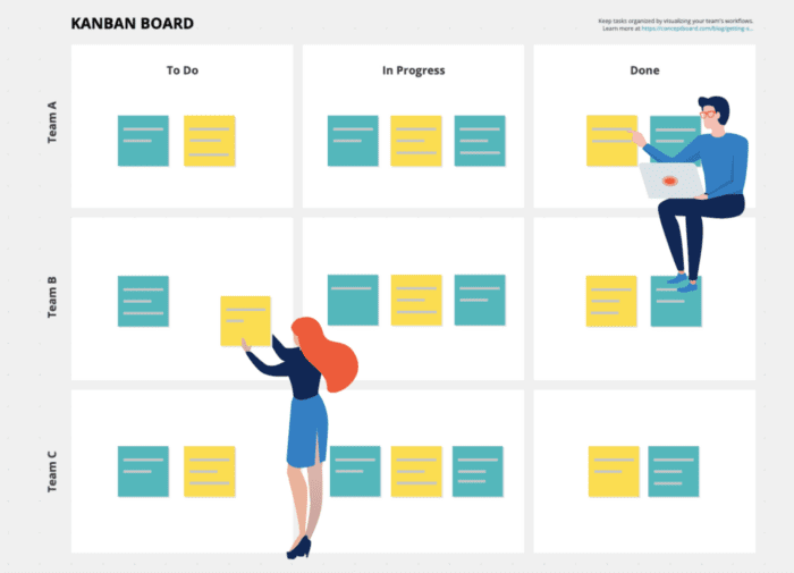
Just like a waterfall, everything flows down—you can't turn around and go back upstream.



It's easy for teams to understand the waterfall process because everything is straightforward. But it doesn't work well with larger teams.

d. **Kanban:** A method for managing the creation of products with an emphasis on continual delivery while not overburdening the development team. Kanban is based on three basic principles: visualize what you do today (workflow), limit the amount of work in progress (WIP), and enhance flow.

Kanban project management is another simple concept to grasp, and the methodology can be used for a wide range of projects—including software.



It's common for kanban boards to be used in other types of project management methodologies. For example, we briefly touched on this earlier when discussing scrum. Lots of scrum teams use kanban boards to manage sprints.

e. **Lean Principles:** Lean principles focus on creating more value for customers with fewer resources. In Android app development, this means understanding customer value, the value stream, flow, pull, and seeking perfection in the development process.
f. **User Stories:** In Android app development, user stories help to focus on the user's needs. They are short, simple descriptions of a feature told from the perspective of the person who desires the new capability, usually a user or customer of the system.
g. **Sprints:** Sprints are a set period during which specific work has to be completed and made ready for review. In the context of Android development, a sprint could involve designing a new feature, coding, testing, or bug fixing.
h. **Version Control:** Essential for managing changes to the app's source code, version control systems like Git help Android development teams track revisions, branch out for new features, and merge code changes.

Continuous Integration/Continuous Deployment (CI/CD): CI/CD is a method to frequently deliver apps to customers by introducing automation into the stages of app development. The main concepts attributed to CI/CD are continuous integration, continuous deployment, and continuous delivery.

i. **Risk Management:** Identifying, analyzing, and responding to risks in the app development process to minimize, monitor, and control the probability or impact of unfortunate events.

These concepts are integral to managing an Android app development project effectively, ensuring that the final product is delivered on time, within budget, and meets or exceeds stakeholder expectations.

Resources are the additional files and static content that your code uses, such as bitmaps, layout definitions, user interface strings, animation instructions, and more.

1.2 Group resource types

Place each type of resource in a specific subdirectory of your project's res/ directory. For example, here's the file hierarchy for a simple project:

```
MyProject/
    src/
        MyActivity.java
    res/
        drawable/
            graphic.png
        layout/
            main.xml
            info.xml
        mipmap/
            icon.png
        values/
            strings.xml
```

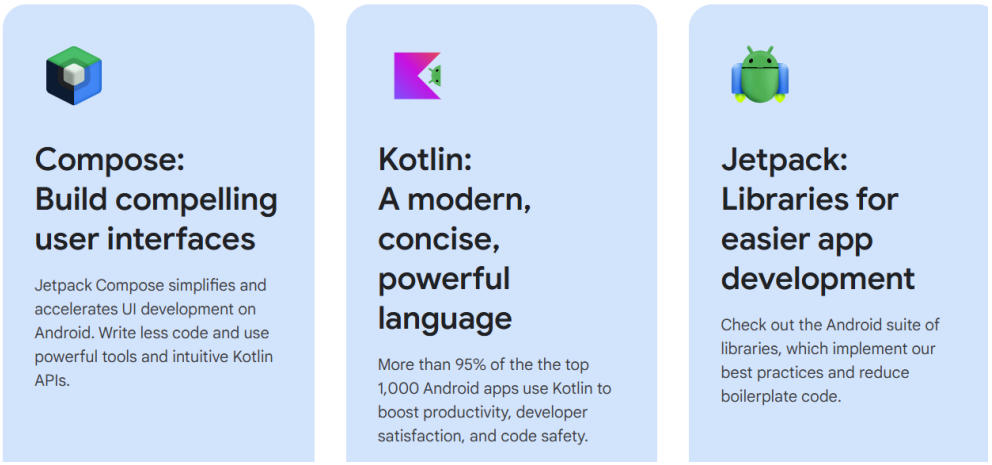
The res/ directory contains all the resources in its subdirectories: an image resource, two layout resources, a mipmap/ directory for launcher icons, and a string resource file. The resource directory names are important and are described in table.

1.3 Process management

Process management in the context of Android app project development involves overseeing the various processes that occur during the lifecycle of an app, from its initial conception to its release and subsequent updates.

1.4 Technology management

Technology management in the context of Android app project development encompasses the strategic planning and execution of technology-related tasks to ensure the successful creation and maintenance of an Android application.



A. **Selection of Development Tools:** Choosing the right set of tools, such as Integrated Development Environments (IDEs), version control systems, and debugging tools, is crucial for efficient development. For Android, this often means using Android Studio, the official IDE for Android development.

B. **Adoption of Programming Languages:** Deciding on the programming languages that will be used, typically Java or Kotlin for Android development, and ensuring the development team is proficient in these languages.
C. **Use of Frameworks and Libraries:** Integrating frameworks and libraries to speed up development and provide functionalities without having to build them from scratch. This includes both native libraries provided by the Android SDK and third-party libraries.
D. **Application Lifecycle Management:** Managing the stages of the app's lifecycle, from development to deployment, and updates. This involves understanding the Android activity lifecycle and how it affects the user experience2.
E. **Performance Optimization:** Ensuring the app performs well across a variety of devices with different hardware capabilities. This includes managing memory usage, battery life, and responsiveness.
F. **Security Management:** Implementing best practices to keep user data secure and the app safe from vulnerabilities. This includes using encryption, secure communication protocols, and proper authentication methods.
G. **Compliance with Standards:** Adhering to Google's guidelines and standards for Android development, including design guidelines, user interface best practices, and the use of Google Play Services.
H. **Testing and Quality Assurance:** Establishing a robust testing process to identify and fix bugs early. This includes unit testing, integration testing, and user acceptance testing.
I. **Continuous Integration/Continuous Deployment (CI/CD**): Setting up CI/CD pipelines to automate the building, testing, and deployment processes, which helps in delivering updates and new features more rapidly and reliably.

Monitoring and Analytics: Using tools to monitor the app's performance in real-time and gather analytics on user behavior to inform future development decisions.

1.5 Team communication and reporting

Team communication and reporting are critical components in the development of Android apps, ensuring that all team members are aligned and informed throughout the project lifecycle.

Concepts apply in this context:

a. **Clear Communication Channels:** Establishing clear and open channels for communication is vital. This can include instant messaging platforms, video conferencing tools, and regular meetings to discuss progress and challenges1.
b. **Roles and Responsibilities:** Each team member should have a clear understanding of their roles and responsibilities. This clarity helps in setting expectations and accountability, which is essential for effective communication and project success1.
c. **Regular Reporting:** Regular status reports are crucial for keeping stakeholders informed about the project's progress. These reports can include updates on completed tasks, upcoming tasks, any blockers, and overall project health.

d. **Agile Ceremonies:** Many Android app development teams use Agile methodologies, which include ceremonies like daily stand-ups, sprint planning, reviews, and retrospectives. These ceremonies facilitate ongoing communication and continuous improvement2.
e. **Documentation:** Maintaining up-to-date documentation, including project plans, design documents, and technical specifications, ensures that team members can access and share important information as needed.
f. **Feedback Loops:** Implementing feedback loops within the team and with stakeholders, including users, helps in identifying issues early and adjusting plans quickly to address any concerns.
g. **Project Management Tools:** Utilizing project management tools can help in tracking tasks, deadlines, and dependencies. These tools often come with reporting features that provide insights into project metrics and progress3.
h. **Risk Communication:** Communicating potential risks and their mitigation plans is essential for transparency and preparedness. This involves identifying risks early and reporting them to relevant stakeholders.
i. **Collaborative Environment:** Fostering a collaborative environment encourages team members to share knowledge, ask questions, and offer solutions, which can lead to more innovative and effective outcomes.
j. **Continuous Learning:** Encouraging continuous learning and sharing of new technologies, tools, and best practices among team members can help in improving processes and the quality of the Android app.

Self-Check1: Interpret Project Management Basics

1. What is project management?
   Answer
2. Why is resource management important in projects?
   Answer
3. How does process management contribute to project development?
   Answer
4. What role does technology management play in project development?
   Answer
5. How can team communication be optimized in project development?
   Answer
6. What is a key principle of project management?
   Answer
7. What is a common challenge in resource management?
   Answer
8. What is an example of process management in action?
   Answer
9. How does technology management affect project risk?
   Answer
10. Why is acknowledging team communication and reporting important?
    Answer

Answer Key 1: Interpret Project Management Basics

1. **What is project management?**
   Answer: Project management is the application of knowledge, skills, tools, and techniques to project activities to meet the project requirements.

2. **Why is resource management important in projects?**
   Answer: Resource management is crucial for ensuring that the right resources are available at the right times to support the successful completion of a project.

3. **How does process management contribute to project development?**
   Answer: Process management helps in streamlining activities, reducing waste, and ensuring that tasks are completed efficiently and effectively.

4. **What role does technology management play in project development?**
   Answer: Technology management involves planning and executing the use of technology to enhance the efficiency and effectiveness of project development.

5. **How can team communication be optimized in project development?**
   Answer: Team communication can be optimized by establishing clear channels, using collaborative tools, and ensuring regular and transparent reporting.

6. **What is a key principle of project management?**
   Answer: A key principle of project management is aligning project objectives with business goals to ensure that the project delivers value.

7. **What is a common challenge in resource management?**
   Answer: A common challenge is balancing resource allocation to avoid overuse or underuse, ensuring all project areas are adequately resourced.

8. **What is an example of process management in action?**
   Answer: An example is using a workflow management system to automate and monitor project tasks, ensuring they are completed on schedule.

9. **How does technology management affect project risk?**
   Answer: Proper technology management can mitigate project risk by ensuring the right technology is used and maintained to support project goals
   .

10. **Why is acknowledging team communication and reporting important?**
    Answer: Acknowledging team communication and reporting is important for maintaining morale, ensuring accountability, and fostering a culture of recognition and continuous improvement.

Activity Sheet 1.1:

1. **Concepts of Project Management**
   Task: Research and document the five phases of project management: initiation, planning, execution, monitoring, and closure.
   Goal: Understand the end-to-end process of managing a project.
   Expected Outcome: A report detailing each phase with examples from Android app development.
2. **Resource Management**
   Task: Create a resource allocation plan for an Android app project, including human resources, technology, and physical assets.
   Goal: Learn to efficiently allocate and utilize resources.
   Expected Outcome: A resource plan that maximizes efficiency without overextending capabilities.
3. **Process Management**
   Task: Map out the process flow for an Android app's development lifecycle, from concept to launch.
   Goal: Understand the importance of process optimization.
   Expected Outcome: A process flow diagram that identifies key stages and activities.
4. **Technology Management**
   Task: Evaluate and select appropriate technologies for different aspects of Android app development, such as databases, front-end frameworks, and testing tools.
   Goal: Make informed decisions about technology use.
   Expected Outcome: A technology stack document justifying each choice.
5. **Team Communication and Reporting**
   Task: Develop a communication plan that outlines methods, frequency, and channels for team interaction and progress reporting.
   Goal: Foster effective communication and timely reporting.
   Expected Outcome: A communication plan that ensures all team members are informed and engaged.

Learning Outcome 2: Develop An Application In Android

Assessment Criteria:

2.1 Android Architecture Components are identified
2.2 Project model is selected as per project requirement.
2.3 Key principles of selected model are implemented.
2.4 User interface of a mobile application is designed.
2.5 Git as a source control system is used.
2.6 Android application is developed.

Content:

1. Android Architecture Components
    a. Separation of Concern
    b. MVVM Pattern
    c. Lifecycle of a View Model
    d. Live Data
    e. Data Binding
    f. Pagination
2. Project model selection as per project requirement.
3. Key principles of android project model.
4. Designing User interface of a mobile application.
5. Git as a source control system.

Resources Required/ Conditions:

The trainees must be provided with the following:

- Handouts or reference materials/books/ CBLMs on the above stated contents
- PCs/printers or laptop/printer with internet access
- Digital projector and Screen
- Bond paper
- Ball pens/pencils and other office supplies and materials
- Relevant learning materials
- Workplace or simulated environment

Methodologies

- Lecture/discussion
- Demonstration/application
- Presentation
- Blended delivery methods

Assessment Methods

- Written test
- Demonstration
- Observation with checklist
- Oral questioning
- Portfolio

Learning Experience 2: Develop An Application In Android

In order to achieve the objectives stated in this learning guide, you must perform the learning steps below. Beside each step are the resources or special instructions you will use to accomplish the corresponding activity.

Learning Steps	Resources specific instructions
1. Student will ask the instructor about develop an application in Android	1. Instructor will provide the learning materials "**Performing Project Work with Android"**
2. Read the Information sheet/s	2. Information Sheet No: 2 Develop an application in Android
3. Complete the Self Checks & Check answer sheets.	3. Self-Check/s Self-Check No: 2 - Develop an application in Android Answer key No. 2 - Develop an application in Android
4. Read the Job Sheet and Specification Sheet and perform job	4. Job- Sheet No: 2 - Develop an application in Android Specification Sheet: 2- Develop an application in Android

Information Sheet 2: Develop an Application in Android

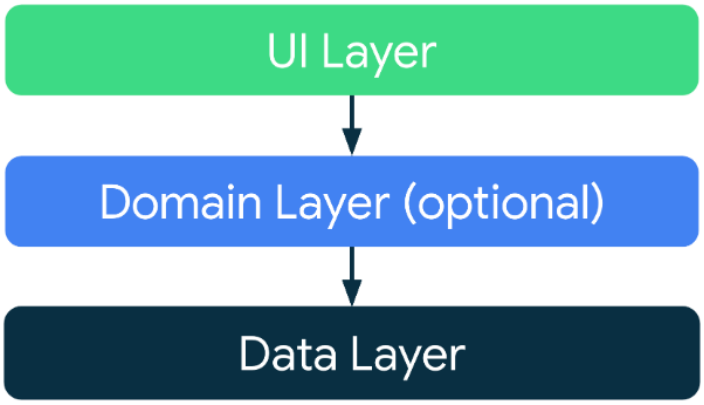
Learning Objectives:

After completion of this information sheet, the learners will be able to:

2.1 Identify Android Architecture Components
2.2 Select Project model as per project requirement.
2.3 Implement Key principles of selected model.
2.4 Design user interface of a mobile application.
2.5 Use Git as a source control system
2.6 Develop Android application.

2.1 Android Architecture Components

Android Architecture Components are a collection of libraries that help you design robust, testable, and maintainable apps. They serve as the building blocks for creating apps that have a solid architecture and are less prone to bugs or crashes.



Overview of the key components:

a. **LiveData:** LiveData is an observable data holder class that is lifecycle-aware, meaning it respects the lifecycle of other app components, such as activities, fragments, or services. This characteristic ensures that LiveData only updates app component observers that are in an active lifecycle state.
b. **ViewModel:** ViewModel is designed to store and manage UI-related data in a lifecycle-conscious way. It allows data to survive configuration changes such as screen rotations.
c. **Room:** Room is an abstraction layer over SQLite to allow for more robust database access while harnessing the full power of SQLite. It includes compile-time checks of SQLite statements and can return LiveData objects.
d. **Data Binding:** Data Binding binds UI components in your layouts to data sources in your app using a declarative format rather than programmatically, reducing boilerplate code.
e. **Navigation:** The Navigation component helps you implement navigation, from simple button clicks to more complex patterns, such as app bars and the navigation drawer.

f. **WorkManager:** WorkManager is a library for managing deferrable, asynchronous tasks that require guaranteed execution. It is the recommended solution for persistent work.
g. **Paging:** The Paging library helps you load and display small chunks of data at a time. It supports both network and database data.
h. **Lifecycle:** Lifecycle components help you produce better-organized, and often lighter-weight code, that is easier to maintain and less prone to memory leaks.
i. **Saved State for ViewModel:** This component helps you save the UI state across app process death and recreation by the system.

These components are part of the Android Jetpack suite, which provides guidance on app architecture and a robust set of libraries to ease the development of Android apps. They are designed to work together seamlessly and can significantly reduce the amount of boilerplate code you need to write.

Selecting the right project model as per the project requirement is a critical step in project management that can significantly influence the success of the project. Here's a high-level explanation of how to approach project model selection:

j. **Understand Project Requirements:** Begin by thoroughly understanding the project's goals, scope, constraints, and stakeholders' expectations. This will help in identifying the key requirements that the project model must satisfy.
k. **Evaluate Project Complexity:** Assess the complexity of the project in terms of size, duration, budget, and technical requirements. Simple projects may benefit from a more straightforward model, while complex projects might require a more robust and flexible approach.
l. **Consider Methodologies:** Familiarize yourself with various project management methodologies such as Agile, Waterfall, Scrum, Kanban, and Lean. Each has its strengths and is suited to different types of projects.

m. **Benefit Measurement Model:** For smaller and less complex projects, the Benefit Measurement Model, which compares the benefits provided by project outcomes to the costs, can be a useful approach.
n. **Constrained Optimization Method:** For larger and more complex projects, the Constrained Optimization Method, which uses mathematical models to find the best project path, may be more appropriate.
o. **Resource Availability:** Take into account the availability of resources, including team skills, technology, and materials. The chosen model should align with the resource capacity of the organization.
p. **Risk Assessment:** Evaluate potential risks associated with the project and select a model that can best mitigate these risks.
q. **Stakeholder Input:** Involve stakeholders in the selection process to ensure their needs and concerns are addressed by the project model.
r. **Flexibility and Scalability:** Ensure that the model is flexible enough to accommodate changes and scalable to adapt to project size variations.
s. **Review and Decide:** Review all gathered information and make a decision based on which model aligns best with the project's requirements, complexity, and constraints.

When you have a number of interesting and challenging projects to choose from, finding a project that is the right fit for your team's skill set, level of competence, and has the best chance of success is the first step in effective project management. Project Selection Methods offer a set of time-tested techniques based on sound logical reasoning to choose a project and filter out undesirable projects with a very low likelihood of success. Project selection methods are an important concept for practicing project managers and aspirants preparing for the PMP® exam alike. In this article, we will discuss the following project selection methods in detail:

- Benefit measurement methods
- Benefit/cost ratio
- Economic model
- Scoring model
- Payback period
- Net present value
- Discounted cash flow
- Internal rate of return
- Opportunity cost
- Constrained optimization methods
- Non-financial considerations

2.2 Select Project model as per project requirement.

Choosing the right project model for your Android application is crucial for ensuring that your app is scalable, maintainable, and efficient. Here are some commonly used project models and how to select the right one based on your project requirements:

1. MVC (Model-View-Controller)

Overview

- Model: Manages data and business logic.
- View: Displays the data to the user.
- Controller: Handles user input and updates the model.

Use Cases

- Simple applications with straightforward user interfaces.
- Projects with a clear separation between UI and data logic.

Pros

- Clear separation of concerns.
- Easy to understand and implement.

Cons

- Can become unwieldy with complex applications.
- Controller can become overloaded with too much logic.

2. MVP (Model-View-Presenter)

Overview

- Model: Manages data and business logic.
- View: Displays the data and forwards user interactions to the Presenter.
- Presenter: Acts as an intermediary between Model and View, handling the logic and updating the View.

Use Cases

- Applications with complex user interfaces.
- Projects requiring a clear separation of responsibilities between UI and logic.

Pros

- Better separation of concerns than MVC.
- Easier to unit test because the Presenter is decoupled from the Android framework.

Cons

- Can introduce additional complexity.
- Requires more boilerplate code.

3. MVVM (Model-View-ViewModel)

Overview

- Model: Manages data and business logic.
- View: Displays data and forwards user interactions to the ViewModel.
- ViewModel: Manages UI-related data and interacts with the Model to update the View.

Use Cases

- Applications with complex UIs and interactions.

- Projects that benefit from two-way data binding.

Pros

- Excellent separation of concerns.
- Simplifies UI-related logic with data binding.
- Easier to test than MVC and MVP.

Cons

- Steeper learning curve due to data binding.
- Can become complex with large applications.

4. Clean Architecture

Overview

- Focuses on separation of concerns by organizing code into layers:
    - Presentation Layer: Handles UI and user interactions.
    - Domain Layer: Contains business logic and use cases.
    - Data Layer: Manages data sources and repositories.

Use Cases

- Large-scale applications with complex business logic.
- Projects requiring high maintainability and testability.

Pros

- Highly scalable and maintainable.
- Clear separation of concerns across different layers.
- Promotes best practices in software design.

Cons

- More complex and requires careful planning.
- Higher initial development cost due to the need for more structure and boilerplate.

Selecting the Right Model

When selecting the right project model, consider the following factors:

1. Application Complexity
    - Simple Applications: MVC might be sufficient.
    - Moderate Complexity: MVP or MVVM can provide a better structure.
    - High Complexity: Clean Architecture is ideal for maintaining a large codebase.
2. Team Size and Experience
    - Small Teams: MVC or MVP can be easier to manage and understand.
    - Experienced Teams: MVVM or Clean Architecture can provide better maintainability and testability.
3. Testing Requirements
    - Basic Testing Needs: MVC or MVP may suffice.
    - Extensive Testing Needs: MVVM or Clean Architecture provides better testability.
4. Maintainability
    - Short-term Projects: MVC might be quick and easy to implement.
    - Long-term Projects: MVVM or Clean Architecture ensures better maintainability.

Examples

Simple To-Do App (MVC)

- Model: Task data and business logic.
- View: XML layouts for displaying tasks.
- Controller: Activity or Fragment handling user interactions.

E-commerce App (MVVM)

- Model: Product data, user data, etc.
- View: XML layouts using data binding.
- ViewModel: Manages UI-related data and communicates with the Model.

Banking App (Clean Architecture)

- Presentation Layer: Activities and Fragments.
- Domain Layer: Use cases and business logic.
- Data Layer: Repositories and data sources (e.g., network, database).

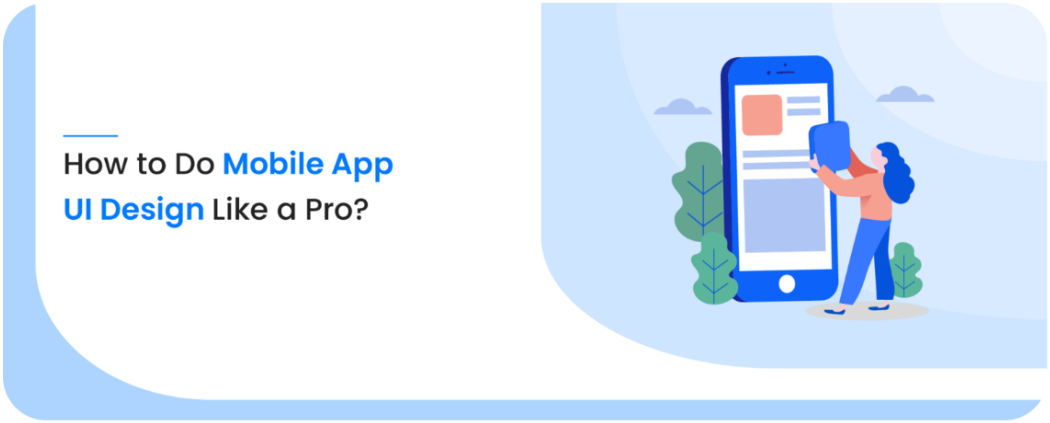
2.3 The key principles of an Android project model

The key principles of an Android project model revolve around creating a robust and maintainable app architecture. Here are some of the foundational principles:

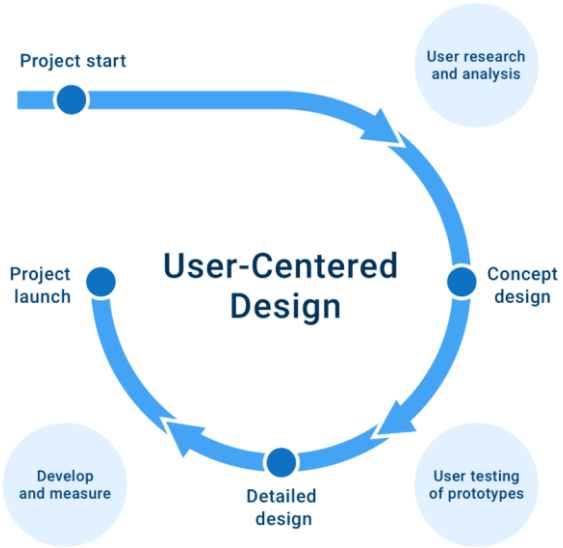
a. **Separation of Concerns:** This principle dictates that different parts of the app should have distinct responsibilities. For instance, UI-based classes like Activities and Fragments should only handle UI and OS interactions, not store data or app state1.
b. **Drive UI from Data Models:** The UI should be a reflection of the data model. This means that changes in the underlying data should automatically update the UI, which can be achieved using components like LiveData1.
c. **Single Source of Truth (SSOT):** Data should have a single source of truth, such as a database or a repository class, to prevent inconsistencies and make data management easier1.
d. **Unidirectional Data Flow:** Data should flow in one direction from the source of truth to the UI, which simplifies the logic and makes it easier to follow and debug1.
e. **Modularization:** Breaking down the app into modules or components can help in managing complexity, making the codebase more maintainable, and facilitating team collaboration.
f. **Scalability:** The architecture should support the growth of the app, allowing for the addition of new features without significant restructuring.
g. **Testability:** The app should be designed with testing in mind, making it easy to write and run tests for various components.
h. **Robustness:** The architecture should handle errors gracefully and maintain performance under different conditions, such as network unavailability or high load.
i. **Maintainability:** The code should be easy to understand and modify, which can be achieved by following consistent coding standards and good documentation practices.
j. **Performance Optimization:** The architecture should promote efficient use of resources to ensure smooth and responsive user experiences.

2.4 Designing the user interface (UI)

Designing the user interface (UI) of a mobile application is a critical process that directly impacts user experience and engagement. Here are some key principles and steps to consider when designing a mobile app UI:



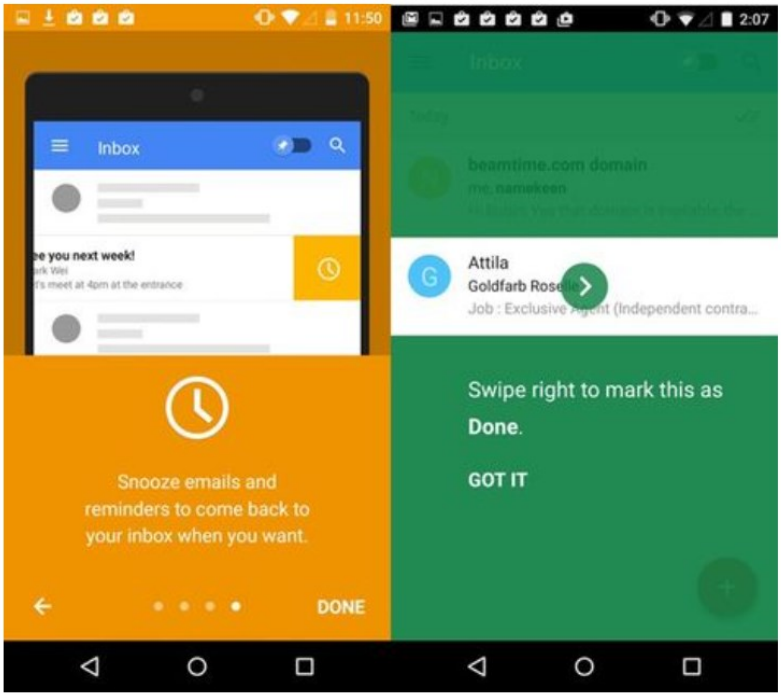
a. **User-Centric Approach:** The design should focus on the user's needs and preferences. Understanding the target audience, their behavior, and how they will interact with the app is fundamental.
b. **Intuitive Design:** The app should be easy to use and navigate. An intuitive design means users can find what they need without confusion or frustration.
c. **Consistency:** Maintain consistency throughout the app with fonts, colors, and button styles. This helps users become familiar with the app's environment and improves usability.
d. **Simplicity:** Keep the design simple and uncluttered. A minimalist approach often leads to a better user experience by reducing cognitive load.
e. **Visibility:** Ensure that all important options and features are visible and easily accessible. Users shouldn't have to search for what they need.
f. **Feedback:** Provide immediate feedback for user actions. If a user performs an action, the app should respond to confirm the action has been recognized.
g. **Accessibility:** Design with accessibility in mind to ensure that the app is usable by people with various disabilities. This includes color contrast, text size, and voice-over compatibility.
h. **Performance:** Optimize UI elements to ensure they do not hinder the app's performance. The app should be responsive and fast.



Testing: Conduct user testing to gather feedback on the UI design. This helps identify any issues or areas for improvement before the final release1.
Iterative Design: Be prepared to make changes based on user feedback. The design process should be iterative, allowing for continual refinement and improvement

Intuitive Design
The mobile app should be easy to use and self-explanatory to the user. So, the main focus of UI design is simplicity. It must provide all the necessary details. Besides, it should not confuse users.

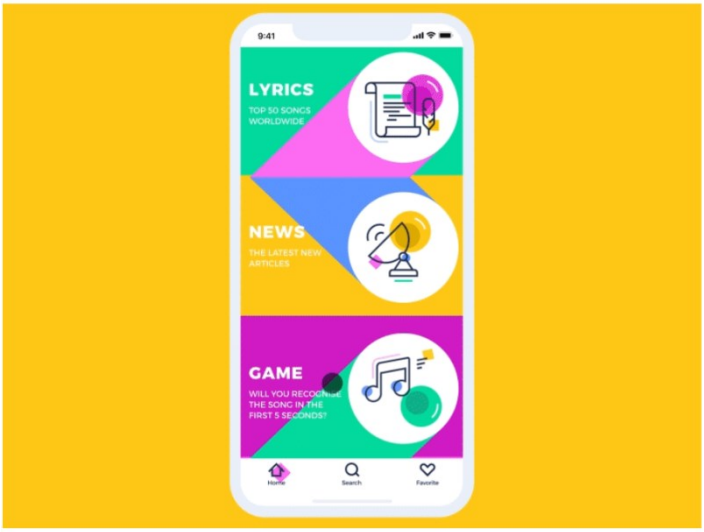
To achieve this, avoid excess details on a single screen of the UI. Ensure accommodating only the necessary features. Simplify complex elements and functionalities by adding details.



Appealing View
The app interface also needs to have a beautiful, pleasing, and comfortable view for the users. In mobile UI design, an additional consideration for an appealing look is that the UI should nicely fit into the small screen.

Prefer simple fonts and high-quality images as these are comfortable to the human eye and understandable to the human brain.

The color theme should match the business purpose and the target users as well. For example, an app that promotes environment-friendly products uses different shades of green color. Note that the color theme also varies for users from diverse cultural and demographic backgrounds. Designing the color scheme of the UI thus needs in-depth research.



4. Seek User Opinion and Feedback

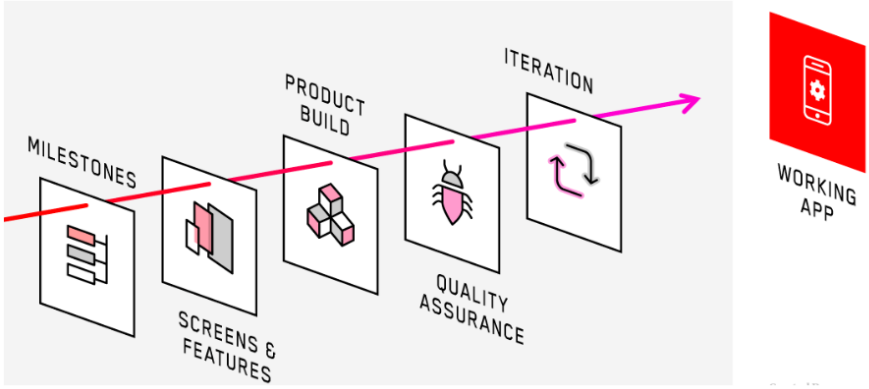
Evaluate the features of the app UI design before deciding what elements to add, exclude, or improve based on user-friendliness. Now, how to determine if the app features are user-friendly or not?

For this, seek opinions and feedback from test users who match the specific target user persona. Request them to use and provide inputs, catch their pain points, and modify the features accordingly.

5. Follow the Iterative Design Methodology

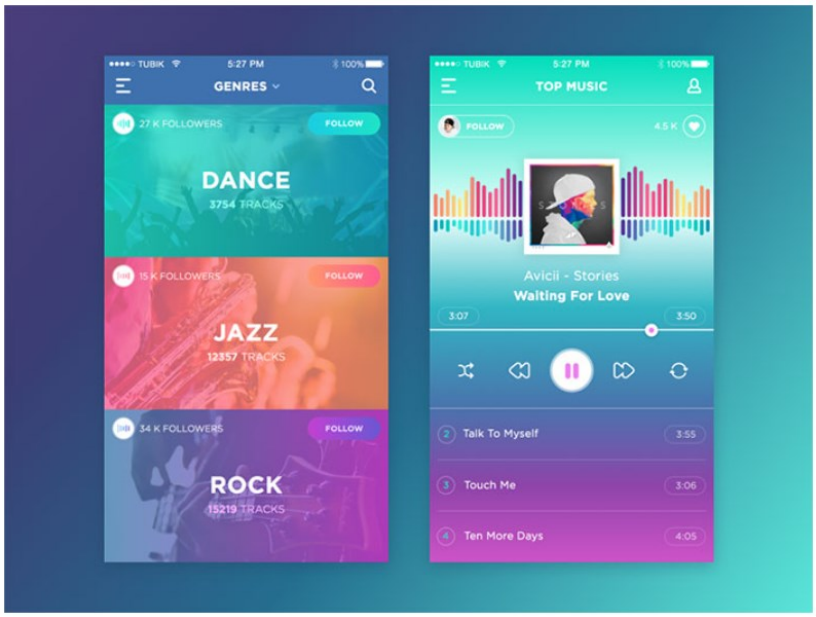
UI design is not a simple task, but involvement of a skilled mobile app developer enhances the process. The process is a virtuous circle of events, including prototyping, testing, analyzing, and improving. After making the changes, the same series of steps starts again from prototyping and goes on. This cyclic process is called iterative design methodology.

Multiple iterations produce the best UI design. Make use of the feedback and opinions gathered to create routine improvements and release new versions at intervals, as needed.



6. Follow and Incorporate the Latest Design Trends

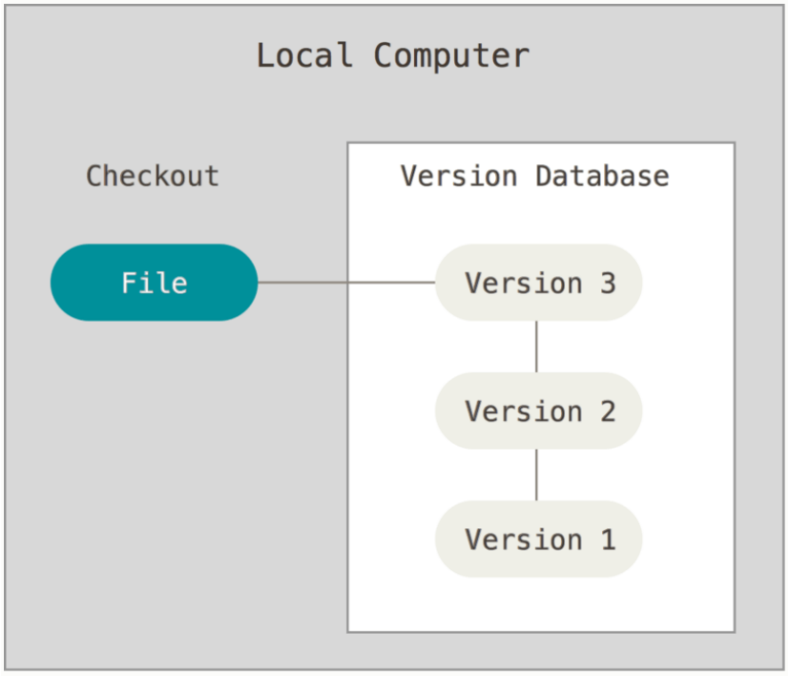
The UI design process is undergoing constant innovations to satisfy changing user needs. Thus, it is foremost important to keep informed about the latest UI design trends gaining popularity.



Wrapping Up

A fantastic, user-centric UI design is the key to the success of a mobile app. Following these simple UI design tips, techniques, and trends, makes it possible to create an amazing, user-centric mobile app experience that would take your business to new heights.

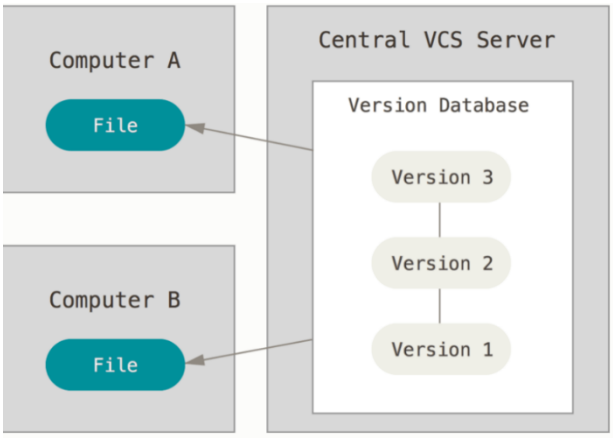
Finally, the last tip! Choose the right development partner for your business app that can offer mobile application development services tailored to your needs.



One of the most popular VCS tools was a system called RCS, which is still distributed with many computers today. RCS works by keeping patch sets (that is, the differences between files) in a special format on disk; it can then re-create what any file looked like at any point in time by adding up all the patches.

Centralized Version Control Systems

The next major issue that people encounter is that they need to collaborate with developers on other systems. To deal with this problem, Centralized Version Control Systems (CVCSs) were developed. These systems (such as CVS, Subversion, and Perforce) have a single server that contains all the versioned files, and a number of clients that check out files from that central place. For many years, this has been the standard for version control.

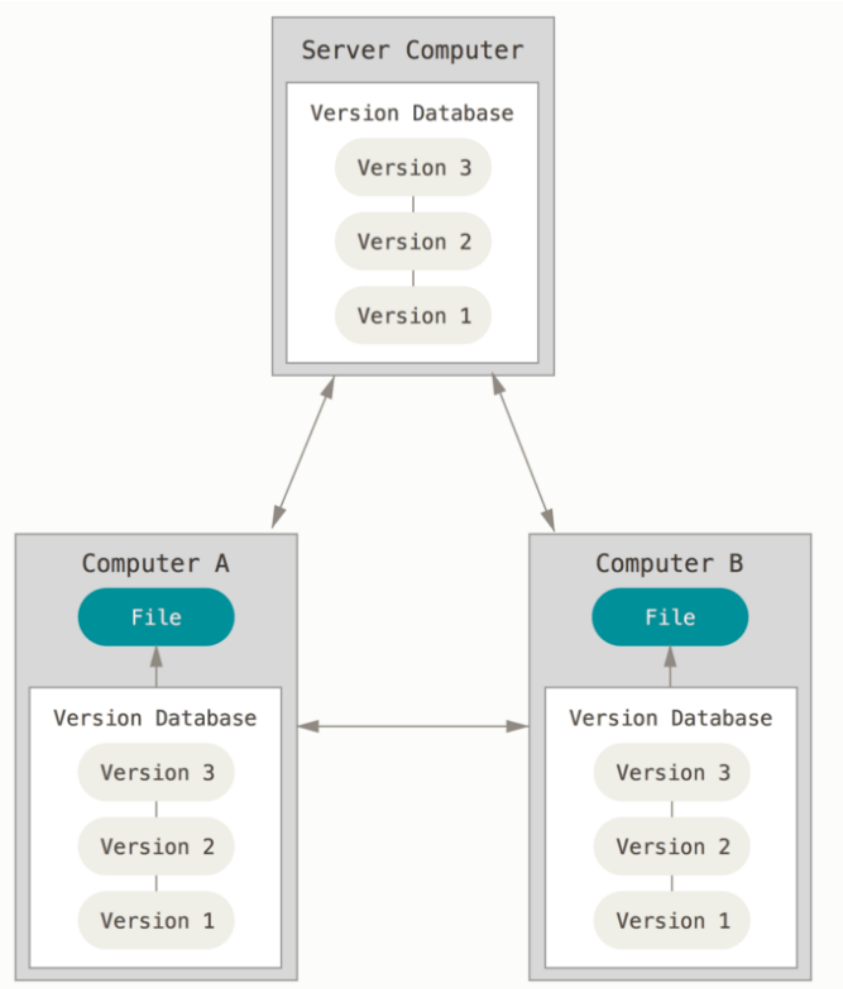


This setup offers many advantages, especially over local VCSs. For example, everyone knows to a certain degree what everyone else on the project is doing. Administrators have fine-grained control over who can do what, and it’s far easier to administer a CVCS than it is to deal with local databases on every client.

However, this setup also has some serious downsides. The most obvious is the single point of failure that the centralized server represents. If that server goes down for an hour, then during that hour nobody can collaborate at all or save versioned changes to anything they're working on. If the hard disk the central database is on becomes corrupted, and proper backups haven't been kept, you lose absolutely everything — the entire history of the project except whatever single snapshots people happen to have on their local machines. Local VCSs suffer from this same problem — whenever you have the entire history of the project in a single place, you risk losing everything.

Distributed Version Control Systems

This is where Distributed Version Control Systems (DVCSs) step in. In a DVCS (such as Git, Mercurial or Darcs), clients don't just check out the latest snapshot of the files; rather, they fully mirror the repository, including its full history. Thus, if any server dies, and these systems were collaborating via that server, any of the client repositories can be copied back up to the server to restore it. Every clone is really a full backup of all the data.



Furthermore, many of these systems deal pretty well with having several remote repositories they can work with, so you can collaborate with different groups of people in different ways

simultaneously within the same project. This allows you to set up several types of workflows that aren't possible in centralized systems, such as hierarchical models.

2.5 The use of Git as a source control system:

Version Control: Git helps in tracking and managing changes to code over time. This allows developers to revert back to previous versions if needed and understand the evolution of their codebase.

Collaboration: Multiple developers can work on the same codebase simultaneously. Git supports branching and merging, allowing teams to work independently and then combine their changes without conflicts.

Compatibility with Existing Systems: Git can be used standalone or integrated with existing systems and protocols like HTTP, FTP, or SSH.

Open Source: Git is open source, which means it's free to use, and its source code is available for anyone to inspect, modify, and enhance

2.6 Develop Android application.

Developing an Android application involves several steps, from setting up your development environment to publishing your app on the Google Play Store. Here is a comprehensive guide to get you started:

1. Set Up Your Development Environment

a. Install Android Studio

1. Download Android Studio: Visit the Android Studio download page.
2. Install Android Studio: Follow the installation instructions for your operating system.

b. Set Up Android Studio

1. Start Android Studio: Launch Android Studio after installation.
2. Configure SDK: Ensure that the Android SDK is installed and configured.
3. Create a New Project: Start a new Android Studio project.

2. Create a New Project

1. Open Android Studio: Launch Android Studio.
2. Start a New Android Studio Project:
    - Click on "Start a new Android Studio project."
    - Choose a project template (e.g., Empty Activity).
    - Name your project and set the package name.
    - Choose a save location.
    - Set the language (Java/Kotlin) and minimum API level.
    - Click "Finish."

3. Understand Project Structure

- app module: Contains all source files.
    - manifests: Contains AndroidManifest.xml.
    - java: Contains Java/Kotlin source files.
    - res: Contains resources such as layouts, strings, and images.

4. Create UI Elements

a. Define Layouts

Layouts define the structure of the user interface.

1. Open res/layout/activity_main.xml: This file defines the layout for the main activity.
2. Add UI Elements: Add UI elements such as buttons, text fields, etc., using XML.

xml

Copy code

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello, World!" />

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Click Me"
        android:layout_below="@id/textView"
        android:layout_marginTop="16dp" />

</RelativeLayout>
```

b. Create Activity

Activities are entry points for interacting with the user.

1. Open MainActivity.java or MainActivity.kt: This file contains the main logic for your activity.
2. Add Logic: Add logic to handle user interactions.

java

Copy code

```
package com.example.myapp;

import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {
    private TextView textView;
    private Button button;
```

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        textView = findViewById(R.id.textView);
        button = findViewById(R.id.button);

        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                textView.setText("Button Clicked!");
            }
        });
    }
}
```

5. Handle User Input

Handle user interactions such as button clicks, text input, etc.

java

Copy code

```
button.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        textView.setText("Button Clicked!");
    }
});
```

6. Manage App Resources

a. Strings

Define strings in res/values/strings.xml.

xml

Copy code

```
<resources>
    <string name="app_name">MyApp</string>
    <string name="hello_world">Hello, World!</string>
    <string name="button_click_me">Click Me</string>
</resources>
```

b. Colors

Define colors in res/values/colors.xml.

xml

Copy code

```
<resources>
    <color name="colorPrimary">#6200EE</color>
    <color name="colorPrimaryDark">#3700B3</color>
    <color name="colorAccent">#03DAC5</color>
</resources>
```

c. Styles
Define styles in res/values/styles.xml.
xml
Copy code
```
<resources>
   <style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
      <item name="colorPrimary">@color/colorPrimary</item>
      <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
      <item name="colorAccent">@color/colorAccent</item>
   </style>
</resources>
```
7. Add Dependencies
Add dependencies in build.gradle.
groovy
Copy code
```
dependencies {
   implementation 'androidx.appcompat:appcompat:1.2.0'
   implementation 'com.google.android.material:material:1.2.1'
   implementation 'androidx.constraintlayout:constraintlayout:2.0.4'
}
```
8. Run Your App
    1. Connect a Device or Use an Emulator: Connect your Android device or start an emulator.
    2. Run Your App: Click the "Run" button in Android Studio to install and run your app.
9. Debugging and Testing
    - Logcat: Use Logcat to view logs and debug your app.
    - Unit Tests: Write unit tests to test your app's logic.
    - Instrumentation Tests: Write instrumentation tests to test UI components.
10. Publish Your App
    1. Generate Signed APK: In Android Studio, go to Build > Generate Signed Bundle / APK.
    2. Sign Your App: Follow the steps to sign your app with a release key.
    3. Upload to Google Play: Create a developer account on the Google Play Console and follow the instructions to upload your APK.

Self-Check 2: Develop an Application in Android

1. What are Android Architecture Components?
   Answer:
2. How do you select a project model based on project requirements?
   Answer:
3. What is a key principle when implementing a selected project model?
   Answer:
4. Why is intuitive design important in mobile app UI?
   Answer:
5. How does Git function as a source control system?
   Answer:
6. What are the first steps in developing an Android application?
   Answer:
7. How does LiveData work within Android Architecture Components?
   Answer: What should be considered for UI design in mobile applications?
8. Can you explain the role of branches in Git?
   Answer:
9. What is the benefit of using Android Studio for app development?
   Answer:

Answer Key 2: Develop An Application In Android

1. **What are Android Architecture Components?**
   Answer: They are a set of libraries that help developers build robust, testable, and maintainable apps by providing a solid app architecture.
2. **How do you select a project model based on project requirements?**
   Answer: You evaluate the project's complexity, methodology, resource availability, and stakeholder input to choose a model that aligns with the project's goals and constraints.
3. **What is a key principle when implementing a selected project model?**
   Answer: One key principle is the Separation of Concerns, ensuring different parts of the app have distinct responsibilities.
4. **Why is intuitive design important in mobile app UI?**
   Answer: Intuitive design is crucial because it makes the app easy to use and navigate, enhancing the user experience.
5. **How does Git function as a source control system?**
   Answer: Git tracks changes to files over time, allowing multiple developers to work on the same codebase and merge changes without conflicts.
6. **What are the first steps in developing an Android application?**
   Answer: The first steps include setting up the development environment, learning the basics of Kotlin, and understanding Android's app components.
7. **How does LiveData work within Android Architecture Components?**
   Answer: LiveData is an observable data holder class that is lifecycle-aware, ensuring UI components only update when they are in an active lifecycle state.
8. **What should be considered for UI design in mobile applications?**
   Answer: Considerations include user-centric design, simplicity, consistency, visibility, feedback, accessibility, performance, and iterative design
9. **Can you explain the role of branches in Git?**
   Answer: Branches in Git allow developers to work on features or fixes in isolation, then merge those changes back into the main branch when ready.
10. **What is the benefit of using Android Studio for app development?**
    Answer: Android Studio provides a comprehensive suite of tools and features that streamline the app development process, from coding to testing and debugging

Task Sheet 2.1: Develop An Application in Android

1. **Identify Android Architecture Components**
   Task: Research and list the Android Architecture Components.
   Goal: Understand the role of each component in building a robust Android app.
   Expected Outcome: A document summarizing each architecture component and its use.
2. **Select Project Model as per Project Requirement**
   Task: Analyze a hypothetical app's requirements and select an appropriate project management model.
   Goal: Learn to align project management methodologies with project needs.
   Expected Outcome: A report detailing the selected project model and the rationale behind the choice.
3. **Implement Key Principles of Selected Model**
   Task: Create a plan to implement the key principles of the chosen project model in the development process.
   Goal: Apply project management principles to an Android app development workflow.
   Expected Outcome: An implementation plan that incorporates the key principles effectively.
4. **Design User Interface of a Mobile Application**
   Task: Design a user interface for a simple Android app.
   Goal: Practice UI design principles to enhance user experience.
   Expected Outcome: Mockups or wireframes of the app's user interface.
5. **Use Git as a Source Control System**
   Task: Set up a Git repository for the Android app project and demonstrate basic Git operations.
   Goal: Become proficient in using Git for version control.
   Expected Outcome: A link to the Git repository with a commit history showcasing various operations.
6. **Develop Android Application**
   Task: Develop a basic Android application that incorporates the identified architecture components.
   Goal: Apply theoretical knowledge to create a functional Android app.
   Expected Outcome: Source code of the developed Android app and a brief user manual.

Job Sheet 1

Job Name: Create An Android Project Based On The Following User Stories.

Working Procedure/ Steps:

1. Follow OSH and use Personal Protective Equipment (PPE).
2. Check Electricity & Internet Connections to your Computer.
3. Start the Computer.
4. Create a folder on the desktop with your registration number and name.
5. Collect the resources from your assessor as per the job requirement.
6. Open Android Studio, Emulator or real Android device.
7. Create a new Android Studio project as per specification sheet.
8. Create necessary file for Location Based Garbage Management System for Smart City.
9. Finish your work as per specification sheet
10. Run app on Emulator or Real device
11. Check functionality of the newly developed apps.
12. Show your work to the trainer.
13. Close Emulator, Android Studio as per standard procedure.
14. Turn off the computer and clean your workplace.

Specification Sheet

Job Name: Create an Android project based on the following user stories.

User Stories

Location Based Garbage Management System for Smart City

There are many Android project ideas for college students that focus on improving the efficiency of city services. One such idea is a location-based garbage management system. This system would use GPS to track the location of garbage trucks and ensure that they are picking up trash from all parts of the city. The system could also provide real-time updates on the level of garbage in each truck, so the city can adjust its collection schedule as needed. In addition, the system could send alerts to residents when their trash has been picked up, so that they can put out their garbage on the correct day.

1.1 User Registration and Authentication:

- User registration via email, phone number, or social media accounts.
- Secure login and logout functionality.
- Profile management for users, including personal details and roles (citizen, waste collector, admin).

1.2 Real-Time Location Tracking:

- GPS-based tracking for waste collection vehicles.
- Display real-time locations of garbage bins on a city map.
- Route optimization for waste collection vehicles.

1.3 Garbage Bin Monitoring:

- Sensor integration for real-time bin status (empty, half-full, full).
- Notifications for bins that require immediate attention.
- Historical data and trends for bin usage.

1.4 Waste Collection Scheduling:

- Automated scheduling based on bin status and location.
- Manual scheduling options for special collections or events.
- Notifications for scheduled pickups to waste collectors and citizens.

1.5 Reporting and Feedback:

- Allow citizens to report overflowing bins or illegal dumping.
- Feedback system for citizens to rate waste management services.
- Ticketing system for tracking reported issues and resolutions.

1.6 Notifications and Alerts:

- Push notifications for bin status updates, collection schedules, and service alerts.
- Customizable notification preferences for users.

1.7 Data Analytics and Insights:

- Dashboard for visualizing waste collection data and performance metrics.
- Reports on collection efficiency, response times, and bin utilization.
- Predictive analytics for future waste management needs.

2 .Technical Requirements:

2.1 Platform:

- Android OS, compatible with versions [specify version range].

2.2 Performance:

- Optimized for speed and efficiency.

- Low battery and data consumption.

2.3 Security:

- Data encryption for user information and transactions.
- Regular security updates and compliance with industry standards.

2.4 Scalability:

- Ability to handle a growing number of users and garbage bins.
- Modular architecture for easy updates and feature additions.

3. Design Requirements:

3.1 User Interface (UI):

- Intuitive and user-friendly design.
- Consistent theme and branding.
- Responsive design for various screen sizes.

3.2 User Experience (UX):

- Smooth navigation and minimal steps to complete tasks.
- Clear call-to-actions and feedback for user interactions.
- Accessibility features for users with disabilities.

4. Development and Testing:

4.1 Development Tools:

- Android Studio, Java/Kotlin for Android development.
- RESTful APIs for backend integration.
- IoT integration for sensor data collection and management.

4.2 Testing:

- Comprehensive testing for functionality, performance, and security.
- User acceptance testing (UAT) before final release.
- Beta testing with a select group of users.

5. Maintenance and Support:

5.1 Updates:

- Regular updates for new features, improvements, and bug fixes.
- User feedback integration for continuous improvement.

5.2 Support:

- Dedicated support team for troubleshooting and assistance.
- Documentation and training materials for users and administrators. .

List of required PPE

S/N	Name of PPE	Specification	Unit	Required Quantity
01	Ergonomic Chair	Wood and foam	No	1
02	Eye protective glass	Metal and Glass	No	1

List of required Materials

S/N	Name of Materials	Specification	Unit	Required Quantity

01	Java JDK	Minimum JDK 11	…	1
02	Android Studio	Latest version	…	1

List of required Tools & Equipment's

S/N	Name	Specification	Unit	Required Quantity
01	Personal Computer or Laptop	Minimum Core i5 7th gen Processor	No	1
04	Internet connection		…	1

Job Sheet 2

Job Name: Create An Android Project on Onroad Vehicle Breakdown Help Assistance.

Time: 2 hours

Working Procedure/ Steps:

1. Follow OSH and use Personal Protective Equipment (PPE).
2. Check Electricity & Internet Connections to your Computer.
3. Start the Computer.
4. Create a folder on the desktop with your registration number and name.
5. Collect the resources from your assessor as per the job requirement.
6. Open Android Studio, Emulator or real Android device.
7. Create a new Android Studio project as per specification sheet.
8. Follow user stories to create the project “OnRoad Vehicle Breakdown Help Assistance”
9. Create necessary file for OnRoad Vehicle Breakdown Help Assistance.
10. Finish your work as per specification sheet
11. Run app on Emulator or Real device
12. Check functionality of the newly developed apps.
13. Show your work to the trainer.
14. Close Emulator, Android Studio as per standard procedure.
15. Turn off the computer and clean your workplace.

Specification Sheet

Job Name: Create an Android project on OnRoad Vehicle Breakdown Help Assistance..

User Stories

OnRoad Vehicle Breakdown Help Assistance

Imagine you are driving on the highway, and your car suddenly breaks down. You're miles away from the nearest town and you have no cell service. What do you do? OnRoad Vehicle Breakdown Help Assistance is an Android project that students can create to help you in just such a situation. The app uses GPS to pinpoint your location and then sends out an SOS signal to a preselected list of contacts. The contacts can then track your location and provide assistance, whether that's coming to pick you up or calling a tow truck. The app can also include a database of nearby service stations, hotels, and restaurants, so you can find help even if you're in an unfamiliar area.

Job Specification Information or App Features

1.1 User Registration and Authentication:

- User registration via email, phone number, or social media accounts.
- Secure login and logout functionality.
- Profile management for users, including personal details, vehicle information, and payment methods.

1.2 Real-Time Location Tracking:

- GPS-based tracking for user location.
- Display nearby service providers on a map.
- Real-time tracking of assistance vehicle en route to the user.

1.3 Service Request:

- Simple and intuitive interface to request breakdown assistance.
- Option to select the type of breakdown (e.g., flat tire, engine trouble, battery issue).
- Estimated time of arrival (ETA) for the service provider.

1.4 Service Provider Management:

- Database of certified service providers.
- Ratings and reviews for service providers.
- Service provider profiles with contact information, services offered, and rates.

1.5 Communication:

- In-app chat and calling feature between users and service providers.
- Automatic SMS updates for request confirmation, service dispatch, and arrival.

1.6 Payment Integration:

- Integration with popular payment gateways for secure transactions.
- Support for multiple payment options (credit/debit card, digital wallets).
- Electronic receipts for completed services.

1.7 Notifications and Alerts:

- Push notifications for request updates, ETA changes, and service completion.
- Customizable notification preferences for users.

1.8 Emergency Contacts:

- Option to add emergency contacts.
- Automatic notification to emergency contacts in case of a breakdown.

1.9 Help and Support:

- In-app help center with FAQs and troubleshooting guides.
- Customer support chat for real-time assistance.
- Feedback system for users to rate and review the service.

2. Technical Requirements:

2.1 Platform:

- Android OS, compatible with versions [specify version range].

2.2 Performance:

- Optimized for speed and efficiency.
- Low battery and data consumption.

2.3 Security:

- Data encryption for user information and transactions.
- Regular security updates and compliance with industry standards.

2.4 Scalability:

- Ability to handle a growing number of users and garbage bins.
- Modular architecture for easy updates and feature additions.

3. Design Requirements:

3.1 User Interface (UI):

- Intuitive and user-friendly design.
- Consistent theme and branding.
- Responsive design for various screen sizes.

3.2 User Experience (UX):

- Smooth navigation and minimal steps to complete tasks.
- Clear call-to-actions and feedback for user interactions.
- Accessibility features for users with disabilities.

4. Development and Testing:

4.1 Development Tools:

- Android Studio, Java/Kotlin for Android development.
- RESTful APIs for backend integration.
- IoT integration for sensor data collection and management.

4.2 Testing:

- Comprehensive testing for functionality, performance, and security.
- User acceptance testing (UAT) before final release.
- Beta testing with a select group of users.

5. Maintenance and Support:

5.1 Updates:

- Regular updates for new features, improvements, and bug fixes.
- User feedback integration for continuous improvement.

5.2 Support:

- Dedicated support team for troubleshooting and assistance.
- Documentation and training materials for users and administrators.

.

List of required PPE

S/N	Name of PPE	Specification	Unit	Required Quantity
01	Ergonomic Chair	Wood and foam	No	1
02	Eye protective glass	Metal and Glass	No	1

List of required Materials:

S/N	Name of Materials	Specification	Unit	Required Quantity
01	Java JDK	Minimum JDK 11	…	1
02	Android Studio	Latest version	…	1

List of required Tools & Equipment's:

S/N	Name	Specification	Unit	Required Quantity
01	Personal Computer or Laptop	Minimum Core i5 7th gen Processor	No	1
04	Internet connection		…	1

Job Sheet 3

Job Name: Create An Android Project On Agri Shop: Farmers Online Selling Application.

Time: 2 hours

Working Procedure/ Steps:

1. Follow OSH and use Personal Protective Equipment (PPE).
2. Check Electricity & Internet Connections to your Computer.
3. Start the Computer.
4. Create a folder on the desktop with your registration number and name.
5. Collect the resources from your assessor as per the job requirement.
6. Open Android Studio, Emulator or real Android device.
7. Create a new Android Studio project as per specification sheet.
8. Follow user stories to create the project “Agri Shop: Farmers Online Selling Application”
9. Create necessary file for Agri Shop: Farmers Online Selling Application.
10. Finish your work as per specification sheet
11. Run app on Emulator or Real device
12. Check functionality of the newly developed apps.
13. Show your work to the trainer.
14. Close Emulator, Android Studio as per standard procedure.
15. Turn off the computer and clean your workplace.

Specification Sheet

Job Name: Create an Android project on Agri Shop: Farmers Online Selling Application.

User Stories

Agri Shop: Farmers Online Selling Application

The Agri Shop application would allow farmers to sell their products directly to consumers through an online platform. This would be one of the best Android project topics for college students as it would provide a real-world opportunity to develop an e-commerce application. In addition to selling agricultural products, an Agri shop can also offer information and resources on topics like sustainable farming practices, organic gardening, and more. There are many challenging aspects to this project, from designing the user interface to integrating payment methods. However, the rewards could be significant, both for the farmers who use the application and for the students who develop it.

Job Specification Information/App Features:

1.1 User Registration and Authentication:

- User registration via email, phone number, or social media accounts.
- Secure login and logout functionality.
- Profile management for users, including personal details and delivery addresses.

1.2 Product Catalog:

- Categorized product listings.
- Detailed product descriptions, images, and specifications.
- Search functionality with filters (price, category, brand, etc.).
- Product reviews and ratings.

1.3 Shopping Cart and Checkout:

- Add/remove items from the shopping cart.
- View cart summary with itemized pricing.
- Apply discount codes and promotions.
- Secure checkout process with multiple payment options (credit/debit card, digital wallets, bank transfers).

1.4 Order Management:

- Order history and status tracking.
- Notifications for order confirmation, dispatch, and delivery.
- Option to cancel or return orders.

1.5 Payment Integration:

- Integration with popular payment gateways for secure transactions.
- Support for multiple currencies.
- Generation of electronic receipts.

1.6 Delivery and Logistics:

- Integration with delivery service providers for real-time tracking.
- Estimated delivery times and shipping cost calculation.
- Option for users to select preferred delivery slots.

1.7 Farmer Support and Community:

- Access to farming guides, tutorials, and best practices.
- Forum or chat feature for community discussions.
- Expert advice and Q&A section.

1.8 Notifications and Alerts:

- Push notifications for order updates, promotions, and new arrivals.
- Customizable notification preferences.

2. Technical Requirements:

2.1 Platform:

- Android OS, compatible with versions [specify version range].

2.2 Performance:

- Optimized for speed and efficiency.
- Low battery and data consumption.

2.3 Security:

- Data encryption for user information and transactions.
- Regular security updates and compliance with industry standards.

2.4 Scalability:

- Ability to handle a growing number of users and garbage bins.
- Modular architecture for easy updates and feature additions.

3. Design Requirements:

3.1 User Interface (UI):

- Intuitive and user-friendly design.
- Consistent theme and branding.
- Responsive design for various screen sizes.

3.2 User Experience (UX):

- Smooth navigation and minimal steps to complete tasks.
- Clear call-to-actions and feedback for user interactions.
- Accessibility features for users with disabilities.

4. Development and Testing:

4.1 Development Tools:

- Android Studio, Java/Kotlin for Android development.
- RESTful APIs for backend integration.

4.2 Testing:

- Comprehensive testing for functionality, performance, and security.
- User acceptance testing (UAT) before final release.
- Beta testing with a select group of users.

5. Maintenance and Support:

5.1 Updates:

- Regular updates for new features, improvements, and bug fixes.
- User feedback integration for continuous improvement.

5.2 Support:

- Dedicated support team for troubleshooting and assistance.
- Documentation and training materials for users and administrators.

.

List of required PPE:

S/N	Name of PPE	Specification	Unit	Required Quantity
01	Ergonomic Chair	Wood and foam	No	1
02	Eye protective glass	Metal and Glass	No	1

List of required Materials:

S/N	Name of Materials	Specification	Unit	Required Quantity
01	Java JDK	Minimum JDK 11	…	1
02	Android Studio	Latest version	…	1

List of required Tools & Equipment's:

S/N	Name	Specification	Unit	Required Quantity
01	Personal Computer or Laptop	Minimum Core i5 7th gen Processor	No	1
04	Internet connection		…	1

Job Sheet 4

Job Name: Create An Android Project On Women's Security With Sms Alert Based Android App.

Time: 2 hours

Working Procedure/ Steps:

1. Follow OSH and use Personal Protective Equipment (PPE).
2. Check Electricity & Internet Connections to your Computer.
3. Start the Computer.
4. Create a folder on the desktop with your registration number and name.
5. Collect the resources from your assessor as per the job requirement.
6. Open Android Studio, Emulator or real Android device.
7. Create a new Android Studio project as per specification sheet.
8. Follow user stories to create the project “Women's Security with SMS Alert based android app”
9. Create necessary file for Women's Security with SMS Alert based android app.
10. Finish your work as per specification sheet
11. Run app on Emulator or Real device
12. Check functionality of the newly developed apps.
13. Show your work to the trainer.
14. Close Emulator, Android Studio as per standard procedure.
15. Turn off the computer and clean your workplace.

Specification Sheet

Job Name: Create an Android project on Women's Security with SMS Alert based android app.

User Stories

Women's Security with SMS Alert based android app

Android project ideas for college students are many but women's security should be given prime importance. Women security with SMS alert based android app would allow women to send an SMS alert to a designated contact in an emergency. The app would also have a GPS tracker that would help to track the location of the user in real-time. This would ensure that help is always available in case of an emergency. Moreover, the app would also provide a list of safe places and helplines that could be useful in an emergency.

Job Specification Information:

1. Job Specification Information or App Features:

1.1 User Registration and Authentication:

- User registration via email or phone number.
- Secure login and logout functionality.
- Profile management including personal details and emergency contacts.

1.2 Emergency SMS Alerts:

- Quick access emergency button to send instant SMS alerts.
- Customizable alert messages.
- Automatic inclusion of real-time location coordinates in SMS alerts.
- Option to send alerts to multiple predefined contacts and emergency services.

1.3 Real-Time Location Tracking:

- GPS-based tracking to provide accurate user location.
- Continuous location sharing with emergency contacts during an alert.
- Real-time tracking visible to selected contacts on a map interface.

1.4 Safety Features:

- Timer-based safety check-in alerts to ensure regular check-ins from the user.
- Fake call feature to simulate an incoming call for avoiding uncomfortable situations.
- Geo-fencing to set safe zones with alerts when the user enters or exits these zones.

1.5 Communication:

- In-app chat feature for real-time communication with emergency contacts.
- Integration with local law enforcement agencies for direct communication in emergencies.

1.6 Notifications and Alerts:

- Push notifications for safety tips, check-in reminders, and app updates.
- Customizable notification preferences for users.

1.7 Help and Support:

- In-app help center with safety tips and emergency procedures.
- Customer support chat for real-time assistance.
- Feedback system for users to report issues and suggest improvements.

2 .Technical Requirements:

2.1 Platform:

- Android OS, compatible with versions [specify version range].

2.2 Performance:

- Optimized for speed and efficiency.
- Low battery and data consumption.

2.3 Security:

- Data encryption for user information and transactions.
- Regular security updates and compliance with industry standards.

2.4 Scalability:

- Ability to handle a growing number of users and garbage bins.
- Modular architecture for easy updates and feature additions.

3. Design Requirements:

3.1 User Interface (UI):

- Intuitive and user-friendly design.
- Consistent theme and branding.
- Responsive design for various screen sizes.

3.2 User Experience (UX):

- Smooth navigation and minimal steps to complete tasks.
- Clear call-to-actions and feedback for user interactions.
- Accessibility features for users with disabilities.

4. Development and Testing:

4.1 Development Tools:

- Android Studio, Java/Kotlin for Android development.
- RESTful APIs for backend integration.

4.2 Testing:

- Comprehensive testing for functionality, performance, and security.
- User acceptance testing (UAT) before final release.
- Beta testing with a select group of users.

5. Maintenance and Support:

5.1 Updates:

- Regular updates for new features, improvements, and bug fixes.
- User feedback integration for continuous improvement.

5.2 Support:

- Dedicated support team for troubleshooting and assistance.
- Documentation and training materials for users and administrators.

.

List of required PPE:

S/N	Name of PPE	Specification	Unit	Required Quantity
01	Ergonomic Chair	Wood and foam	No	1
02	Eye protective glass	Metal and Glass	No	1

List of required Materials:

S/N	Name of Materials	Specification	Unit	Required Quantity
01	Java JDK	Minimum JDK 11	…	1
02	Android Studio	Latest version	…	1

List of required Tools & Equipment's:

S/N	Name	Specification	Unit	Required Quantity
01	Personal Computer or Laptop	Minimum Core i5 7th gen Processor	No	1
04	Internet connection		…	1

Job Sheet 5

Job Name: Create An Android Project on Covid-19 Online Test Results & Availability Booking Of Covid Hospital.

Time: 2 hours

Working Procedure/ Steps:

1. Follow OSH and use Personal Protective Equipment (PPE).
2. Check Electricity & Internet Connections to your Computer.
3. Start the Computer.
4. Create a folder on the desktop with your registration number and name.
5. Collect the resources from your assessor as per the job requirement.
6. Open Android Studio, Emulator or real Android device.
7. Create a new Android Studio project as per specification sheet.
8. Follow user stories to create the project “COVID-19 Online Test Results & availability booking of Covid Hospital”
9. Create necessary file for COVID-19 Online Test Results & availability booking of Covid Hospital.
10. Finish your work as per specification sheet
11. Run app on Emulator or Real device
12. Check functionality of the newly developed apps.
13. Show your work to the trainer.
14. Close Emulator, Android Studio as per standard procedure.
15. Turn off the computer and clean your workplace.

Specification Sheet

Job Name: Create an Android project on COVID-19 Online Test Results & availability booking of Covid Hospital.

User Stories

COVID-19 Online Test Results & availability booking of Covid Hospital

While there are a limited number of testing centers available, many people are turning to online resources to book appointments. However, there is a lot of confusion about how these online tests work and what the results mean. One of the best Android project ideas for final year students could be creating an app that provides information about online COVID-19 tests, including how to book an appointment and interpret the results. This app could also provide information about nearby testing centers and hospitals with available beds for COVID-19 patients.

Job Specification Information:
Job Specification Information or App Features:
1.1 User Registration and Authentication:

- User registration via email, phone number, or social media accounts.
- Secure login and logout functionality.
- Profile management including personal details and health information.

1.2 COVID-19 Test Results:

- Integration with testing labs for automatic retrieval of test results.
- Secure access to COVID-19 test results.
- Notifications for new test results.
- Option to download and share test result reports.

1.3 Hospital Bed Availability:

- Real-time tracking of hospital bed availability (ICU, general, isolation wards).
- Search functionality to find hospitals with available beds based on location.
- Filter options for hospital type, distance, and bed type.

1.4 Booking and Appointments:

- Option to book hospital beds directly through the app.
- Appointment scheduling for COVID-19 testing and consultations.
- Confirmation and reminder notifications for appointments and bookings.

1.5 Emergency Contact and Support:

- Access to emergency contact numbers for COVID-19 helplines.
- In-app chat feature for real-time support and assistance.
- FAQs and guidelines for COVID-19 symptoms, testing, and care.

1.6 Notifications and Alerts:

- Push notifications for test result updates, appointment reminders, and critical alerts.
- Customizable notification preferences for users.

1.7 Data Privacy and Security:

- Data encryption for user information and test results.
- Compliance with health data privacy regulations (e.g., HIPAA, GDPR).

2 .Technical Requirements:
2.1 Platform:

- Android OS, compatible with versions [specify version range].

2.2 Performance:

- Optimized for speed and efficiency.
- Low battery and data consumption.

2.3 Security:

- Data encryption for user information and transactions.
- Regular security updates and compliance with industry standards.

2.4 Scalability:

- Ability to handle a growing number of users and garbage bins.
- Modular architecture for easy updates and feature additions.

3. Design Requirements:

3.1 User Interface (UI):

- Intuitive and user-friendly design.
- Consistent theme and branding.
- Responsive design for various screen sizes.

3.2 User Experience (UX):

- Smooth navigation and minimal steps to complete tasks.
- Clear call-to-actions and feedback for user interactions.
- Accessibility features for users with disabilities.

4. Development and Testing:

4.1 Development Tools:

- Android Studio, Java/Kotlin for Android development.
- RESTful APIs for backend integration with labs and hospitals.

4.2 Testing:

- Comprehensive testing for functionality, performance, and security.
- User acceptance testing (UAT) before final release.
- Beta testing with a select group of users.

5. Maintenance and Support:

5.1 Updates:

- Regular updates for new features, improvements, and bug fixes.
- User feedback integration for continuous improvement.

5.2 Support:

- Dedicated support team for troubleshooting and assistance.
- Documentation and training materials for users and administrators.

.

List of required PPE:

S/N	Name of PPE	Specification	Unit	Required Quantity
01	Ergonomic Chair	Wood and foam	No	1
02	Eye protective glass	Metal and Glass	No	1

List of required Materials:

S/N	Name of Materials	Specification	Unit	Required Quantity
01	Java JDK	Minimum JDK 11	…	1
02	Android Studio	Latest version	…	1

List of required Tools & Equipment's:

S/N	Name	Specification	Unit	Required Quantity
01	Personal Computer or Laptop	Minimum Core i5 7th gen Processor	No	1
04	Internet connection		…	1

Learning Outcome 3: Develop user story

Assessment Criteria:

1. User story is explained.
2. Story estimated.
3. User stories of a project work is defined.
4. Project management tool is used.
5. Project stories are made.

Content:

1. User story.
2. Story estimation.
3. User stories of a project work.
4. Project management tool.
5. Making Project stories.

Resources Required/ Conditions:

The trainees must be provided with the following:

- Handouts or reference materials/books/ CBLMs on the above stated contents
- PCs/printers or laptop/printer with internet access
- Digital projector and Screen
- Bond paper
- Ball pens/pencils and other office supplies and materials
- Relevant learning materials
- Workplace or simulated environment

Methodologies

- Lecture/discussion
- Demonstration/application
- Presentation
- Blended delivery methods

Assessment Methods

- Written test
- Demonstration
- Observation with checklist
- Oral questioning
- Portfolio

Learning Experience 3: Use Presentation Application

In order to achieve the objectives stated in this learning guide, you must perform the learning steps below. Beside each step are the resources or special instructions you will use to accomplish the corresponding activity.

Learning Steps	Resources specific instructions
1. Student will ask the instructor about Develop user story	1. Instructor will provide the learning materials "**Performing Project Work with Android"**
2. Read the Information sheet/s	2. Information Sheet No 3: Develop user story
3. Complete the Self Checks & Check answer sheets.	3. Self-Check/s Self-Check No 3: Develop user story Answer key No. 3: Develop user story
4. Read the Job Sheet and Specification Sheet and perform job	Job- Sheet No 3-1: Develop user story Specification Sheet 3-1: Develop user story

Information Sheet 3: Develop User Story

Learning Objectives:

After completion of this information sheet, the learners will be able to:

3.1 Explain user story.
3.2 Estimate Story.
3.3 Define user stories of a project work.
3.4 Use Project management tool.
3.5 Make Project stories.

3.1 User Story

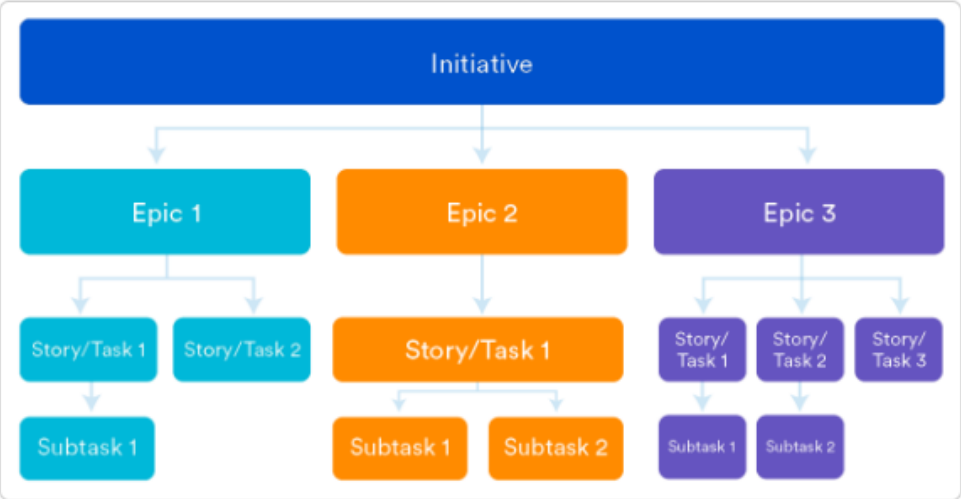
A user story is a concise, informal description of a feature or functionality from the perspective of an end user. It captures the essence of what the user needs and why they need it.

Key components of a user story:

Role (As a [user persona]):
Describes who the user is or their role in the system (e.g., developer, customer, admin).
Feature (I want to [complete an action]):
Specifies the desired action or functionality the user wants to perform (e.g., log in, search for products).
Benefit (So that I can [get some value]):
Explains the value or benefit the user expects from the feature (e.g., save time, improve efficiency).
User stories are commonly used in agile development to facilitate communication between stakeholders, product owners, and development teams. They help prioritize work and ensure alignment with user needs. User stories are a fundamental concept in Agile software development. They provide a way to express development tasks from the perspective of end users or customers. It is an informal, concise explanation of a software feature.
It's written from the viewpoint of the end user or customer.
User stories focus on why and what—not technical details.

Structure of a User Story:
A typical user story follows this format:
"As a [user persona], I want [to perform an action] so that [I achieve a specific value]."
For example:
"As a customer, I want to search for products by category so that I can quickly find relevant items."
Why Use User Stories?
They provide context for development teams.
Teams understand why they're building something and what value it brings.
User stories help prioritize work and drive collaboration.

Remember, user stories break down big projects into actionable steps, making development more manageable and user-focused!



3.2 Story point estimation

Story point estimation is a technique used in Agile project management to replace traditional time or monetary estimates. Instead of estimating tasks in hours or days, teams assign story points to user stories based on their complexity and effort required for implementation.

Here's how it works:

Story points are units of measure that estimate the overall effort needed to complete a user story or any other work item.

They represent the relative size and complexity of a task rather than an absolute time value.

How Does It Work?

Teams start by assigning a base story (usually the simplest) with a certain number of story points (e.g., 1 point).

Then, other stories are estimated relative to this base story. For example:

If a more complex story is twice as challenging, it might be assigned 2 story points.

If another story is even more complex, it could receive 3 or more points.

Collaboration and Perspective.

3.3 User stories are crucial in Android app development:

A. **User-Centric Approach:** They ensure the app prioritizes user needs and pain points, leading to a more valuable product.

B. **Enhanced Communication:** Acting as a common language, user stories foster better understanding between developers, designers, and stakeholders.

C. **Prioritization and Flexibility:** User stories aid in prioritizing features based on user value and business goals, allowing for adaptation throughout development.

D. **Validation and Testing:** They provide a foundation for creating acceptance criteria, which helps gauge if the app meets user expectations.

E. **Structure of a user story:**

As a [user persona], I want to [action] so that [benefit].

For example:

As a fitness enthusiast, I want to track my daily steps and workouts within the app so that I can monitor my progress and stay motivated.

F. **User stories can be further broken down into smaller, more manageable tasks. This helps with:**
   a. **Work Breakdown:** Complex functionalities are divided into smaller chunks, simplifying the development process.
   b. **Estimation and Planning:** Development teams can estimate the effort required for each task, facilitating better planning of development cycles (sprints).

3.4 Project management tool

When it comes to project management tools for Android application development, there are several options available.

Below discussed on Project management tool:

Android Studio:

Android Studio is the official Integrated Development Environment (IDE) for Android app development.

It's based on IntelliJ IDEA and provides a comprehensive set of tools for coding, debugging, and building Android apps.

Key features include:

Gradle Build System: Automates the build process.

Resource Manager: Helps manage app resources like layouts, strings, and images.

Emulator: Allows testing on virtual devices.

Code Editor: Supports Java, Kotlin, and other languages.

Asana (Web and Android App):

Asana is a popular project management tool that balances task management with team collaboration.

You can organize, assign, and complete tasks from your Android device.

Features include task lists, due dates, and team communication.

Remember, choosing the right project management tool depends on your team's needs and preferences. Android Studio is essential for development, while tools like Asana enhance collaboration!

When it comes to Android app development, creating project stories (also known as user stories) is essential for effective communication and successful project execution. Let's dive into the details:

3.5 Project stories

A. To Make a Project stories

a. **Define User Personas:** Before diving into stories, identify your target users. Create user personas representing different user groups who will interact with your app. Each persona should have a clear description of their demographics, goals, and pain points.

b. **Brainstorm User Needs:** Once you understand your user personas, brainstorm the functionalities they'd need to achieve their goals within the app. This could involve workshops or individual brainstorming sessions.
c. **Craft User Stories:** Using the user needs identified, formulate user stories following the INVEST principle:
d. **Independent:** Each story should be self-contained and achievable independently.
e. **Negotiable:** Stories can be refined and adjusted based on discussions and changing needs.
f. **Valuable:** Each story must deliver value to the user and the overall app.
g. **Estimable:** The effort required to complete the story should be reasonably assessable.
h. **Sized Appropriately:** Stories should be small enough to be completed within a sprint (development cycle) but large enough to deliver a meaningful piece of functionality.
i. **Testable:** Acceptance criteria should be defined for each story to ensure successful implementation.

B. Steps of making a Project stories

When creating user stories for an Android app development project, follow these steps:

C. Establish the "You" Behind the App:

Understand your vision, goals, and objectives for the app. Consider your target audience and the problem you aim to solve.

D. Outline Concrete Goals & Objectives:

Clearly define what you want the app to achieve. Identify specific features, functionalities, and outcomes.

E. Assemble Your Development Stack:

Decide on the technology stack, tools, and frameworks you'll use for development.

F. Define the Target Audience:

Understand your users' needs, preferences, and pain points. This will guide your user stories.

Write User Stories:

G. **Use the user story template:** "As a [user persona], I want to [complete an action] so that I can [get some value]."

For example: "As a project manager, I want to stay organized so I can keep my entire team on track."

H. Sketch a UI Design Prototype:

Create rough wireframes or mockups to visualize how the app's interface will look.

I. Address Competitive Apps & Brands:

Research existing apps in your niche. Understand their strengths and weaknesses to inform your user stories.

J. Establish a Development Timeline:

Break down tasks based on user stories. Set milestones and deadlines for each development phase.

Self-Check 3: Develop User Story

1. Question: What is a user story?

2. Question: How do you estimate a user story's effort?

3. Question: What are user stories in the context of project work?

4. Question: How can project management tools help?

5. Question: How do you make project stories?

Answer Key 3: Develop User Story

1. **What is a user story?**
   Answer: A user story is a brief, plain-language description of a feature or functionality from the end-user perspective. It follows the format: "As a [user persona], I want to [complete an action] so that I can [get some value]."

2. **How do you estimate a user story's effort?**
   Answer: Teams estimate user stories using techniques like story points or relative sizing. It helps prioritize work and allocate resources effectively.

3. **What are user stories in the context of project work?**
   Answer: User stories describe what users want to achieve within a software system. They simplify complex requirements and focus on user needs.

4. **How can project management tools help?**
   Answer: Project management tools assist in planning, tracking progress, managing tasks, and collaborating with teams during software development.

5. **How do you make project stories?**
   Answer: To create project stories, follow these steps: define goals, write user stories, sketch UI designs, and establish a development timeline.

Task Sheet 3.1: Develop User Story

Title: Create User Stories for a Project:
Performance Objective: By the end of this task, the trainee should be able to Creating User Stories for a Project.
1. Define a hypothetical project (e.g., building an e-commerce app).
2. Write three user stories related to essential features of the app (e.g., user registration, product search, checkout process).
3. Explore a project management tool (e.g., Asana, Jira, or Trello).
4. Create a sample project board with user stories, tasks, and deadlines.
5. Include acceptance criteria, user roles, and expected outcomes.

Specification Sheet 3.1

A. Tools and Material required:

- Notebook
- Handbook
- Project management tool

B. Equipment:

- Laptop/Computer

Learning Outcome 4: Perform Unit Testing

Assessment Criteria:

1. Test cases are identified
2. Testing tools are used

Content:

1. Test cases
2. Testing tools
    a. JUnit
    b. Mockito

Resources Required/ Conditions:

The trainees must be provided with the following:

- Handouts or reference materials/books/ CBLMs on the above stated contents
- PCs/printers or laptop/printer with internet access
- Digital projector and Screen
- Bond paper
- Ball pens/pencils and other office supplies and materials
- Relevant learning materials
- Workplace or simulated environment

Methodologies

- Lecture/discussion
- Demonstration/application
- Presentation
- Blended delivery methods

Assessment Methods

- Written test
- Demonstration
- Observation with checklist
- Oral questioning
- Portfolio

Learning Experience 4: Perform Unit Testing

In order to achieve the objectives stated in this learning guide, you must perform the learning steps below. Beside each step are the resources or special instructions you will use to accomplish the corresponding activity.

Learning Steps	Resources specific instructions
1. Student will ask the instructor about Perform unit testing	1. Instructor will provide the learning materials "**Performing Project Work with Android"**
2. Read the Information sheet/s	2. Information Sheet No: 4 Perform unit testing
3. Complete the Self Checks & Check answer sheets.	3. Self-Check/s Self-Check No: 4 Perform unit testing Answer key No. 4 Perform unit testing
4. Read the Job Sheet and Specification Sheet and perform job	4. Job- Sheet No: 4 Perform unit testing Specification Sheet: 4 Perform unit testing

Information Sheet 4: Perform Unit Testing

Learning Objectives:

After completion of this information sheet, the learners will be able to:

4.1 Identify test cases
4.2 Use Testing tools

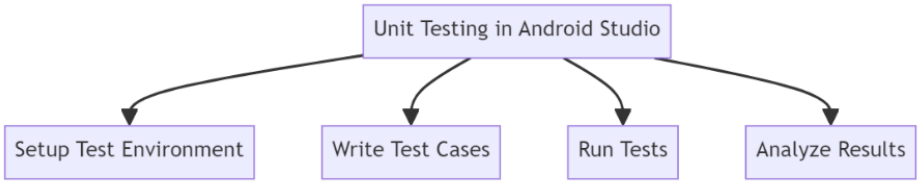
4.1 Unit testing

Unit testing in Android refers to the practice of writing and executing tests to validate the individual units or components of an Android application. These units can include methods, functions, classes, or even small parts of the user interface. The objective is to ensure that each unit of code functions correctly and behaves as intended in isolation from the rest of the application.

When units are testable in isolation, it generally indicates that the code is well-structured and adheres to the Single Responsibility Principle (SRP). This improves code quality, makes it easier to maintain, and enhances the overall architecture of the Android application.

Unit testing, when combined with other testing approaches like integration testing and Android UI testing, contributes to the overall quality, reliability, and success of Android apps.

Unit testing is a crucial step in Android Studio, ensuring that each component of your app functions correctly. By isolating and testing individual units of code, developers can identify and fix issues early, enhancing the quality and reliability of their Android applications. This practice streamlines the development process, making your codebase robust and maintainable.



4.2 Common Challenges in Unit Testing for Android

While unit testing in Android offers numerous benefits, developers may encounter certain challenges during the process.

Dependency management: Android apps often depend on external libraries, system components, or APIs. Managing these dependencies and ensuring they are properly mocked or stubbed during unit testing can be challenging. Dependency injection frameworks like Dagger can help alleviate this challenge by providing a way to manage and mock dependencies effectively.

Android-specific components: Android applications rely on various Android-specific components such as Context, Intent, and SharedPreferences. These components are tightly coupled with the Android framework and can be difficult to mock or simulate in unit tests. Specialized libraries like Robolectric or Mockito-Android can assist in creating mock objects or shadowing Android components for testing purposes.

Asynchronous operations: Android applications often involve asynchronous operations like network requests, database queries, or interactions with the user interface. Testing these asynchronous operations can be tricky as traditional unit tests are typically synchronous.

Tools like Mockito's when().thenReturn() and verify() can help simulate asynchronous behavior and ensure proper testing of these operations.

Legacy code and tight coupling: Legacy codebases or codes with high coupling can present challenges in unit testing. Tight coupling makes it difficult to isolate individual units for testing, as changes in one unit may affect others. Refactoring and introducing dependency injection can help mitigate this challenge and make code more testable.

Resource access and configuration: Android applications often rely on resources like strings, dimensions, or drawable. Accessing these resources directly in unit tests can be problematic, as they require a proper Android context. Libraries like AndroidX Test can provide utilities to create a test-specific application context and access resources during unit testing.

Time-consuming test execution: As the size and complexity of an Android application grow, the number of unit tests can also increase significantly. Running a large number of tests can consume substantial time and resources, affecting the development workflow. Leveraging tools like Gradle's build caching and running tests in parallel can help optimize test execution time.

Test maintenance: Maintaining unit tests can be challenging, especially when the application undergoes frequent changes or refactorings. Tests may need to be updated or modified to reflect the changes in the codebase. Regular refactoring and ensuring tests have clear and concise assertions can make test maintenance more manageable.

Testing Android Components

Testing Android components involves verifying the behavior and functionality of various Android-specific classes and components, such as Activities, Fragments, Services, Broadcast Receivers, and Content Providers.

Activity testing

Use ActivityScenario or Robolectric to create instances of Activity and simulate its lifecycle events.

Test different scenarios, such as launching an Activity, interacting with UI elements, and verifying expected outcomes.

Use Espresso or UIAutomator to perform UI testing and interact with UI elements.

Fragment testing

Use FragmentScenario or Robolectric to create instances of Fragment and test its lifecycle events.

Mock dependencies and use Mockito to simulate interactions between the Fragment and its dependencies.

Test Fragment interactions with its hosting Activity and other Fragments.

Service testing

Create an instance of Service and use local or bound service testing techniques to invoke its methods and verify behavior.

Mock dependencies and use Mockito to simulate interactions with other components.

Test the Service's response to various scenarios, such as receiving Intents or handling background tasks.

Broadcast Receiver testing

Create a mock Context and Intent to simulate the broadcast event.

Register the Broadcast Receiver programmatically or use the Context's registerReceiver() method for testing.

Test the Receiver's behavior when receiving different types of Intents and validate the expected results.

Content Provider testing:

Use a test-specific ContentResolver and mock the underlying data source.

Set up test data and perform CRUD (Create, Read, Update, Delete) operations on the Content Provider.

Verify that the Content Provider behaves correctly, such as returning the expected data or handling query parameters appropriately.

Testing Android-specific behaviors

Test specific Android behaviors, such as handling permissions, managing app lifecycle, handling configuration changes, or accessing system resources.

Use appropriate test doubles (e.g., Mockito mocks or Robolectric shadows) to simulate these behaviors during testing.

Integration testing

In Android Integration Testing, interactions and collaborations between multiple Android components, such as Activities, Fragments, and Services.

Mock dependencies and simulate the necessary environment for integration testing.

Verify that the components work together as expected and the desired system behavior is achieved.

UI testing:

Use UI testing frameworks like Espresso, UIAutomator, or Robolectric to perform UI testing and validate the UI behavior of your app's components.

Interact with UI elements, simulate user actions, and assert expected UI states or outcomes.

By combining unit tests, integration tests, and UI tests for different Android components, you can ensure comprehensive testing coverage and improve the quality and reliability of your Android application.

Setup Test Environment

It involves configuring the necessary dependencies in your build.gradle file. Ensure to include the appropriate testing libraries such as JUnit and Mockito for streamlined testing.

```
dependencies {
    // JUnit test framework
    testImplementation 'junit:junit:4.13.2'
    // Mockito framework for mocking in tests
    testImplementation 'org.mockito:mockito-core:3.9.0'
}
```

Here, testImplementation ensures that these libraries are only used during testing, keeping the app's production build lightweight.

Configuring The SDK And Emulator

The Android SDK and Emulator need to be set up correctly. This requires selecting an API level compatible with your tests and ensuring the emulator is responsive and fast enough to run them efficiently.

```
android {
    compileSdkVersion 30
    testOptions {
```

```
        unitTests.includeAndroidResources = true
    }
}
```

This code snippet sets the compile SDK version and includes Android resources during unit tests, enabling a seamless testing experience.

Creating A Test Class

Create a test class in the src/test/java directory. This class should mirror the structure of the class you intend to test. For example, if you have a LoginViewModel, create a LoginViewModelTest class.

```
public class LoginViewModelTest {
    @Before
    public void setUp() {
        // Set up the environment before each test
    }

    @Test
    public void login_Success() {
        // Test code here
    }
}
```

The @Before annotation marks a method to run before each test, while @Test denotes a test case. This structure is essential for maintaining organized test cases.

Running The Setup

With the environment set up, run the tests to verify the setup is correct. Use the following command in the terminal or run them directly from Android Studio.

```
./gradlew test
```

This command initiates the testing process. Look out for any errors that may indicate issues with the environment setup.

If all goes well, you're ready to write test cases.

Write Test Cases

Writing Test Cases is a methodical process that requires understanding the functionality of the code you're testing. Each test case should focus on one particular Aspect of Functionality and verify that it works as expected under various conditions.

- Asserting Test Conditions
- Mocking Dependencies
- Testing Exceptions
- Edge Cases and Boundary Conditions
- Testing Asynchronous Code

Asserting Test Conditions

Tests assert conditions with the JUnit framework's assert methods. For instance, to check if a method returns the expected value, you would use assertEquals.

```
@Test
public void addition_IsCorrect() {
    assertEquals(4, 2 + 2);
}
```

Mocking Dependencies

Often, classes have dependencies that need to be mocked. Mockito is a popular framework that allows you to create and configure mock objects.

```
@Test
public void userFetch_Success() {
    // Arrange
    UserRepository mockRepo = mock(UserRepository.class);
    User dummyUser = new User("john_doe");
    when(mockRepo.fetchUser()).thenReturn(dummyUser);

    // Act
    User result = mockRepo.fetchUser();

    // Assert
    assertEquals("john_doe", result.getUsername());
```

```
}
```

Here, a UserRepository is mocked to return a dummy User. The when...thenReturn pattern is crucial for defining mock behavior.

Testing tools

JUnit is a widely-used testing framework for Java applications that provides a set of annotations and assertions to help you write tests for your code. Mockito, on the other hand, is a mocking framework that allows you to create and manage mock objects to simulate the behavior of real objects during testing.

To follow along, make sure you have IntelliJ IDEA installed, and let's get started!

Setting up the project

First, create a new Java project in IntelliJ IDEA. Then, add the JUnit and Mockito dependencies to your project's build.gradle or pom.xml file.

For Gradle:

```
dependencies {
    testImplementation 'org.junit.jupiter:junit-jupiter-api:5.8.2'
    testImplementation 'org.mockito:mockito-core:4.2.0'
    testRuntimeOnly 'org.junit.jupiter:junit-jupiter-engine:5.8.2'
}
```

For Maven:

```
<dependencies>
    <dependency>
        <groupId>org.junit.jupiter</groupId>
        <artifactId>junit-jupiter-api</artifactId>
        <version>5.8.2</version>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>org.mockito</groupId>
        <artifactId>mockito-core</artifactId>
        <version>4.2.0</version>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>org.junit.jupiter</groupId>
        <artifactId>junit-jupiter-engine</artifactId>
        <version>5.8.2</version>
        <scope>test</scope>
    </dependency>
</dependencies>
```

Writing a simple JUnit test

Let’s say we have a simple Calculator class with an add method:

```
public class Calculator {
   public int add(int a, int b) {
      return a + b;
   }
}
```

To write a JUnit test for the add method, create a new class named CalculatorTest in the same package as the Calculator class, and add the following code:

```
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.assertEquals;
public class CalculatorTest {
   @Test
   public void testAdd() {
      Calculator calculator = new Calculator();
      int result = calculator.add(2, 3);
      assertEquals(5, result, "2 + 3 should equal 5");
   }
}
```

Using Mockito for mocking dependencies

Now let’s say we have a class UserService that depends on a UserRepository:

```
public class UserService {
   private UserRepository userRepository;
   public UserService(UserRepository userRepository) {
      this.userRepository = userRepository;
   }
   public boolean isUserActive(int userId) {
      User user = userRepository.findById(userId);
      return user != null && user.isActive();
   }
}
```

To test the isUserActive method, we can use Mockito to create a mock UserRepository and define its behavior. Create a new class named UserServiceTest and add the following code:

```
import org.junit.jupiter.api.Test;
import static org.mockito.Mockito.*;
import static org.junit.jupiter.api.Assertions.*;
public class UserServiceTest {
   @Test
   public void testIsUserActive() {
      // Create a mock UserRepository
      UserRepository userRepository = mock(UserRepository.class);
      // Define the behavior of the mock UserRepository
      User activeUser = new User(1, "John Doe", true);
```

```
        when(userRepository.findById(1)).thenReturn(activeUser);
        // Instantiate UserService with the mock UserRepository
        UserService userService = new UserService(userRepository);
        // Test the isUserActive method
        assertTrue(userService.isUserActive(1), "User with ID 1 should be active");
        // Verify the mock UserRepository's findById method was called with the correct argument
        verify(userRepository, times(1)).findById(1);
    }
    @Test
    public void testIsUserInactive() {
        // Create a mock UserRepository
        UserRepository userRepository = mock(UserRepository.class);

        // Define the behavior of the mock UserRepository
        User inactiveUser = new User(2, "Jane Doe", false);
        when(userRepository.findById(2)).thenReturn(inactiveUser);
        // Instantiate UserService with the mock UserRepository
        UserService userService = new UserService(userRepository);
        // Test the isUserActive method
        assertFalse(userService.isUserActive(2), "User with ID 2 should be inactive");
        // Verify the mock UserRepository's findById method was called with the correct argument
        verify(userRepository, times(1)).findById(2);
    }
}
```

With these tests in place, we have effectively used Mockito to mock the UserRepository dependency and test the UserService class in isolation. This allows us to focus on the specific behavior of the UserService class without worrying about the implementation details of the UserRepository.

Self-Check 4: Perform Unit Testing

1. Question: What are test cases?

2. Question: What are testing tools?

3. Question: What’s the difference between positive and negative test cases?

4. Question: What are JUnit and Mockito?

5. Question: Name a popular testing tool for Android.

Answer Key 4: Perform Unit Testing

1. **What are test cases?**
   Answer: Test cases are specific scenarios or conditions used to validate software behavior.

2. **What are testing tools?**
   Answer: Testing tools assist in creating, executing, and managing tests during software development.

3. **What's the difference between positive and negative test cases?**
   Answer: Positive test cases validate expected behavior, while negative ones check error handling.

4. **What are JUnit and Mockito?**
   Answer: JUnit is a Java testing framework, and Mockito is a mocking framework for managing dependencies.

5. **Name a popular testing tool for Android.**
   Answer: Appium is widely used for cross-platform mobile app automation testing.

Task Sheet 4.1: Perform Unit Testing

Performance Objective: By the end of this task, the trainee should be able to perform unit testing.
1. Choose a simple feature (e.g., login functionality).
2. Write three manual test cases for this feature: • Validate successful login with valid credentials. • Verify error message display for incorrect password. • Test account lockout after multiple failed login attempts.
3. Install a test management tool (e.g., TestRail or Jira).
4. Create a test project.
5. Add the test cases from Task 1 to the project.
6. Execute the test cases and mark the results (pass/fail).

Learning Outcome 5: Create Project Presentation

Assessment Criteria:
1. Project document is created.
2. Final project presentation in a group and/or individual is created.

Content:
1. Project documentation
2. Final project presentation

Resources Required/ Conditions:
The trainees must be provided with the following:

- Handouts or reference materials/books/ CBLMs on the above stated contents
- PCs/printers or laptop/printer with internet access
- Digital projector and Screen
- Bond paper
- Ball pens/pencils and other office supplies and materials
- Relevant learning materials
- Workplace or simulated environment

Methodologies

- Lecture/discussion
- Demonstration/application
- Presentation
- Blended delivery methods

Assessment Methods

- Written test
- Demonstration
- Observation with checklist
- Oral questioning
- Portfolio

Learning Experience 5: Create Project Presentation

In order to achieve the objectives stated in this learning guide, you must perform the learning steps below. Beside each step are the resources or special instructions you will use to accomplish the corresponding activity.

Learning Steps	Resources specific instructions
1. Student will ask the instructor about Create project presentation	1. Instructor will provide the learning materials "**Performing Project Work with Android"**
2. Read the Information sheet/s	2. Information Sheet No: 5 Use internet to access information
3. Complete the Self Checks & Check answer sheets.	3. Self-Check/s Self-Check No: 5 Create project presentation Answer key No. 5 Create project presentation
4. Read the Job Sheet and Specification Sheet and perform job	4. Job- Sheet No: 5 Create project presentation Specification Sheet: 5 Create project presentation

Information Sheet 5: Create Project Presentation

Learning Objectives:

After completion of this information sheet, the learners will be able to:

5.1 Create project document.
5.2 Create final project presentation in a group and/or individual.

5.1 Project Documentation

In Android app development, project documentation refers to all the written information that details your app's creation process, functionalities, and design choices. It's essentially a roadmap that keeps the project organized and ensures everyone involved understands the app's purpose and inner workings.

Here's why project documentation is important:

A. **Clarity and Communication:** Good documentation helps developers (including future ones joining the project) understand the app's architecture, features, and decisions made during development. This promotes clear communication and avoids confusion.
B. **Version Control and Maintenance:** As the app evolves, documentation helps track changes made over different versions. This simplifies maintenance and debugging processes.
C. **Collaboration:** If multiple developers are working on the app, proper documentation ensures everyone stays on the same page regarding the overall design and functionalities.

Here are some common components of project documentation in Android app development:

D. **Project Requirements:** This outlines the app's goals, target audience, and desired functionalities.
E. **Technical Specifications:** This details the technologies used (programming languages, libraries, frameworks), development tools, and API integrations.
F. **App Architecture:** This explains how the app is structured, including components, data flow, and class interactions.
G. **User Interface (UI) Design Documents:** This describes the app's user interface (UI) layouts, screen designs, and user interaction flows.
H. **API Documentation (if applicable):** If the app uses APIs, this section explains how they are integrated and used within the app.
I. **Testing Procedures:** This documents the app's testing strategy, including unit tests, integration tests, and UI tests.

J. Creating project documentation

Creating project documentation for Android app development involves several steps to ensure clarity, maintainability, and ease of understanding for all stakeholders involved. Here's a general outline of the steps:

K. Project Overview:

Provide a brief introduction to the project, including its purpose, goals, and scope.

Outline the target audience and any specific requirements or constraints.

L. Functional Requirements:

Detail the functional specifications of the app, including its core features and functionalities.

Use use cases, user stories, or diagrams to describe how users will interact with the app and achieve their goals.

M. Technical Architecture:

Describe the overall architecture of the app, including its components, modules, and dependencies.

Provide diagrams, such as UML diagrams or architectural diagrams, to visualize the app's structure and relationships between components.

N. Data Model:

Define the data model used by the app, including entities, relationships, and attributes.

Specify how data will be stored, retrieved, and manipulated within the app, such as using databases, APIs, or local storage.

O. User Interface Design:

Present the user interface design of the app, including wireframes, mockups, or UI flow diagrams.

Describe the visual elements, layout, and navigation patterns used in the app's interface.

P. Development Environment:

Document the development tools, frameworks, and technologies used to build the app.

Include setup instructions for developers to configure their development environment and build the app locally.

Q. Coding Standards:

Define coding standards and best practices to maintain consistency and readability across the codebase.

Specify naming conventions, code formatting guidelines, and documentation requirements.

R. Testing Strategy:

Outline the testing approach for the app, including unit testing, integration testing, and UI testing.

Describe test cases, test scenarios, and testing tools used to ensure the app's quality and reliability.

S. Deployment Plan:

Detail the deployment process for releasing the app to production or distribution channels, such as Google Play Store.

Include steps for code review, continuous integration, beta testing, and app store submission.

T. Maintenance and Support:

Provide guidelines for maintaining and updating the app after its initial release.

Document troubleshooting tips, known issues, and procedures for handling user feedback and bug reports.

U. Glossary and References:

Define any technical terms or acronyms used throughout the documentation.
Include references to external documentation, APIs, libraries, or third-party services used in the project.

V. **Version Control and Change History:**
Maintain version control of the documentation itself using a version control system like Git.
Keep a change log or revision history to track updates, revisions, and contributions to the documentation.

5.2 The final project presentation

The final project presentation is the culmination of the entire project lifecycle, where the team showcases the completed work to stakeholders, clients, or other relevant parties. This presentation serves to summarize the project's objectives, achievements, challenges faced, and outcomes.

During a final project presentation, we need to consider:

A. **Introduction:**
Begin with a brief introduction to the project, including its purpose and goals.
Introduce the project team members and their roles in the project.

B. **Project Overview:**
Provide an overview of the project, including its background, scope, and objectives.
Highlight any key milestones or achievements reached during the project timeline.

C. **Problem Statement:**
Clearly define the problem or need that the project aims to address.
Explain why this problem is important and how the project contributes to solving it.

D. **Approach and Methodology:**
Describe the approach and methodology used to tackle the project, including any research, analysis, or planning conducted.
Explain the rationale behind the chosen strategies and methodologies.

E. **Technical Details:**
Present technical aspects of the project, such as the architecture, technologies used, and implementation details.
Discuss any challenges encountered during development and how they were overcome.

F. **Demo:**
Demonstrate the functioning of the final product, showcasing its features and functionalities.
Walk through different use cases or scenarios to illustrate how users interact with the product.

G. **Results and Outcomes:**
Summarize the results and outcomes achieved through the project.
Highlight any metrics or key performance indicators (KPIs) that demonstrate the project's success.

H. **Lessons Learned:**

Reflect on lessons learned throughout the project, including successes, failures, and areas for improvement.

Discuss any insights gained that could be applied to future projects.

I. Future Recommendations:

Provide recommendations for future iterations or enhancements of the project.

Discuss potential areas for further development or expansion.

J. Conclusion:

Conclude the presentation by summarizing the key points and emphasizing the project's significance.

Thank the audience for their attention and invite questions or feedback.

K. Q&A Session:

Open the floor for questions from the audience.

Address any inquiries or concerns raised by stakeholders.

Self-Check 5: Create Project Presentation

1. Why is it important to create project documentation?

2. What are the benefits of creating a final project presentation?

3. Should project documentation be created collaboratively or by individuals?

4. How can project documentation facilitate project management?

5. What elements should be included in a final project presentation?

Answer Key 5: Create Project Presentation

1. **Why is it important to create project documentation?**
   Answer: Project documentation serves as a comprehensive reference for project stakeholders, ensuring clarity, consistency, and effective communication throughout the project lifecycle.

2. **What are the benefits of creating a final project presentation?**
   Answer: A final project presentation allows the project team to showcase their achievements, highlight key results, and receive feedback from stakeholders. It also helps in summarizing the project's journey and outcomes.

3. **Should project documentation be created collaboratively or by individuals?**
   Answer: Project documentation can be created collaboratively by the project team to ensure diverse perspectives are considered. However, individual contributions may also be necessary for specific sections or tasks.

4. **How can project documentation facilitate project management?**
   Answer: Project documentation provides a roadmap for project management activities, including planning, monitoring, and controlling. It helps in tracking progress, managing risks, and ensuring alignment with project objectives.

5. **What elements should be included in a final project presentation?**
   Answer: A final project presentation should include an overview of the project, its objectives, achievements, challenges, technical details, results, lessons learned, and future recommendations. It should be tailored to the audience's interests and level of understanding.

Task Sheet 4.1: Create Project Presentation

Title: Perform Project Documentation and Presentation Creation

Performance Objective: By the end of this task, the trainee should be able to Perform Project Documentation and Presentation Creation

1. Create project documentation for a hypothetical project of your choice (e.g., developing a mobile app, designing a website, etc.).
2. Include the following sections in your project documentation.
    - Project Overview
    - Functional Requirements
    - Technical Architecture
    - Data Model
    - User Interface Design
    - Development Environment
    - Coding Standards
    - Testing Strategy
    - Deployment Plan
    - Maintenance and Support
    - Glossary and References
    - Version Control and Change History
3. Collaborate with your team members to gather information and contribute to the documentation.
4. Use appropriate tools and formats to create professional-looking documentation (e.g., Microsoft Word, Google Docs, Confluence, etc.).
5. Develop a final project presentation based on the project documentation created in the previous step.
6. Structure your presentation to cover the following key points:
    - Introduction to the Project
    - Project Overview and Objectives
    - Problem Statement and Approach
    - Technical Details and Implementation
    - Demo of the Final Product (if applicable)
    - Results and Outcomes
    - Lessons Learned
    - Future Recommendations
7. Decide whether the presentation will be delivered by an individual or a group.
8. Practice delivering the presentation to ensure clarity, coherence, and professionalism.

Learning Outcome 6: Develop Awareness About Rights

Assessment Criteria:

1. The policies, rules and regulations that govern the work and workplace are upheld.
2. Illegal conduct or illegitimate action is reported to appropriate management.
3. Propriety or confidential information is protected.

Content:

1. The policies, rules and regulations that govern the work and workplace.
2. Illegal conduct or illegitimate action.
3. Propriety or confidential information.

Resources Required/ Conditions:

The trainees must be provided with the following:

- Handouts or reference materials/books/ CBLMs on the above stated contents
- PCs/printers or laptop/printer with internet access
- Digital projector and Screen
- Bond paper
- Ball pens/pencils and other office supplies and materials
- Relevant learning materials
- Workplace or simulated environment

Methodologies

- Lecture/discussion
- Demonstration/application
- Presentation
- Blended delivery methods

Assessment Methods

- Written test
- Demonstration
- Observation with checklist
- Oral questioning
- Portfolio

Learning Experience 6: Develop Awareness About Rights

In order to achieve the objectives stated in this learning guide, you must perform the learning steps below. Beside each step are the resources or special instructions you will use to accomplish the corresponding activity.

Learning Steps	Resources specific instructions
5. Student will ask the instructor about developing awareness about rights	5. Instructor will provide the learning materials "**Performing Project Work with Android"**
6. Read the Information sheet/s	6. Information Sheet No: 6 Develop awareness about rights
7. Complete the Self Checks & Check answer sheets.	7. Self-Check/s Self-Check No: 6 Develop awareness about rights Answer key No. 6 Develop awareness about rights
8. Read the Job Sheet and Specification Sheet and perform job	8. Job- Sheet No: 6 Develop awareness about rights Specification Sheet: 6 Develop awareness about rights

Information Sheet 6: Develop Awareness About Rights

Learning Objectives: After completion of this information sheet, the learners will be able to:

6.1 Upheld the policies, rules and regulations that govern the work and workplace
6.2 Report Illegal conduct or illegitimate action to appropriate management.
6.3 Protect propriety or confidential information.

6.1 The policies, rules and regulations that govern the work and workplace

In the context of Android app development, the policies, rules, and regulations that govern work and workplaces can be found at two levels:

A. General Workplace Policies:

These are the standard regulations that apply to any job and are established by the company you work for. They are likely outlined in an employee handbook and cover areas like:

a. **Work Hours and Breaks:** Standard work hours per day/week, overtime policies, breaks, and time-off procedures.
b. **Acceptable Use Policy:** Guidelines for using company devices, internet access, software, and electronic communications.
c. **Data Security and Confidentiality:** Protocols for protecting sensitive company information and client data.
d. **Intellectual Property:** Ownership rights regarding any code, designs, or materials created during work hours for the company.
e. **Anti-Discrimination and Anti-Harassment:** Policies ensuring a respectful and inclusive work environment free from discrimination or harassment.

B. Industry-Specific Regulations:

While there are no specific regulations solely for Android app development, there might be intellectual property considerations depending on the nature of the app and any external resources used. Here are some areas to keep in mind:

a. **Copyright and Trademark Laws:** Ensure proper licensing and usage rights for any third-party libraries, frameworks, or code snippets used in the app.
b. **Open-Source Licenses:** If using open-source libraries, comply with their specific licensing requirements, which might involve attribution or limitations on usage.
c. **Data Privacy Regulations:** Depending on the app's functionality and target audience, consider data privacy regulations like GDPR (General Data Protection Regulation) or CCPA (California Consumer Privacy Act) if they apply. These might dictate how user data is collected, stored, and used within the app.

Illegal conduct or illegitimate actions in Android app development encompass activities that violate laws, ethical boundaries, or the policies set by Google Play (the official app store for Android). Here's a breakdown of some key areas to avoid:

6.2 Illegal Activities:

Copyright Infringement: Using copyrighted content (music, images, code) without proper permission.

a. **Trademark Infringement:** Using logos or brand names that belong to others without authorization.
b. **Distribution of Malware or Spyware:** Creating apps that steal user data, damage devices, or infiltrate systems.
c. **Facilitating Illegal Activities:** Apps promoting gambling, drug sales, or other illegal actions.

A. Unethical Practices:

a. **Deception and Spam:** Apps with misleading descriptions, fake reviews, or functionality that spams users with unwanted content.
b. **Data Collection and Privacy Violations:** Collecting excessive user data without proper consent or selling user data to third parties without transparency.
c. **Predatory Monetization:** Deceptive in-app purchases, hidden fees, or manipulative tactics to pressure users into spending money.
d. **Unethical Advertising:** Displaying intrusive or misleading ads, especially those targeting children.

B. Violations of Google Play Policies:

a. **Security Issues:** Apps with vulnerabilities that can expose user data or compromise devices.

   Inappropriate Content: Apps containing hate speech, violence, or pornography that violate Google Play's content guidelines.
b. **App Compatibility Issues:** Apps that crash frequently, have bugs, or are not compatible with a reasonable range of Android devices.
c. **App Functionality Misrepresentation:** Apps that don't deliver the promised functionality as described in the app store listing.
d. **Consequences of Illegal or Illegitimate Actions:**
e. **Legal Action:** Copyright holders or authorities can pursue legal action for copyright infringement or other illegal activities.
f. **App Removal:** Google Play can remove apps that violate their policies.
g. **Account Suspension:** Developers engaging in repeated violations might face suspension or termination of their developer account.
h. **Reputational Damage:** Negative reviews and public backlash can damage the developer's reputation.

A. Developing Ethical and Legal Apps:

a. **Be Transparent:** Clearly disclose app functionalities, data collection practices, and monetization methods.
b. **Respect User Privacy:** Obtain informed consent for data collection and use it responsibly according to privacy regulations.
c. **Prioritize Security:** Implement robust security measures to protect user data and device security.

Follow Google Play Policies: Familiarize yourself with Google Play's Developer Policy Center (https://play.google/developer-content-policy/) to ensure your app complies with their guidelines.

6.3 Propriety or confidential information

In the context of Android app development, proprietary or confidential information refers to any sensitive data or knowledge that gives your company or project a competitive advantage.

a. **Trade Secrets:** This can include unique algorithms, design patterns, or source code that give your app a specific functionality or edge over competitors.
b. **App Design and Functionality Details:** The inner workings, design choices, and features planned for your app can be considered confidential, especially during development, to prevent competitors from copying your ideas.
c. **Client Data and Information:** If your app handles any user data or information specific to your clients (e.g., login credentials, financial data), this information is confidential and needs to be protected.
d. **API Keys and Credentials:** Certain APIs used in your app development process might require unique keys or credentials for authentication. Keeping these credentials confidential ensures unauthorized access is prevented.
e. **Non-Public Documentation:** Internal documents like project roadmaps, design mockups, or technical specifications not intended for public release can be considered confidential.

Importance of Protecting Proprietary Information:

a. **Competitive Advantage:** Safeguarding these secrets prevents competitors from replicating your app's unique features or functionality.
b. **Client Trust:** Protecting client data upholds user privacy and builds trust in your app's security practices.
c. **Legal Issues:** Leaking trade secrets or violating client data privacy laws can lead to lawsuits and hefty fines.

How to Protect Proprietary Information in Android App Development:

a. **Non-Disclosure Agreements (NDAs):** Have all employees, contractors, or anyone with access to confidential information sign NDAs to ensure legal recourse in case of breaches.
b. **Secure Code Repositories:** Use version control systems with access control features to restrict who can view or modify the app's codebase.
c. **Data Encryption:** Encrypt sensitive user data both at rest (stored on servers) and in transit (during transmission) to minimize the risk of breaches.
d. **Limited Access Controls:** Implement role-based access controls within your development team. Grant access to information only to those who need it for their specific tasks.
e. **Regular Security Audits:** Conduct periodic security audits to identify and address any vulnerabilities in your app's security posture.

Self-Check Sheet 6: Develop Awareness About Rights

1. What does it mean to uphold the policies, rules, and regulations of the workplace?

   Answer

2. When should you report illegal conduct or illegitimate actions in the workplace?

3. Why is it important to protect proprietary or confidential information?

   Answer

4. How can you uphold data privacy regulations in your work?

   Answer

5. What are some benefits of reporting illegal or illegitimate actions?

   Answer

Answer Key 6: Develop Awareness About Rights

1. **What does it mean to uphold the policies, rules, and regulations of the workplace?**
   Answer: Following the company handbook and guidelines related to work hours, acceptable behavior, data security, and other established procedures to maintain a safe and professional work environment.

2. **When should you report illegal conduct or illegitimate actions in the workplace?**
   Answer: If you witness any activity that violates the law (like copyright infringement) or unethical practices (like data privacy breaches), report it to your manager or a designated compliance officer.

3. **Why is it important to protect proprietary or confidential information?**
   Answer: This information, like trade secrets or client data, gives your company a competitive edge. Protecting it prevents leaks and ensures responsible handling of sensitive information.

4. **How can you uphold data privacy regulations in your work?**
   Answer: Only collect user data necessary for the app's function, obtain informed consent, and store and use data securely according to relevant regulations (e.g., GDPR, CCPA).

5. **What are some benefits of reporting illegal or illegitimate actions?**
   Answer: It promotes a safe and ethical work environment, protects the company from legal repercussions, and fosters trust and transparency within the organization.

Review Of Competency

Below is yourself assessment rating for module **"Performing Project Work with Android"**

SL no	**Assessment of performance Criteria**	Yes	No
1.	Concepts of project management is interpreted		
2.	Resource management is interpreted		
3.	Process management is interpreted		
4.	Technology management is interpreted		
5.	Team communication and reporting are acknowledged		
6.	Android Architecture Components are identified		
7.	Project model is selected as per project requirement.		
8.	Key principles of selected model are implemented.		
9.	User interface of a mobile application is designed.		
10.	Git as a source control system is used.		
11.	Android application is developed		
12.	User story is explained.		
13.	Story estimated.		
14.	User stories of a project work is defined.		
15.	Project management tool is used.		
16.	Project stories are made.		
17.	Test cases are identified		
18.	Testing tools are used		
19.	Project document is created.		
20.	Final project presentation in a group and/or individual is created.		
21.	The policies, rules and regulations that govern the work and workplace are upheld.		
22.	Illegal conduct or illegitimate action is reported to appropriate management.		
23.	Propriety or confidential information is protected.		

I now feel ready to undertake my formal competency assessment.

Signed:

Date:

Development of CBLM

The Competency based Learning Material (CBLM) of "**Performing Project Work with Android**" (Occupation: Android Mobile Application Development, Level-4) for National Skills Certificate is developed by NSDA with the assistance of SIMEC System Ltd., ECF Consultancy & SIMEC Institute of Technology JV (Joint Venture Firm) in the month of July, 2024 under the contract number of package SD-9B dated 15th January 2024.

SL No.	Name & Address	Designation	Contact Number
1	Md. Abdul Al Hossain	Writer	01778-926438
2	Md B**elayet** Hossain	Editor	01714-117032
3	Engr Md. Zuwel Parves	Co-Ordinator	01737-278906
4	Md. Abdur Razzaque	Reviewer	01713-304824

Reference

1. Project Management Methodologies For Mobile App Development - BuildFire
2. Top 40 Android Project Ideas & Topics of 2024 [Source Code] (knowledgehut.com)
3. App resources overview | Android Developers
4. How to Design a Mobile App User Interface in 2024 (softwaresuggest.com)
5. Design & Plan | Android Developers
6. https://chatgpt.com/
7. https://gemini.google.com