

KISAN BUDDY MAHINDRA FARMEQ

A PROJECT REPORT

Submitted by,

Ms. A Siva Sahithi - 20211CSE0164

Mr. Kakarla Manoj Kumar – 20211CSE0884

Mr. Shreyank Shankar Sarwadh – 20221LCS0005

Under the guidance of,

Dr. Abdul Khadar A

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

At



PRESIDENCY UNIVERSITY

BENGALURU

DECEMBER 2024-25

PRESIDENCY UNIVERSITY
SCHOOL OF COMPUTER SCIENCE ENGINEERING
CERTIFICATE

This is to certify that the Project report “**Kisan Buddy Mahindra Farmeq**” being submitted by “ASivaSahithi, Kakarla Manoj Kumar, Shreyank Shankar Sarwade ” bearing roll number(s) “20211CSE0164, 20211CSE0884, 20221LCS0005” in partial fulfillment of the requirement for the award of the degree of Bachelor of Technology in Computer Science and Engineering is a bonafide work carried out under my supervision.

Dr. Abdul Khadar A
Associate Professor
School of CSE
Presidency University

Dr. Dr. Asif Mohammed
HOD& Professor
School of CSE
Presidency University

Dr. L. SHAKKEERA
Associate Dean
School of CSE
Presidency University

Dr. MYDHILI NAIR
Associate Dean
School of CSE
Presidency University

Dr. SAMEERUDDIN KHAN
Pro-Vc School of Engineering
Dean -School of CSE
Presidency University

PRESIDENCY UNIVERSITY
SCHOOL OF COMPUTER SCIENCE ENGINEERING
DECLARATION

We hereby declare that the work, which is being presented in the project report entitled **Kisan Buddy Mahindra Farmeq** in partial fulfillment for the award of Degree of **Bachelor of Technology in Computer Science and Engineering**, is a record of our own investigations carried under the guidance of **Dr. Abdul Khadar A, Associate Professor, School of Computer Science Engineering, Presidency University, Bengaluru.**

We have not submitted the matter presented in this report anywhere for the award of any other Degree.

NAME	ROLL NUMBER	SIGNATURE
Aluru Siva Sahithi	20211CSE0164	
Kakarla Manoj Kumar	20211CSE884	
Shreyank Shankar Sarwade	20221LCS0005	

ABSTRACT

The Android application aims to streamline the interaction between farmers and mandi shops by offering a user-friendly platform. The app provides three key modules: Admin, Mandi Shops, and Farmers. The Admin module allows administrators to log in, add mandi shops, and view registered farmers. Mandi shops can log in, view farmer requests, update prices for produce, and update request statuses. Farmers can register, log in, add requests for selling their produce, view the history of their transactions, and make payments through the app. The application also integrates location services to ensure farmers relate to the nearest mandi shops. This efficient system enhances communication between farmers and markets, making the process of selling agricultural products easier and more transparent.

KEYWORDS: Mobile application, Android, Admin, Mandi Shops, Farmers

ACKNOWLEDGEMENT

First of all, we indebted to the **GOD ALMIGHTY** for giving me an opportunity to excel in our efforts to complete this project on time.

We express our sincere thanks to our respected dean **Dr. Md. Sameeruddin Khan**, Pro-VC, School of Engineering and Dean, School of Computer Science & Engineering, Presidency University for getting us permission to undergo the project.

We express our heartfelt gratitude to our beloved Associate Deans **Dr. Shakkeera L and Dr. Mydhili Nair**, School of Computer Science Engineering & Information Science, Presidency University, and **Dr. Asif Mohammed** Head of the Department, School of Computer Science & Engineering, Presidency University, for rendering timely help in completing this project successfully. We are greatly indebted to our guide **Dr. Abdul Khadar A, Associate Professor** and Reviewer, School of Computer Science & Engineering, Presidency University for his inspirational guidance, and valuable suggestions and for providing us a chance to express our technical capabilities in every respect for the completion of the project work. We would like to convey our gratitude and heartfelt thanks to the PIP2001 Capstone Project Coordinators **Dr. Sampath A K, Dr. Abdul Khadar A and Mr. Md Zia Ur Rahman**, department Project Coordinators and Git hub coordinator **Mr. Muthuraj**. We thank our family and friends for the strong support and inspiration they have provided us in bringing out this project.

A Siva Sahithi (1)

Kakarla Manoj Kumar (2)

Shreyank Shankar Sarwade (3)

LIST OF TABLES

Sl. No.	Table Name	Table Caption	Page No.
1	Table 1.1	Test Cases	43

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	i
	ACKNOWLEDGMENT	ii

1.	INTRODUCTION	1
	1.1 Motivation	1
	1.2 Problem Statement	1
	1.3 Objective of the Project	1
	1.4 Scope	1
	1.5 Project Introduction	2
2.	LITERATURE SURVEY	...
	2.1 Related Work	3
3.	RESEARCH GAPS OF EXISTING METHODS	6
4.	PROPOSED METHODOLOGY	8
	4.1 Requirement Analysis	8
	4.2 System Design and Architecture	8
	4.3 Development	8
	4.4 Testing	9
	4.5 Deployment	9
	4.6 Maintenance	9
5.	OBJECTIVES	11
6.	SYSTEM DESIGN AND IMPLEMENTATION	13
7.	ANDROID ENVIRONMENT	24
8.	SYSTEM STUDY AND TESTING	43
9.	TIMELINE OR EXECUTION O PROECT	48

10.	OUTCOMES	49
11.	RESULTS AND DISCUSSIONS	50
12.	CONCLUSION	51
13.	FUTURE ENHANCEMENT	52
14.	REFERENCES	53
15.	APPENDIX-A—PSUEDOCODE	54
16.	APPENDIX-B—SCREENSHOTS	56
17.	APPENDIX-C---ENCLOSURES	60

CHAPTER-1

INTRODUCTION

1.1 Motivation

The driving force Millions of farmers around the world depend heavily on agriculture for their livelihood. However, inefficient practices including opaqueness, late payments, and restricted access to local markets (mandi shops) are common in the traditional methods used by farmers to sell their produce. By giving farmers, a smooth way to communicate with mandi shops, our Android app seeks to solve these problems by improving their access to better market prices and services while streamlining the entire transaction process.

1.2 Problem Statement

The traditional agricultural market system lacks efficiency, transparency, and accessibility for farmers. Farmers often face challenges in selling their produce due to limited communication with mandi shops, delayed payments, and difficulty in accessing local markets. This manual process not only causes inefficiencies in transactions but also limits farmers' ability to secure fair prices for their products. The need for a digital platform that bridges the gap between farmers and mandi shops is crucial to improve the agricultural supply chain and farmer livelihoods

1.3 Objective of the Project

The primary objective of the system is to provide a digital platform that connects farmers to mandi shops, simplifying the process of selling agricultural produce. Empower farmers with easy access to local mandi shops. Improve the communication between farmers and mandi shops for real-time updates on produce costs and status. Streamline transactions, ensuring timely payments and enhanced transparency.

1.4 Scope

The application is designed for farmers, mandi shop owners, and administrators in rural and urban regions. It enables farmers to sell their produce directly to nearby mandi shops, view their transaction history, and make payments. Mandi shops can view and update farmer requests, manage prices, and track transactions, while administrators have control over managing mandi shops and viewing farmers. The application aims to enhance transparency, speed up communication, and provide real-time market access for farmers and vendors.

1.5 Project Introduction

Agriculture remains the backbone of many economies, especially in rural areas where farmers rely on traditional methods to sell their produce. However, these conventional systems are often plagued by inefficiencies, including limited market access, communication barriers between farmers and mandi shops, and delayed payments. The growing need to modernize and streamline agricultural transactions has prompted the development of digital solutions that empower farmers with direct access to markets and better control over their produce sales. This Android application seeks to bridge the gap between farmers and mandi shops by offering a comprehensive platform that facilitates seamless communication and transactions. The app is designed with three core modules—Admin, Mandi Shops, and Farmer each tailored to meet the specific needs of the respective users. Farmers can easily register, log in, submit requests for selling produce, view their transaction history, and make payments. Mandi shops can log in, view and update requests, manage costs, and track the status of transactions, while the Admin module ensures efficient oversight of mandi shops and farmers. By integrating location-based services, the application helps farmers connect with nearby mandi shops, improving market accessibility. The platform also ensures transparency and security in transactions, enabling farmers to receive timely payments and track their requests. With its user-friendly design and streamlined process, the app aims to revolutionize agricultural transactions by providing a digital solution that benefits both farmers and mandi shop owners, enhancing the efficiency of the overall system.

CHAPTER-2

LITERATURE SURVEY

2.1 Related Work

2.1.1 Agrawal, M., & Pandey, D. (2021). "Digital Solutions for Enhancing Market Access for Farmers in India."

Introduction:

This paper explores the role of digital platforms and technologies in enhancing market access for farmers in India. The authors discuss the key digital solutions being implemented, such as mobile apps and e-commerce platforms, and their impact on improving the market reach and income of farmers, particularly smallholder farmers.

Summary:

Agrawal and Pandey highlight the transformative potential of digital solutions in India's agricultural sector. The study reveals how mobile applications and online platforms have empowered farmers by providing real-time market information, access to buyers, and better price realization. It emphasizes the role of digital literacy and infrastructure development in ensuring these solutions benefit all farmers, especially those in remote areas. The authors conclude that digital innovations are critical to bridging the gap between farmers and markets, thereby enhancing agricultural productivity and income.

2.1.2 Reddy, A. A., & Mishra, D. (2020). "Challenges and Opportunities in the Indian Agricultural Market: A Farmer's Perspective."

Introduction:

This article discusses the challenges Indian farmers face in accessing markets, with a particular focus on smallholder farmers. It examines the structural issues within the agricultural supply chain and offers insights into potential opportunities that could alleviate these barriers.

Summary:

Reddy and Mishra identify several challenges that limit market access for Indian farmers, including inadequate infrastructure, middlemen exploitation, and a lack of transparency in market prices. The study also outlines opportunities for improving the agricultural market, such as policy reforms, greater use of technology, and the promotion of farmer producer organizations (FPOs). The authors argue that while these opportunities are promising, addressing the systemic issues within the agricultural market is necessary to ensure equitable access and fair pricing for all farmers.

2.1.3 Kumar, P., & Joshi, P. K. (2019). "Technological Innovations for Market Linkages in Indian Agriculture."

Introduction:

This paper explores the role of technological innovations in creating effective market linkages for Indian farmers. The authors focus on how technology-driven initiatives, such as e-marketplaces, are revolutionizing the agricultural value chain in India.

Summary:

Kumar and Joshi discuss the impact of technological innovations in improving market access for farmers, highlighting the role of e-marketplaces, precision agriculture tools, in creating more transparent and efficient market systems. The study illustrates how these innovations have helped reduce transaction costs, eliminate middlemen, and ensure better prices for farmers. The authors advocate for further government investment in technology infrastructure and the integration of small farmers into digital platforms to sustain long-term agricultural growth.

2.1.4 Singh, R., & Sharma, K. (2020). "Role of Mobile Applications in Enhancing Market Efficiency for Small Farmers."

Introduction:

This study focuses on the role of mobile applications in improving market efficiency and access for small-scale farmers in India. It examines how mobile technology is reshaping agricultural markets by providing farmers with critical market data, weather forecasts, and direct access to buyers.

Summary:

Singh and Sharma illustrate the positive impact of mobile applications in streamlining agricultural marketing processes, reducing information asymmetry, and connecting farmers directly with buyers. The paper presents case studies of successful mobile platforms that have enabled farmers to get real-time price updates, reduce dependence on intermediaries, and secure better deals. The authors conclude that mobile technology is a key enabler of market efficiency, but stress the importance of increasing mobile literacy among farmers to fully realize its potential.

2.1.5 Mukherjee, S., & Roy, A. (2018). "Improving Agricultural Market Access through ICT: A Review of Success Stories in India."

Introduction:

This paper reviews the role of Information and Communication Technology (ICT) in improving market access for Indian farmers. It presents several case studies showcasing how ICT interventions have enhanced market linkages and enabled farmers to receive better prices for their produce.

Summary:

Mukherjee and Roy provide a comprehensive review of successful ICT interventions in Indian agriculture, including initiatives like eNAM (National Agriculture Market) and mobile-based advisory services. The authors emphasize the importance of ICT in providing farmers with market information, price transparency, and access to broader markets. The review concludes that while ICT has proven to be a powerful tool for improving market access, more investment in rural ICT infrastructure is necessary to ensure that all farmers, particularly those in remote areas, can benefit. The study highlights the need for sustained government support and private sector collaboration to scale ICT-driven market solutions.

CHAPTER-3

RESEARCH GAPS OF EXISTING METHODS

Limited Digital Infrastructure for Farmers:

- Many existing solutions focus on urban or semi-urban areas and often neglect rural farmers who may lack access to stable internet or smartphones.

Fragmentation in Mandi Shop Integration:

- Current platforms do not fully integrate real-time data from multiple mandi shops, leading to incomplete or outdated pricing and stock information for farmers.

Transparency Issues:

- While some systems aim to reduce middlemen exploitation, few provide end-to-end transparency, especially regarding transaction tracking and payment processing.

Underutilization of Advanced Technologies:

- Existing solutions do not incorporate advanced analytics, AI for price recommendations, or blockchain for secure and transparent transactions.

Lack of Multilingual and User-Friendly Interfaces:

- Many platforms fail to address language diversity and ease of use for farmers who may not be literate in the application's default language.

Challenges in Real-Time Interaction:

- Most systems do not include features for direct, real-time communication between farmers and mandi shops, such as chat or video support.

Limited Focus on Predictive Insights:

- Few existing systems leverage predictive analytics for helping farmers decide the best time to sell or guiding them on crop selection based on market trends.

Inadequate Support for Diverse Payment Systems:

- Current applications often provide limited options for digital payments, reducing their accessibility and usability for all farmers.

Insufficient Post-Sale Support:

Platforms typically focus on facilitating sales but overlook post-sale support like customer service for resolving disputes or delayed payments.

Recommendations to Address Gaps:

- **Enhanced Digital Infrastructure:** Invest in low-bandwidth solutions to ensure accessibility for farmers in remote areas.
- **Integration of Advanced Technologies:** Use AI for pricing insights, blockchain for secure payments, and predictive analytics for market trends.
- **Multilingual Support:** Develop applications in multiple languages and use voice-based navigation for accessibility.
- **Real-Time Features:** Add real-time communication tools and dynamic market updates.
- **Robust Payment Ecosystems:** Expand digital payment options, including mobile wallets, UPI, and offline payment methods.

Invest in low-bandwidth solutions and multilingual support to ensure accessibility for remote farmers. Incorporate AI, blockchain, and predictive analytics for market trends, along with real-time communication tools and robust payment options.

CHAPTER-4

PROPOSED METHODOLOGY

The methodology for developing the Android application involves a systematic approach that incorporates design, development, integration, and testing phases to ensure the efficient functioning of the three key modules: Admin, Mandi Shops, and Farmers. The methodology can be broken down as follows

4.1 Requirements Analysis

Objective: Understand the core functionalities required by each user group (Admin, Mandi Shops, and Farmers) and gather requirements for the integration of location services.

Approach:

- Conduct stakeholder interviews with administrators, mandi shop owners, and farmers to identify needs and challenges.
- Document user stories for each module.
- Determine key interactions such as login, registration, produce requests, and location-based connections.

4.2 System Design and Architecture

Objective: Design a scalable and efficient architecture to support the three modules while ensuring seamless integration between them and location services.

Approach:

- **Modular Design:** Define clear boundaries and interfaces for the Admin, Mandi Shops, and Farmers modules.
- **Database Schema:** Create a relational database schema to handle user data, transaction history, and produce details.
- **Location Services Integration:** Leverage Android's location APIs to ensure that farmers are matched with the nearest mandi shops.
- **UI/UX Design:** Design user-friendly interfaces for each user role, focusing on simplicity and ease of navigation.

4.3 Development

Objective: Build and implement the functionalities of the three modules, ensuring proper interaction and integration of location services.

Approach:

- **Admin Module:**
 - ☐ Implement login functionality with role-based access control.
 - ☐ Allow admins to manage mandi shop listings and view farmer details.
- **Mandi Shops Module:**
 - ☐ Implement login and authentication for mandi shop owners.
 - ☐ Provide features to view farmer requests, manage produce costs, and update request statuses.

-
- **Farmers Module:**
 - Allow farmers to register, log in, and submit requests for selling produce.
 - Implement features to view transaction history and facilitate payment processing.
 - **Location Services Integration:**
 - Implement GPS-based features to determine the farmer's location and suggest the nearest mandi shops.
 - **Backend Services:**
 - Develop backend APIs to handle data requests and responses, ensuring real-time updates between users.

4.4 Testing

Objective: Ensure that the application functions as expected across different user roles, with all features working correctly.

Approach:

- **Unit Testing:** Test individual modules and components (Admin, Mandi Shops, Farmers) to ensure correctness.
- **Integration Testing:** Verify that interactions between modules (e.g., farmer requests, mandi shop updates) work seamlessly.
- **Location Testing:** Simulate various geographic locations to ensure proper integration with location services and correct matching of farmers to nearest mandi shops.
- **Usability Testing:** Conduct testing with real users (farmers, mandi shop owners, and admins) to identify any usability issues and improve the interface.

4.5 Deployment

Objective: Deploy the Android application to the Google Play Store and provide users with access to the system.

Approach:

- **Beta Testing:** Conduct closed beta testing with selected users to ensure functionality, performance, and bug-free operation.
- **Final Deployment:** After successful beta testing, release the application to the Google Play Store for public access.
- **Monitoring and Support:** Continuously monitor the app's performance and user feedback, offering timely bug fixes and updates.

4.6. Maintenance and Updates

Objective: Ensure the application remains up-to-date, bug-free, and functional over time.

Approach:

- Regularly update the app with new features and enhancements based on user feedback.
- Address any technical issues or bugs reported by users.
- Ensure the app stays compatible with the latest Android versions and devices.

The proposed methodology focuses on developing an Android-based application with three key user modules—Admin, Mandi Shops, and Farmers—to streamline agricultural transactions. The methodology includes the following steps:

1. System Design:

Develop user-friendly interfaces for each module tailored to specific user needs:

Admin Module: Manages mandi shops, oversees farmer registrations, and monitors transactions.

Mandi Shops Module: Allows shop owners to view and respond to farmer requests, update produce prices, and track transactions.

Farmer Module: Enables farmers to register, log in, submit requests for selling produce, view transaction histories, and make payments.

2. Integration of Location Services:

- a. Utilize GPS-based location services to connect farmers with the nearest mandi shops efficiently.

3. Technology Framework:

- a. **Backend:** Utilize Kotlin programming language for development in the Android environment.
- b. **Database:** Implement an SQL-based database for data management and transaction storage.
- c. **Frontend:** Design user interfaces using Android Studio with a focus on simplicity and usability.

4. Features and Functionalities:

- a. Enable real-time updates for produce prices and request statuses.
- b. Integrate secure payment gateways for timely transactions.
- c. Provide transaction tracking and historical data for transparency.

5. Testing and Validation:

- a. Conduct unit testing, integration testing, and user acceptance testing to ensure functionality and user satisfaction.
- b. Validate system scalability, security, and performance under varying load conditions.

This methodology ensures an organized, scalable, and user-centric approach to bridging the gap between farmers and mandi shops.

CHAPTER-5

OBJECTIVES

The primary objective of this system is to create a comprehensive digital platform that bridges the gap between farmers and mandi shops, simplifying and streamlining the process of selling agricultural produce. In traditional markets, farmers often face significant challenges in connecting with local mandi shops, including long distances, inefficient communication, and delayed updates on market prices and transaction statuses. This platform aims to empower farmers by providing them with easy access to nearby mandi shops, ensuring that they can quickly and efficiently engage in the selling process. By leveraging technology, the system improves communication between farmers and mandi shop owners, enabling real-time updates on the availability of produce, market demands, and the costs of agricultural products. This not only helps farmers make more informed decisions about when and where to sell their produce but also ensures that mandi shop owners can access a steady supply of fresh, local produce based on demand.

Furthermore, the system significantly enhances the transparency and efficiency of financial transactions between farmers and mandi shops. It offers a streamlined approach to manage pricing, order requests, and payment processing, reducing the potential for disputes or misunderstandings. Farmers can submit their produce requests and track the status of these requests from submission to payment. Mandi shop owners, in turn, can view these requests, confirm orders, and update their status in real-time. By implementing a secure and reliable payment gateway, the system ensures that payments are processed in a timely manner, offering both farmers and shop owners peace of mind regarding financial transactions. The transparency embedded within the system also means that both parties can view transaction histories, providing a clear record of exchanges that can be referred to in case of discrepancies.

This digital platform not only benefits individual farmers and mandi shop owners but also contributes to the broader agricultural ecosystem. It encourages a more organized and accessible market for agricultural produce, reducing the inefficiencies of traditional, offline trading systems. The integration of location-based services ensures that farmers relate to the nearest mandi shops, further optimizing the supply chain and minimizing logistical costs. By automating and centralizing the flow of information between farmers, mandi shop owners,

and administrators, the system fosters a more competitive, transparent, and sustainable market environment. Ultimately, the system aims to create a digital infrastructure that enhances the livelihoods of farmers, ensures fair pricing, and fosters a more robust agricultural market that can better withstand the challenges of the modern economy.

CHAPTER-6

SYSTEM DESIGN & IMPLEMENTATION

6.1 Introduction of Input design

6.1.1 INPUT DESIGN

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data into a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- What data should be given as input?
- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when error occur.

6.1.2 OBJECTIVES

1. Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.
2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.
3. When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow

6.1.3 OUTPUT DESIGN

A quality output is one which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to

other systems through outputs. In output design it is determined how the information is to be displaced for immediate need and the hard copy output. It is the most important and direct source of information for the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can be use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.

2. Select methods for presenting information.

3. Create document, report, or other formats that contain information produced by the system. The output form of an information system should accomplish one or more of the following objectives.

- Convey information about past activities, current status or projections of the
- Future.
- Signal important events, opportunities, problems, or warnings.
- Trigger an action.
- Confirm an action.

6.2 UML Diagram

UML stands for Unified Modelling Language. UML is a standardized general-purpose modelling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group. The goal is for UML to become a common language for creating models of object-oriented computer software. In its current form UML is comprised of two major components: A Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML. The Unified modelling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modelling and other non-software systems. The UML represents a collection of the best engineering practices that have proven successful in the modelling of large and complex systems. UML is a very important part of developing objects-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

The Unified Modelling Language (UML) serves as a standardized, general-purpose modelling language within the realm of object-oriented software engineering, overseen and created by

the Object Management Group (OMG). Its primary objective is to establish a universal language for modelling object-oriented computer software, aiming to provide a common ground for software developers to communicate and collaborate effectively. UML consists of two main components: a Meta-model, which defines the structure and semantics of UML itself, and a notation, which encompasses the graphical symbols and diagrams used to represent various aspects of software systems. While currently focused on these components, UML may incorporate additional methods or processes in the future. As a standard language, UML facilitates the specification, visualization, construction, and documentation of software artifacts, along with applications in business modelling and other non-software domains. It encapsulates a collection of best engineering practices proven effective in modelling large and intricate systems. In the software development process, UML plays a pivotal role by enabling developers to express the design of software projects using graphical notations. Its adoption promotes clarity, consistency, and efficiency in communication, aiding in the development of robust and scalable object-oriented software systems. Thus, UML stands as a cornerstone in the development of object-oriented software and the broader software engineering process.

6.2.1 GOALS:

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modelling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modelling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

6.2.2 CLASS DIAGRAM:

In software engineering, a class diagram in the Unified Modelling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

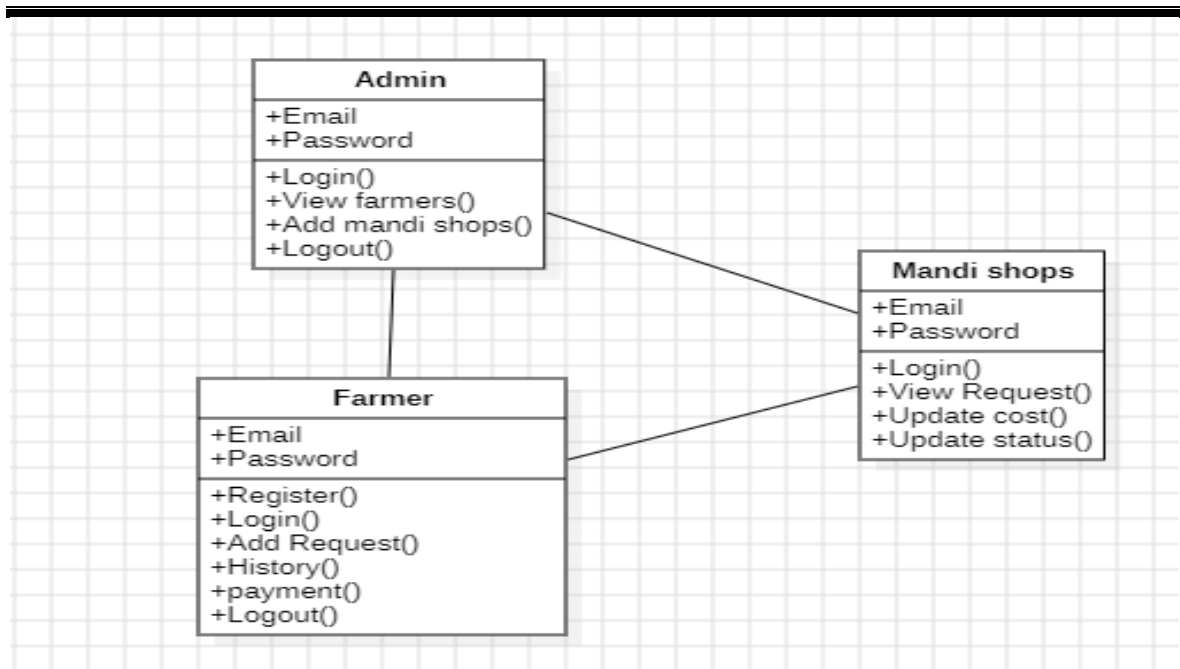


Figure 1: CLASS DIAGRAM

6.2.3 USE CASE DIAGRAM:

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

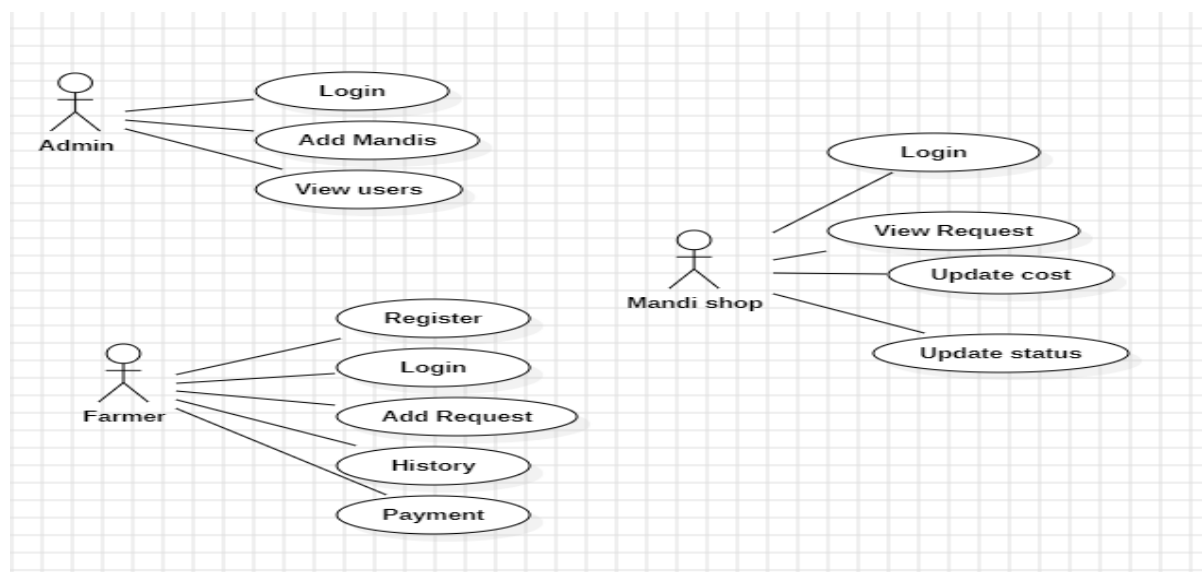


Figure 2: USE CASE DIAGRAM

6.2.4 SEQUENCE DIAGRAM:

A sequence diagram in Unified Modelling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

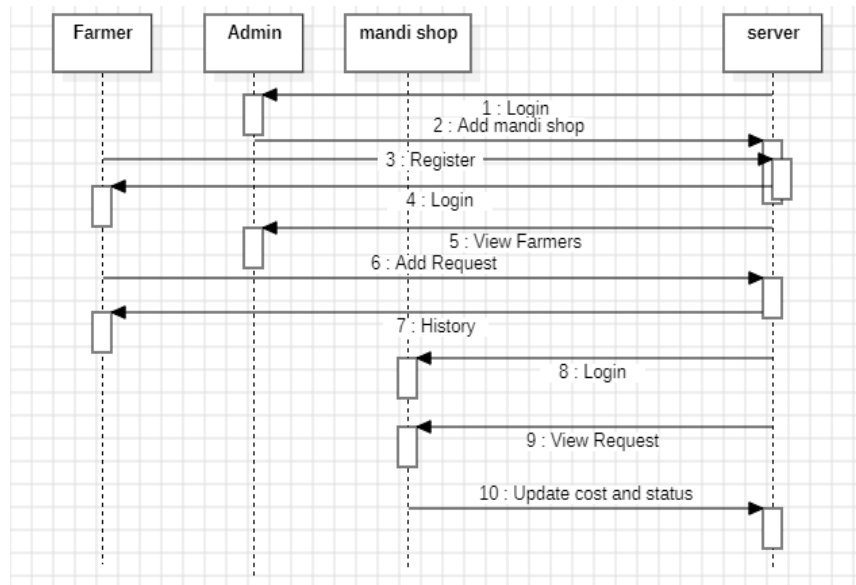


Figure 3: SEQUENCE DIAGRAM

6.2.5 COLLABORATION DIAGRAM:

In collaboration diagram the method call sequence is indicated by some numbering technique as shown below. The number indicates how the methods are called one after another. We have taken the same order management system to describe the collaboration diagram. The method calls are similar to that of a sequence diagram. But the difference is that the sequence diagram does not describe the object organization whereas the collaboration diagram shows the object organization.

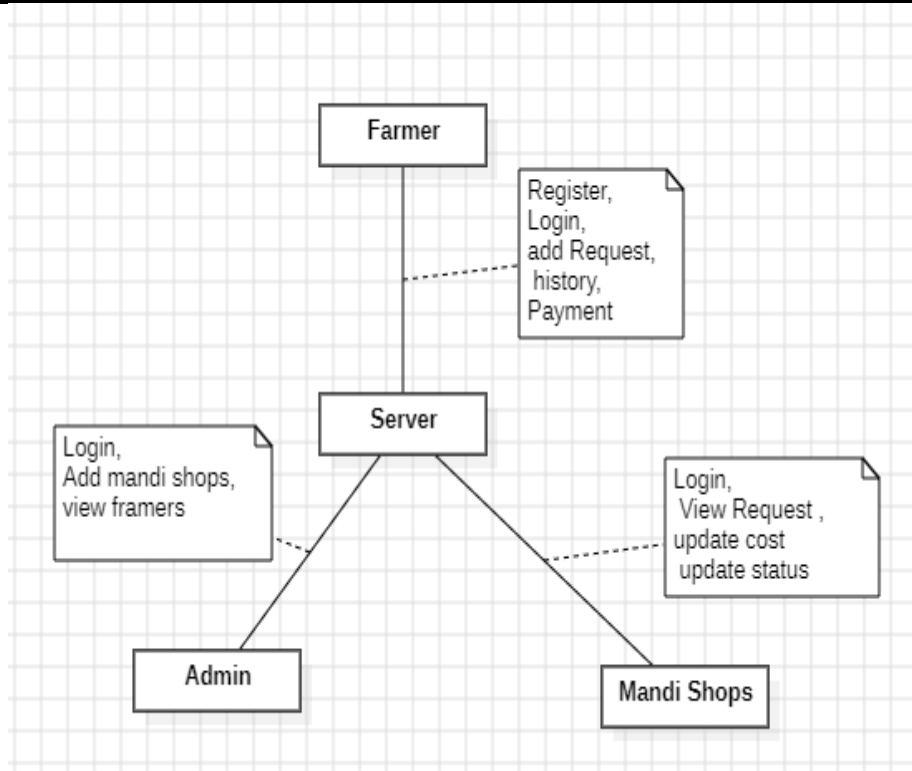


Figure 4: COLLABORATION DIAGRAM

6.2.6 ACTIVITY DIAGRAM:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modelling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

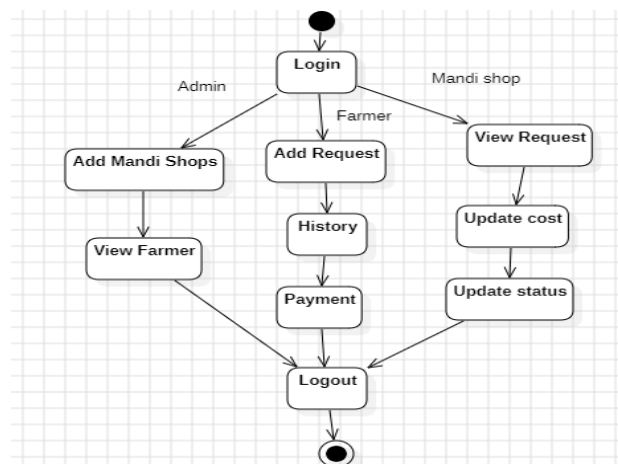


Figure 5: ACTIVITY DIAGRAM

6.2.7 COMPONENT DIAGRAM:

A component diagram in software engineering illustrates the components of a system and their relationships. Components represent modular units of functionality, such as classes, modules, or libraries, and are depicted as rectangles with the component's name inside. Relationships between components are shown with lines connecting them, indicating dependencies, associations, or interfaces. Component diagrams help visualize the architecture of a system, including how components interact and communicate with each other. They are useful for understanding the structure of a software system and for communicating design decisions to stakeholders.

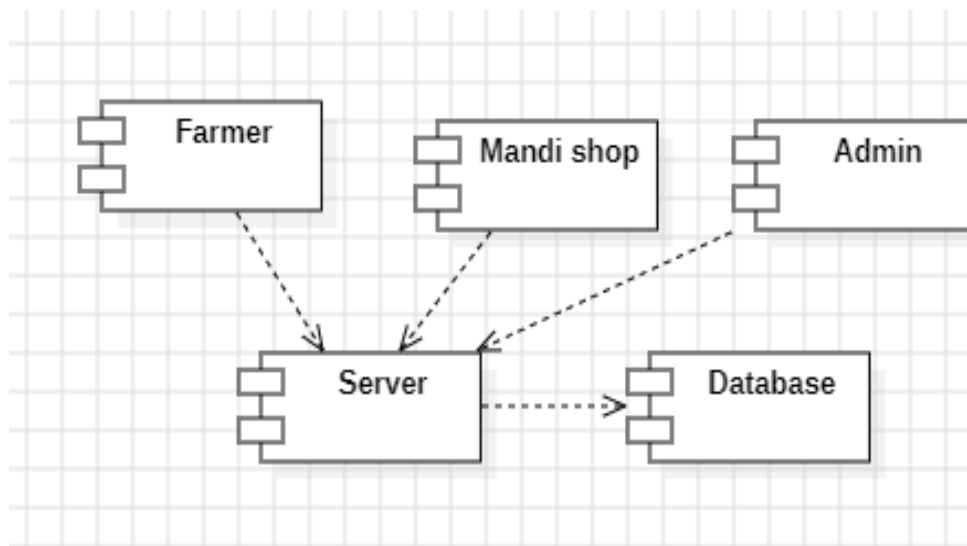


Figure 6: COMPONENT DIAGRAM

6.2.9 DEPLOYMENT DIAGRAM:

A deployment diagram in software engineering visualizes the physical deployment of software components onto hardware nodes in a distributed system. Nodes represent hardware devices, such as servers, computers, or mobile devices, depicted as rectangles with the node's name inside. Components, represented by rectangles with the component's name inside, are deployed onto nodes, showing how software elements are distributed across the hardware infrastructure. Deployment diagrams illustrate the configuration and deployment topology of a system, including the relationships between software components and the hardware resources they utilize. They aid in understanding system deployment and resource allocation in distributed environments.

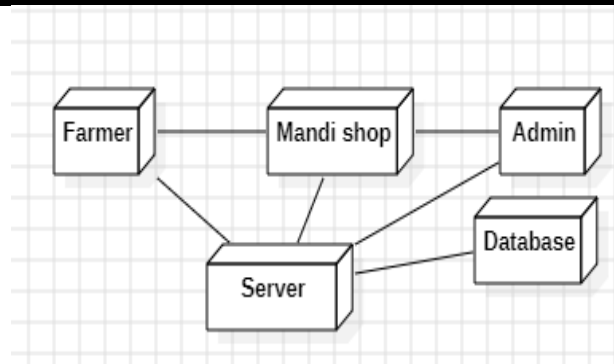


Figure 7: DEPLOYMENT DIAGRAM

6.2.10 ER Diagram:

An Entity-Relationship (ER) diagram in database design illustrates the relationships between entities within a database schema. Entities represent real-world objects or concepts, such as customers, orders, or products, depicted as rectangles with the entity's name inside. Relationships between entities are shown with lines connecting them, indicating associations or dependencies. Cardinality and participation constraints may also be included to specify the nature of the relationships. ER diagrams help visualize the structure of a database schema, including the entities, attributes, and relationships between them. They serve as a blueprint for designing and implementing relational databases effectively.

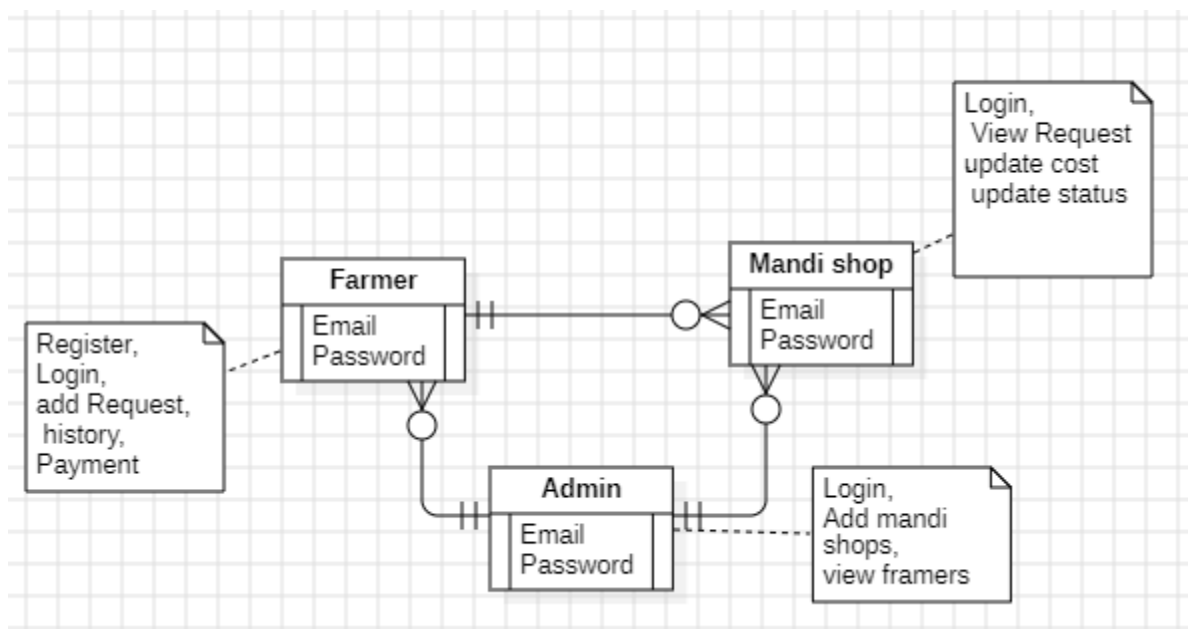


Figure 8: ER DIAGRAM

6.3 Data Flow Diagram:

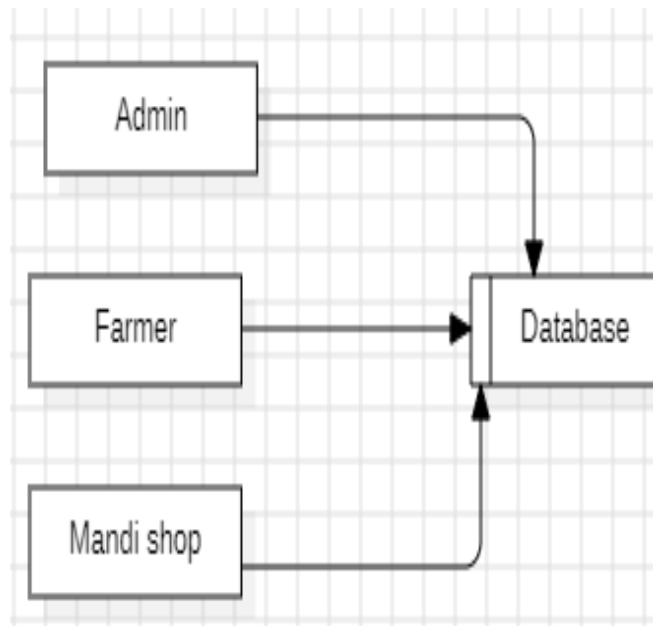


Figure 9: DATA FLOW DIAGRAM

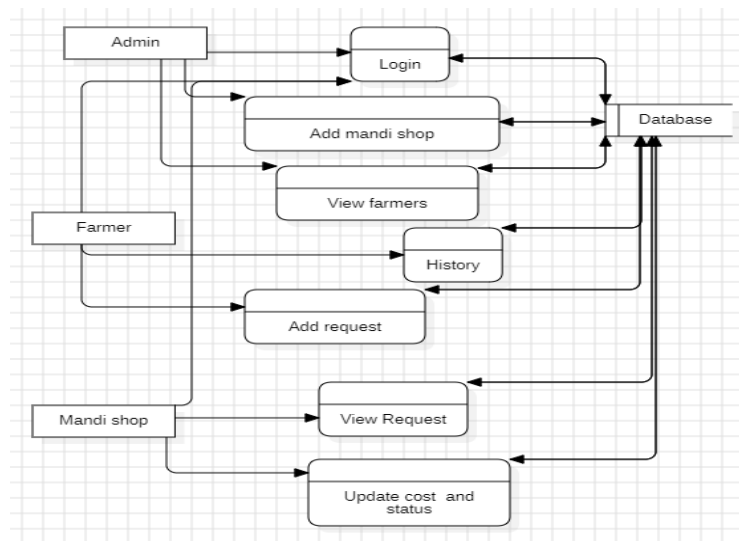


Figure 10: DATA FLOW DIAGRAM

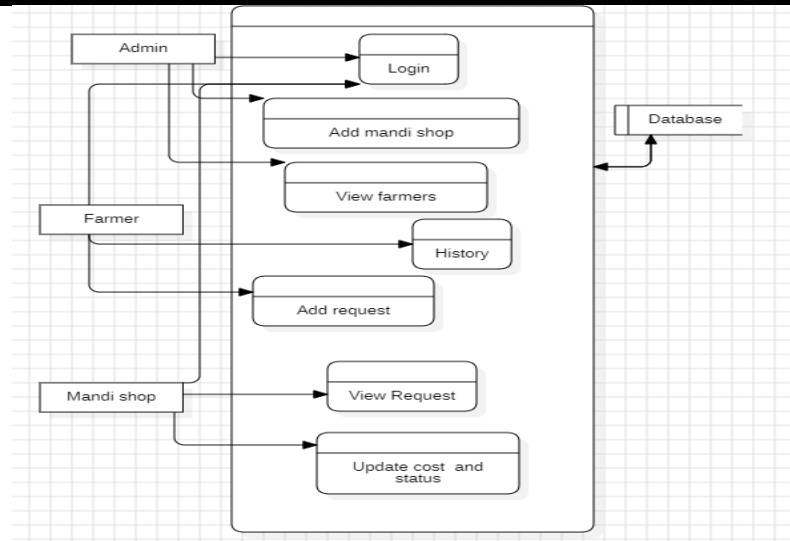


Figure 11: DATA FLOW DIAGRAM

6.4 Modules

6.4.1 Admin Module

The Admin module is responsible for managing the overall system and ensuring seamless operations between farmers and mandi shops. Administrators can log in with secure credentials to access the system and perform essential functions such as adding new mandi shops, monitoring registered farmers, and overseeing transactions. This module plays a vital role in maintaining the platform's integrity and ensuring that both farmers and mandi shop owners are registered and verified properly. The admin also has access to view data, ensuring that the system runs efficiently and effectively.

6.4.2 Mandi Shops Module

The Mandi Shops module is designed to enable mandi shop owners to interact directly with farmers. After logging in, mandi shop owners can view requests submitted by farmers, update the cost of produce based on market conditions, and change the status of requests as they are processed. This module streamlines the communication process by allowing mandi shops to respond to farmer requests in real time. Additionally, mandi shop owners can manage transactions and keep track of the produce they have received, providing a clear view of business operations.

6.4.3 Farmer Module

The Farmer module is tailored to the needs of farmers, enabling them to easily register, log in, and use the platform to sell their produce. Farmers can add new requests for selling their crops, specifying the details of their produce and sending them to nearby mandi shops. They also

have access to their transaction history, allowing them to view past interactions and payments. This module offers a payment gateway to facilitate secure and timely payments, ensuring farmers are paid promptly for their produce. Through this module, farmers can efficiently manage their sales, leading to better market access and financial stability.

CHAPTER-7

ANDROID ENVIRONMENT

7.1 Software Installation

Installing Android Studio on Windows is a comprehensive process that allows you to set up a complete development environment for building Android applications. Here's a detailed step-by-step guide to help you through the entire installation process:

7.1.1 Download Android Studio:

- First, navigate to the official Android Studio website at <https://developer.android.com/studio>. On the homepage, you'll find the option to download Android Studio for Windows. Click on the "Download Android Studio" button.
- **Run the Installer:**
 - Once the download is complete, locate the downloaded executable file (usually named something like android-studio-ide-xxxxxxx.exe) in your Downloads folder or wherever you saved it. Double-click the file to launch the Android Studio Setup Wizard.
- **Choose Installation Type:**
 - The Android Studio Setup Wizard will prompt you to choose an installation type. Select "Standard" if you want to install Android Studio with the default settings. If you prefer to customize the installation, choose "Custom" and specify the components you want to install. Click "Next" to continue.
- **Select Components:**
 - In the next step, you'll be asked to choose the components you want to install. This typically includes Android Studio, the Android SDK, and the Android Virtual Device (AVD) emulator. Ensure that these components are selected and click "Next" to proceed.
- **Choose Installation Location:**
 - Now, you'll need to choose the installation location for Android Studio and the Android SDK. The default location is usually fine, but you can specify a different directory if you prefer. Once you've chosen the installation location, click "Next" to continue.
- **Select Start Menu Folder:**
 - In this step, you can choose the Start Menu folder where Android Studio

shortcuts will be created. You can leave the default folder or specify a different one. Click "Install" to begin the installation process.

- **Install Android Studio:**
 - Android Studio will now begin installing the selected components. This process may take some time depending on your internet connection speed and the performance of your computer. You'll see a progress bar indicating the installation progress.
- **Complete Installation:**
 - Once the installation is complete, you'll see a confirmation message. Click "Next" to proceed. You may also be prompted to import previous settings from a previous installation of Android Studio. If this is your first time installing Android Studio, you can choose the default settings. Click "Finish" to complete the installation.
- **Launch Android Studio:**
 - After installation, you can launch Android Studio from the Start menu or by double-clicking the Android Studio icon on your desktop. The first time you launch Android Studio, it may take some time to set up the initial configuration.
- **Set Up Android Studio:**
 - When you first launch Android Studio, you'll be prompted to set up the Android SDK, download additional components, and configure emulator settings. Follow the on-screen instructions to complete the setup process.
- **Start Developing:**
 - Once Android Studio is set up, you're ready to start developing Android applications. Create a new project, explore the features of Android Studio, and begin coding your app!

By following these steps, you can install Android Studio on Windows and set up a complete development environment for building Android applications.

7.1.2 Download and Install Android Studio

Step 1. To download the Android Studio, visit the official Android Studio website in your web browser.

Step 2. Click on the "Download Android Studio" option.

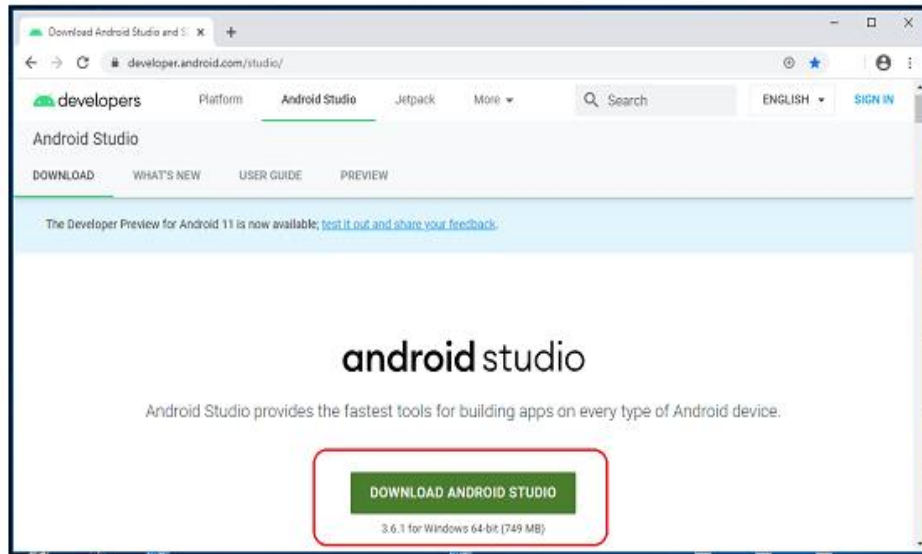


Figure 12: INSTALLATION

Step 3. Double-click on the downloaded "Android Studio-ide.exe" file.

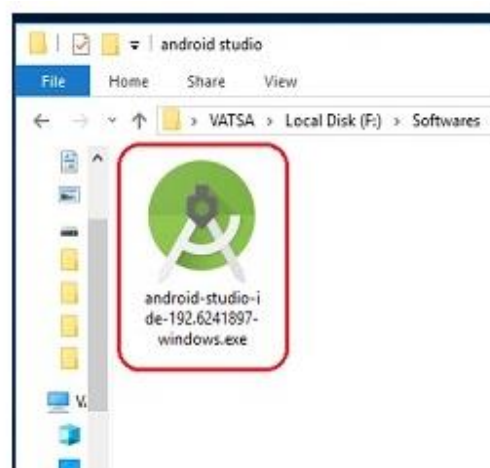


Figure 13: Android Studio-ide.exe

Step 4. "Android Studio Setup" will appear on the screen and click "Next" to proceed.

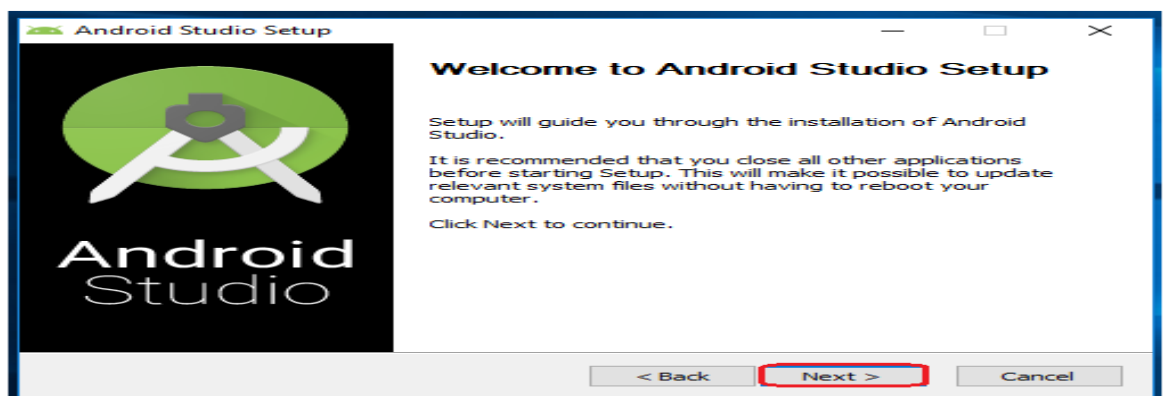


Figure 14: Android Studio Setup

Step 5. Select the components that you want to install and click on the "Next" button.

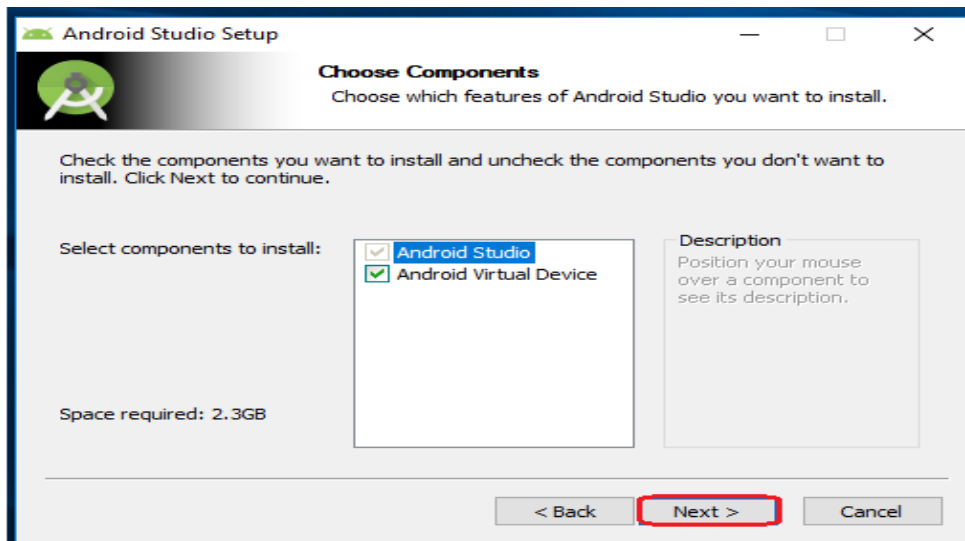


Figure 15: Next

Step 6. Now, browse the location where you want to install the Android Studio and click "Next" to proceed.

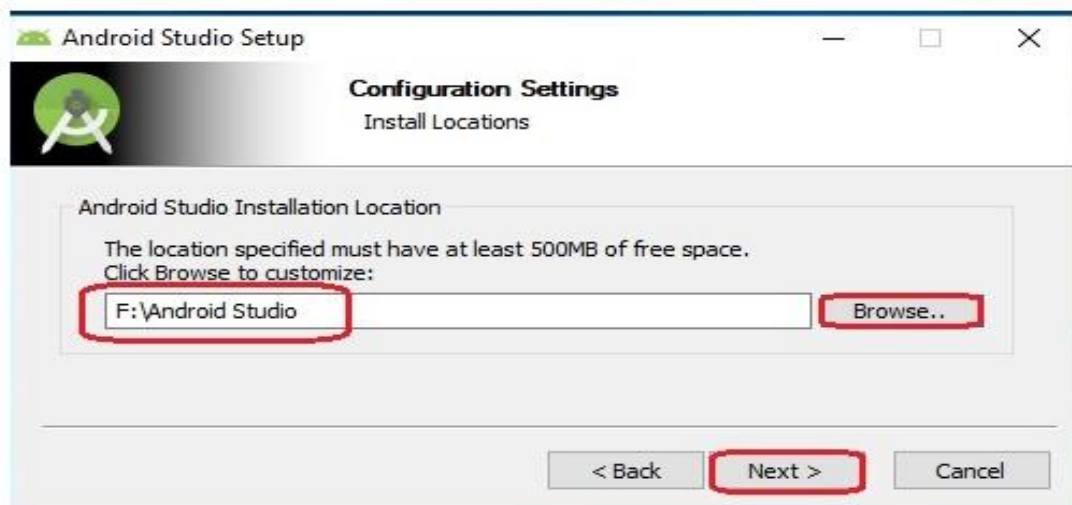


Figure 16: BROWSE LOCATION

Step 7. Choose a start menu folder for the "Android Studio" shortcut and click the "Install" button to proceed.

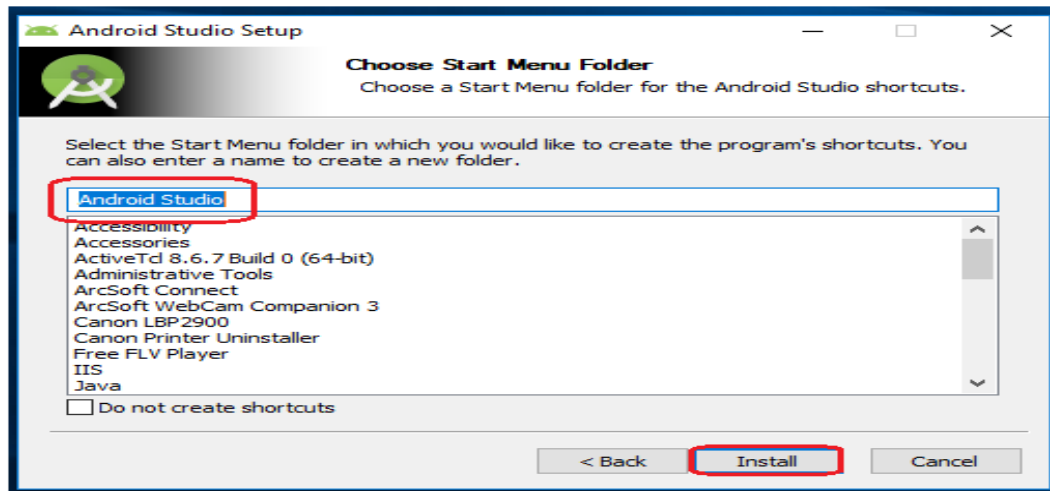


Figure 17: CLICK INSTALL

Step 8. After the successful completion of the installation, click on the "Next" button.

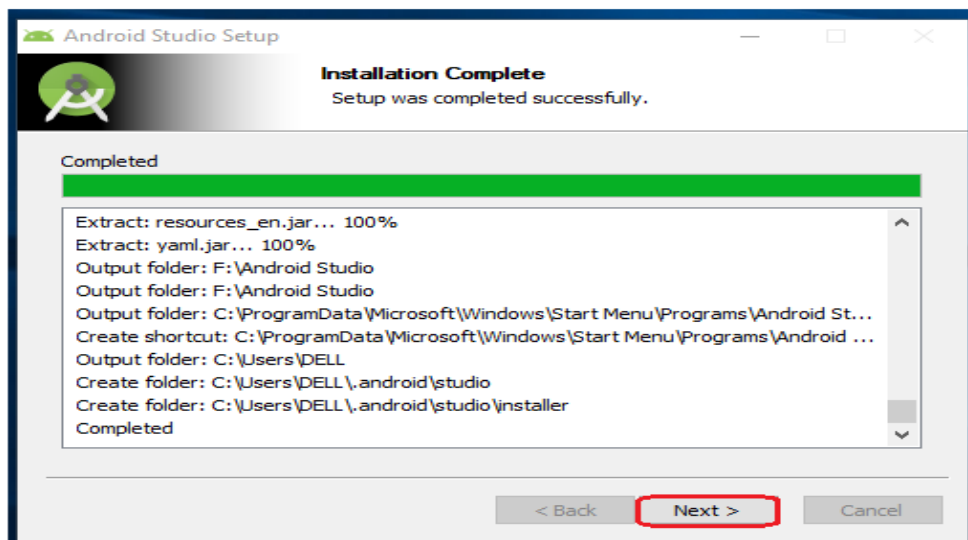


Figure 18: AFTER INSTALL

Step 9. Click on the "Finish" button to proceed.

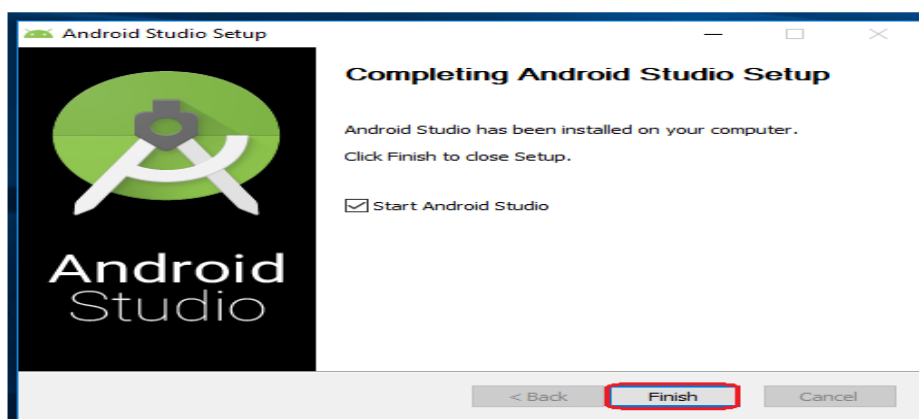


Figure 19 : FINISH

Now, your Android studio welcome screen will appear on the screen.



Figure 20: WELCOME SCREEN

7.1.3 Android Studio Setup Configuration

Step 10. "Android Studio Setup Wizard" will appear on the screen with the welcome wizard.

Click on the "Next" button.

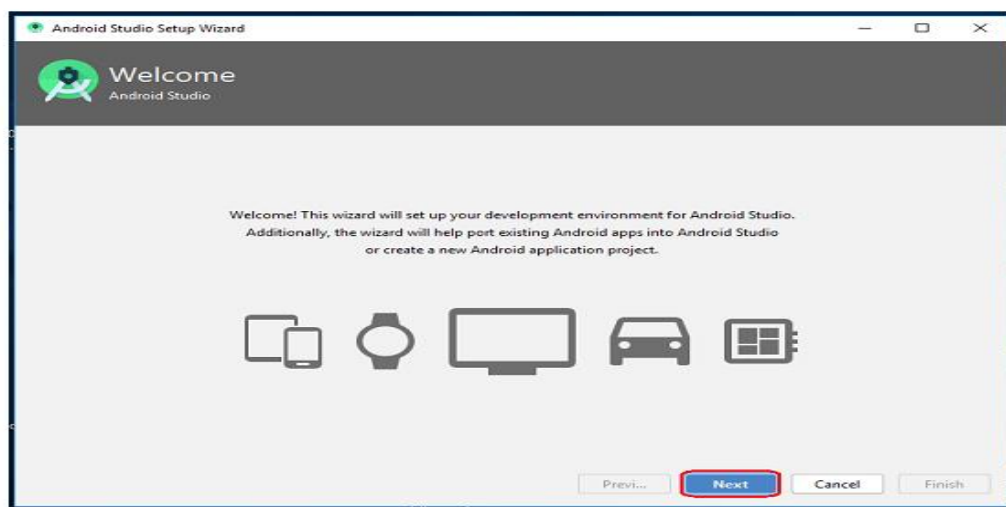


Figure 21: SETUP CONFIGURATION

Step 11. Select (check) the "Standard" option if you are a beginner and do not have any idea about Android Studio. It will install the most common settings and options for you. Click "Next" to proceed.

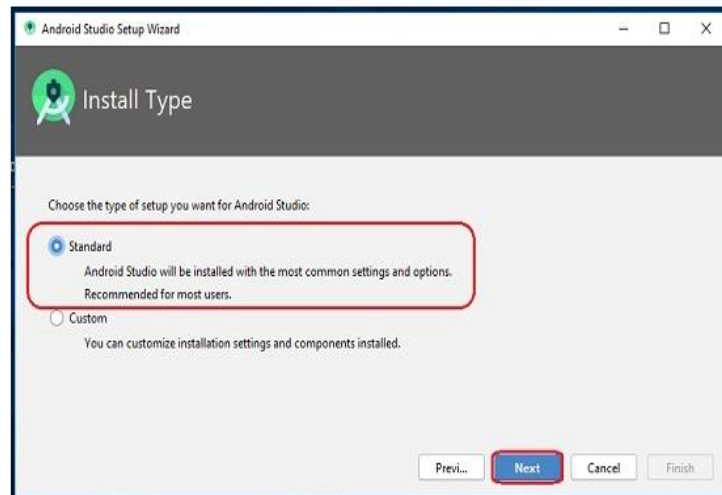


Figure 22: SELECT STANDARD

Step 12. Now, select the user interface theme as you want. (I prefer the Dark theme (Dracula) that is most liked by the coders). Then, click on the "Next" button.

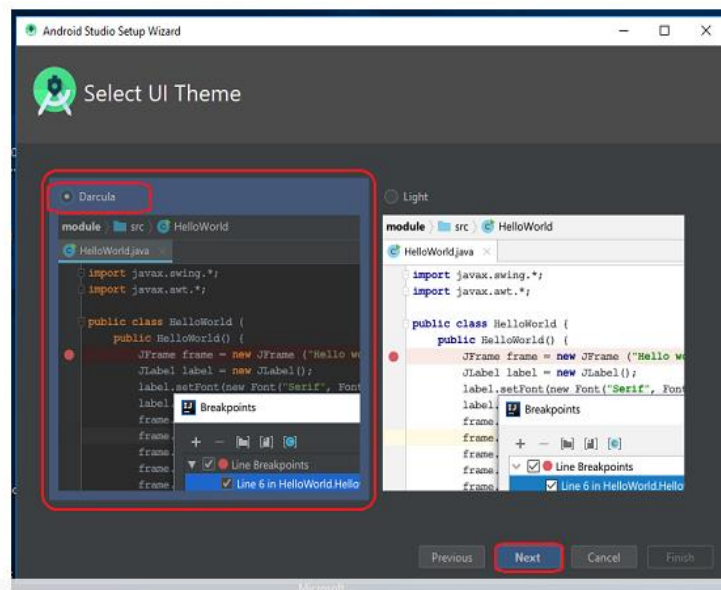


Figure 23: THEMES

Step 13. Now, click on the "Finish" button to download all the SDK components.

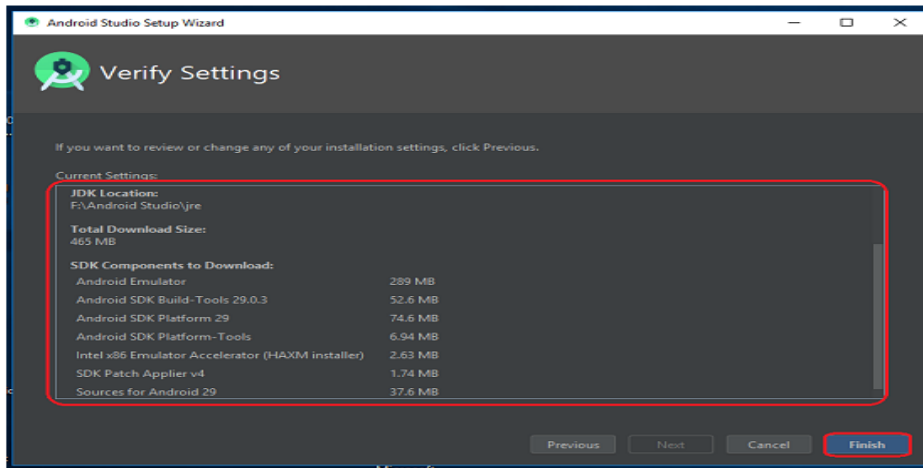


Figure 24: FINISH

And, the downloading and installation process of components gets started.

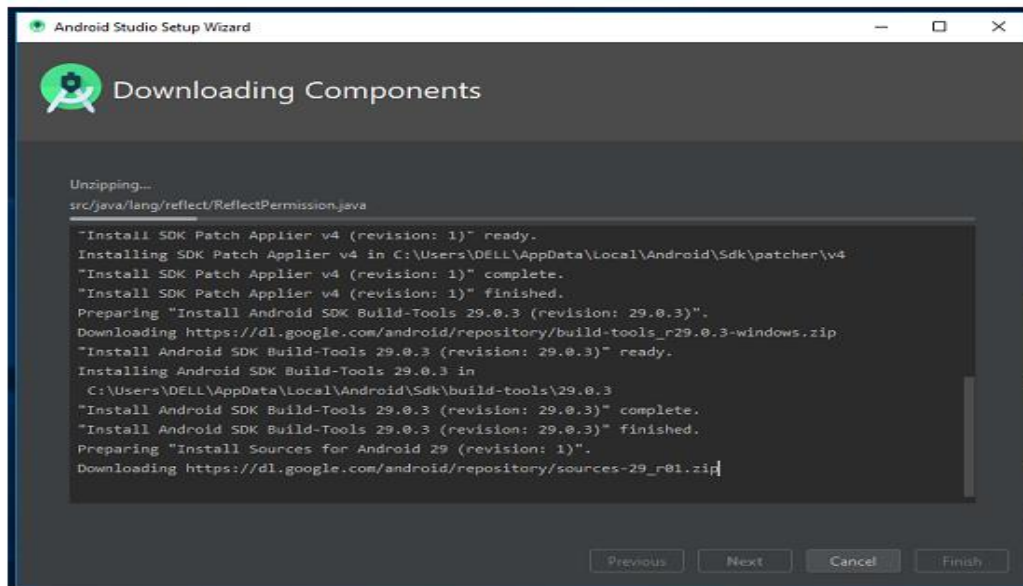


Figure 25: DOWNLOADING

Step 14. After downloading all the necessary components, click on the "Finish" button.

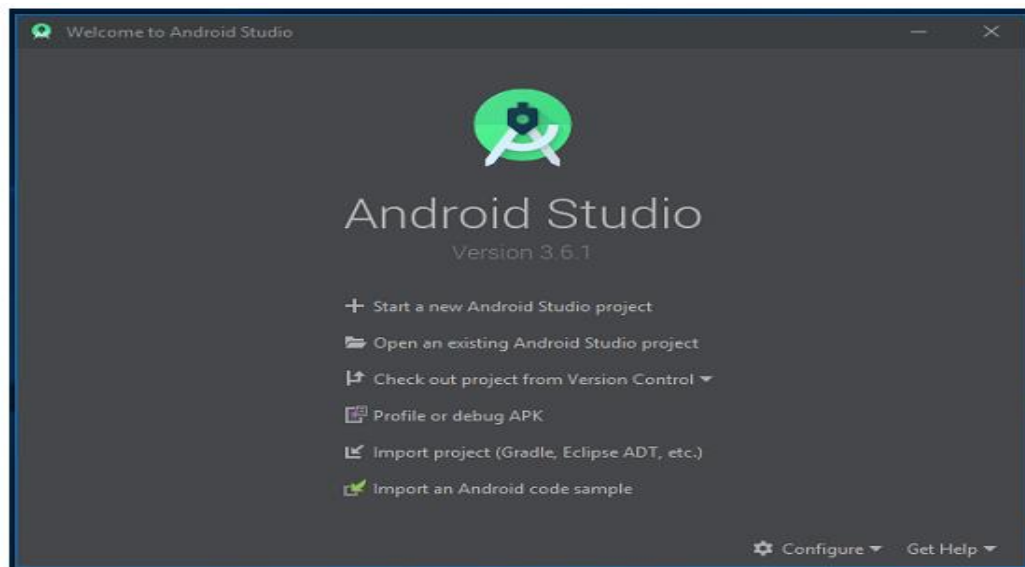


Figure 26: FINISH DOWNLOADING

7.2 SOFTWARE DEVELOPMENT LIFE CYCLE

The meaning of Agile is swift or versatile. "Agile process model" refers to a software development approach based on iterative development. Agile methods break tasks into smaller iterations, or parts do not directly involve long term planning. The project scope and requirements are laid down at the beginning of the development process. Plans regarding the number of iterations, the duration and the scope of each iteration are clearly defined in advance. Each iteration is considered as a short time "frame" in the Agile process model, which typically lasts from one to four weeks. The division of the entire project into smaller parts helps to minimize the project risk and to reduce the overall project delivery time requirements. Each iteration involves a team working through a full software development life cycle including planning, requirements analysis, design, coding, and testing before a working product is demonstrated to the client.

Actually, Agile model refers to a group of development processes. These processes share some basic characteristics but do have certain subtle differences among themselves. A few Agile SDLC models are given below: Crystal A tern Feature-driven development Scrum Extreme programming (XP) Lean development Unified process In the Agile model, the requirements are decomposed into many small parts that can be incrementally developed.

The Agile model adopts Iterative development. Each incremental part is developed over an iteration. Each iteration is intended to be small and easily manageable and that can be completed within a couple of weeks only. At a time one iteration is planned, developed and

deployed to the customers. Long-term plans are not made.

Agile model is the combination of iterative and incremental process models. Steps involve in agile SDLC models are:

- Requirement gathering
- Requirement Analysis
- Design Coding
- Unit testing
- Acceptance testing

The time to complete an iteration is known as a Time Box. Time-box refers to the maximum amount of time needed to deliver an iteration to customers. So, the end date for an iteration does not change. Though the development team can decide to reduce the delivered functionality during a Time-box if necessary to deliver it on time. The central principle of the Agile model is the delivery of an increment to the customer after each Time-box.

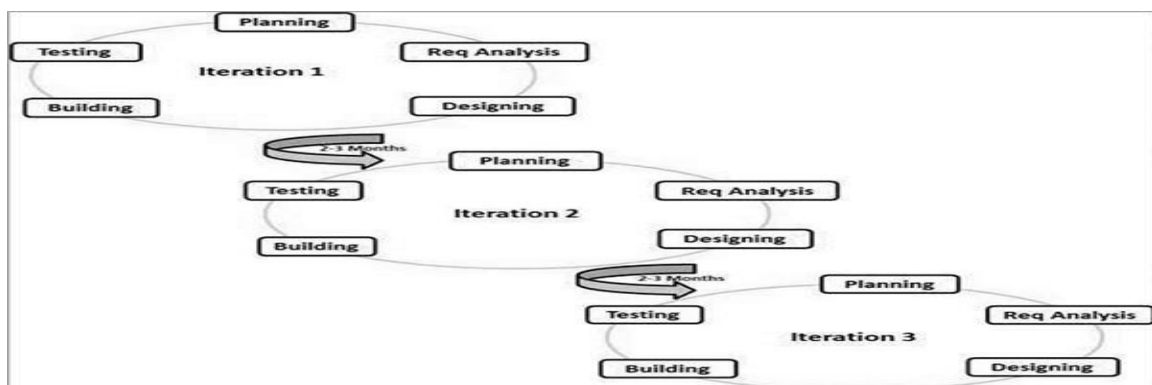


Figure 27: SOFTWARE DEVELOPMENT LIFE CYCLE

7.3 Principles of Agile model:

- To establish close contact with the customer during development and to gain a clear understanding of various requirements, each Agile project usually includes a customer representative on the team. At the end of each iteration stakeholders and the customer representative review, the progress made and re-evaluate the requirements.
- Agile model relies on working software deployment rather than comprehensive documentation.
- Frequent delivery of incremental versions of the software to the customer representative in intervals of few weeks.
- Requirement change requests from the customer are encouraged and efficiently incorporated.

-
- It emphasizes on having efficient team members and enhancing communications among them is given more importance. It is realized that enhanced communication among the development team members can be achieved through face-to-face communication rather than through the exchange of formal documents.
 - It is recommended that the development team size should be kept small (5 to 9 people) to help the team members meaningfully engage in face-to-face communication and have collaborative work environment.
 - Agile development process usually deploys Pair Programming. In Pair programming, two programmers work together at one work-station. One does code while the other reviews the code as it is typed in. The two programmers switch their roles every hour or so.

7.4 Advantages:

- Working through Pair programming produce well written compact programs which has fewer errors as compared to programmers working alone.
- It reduces total development time of the whole project. Customer representatives get the idea of updated software products after each iteration. So, it is easy for him to change any requirement if needed.

7.5 Disadvantages:

- ☐ Due to lack of formal documents, it creates confusion and important decisions taken during different phases can be misinterpreted at any time by different team members.
- ☐ Due to the absence of proper documentation, when the project completes and the developers are assigned to another project, maintenance of the developed project can

7.6 SOFTWARE ENVIRONMENT

7.6.1 Software Environment

The software environment for Android development typically consists of the following components:

7.6.2 Operating System: You can develop Android applications on various operating systems, including Windows, macOS, and Linux. However, macOS and Linux are often preferred by developers due to their Unix-based nature, which provides better compatibility with Android development tools.

7.6.3 Integrated Development Environment (IDE): The primary tool for Android development is Android Studio, which is an official IDE provided by Google. Android Studio is based on IntelliJ IDEA and offers a feature-rich environment for coding, debugging, and testing Android applications.

7.6.4 Android SDK (Software Development Kit): The Android SDK provides the necessary tools, libraries, and APIs for building Android applications. It includes tools for compiling code, debugging, and packaging applications, as well as platform-specific APIs for interacting with device features.

7.6.5 Java Development Kit (JDK): Android applications are primarily developed using the Java programming language. Therefore, you need to have the Java Development Kit (JDK) installed on your system to compile and run Java code. Android Studio comes bundled with OpenJDK, which is sufficient for most development purposes.

7.6.6 Android Virtual Device (AVD): The Android Virtual Device is an emulator that allows you to test your Android applications on virtual devices with different configurations. You can create and manage virtual devices using the AVD Manager in Android Studio.

7.6.7 Version Control System: Version control systems like Git are essential for managing source code and collaborating with other developers. Android Studio has built-in support for Git, allowing you to easily commit, push, and pull changes to and from remote repositories.

7.6.8 Build Automation Tools: Gradle is the official build system for Android development. It is responsible for compiling code, resolving dependencies, and packaging applications into APK (Android Package) files. Android Studio integrates with Gradle, providing a user-friendly interface for configuring build settings.

7.6.9 Device Testing Tools: In addition to testing applications on virtual devices, it's essential to test them on real Android devices to ensure compatibility and performance. Android Studio supports deploying and debugging applications on physical devices connected via USB or over a network.

7.6.10 Third-Party Libraries and Frameworks: Developers often use third-party libraries and frameworks to streamline development and add functionality to their applications. Common examples include Retrofit for networking, Picasso for image loading, and Dagger for dependency injection.

By setting up this software environment, you'll have everything you need to develop, test, and deploy Android applications effectively.

7.7 Features: -

1. **Application framework** enabling reuse and replacement of components
2. **Dalvik virtual machine** optimized for mobile devices
3. **Integrated browser** based on the open source Web Kit engine
4. **Optimized graphics** powered by a custom 2D graphics library; 3D graphics based on the OpenGL ES 1.0 specification (hardware acceleration optional)
5. **SQLite** for structured data storage
6. **Media support** for common audio, video, and still image formats (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF)
7. **GSM Telephony** (hardware dependent)
8. **Bluetooth, EDGE, 3G, and WIFI** (hardware dependent)
9. **Camera, GPS, compass, and accelerometer** (hardware dependent)
10. **Rich development environment** including a device emulator, tools for debugging, memory and performance profiling, and a plugin for the Eclipse IDE

Android Architecture:

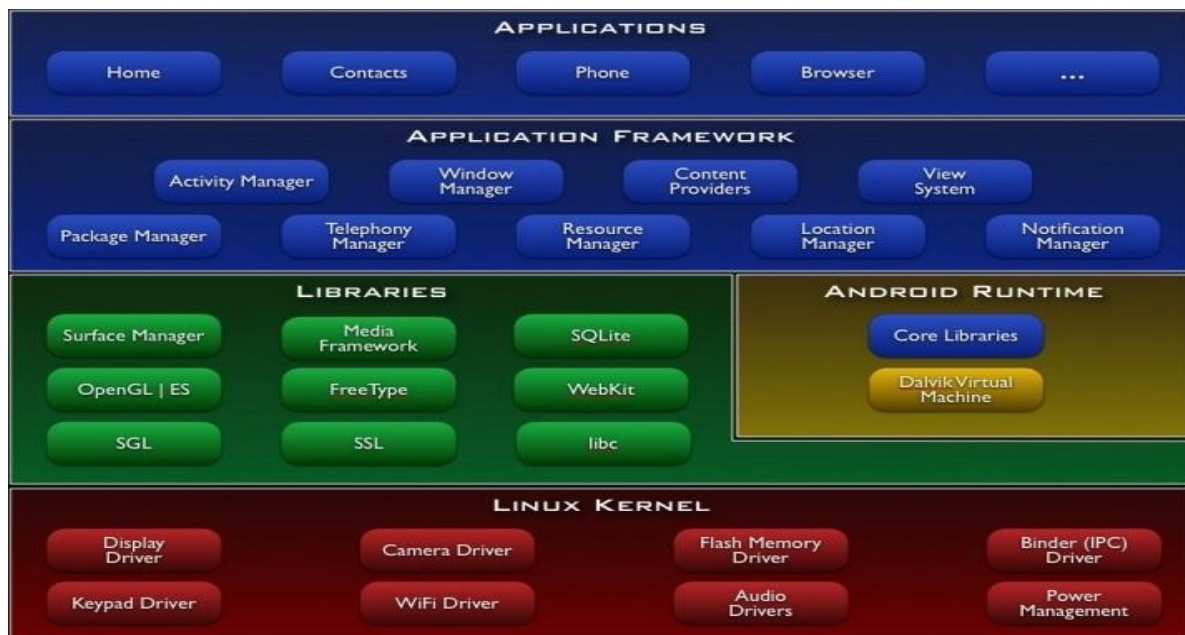


Figure 28: ANDROID ARCHITECTURE

The Android architecture consists of several layers that work together to provide a robust platform for developing and running mobile applications. Understanding the Android architecture is crucial for developers to build efficient and scalable applications. Here's an overview of the Android architecture:

Linux Kernel: At the core of the Android architecture lies the Linux kernel. Android is built on top of the Linux operating system, which provides core services such as hardware abstraction, memory management, process management, and security features.

Hardware Abstraction Layer (HAL): The Hardware Abstraction Layer (HAL) provides a standardized interface for accessing hardware components such as camera, audio, display, and sensors. It allows Android to support a wide range of hardware configurations and device types.

Android Runtime (ART): Android applications are primarily written in Java or Kotlin and run on the Android Runtime (ART). ART is a managed runtime environment that executes application code and provides features such as memory management, garbage collection, and runtime optimizations.

Native C/C++ Libraries: Android includes a set of native C/C++ libraries that provide core system functionality, such as graphics rendering (OpenGL ES), database management (SQLite), and multimedia playback (Media framework).

Java API Framework: The Java API Framework consists of a set of high-level Java APIs that developers use to build Android applications. These APIs provide access to system services and features, such as user interface components (Views and ViewGroups), data storage (SharedPreferences, SQLite), connectivity (Wi-Fi, Bluetooth), and location services (GPS, Maps).

System Services: Android includes a set of system services that run in the background and provide essential functionalities to applications. These services include Activity Manager, Content Provider, Location Manager, Notification Manager, and Telephony Manager.

Application Framework: The Application Framework provides a set of reusable components and libraries that developers use to build Android applications. It includes components such as Activities, Fragments, Services, Broadcast Receivers, and Content Providers, which form the building blocks of Android applications.

User Interface (UI) Toolkit: Android provides a rich set of UI components and widgets that developers use to create the user interface of their applications. These components include layouts, views, widgets, and resource files (XML) for defining UI layouts and styles.

Android System Apps: Android comes pre-installed with a set of system apps that provide core functionality, such as Phone, Contacts, Messaging, Browser, and Settings. These apps are built using the same APIs and frameworks available to third-party developers.

By understanding the Android architecture, developers can leverage the platform's features and components effectively to build high-quality and performant applications for a wide range of devices and use cases.

Hardware running Android

The Android operating system, initially released in 2008, has become one of the most popular and versatile platforms for mobile devices, including cellphones, tablets, netbooks, TVs, and more. The main supported platform for Android is the ARM architecture, which provides efficient performance and power consumption suitable for mobile devices.

The first commercially available phone to run Android was the HTC Dream, released in October 2008. Since then, numerous devices from various manufacturers, including Samsung,

Google, and HTC, have adopted Android as their operating system.

In early 2010, Google collaborated with HTC to launch its flagship Android device, the Nexus One, followed by the Samsung-made Nexus S later that year. Despite some initial challenges with bugs, documentation, and QA infrastructure, developers quickly began creating applications for the Android platform.

The Android Dev Phone, a SIM-unlocked and hardware-unlocked device designed for advanced developers, provides an environment for testing and developing Android applications. Additionally, the Android Software Development Kit (SDK) includes a comprehensive set of development tools, such as a debugger, libraries, emulator, documentation, and sample code.

Supported development platforms for building Android applications include Linux, macOS, and Windows. The officially supported integrated development environment (IDE) is Eclipse, though developers can use any text editor and command line tools to create, build, and debug Android applications.

Android applications are packaged in .apk (Android Package) format, which contains compiled byte code files (.dex), resource files, and other assets. These .apk files are stored in the /data/app folder on the Android OS, accessible only to the root user for security reasons. Overall, the Android ecosystem offers a robust platform for developers to create a wide range of applications for diverse hardware devices, leveraging the flexibility and power of the ARM architecture and the comprehensive development tools provided by the Android SDK.

Android operation System

Android is an operating system based on Linux with a Java programming interface. It provides tools, e.g., a compiler, debugger and a device emulator as well as its own Java Virtual machine (Dalvik Virtual Machine - DVM). Android is created by the Open Handset Alliance which is led by Google. Android uses a special virtual machine, e.g., the Dalvik Virtual Machine. Dalvik uses special bytecode. Therefore, you cannot run standard Java bytecode on Android. Android provides a tool "dx" which allows to convert Java Class files into "dex" (Dalvik Executable) files. Android applications are packed into an .apk (Android Package) file by the program "aapt" (Android Asset Packaging Tool) To simplify development Google provides the Android Development Tools (ADT) for Eclipse . The ADT performs automatically the conversion from class to dex files and creates the apk during deployment. Android supports 2-D and 3-D graphics using the OpenGL libraries and supports data storage in a SQLite database. Every Android applications run in its own process and under its own user id which

is generated automatically by the Android system during deployment. Therefore, the application is isolated from other running applications and a misbehaving application cannot easily harm other Android applications.

Important Android Components

Several key components form the foundation of Android development. Understanding these components is essential for building robust and feature-rich Android applications. Here are some important Android components:

Activities: Activities represent the UI (user interface) of an application. Each screen in an Android app is typically represented by an activity. Activities manage user interactions, such as receiving input and rendering views.

Fragments: Fragments are modular components that represent a portion of a user interface or behavior within an activity. They enable developers to build flexible and responsive UIs that can adapt to different screen sizes and orientations.

Views and View Groups: Views are UI elements, such as buttons, text fields, and images, that are used to construct the user interface of an Android app. View Groups are containers that hold and arrange multiple views. Common View Groups include Linear Layout, Relative Layout, and Constraint Layout.

Intents: Intents are messaging objects used to communicate between components of an Android application and between different applications. They can be used to start activities, broadcast messages, and perform other actions.

Services: Services are background processes that can run independently of the user interface. They are often used for tasks that need to continue running even when the app is not in the foreground, such as playing music, fetching data from the internet, or performing background computations.

Broadcast Receivers: Broadcast Receivers are components that respond to system-wide broadcast messages. They enable applications to receive and react to events or messages from other applications or the system itself.

Content Providers: Content Providers manage access to a structured set of data. They allow different applications to share data with each other in a secure and standardized way. Content Providers are commonly used to store and retrieve data from databases, files, or other sources.

Shared Preferences: Shared Preferences are a simple key-value storage mechanism provided by the Android framework. They are often used to store small amounts of persistent data, such as user preferences or settings.

SQLite Database: SQLite is a lightweight relational database management system included as a core component of the Android platform. It enables developers to store and manage structured data in their applications.

Manifest File: The `AndroidManifest.xml` file is a crucial component of every Android application. It contains essential information about the application, such as its package name, version, permissions, and the components it contains.

Understanding how these components work together is essential for building successful Android applications. Developers leverage these components to create engaging, responsive, and efficient user experiences on the Android platform.

Android Source Code

The following step is optional.

During Android development it is very useful to have the Android source code available as Android uses a lot of defaults.

Haris Peco maintains plugins with provides access to the Android Source code. Use the Eclipse update manager to install two of his plugins.

Create an Android Emulator Device

Creating an Android emulator device allows you to test your Android applications on virtual devices with different configurations. Here's how you can create an Android emulator device using Android Studio:

- **Open Android Studio:** Launch Android Studio on your computer.
- **Open AVD Manager:** Click on the "AVD Manager" icon in the toolbar. It looks like a phone with an Android logo on it.

-
- **Create a New Virtual Device:** In the AVD Manager window, click on the "Create Virtual Device" button.
 - **Select Hardware Profile:** Choose a hardware profile for your virtual device. This defines the device's specifications, such as screen size, resolution, and hardware capabilities. You can select a predefined profile or create a custom one.
 - **Select System Image:** Choose a system image for your virtual device. The system image represents the version of Android that the device will run. You can download system images for different Android versions and device types from the list provided.
 - **Configure Device Settings:** Customize additional settings for your virtual device, such as the amount of RAM, the amount of internal storage, and whether to enable or disable hardware acceleration.
 - **Finish Setup:** Once you've configured the device settings, click on the "Finish" button to create the virtual device.
 - **Launch the Virtual Device:** In the AVD Manager window, you'll see the newly created virtual device listed. Click on the "Play" button next to the device to launch it.
 - **Wait for Boot:** The virtual device will start booting up, just like a real Android device. This process may take a few minutes, depending on your computer's performance.
 - **Test Your App:** Once the virtual device has finished booting, you can test your Android application on it. Simply run your app from Android Studio, and it will be installed and launched on the virtual device.

By following these steps, you can create an Android emulator device and test your applications on virtual devices with different configurations. This allows you to ensure that your app works correctly across a variety of Android devices and screen sizes.

CHAPTER-8

SYSTEM STUDY AND TESTING

8.1 FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

- **ECONOMICAL FEASIBILITY**
- **TECHNICAL FEASIBILITY**
- **SOCIAL FEASIBILITY**

8.2 ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus, the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

8.3 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

8.4 SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

8.5 SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

8.6 Types of test & Test Cases

8.6.1 UNIT TESTING

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. It is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

8.6.2 INTEGRATION TESTING

Integration tests are designed to test integrated software components to determine if they run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfactory, as shown by successful unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

8.6.3 FUNCTIONAL TEST

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centred on the following items:

- Valid Input : identified classes of valid input must be accepted.
- Invalid Input : identified classes of invalid input must be rejected.
- Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked. Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified, and the effective value of current tests is determined.

8.6.4 SYSTEM TEST

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

8.6.5 WHITE BOX TESTING

White Box Testing is a testing in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is used to test areas that cannot be reached from a black box level.

8.6.6 BLACK BOX TESTING

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box. you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

8.6.7 UNIT TESTING:

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

All field entries must work properly.

Pages must be activated from the identified link.

The entry screen, messages and responses must not be delayed.

Features to be tested

Verify that the entries are of the correct format

No duplicate entries should be allowed

All links should take the user to the correct page.

8.7 INTEGRATION TESTINGSS

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g., components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

8.7.1 ACCEPTANCE TESTING

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

8.7.2 TESTING CASES

Test case id	Test Scenario	Test Steps	Prerequisites	Test Data	Expected result	Actual result	Test status
#CVD001	To authenticate a successful signup with user data	<ul style="list-style-type: none"> User navigate the signup page Enter the valid user data 	User data	Username Password Mobile Email location	When the user submits the user data, data should be store in database successfully	As Expected,	Pass

		<ul style="list-style-type: none"> Click on signup button 					
#CVD002	To authenticate a successful login with user data	<ul style="list-style-type: none"> User navigate the login page Enter the valid username, password Click on login button 	Username , password	Username, password	When the user submits the user data, data should be authenticated successfully	As Expected,	Pas s

Table 1: TEST CASES

CHAPTER-9

TIMELINE FOR EXECUTION OF PROJECT (GANTT CHART)

A Gantt chart is a project management tool used to visually represent a project schedule. It displays tasks, activities, or events along a timeline, helping teams track progress, allocate resources, and manage deadlines effectively.

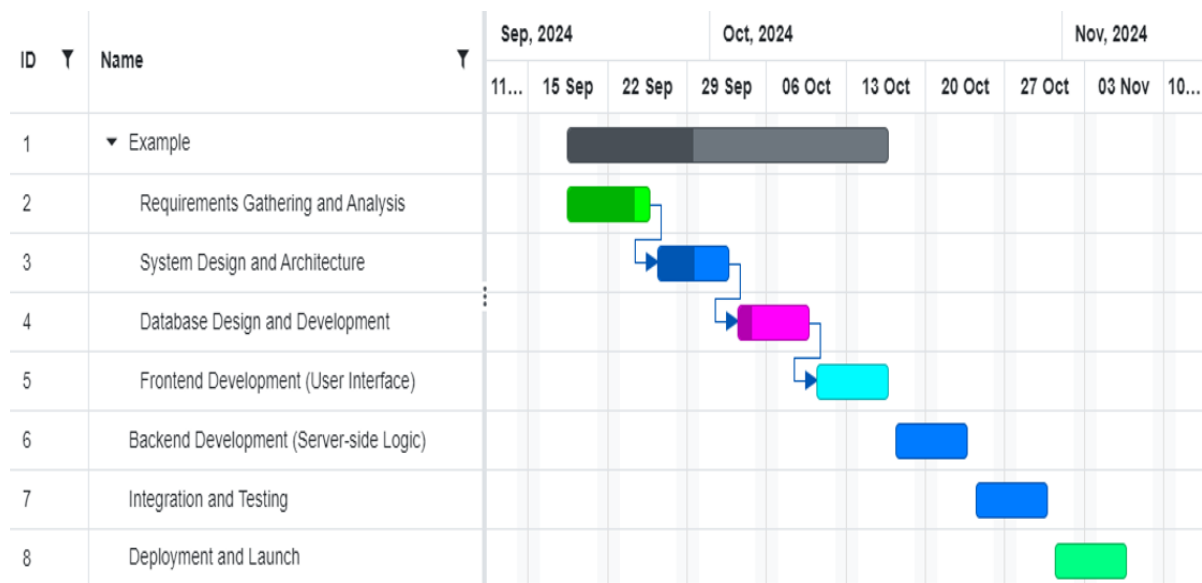


Figure 29: GANTT CHART

CHAPTER-10

OUTCOMES

The Android application achieves several critical outcomes aimed at enhancing the agricultural value chain.

1. **Streamlined Communication:** Farmers and mandi shops benefit from direct and real-time interaction facilitated by the application. This removes the need for intermediaries, reducing delays and misunderstandings. Farmers can view updated produce prices and mandi shops can process requests seamlessly.
2. **Enhanced Transparency:** Transparency is significantly improved through features like transaction tracking, secure payment systems, and detailed records of interactions. This ensures both farmers and mandi shops have a clear view of their operations, building trust in the system.
3. **Improved Market Access:** By utilizing location services, farmers are connected with the nearest mandi shops. This enables them to reduce travel time and costs, ensuring they can quickly and conveniently sell their produce.
4. **Operational Efficiency:** The app's modular design provides tailored solutions for farmers, mandi shops, and administrators. Farmers can manage requests and payments efficiently, mandi shops can track produce costs and requests, and administrators can oversee the platform's operations to maintain integrity and smooth functionality.
5. **Empowerment of Stakeholders:** Through this application, farmers gain access to timely payments and a larger customer base, while mandi shops can improve their business operations. The administrators ensure a well-regulated platform, fostering a balanced ecosystem.

CHAPTER-11

RESULTS AND DISCUSSIONS

The results of the Android application demonstrate a successful transition from a traditional, inefficient agricultural system to a digital, streamlined process. The key achievements are grounded in addressing historical challenges faced by farmers and mandi shops, including delayed payments, poor market accessibility, and lack of price transparency.

By providing a platform that enables real-time updates and integrates location services, the app ensures farmers are always informed about the best opportunities to sell their produce. This helps them secure fair prices and reduces dependence on middlemen, fostering financial independence. Mandi shops, on the other hand, benefit from efficient management tools to handle farmer requests and update prices based on market trends. The result is a harmonious interaction between stakeholders, eliminating barriers like miscommunication and delays.

Additionally, the app's user-friendly interface ensures that even users with limited technical knowledge can navigate its features effectively. This simplicity is critical in rural areas, where technology adoption may be slower. However, challenges such as limited access to smartphones and digital literacy among some farmers must be addressed. Initiatives such as training programs or partnerships with local organizations could amplify the app's reach and impact.

The potential for future enhancements is another strong point of discussion. Integrating advanced features such as AI-based pricing recommendations, weather forecasting, and blockchain technology can add more value. Multilingual support and predictive analytics could broaden the app's appeal across diverse regions, making it an indispensable tool in the agricultural sector.

Overall, the app demonstrates a transformative impact, bridging the gap between farmers and markets while providing a scalable solution for future growth. It not only resolves existing inefficiencies but also paves the way for a more connected and empowered agricultural ecosystem.

CHAPTER-12

CONCLUSION

In conclusion, The Android application successfully bridges the gap between farmers and mandi shops, providing a streamlined and user-friendly platform for interactions. By incorporating features such as location services, real-time price updates, and transaction tracking, the app enhances communication and transparency in the agricultural value chain. Farmers can now easily connect with nearby mandi shops, manage their transactions, and make informed decisions, ultimately leading to better market efficiency and improved livelihoods for farmers. The application ensures that the process of selling agricultural products becomes more accessible and less cumbersome for all stakeholders involved.

CHAPTER-13

FUTURE ENHANCEMENT

Future enhancements for the application could include integrating AI-based recommendations for optimal crop pricing, weather forecasts, and crop health monitoring. A digital payment gateway can be expanded to include more options, improving financial convenience for farmers. The app could also offer multilingual support for greater accessibility across India, as well as a real-time chat feature between farmers and mandi shops for direct communication. Additionally, blockchain technology could be integrated to ensure transparency in transactions, while predictive analytics could help farmers decide the best time to sell their produce, ultimately maximizing their profitability and market reach.

REFERENCES

- Agrawal, M., & Pandey, D. (2021). "Digital Solutions for Enhancing Market Access for Farmers in India." *Journal of Agricultural Sciences and Technology*, 10(2), 150-160.
- Reddy, A. A., & Mishra, D. (2020). "Challenges and Opportunities in the Indian Agricultural Market: A Farmer's Perspective." *Agricultural Economics Review*, 25(3), 120-132.
- Kumar, P., & Joshi, P. K. (2019). "Technological Innovations for Market Linkages in Indian Agriculture." *Agriculture and Food Security Journal*, 8(1), 45-53.
- Singh, R., & Sharma, K. (2020). "Role of Mobile Applications in Enhancing Market Efficiency for Small Farmers." *International Journal of Agricultural Extension*, 17(2), 95-108.
- Mukherjee, S., & Roy, A. (2018). "Improving Agricultural Market Access through ICT: A Review of Success Stories in India." *Journal of Rural Development Studies*, 14(2), 234-247.
- Jain, S., & Patel, A. (2021). "Mandi Connect: A Mobile Application to Bridge the Gap Between Farmers and Markets." *International Journal of Digital Transformation*, 5(4), 102-112.
- Gupta, V., & Singh, H. (2022). "Digitization of Agricultural Market Systems: A Case Study of Mandi Networks in India." *Journal of Applied Agricultural Research*, 9(1), 77-89.
- Deshmukh, A., & Rao, S. (2019). "Connecting Farmers to Markets through Technology: The Impact of Digital Platforms on Rural Economies." *Rural Development Journal*, 12(3), 163-174.
- Mishra, S., & Choudhury, P. (2020). "Addressing Market Inefficiencies for Farmers: A Review of Digital Agricultural Solutions." *International Journal of Agribusiness Studies*, 18(2), 112-122.
- Narayan, G., & Verma, R. (2021). "Enhancing Farmer Market Linkages Using Mobile Applications: A Review of Case Studies in India." *Journal of Agricultural Economics and Development*, 10(4), 182-194.

APPENDIX-A

PSUEDOCODE

1. Initialize Application

START

IMPORT necessary libraries (e.g., Firebase SDK, Google Maps API, UI libraries)

CONFIGURE Firebase authentication and database

SET UP Google Maps API with API key

DEFINE main activity and navigation

2. User Authentication

FUNCTION authenticateUser(email, password):

 IF email and password are valid:

 CALL Firebase Authentication to verify credentials

 RETURN success message and navigate to dashboard

 ELSE:

 RETURN error message

3. Dashboard

FUNCTION loadDashboard(userID):

 FETCH user-specific data from Firebase Database

 DISPLAY:

 - Weather updates (API call to weather service)

 - Crop information (data from Firebase)

 - Alerts (e.g., notifications on pest outbreaks)

4. Geolocation and Mapping

FUNCTION locateFarm():

 REQUEST user permission for location access

 IF permission granted:

 FETCH user's geolocation using Google Maps API

 MARK location on the map

 ALLOW user to add or modify farm location

 ELSE:

 DISPLAY error message or fallback location

5. Resource Center

FUNCTION showResources(category):

 INPUT category of interest (e.g., Crop Management, Irrigation)

 FETCH corresponding data or videos from Firebase

DISPLAY the resources in the selected category

6. Market Price Updates

FUNCTION getMarketPrices(crop):

 INPUT crop name

 CALL external API to fetch current market prices

 DISPLAY the prices for user-selected markets

7. Notification System

FUNCTION submitFeedback(userID, feedback):

 STORE feedback in Firebase database under userID

 RETURN confirmation message

8. Feedback System

FUNCTION submitFeedback(userID, feedback):

 STORE feedback in Firebase database under userID

 RETURN confirmation message

9. Settings

FUNCTION updateSettings(userID, settings):

 INPUT new settings (e.g., language, notification preferences)

 UPDATE Firebase database with new settings

 RETURN success message

10. Error Handling

FUNCTION handleError(errorCode):

 DISPLAY appropriate error message based on errorCode

11. Application Termination

FUNCTION onExit():

 SAVE user session data if necessary

 CLOSE all active connections

 TERMINATE application

END

APPENDIX-B

SCREENSHOTS

SIGN UP

Name

Number

Email

Address

City

Password

Sign in

PROFILE

Available

Name

mandi

Number

9898989898

Address

sssss

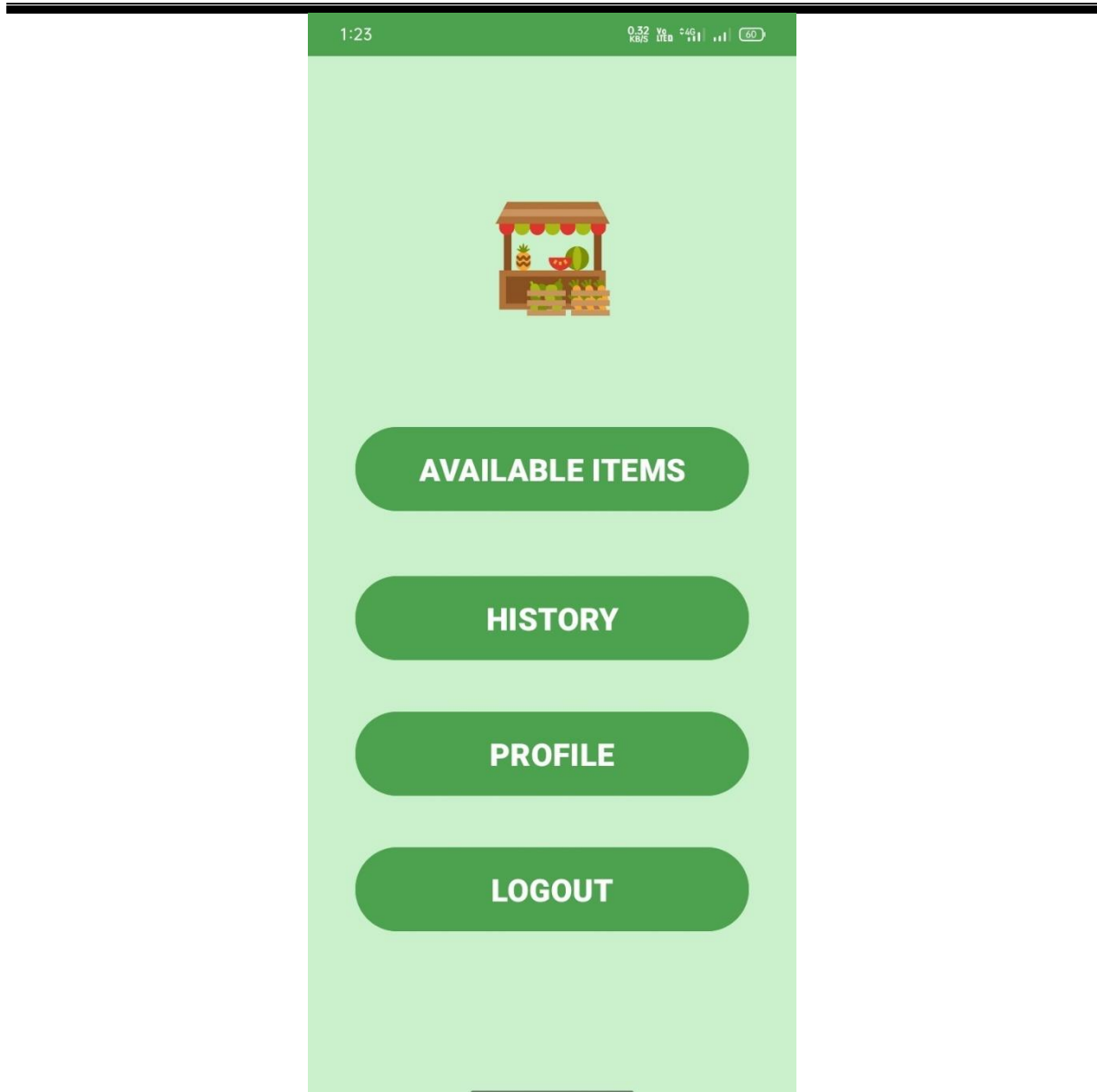
City

Tirupati

Password

....

Submit




1:2310.0 KbpsVoLTE4G601:240.97 KbpsVoLTE4G60

AVAILABLE ITEMS

Enter Your City
b

Names of Vegetables in English



www.mahindra.com

MIX VEGETABLES

I have various types of vegetables which are naturally planted and reasonable prices


450

Available

farmer Name :shreyank

call

Names of Vegetables in English



www.mahindra.com

USER

NAMEsiva

EMAILaluru.sahithi@gmail.com

NUMBER9063259740

CITYMiyapur

NAMEshreyank

EMAILsarwadeshshreyank@gmail.com

NUMBER9480315034

CITYBangalore

NAMEranga

EMAILmanojkumarkakarla2003@gmail.com

NUMBER9346604127

CITYrajankunte

NAMEkuswanth

EMAILlasya.satya.123@gmail.com

NUMBER8121059288

CITYvijayawada

NAMEtest

EMAILtester@gmail.com

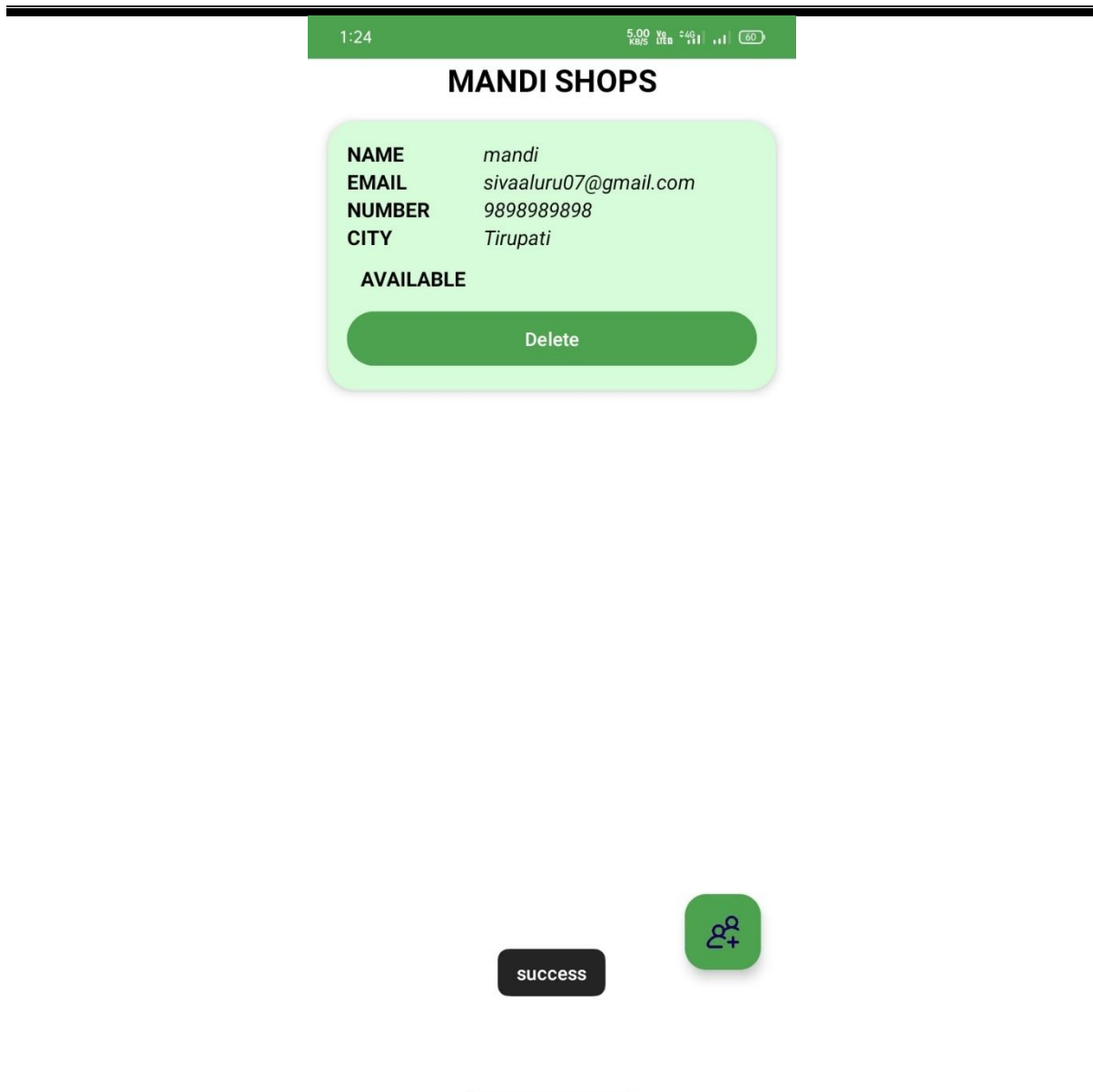
NUMBER89

CITYTirupati

NAMEtest

School of Computer Science & Engineering

58



APPENDIX-C

ENCLOSURES

1. Journal publication/Conference Paper Presented Certificates of all students.















2. Similarity Index / Plagiarism Check report clearly showing the Percentage (%). No need for a page-wise explanation.

Match Groups

-  **56 Not Cited or Quoted 19%**
Matches with neither in-text citation nor quotation marks
-  **0 Missing Quotations 0%**
Matches that are still very similar to source material
-  **0 Missing Citation 0%**
Matches that have quotation marks, but no in-text citation
-  **0 Cited and Quoted 0%**
Matches with in-text citation present, but no quotation marks

Top Sources

- 10%  Internet sources
- 2%  Publications
- 18%  Submitted works (Student Papers)

Top Sources

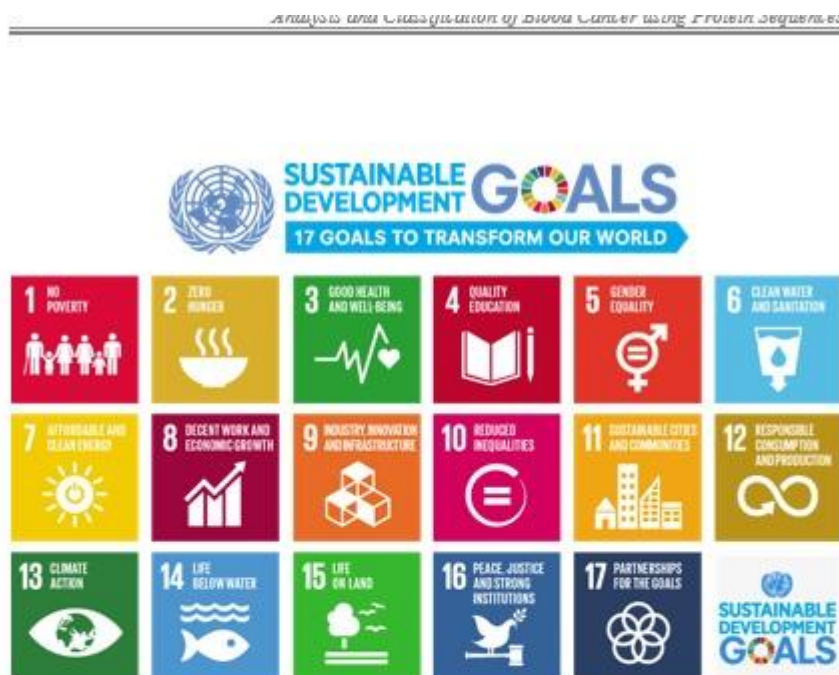
The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	Submitted works	Presidency University on 2024-01-11	3%
2	Submitted works	Bannari Amman Institute of Technology on 2024-12-17	2%
3	Internet	www.kluniversity.in	1%
4	Submitted works	Jawaharlal Nehru Technological University on 2024-11-14	1%
5	Internet	www.jetir.org	1%
6	Internet	archive.org	<1%
7	Submitted works	Presidency University on 2024-01-13	<1%
8	Internet	www.coursehero.com	<1%
9	Submitted works	Milwaukee School of Engineering on 2022-05-06	<1%
10	Submitted works	Northern Arizona University on 2024-04-15	<1%

11	Internet	www.slideshare.net	<1%
12	Submitted works	University of East London on 2023-05-10	<1%
13	Submitted works	Visvesvaraya Technological University, Belagavi on 2022-07-11	<1%
14	Submitted works	University of Wales Institute, Cardiff on 2024-08-20	<1%
15	Internet	ijarcsee.org	<1%
16	Submitted works	Jawaharlal Nehru Technological University on 2024-07-06	<1%
17	Submitted works	Hogeschool Utrecht - Tii on 2024-12-02	<1%
18	Submitted works	Midlands State University on 2014-05-13	<1%
19	Submitted works	UCL on 2025-01-02	<1%
20	Submitted works	University of Wollongong on 2023-11-21	<1%
21	Internet	krishikosh.egranth.ac.in	<1%
22	Submitted works	Regent Independent School and Sixth Form College on 2023-11-09	<1%
23	Submitted works	Texas A&M University, Galveston on 2023-06-22	<1%
24	Submitted works	Federal University of Technology on 2022-11-29	<1%

25	Submitted works	Manipal University on 2023-11-15	<1%
26	Submitted works	Sunway Education Group on 2023-04-28	<1%
27	Submitted works	Taylor's Education Group on 2024-12-09	<1%
28	Submitted works	University of Sydney on 2023-05-12	<1%
29	Internet	tms-outsource.com	<1%
30	Submitted works	California Southern University on 2024-11-07	<1%
31	Submitted works	Danford College on 2024-06-23	<1%
32	Submitted works	Manipal University on 2022-07-05	<1%
33	Submitted works	Presidency University on 2024-01-11	<1%
34	Submitted works	SASTRA University on 2014-04-10	<1%
35	Internet	www.it.itb.ac.in	<1%
36	Submitted works	Sreenidhi International School on 2017-05-08	<1%
37	Submitted works	Jawaharlal Nehru Technological University Kakinada on 2020-06-17	<1%
38	Submitted works	Sreenidhi International School on 2023-06-08	<1%
39	Submitted works	University of Duhok on 2014-03-13	<1%
40	Submitted works	islingtoncollege on 2024-12-24	<1%

4. Details of mapping the project with the Sustainable Development Goals (SDGs).



The Project work carried out here is mapped to SDG-3 Good Health and Well-Being.

The project work carried here contributes to the well-being of the human society. This can be used for Analyzing and detecting blood cancer in the early stages so that the required medication can be started early to avoid further consequences which might result in mortality.

This project connects farmers to mandi via an app, the project can be mapped to the following Sustainable Development Goals (SDGs):

1. SDG-2 Zero Hunger:

This project primarily focuses on empowering farmers by improving their access to mandi(markets), which enhances agriculture productivity, ensures fair pricing, and improves the food supply chain. These factors directly contribute to achieving food security and sustainable agriculture, which are the core objectives of SDG-2.