DIARY DELIGHTS MANAGEMENT SYSTEM

A PROJECT REPORT
Submitted by

KARUPPUSAMY N
(REG. NO: 21BSR019)

KRISHNA VARATHAN.B
(REG. NO: 21BSR022)

MOHAN NIVASH.C
(REG. NO: 21BSR028)

In partial fulfillment of the requirements for the award of the degree of

BACHELOR OF SCIENCE

IN

SOFTWARE SYSTEMS

DEPARTMENT OF COMPUTER TECHNOLOGY -UG

KONGU ENGINEERING COLLEGE
(Autonomous)

PERUNDURAI ERODE – 638060

NOVEMBER-2023

DEPARTMENT OF COMPUTER TECHNOLOGY-UG

KONGU ENGINEERING COLLEGE

(Autonomous)

PERUNDURAI ERODE – 638060

NOVEMBER 2023

BONAFIDE CERTIFICATE

This is to certify that the project report titled **“DIARY DELIGHTS MANAGEMENT SYSTEM”** is the approved record of work done by **KARUPPUSAMY N (REG. NO: 21BR019) KRISHNA VARATHAN B (REG NO: 21BSR022) MOHAN NIVASH C (REG NO: 21BR028)** in partial fulfillment for the award of Degree of Bachelor of Science in (**SOFTWARE SYSTEMS)** of Anna University Chennai during the year 2023-2024.

SUPERVISOR **HEAD OF THE DEPARTMENT**

(Signature with seal)

Date:

Submitted for the end semester viva-voce examination held on_____.

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DEPARTMENT OF COMPUTER TECHNOLOGY-UG

KONGU ENGINEERING COLLEGE

(AUTONOMOUS)

PERUNDURAI ERODE-638060

NOVEMBER 2023

DECLARATION

We affirm that the project report titled "**DIARY DELIGHTS MANAGEMENT SYSTEM**" is being submitted in partial fulfillment of the requirements for the award of **B.Sc. degree in SOFTWARE SYSTEMS** is the original work carried out by us. It has not formed part of any other project report or dissertation based on which degree or award was conferred on an earlier occasion of any other candidate.

KARUPPUSAMY N (REG. NO: 21BSR019)

KRISHNA VARATHAN B (REG. NO: 21BSR022)

MOHAN NIVASH C (REG. NO: 21BSR028)

Date:

I certify that the declaration made above by the candidates is true to the best of my knowledge.

Name and Signature of the Supervisor with seal

ABSTRACT

In our day to day life we are using dairy products ,every time the customer go to the dairy shop and purchase dairy products this makes more for complicated wakeup early morning to go and consuming time.By creating online application is to enhance the process of procuring and delivering the fresh dairy products to customers.

This application aims to revolutionize the traditional product delivery by providing convenience, efficiency, Time balancing and fresh products to customer’s doorsteps.As per the requirements we have to create the Mobile Application using the Android Studio for Application Creation and we are using firebase for storing data.

The software features an easy-to-use interface that makes it possible for users to quickly peruse a carefully chosen range of milk alternatives. By choosing options for kind, amount, and delivery, users can customize their orders. By giving users access to real-time product availability updates, the application further improves user experience by empowering users to make well-informed decisions.

The software features an easy-to-use interface that makes it possible for users to quickly peruse a carefully chosen range of milk alternatives. By choosing options for kind, amount, and delivery, users can customize their orders. By giving users access to real-time product availability updates, the application further improves user experience by empowering users to make well-informed decisions.

ACKNOWLEDGEMENT

We express our sincere thanks \to our beloved Correspondent **THIRU.A.K.ILANGO B.Com., M.B.A., LLB,** and other philanthropic Trust members of the Kongu Vellalar Institute of Technology Trust for having provided with necessary resources to complete this project.

We are always grateful to our beloved visionary Principal **Dr.V.BALUSAMY B.E.,M.Tech., Ph.D.,** and thank him for his motivation and moral support.

We express our deep sense of gratitude and profound thanks to **Dr. S.KALAISELVI MCA.,M.E.,Ph.D,** Head of the Department, Computer Technology-UG for his invaluable commitment and guidance for this project.

We are in immense pleasure to express our hearty thanks to our Project Coordinator **Ms.P.GOKILABRINDHA M.Sc.,M.Phil** and our guide **Dr.S.KAVITHA BHARATHI., B.Sc ,M.Phil., Ph.D.,** for providing valuable guidance and constant support throughout our project. We also thank the teaching, and non-teaching staff members, fellow students, and our parents who stood with us to complete our project successfully.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

ABBREVIATION	EXPANSION
No-SQL	NOT ONLY SQL
UI	USER INTERFACE
HTTP	HYPERTEXT TRANSFER PROTOCOL
UX	USER EXPERIENCE
FCM	FIREBASE CLOUD MESSAGING
XML	EXTENSIBLE MARKUP LANGUAGE

CHAPTER 1
INTRODUCTION

1.1 PROBLEM DEFINTION

The Dairy Order Application appears to be a promising innovation that brings efficiency and convenience to the procurement and delivery of fresh dairy products. The application operates as a digital platform, eliminating the need for traditional, manual processes. This transition to a digital format suggests increased efficiency and a streamlined approach to dairy product procurement. The user-friendly interface makes it easy for customers to navigate the application. This is crucial for attracting a wide range of users, including those who may not be technology. Being accessible through a mobile application suggests convenience for users on the go. Mobile accessibility is especially important in today's fast-paced society where people rely heavily on their smartphones. The ability for users to create accounts indicates a personalized experience. Account creation may also facilitate features such as order tracking, personalized recommendations, and loyalty programs.

The application's offering of a variety of dairy products suggests a selection for consumers. This can cater to different preferences and dietary needs, attracting a broader customer base. The claim of placing orders with just a few clicks indicates a streamlined and efficient ordering process. This convenience is likely to appeal to busy individuals who value time-saving solutions. Serving as a bridge between consumers and local dairy farmers and suppliers .This can contribute to supporting local businesses, promoting product sustainability.

The goal of modifying the traditional milk delivery model suggests a departure from conventional methods. This could involve faster delivery times, fresher products, and reduced environmental impact. The application's core objectives include providing convenience, efficiency, and freshness. These are key factors that can significantly enhance the overall customer experience. The promise of delivering products to customers' doorsteps aligns with the current trend of doorstep services. This can save customers time and effort, contributing to the overall appeal of the application. The Dairy Order Application appears to address key challenges in the dairy product supply by digital technology to create a more efficient and user-friendly experience for both consumers and local suppliers.

1.2 OBJECTIVE OF PROJECT

- Procurement process
- Digital transformation
- Creating User friendly Interface
- Cost reduction
- Communicate and promote products
- Online Accessibiltiy

CHAPTER 2

SYSTEM ANALYSIS

2.1 EXISTING SYSTEM

The current system is manually operated. The current method has several flaws and as a result, considered is given to monitoring the product on hand . Due to this method they extend their brand name. Therefore a proposed system is required to address the shortcomings of the current system and is gets computerized.

2.1.1 Drawbacks of Existing System

The disadvantages of the existing system are:

- Time consumption is more for purchase.

.

- Lack of product availability and continuity of service.
- Stock maintained manually.
- Doesn't have the proper ordering of products.

2.2 PROPOSED SYSTEM

The suggested system aids in handling quick report generation and data processing. The current system's shortcomings, such as checking for open orders, are taken into account, and it displays stock information. The goal of the project is to make users and administrators more participatory and user-friendly. It gets around the challenges of selling dairy products in E-commerce platform. It is an effective forum for all users.

2.2.1 Advantages of Proposed System

The proposed system has following advantages

- ➢ User interface is designed efficiently.
- ➢ The data are maintained for long years in cloud.
- ➢ This project reduces the human work time and cost.

2.3 FEASIBILITY STUDY

Feasibility studies aim to objectively and rationally uncover the strength and weakness of the existing business or proposed venture, opportunities and threats as presented by the environment, the resources required to carry through and ultimately the prospects for success.

2.3.1 Economical Feasibility

This study is carried out to check the economic impact that the system will have on the organization. The designed application is found to be more economical than the current system. Instead of allocating more space is reduced and maintenance cost is reduced.

2.3.2 Operational Feasibility

To complete this assessment, research must be conducted to analyze and assess whether and how effectively the project will satisfy the needs of the company. The proposed system's accessing method addresses issues that cropped up in the old system. The company's present inventory management procedures can be included in this system. The project plan's compliance with the requirements determined throughout the system development process requirements analysis phase is also examined.

2.3.3 Technical Feasibility

The technical resources that the organization has access to are the main focus of this application. It aids organizations in determining whether the technical resources are adequate and whether the technical team has the skills necessary to turn concepts into functional systems. The current system seeks to fix the issues with the previous one. It also entails assessing the proposed system's technical requirements for hardware, software, and other areas.

2.4 SYSTEM SPECIFICATION

Frontend	Java (XML)
Backend	FIREBASE

2.4.1 HARDWARE REQUIREMENTS

This section gives the details and specification of the hardware on which the system is expected to work.

Processor	Intel(R) Core(TM) i3-7020U CPU
RAM	4.00 GB
Hard disk	1 TB
System type	64-bit operating system, x64-based processor

2.4.2. SOFTWARE REQUIREMENTS

This section gives the details of the software that are used for the development.

Operating System	Windows 10 and above
Environment	Android studio
Frontend	Java (XML)
Backend	FIREBASE

2.4.1 HARDWARE SPECIFICATION

Processor	Intel(R) Core(TM) i3-7020U CPU
RAM	4.00 GB
Hard disk	1 TB
System type	64-bit operating system, x64-based processor

2.4.2 SOFTWARE REQUIREMENT

Operating System	Windows 10 and above
Environment	Android studio

2.4.3 SOFTWARE DESCRIPTION

FRONT END

Android Studio is a popular integrated development environment (IDE) used for developing Android applications. It's primarily based on Java and Kotlin programming languages. When you're working on the frontend of an Android app in Android Studio using Java, you're essentially dealing with the user interface (UI) and user experience (UX) elements.

XML Layouts: The frontend in Android Studio is typically constructed using XML files. These files define the layout and structure of the app's user interface, including elements such as buttons, text fields, images, etc.

Activities and Fragments: Activities represent the screens with which users interact, while fragments are smaller components that can be used within activities. You'll create and manage these elements to build the user interface.

Views and ViewGroups: Views are UI elements like buttons, text fields, etc., while ViewGroups are containers that hold multiple views.Event Handling: In the frontend, you handle user interactions, like button clicks or text input, by writing Java code to respond to these events.

Resource Management: Android Studio facilitates the management of various resources, such as images, strings, and other assets used in the app's frontend.

UI Customization : You can customize the appearance of your app by applying different styles, themes, and layouts to create a unique user experience. Remember, Android Studio provides various tools and a powerful layout editor to help in designing and previewing the UI, making it easier to create a visually appealing frontend for your Android application.

BACKEND

Firebase:

Firebase is a comprehensive platform provided by Google that offers a variety of services to help developers build and manage apps. When it comes to the backend of an application, Firebase provides a range of tools and services to handle server-side functionalities, database management, authentication, storage, and more. Here's a breakdown:

Realtime Database and Firestore: Firebase offers both the Realtime Database and Firestore, which are NoSQL databases. These allow developers to store and sync data between users in real time. They are document-based databases where data is stored in JSON-like formats.

Authentication: Firebase provides authentication services that support various methods of user authentication like email/password, social logins (Google, acebook, etc.), phone number authentication, and more. This is crucial for managing user access and data security.

Cloud Functions: These allow developers to run backend code in response to events triggered by Firebase features and HTTPS requests. It enables serverless computing, where you can execute code without managing the server.

Cloud Storage: Firebase Storage allows you to store and serve user-generated content, such as images, audio, and video files. It integrates with Firebase Authentication to control access to these files.

Cloud Messaging (FCM): Firebase Cloud Messaging enables you to send notifications and messages to users across various platforms (Android, iOS, web) from a single backend service.

Hosting and Cloud Firestore: Firebase Hosting offers a secure and easy way to deploy web apps and static content. Cloud Firestore is a more scalable NoSQL database that works well for larger applications

CHAPTER 3

SYSTEM DESIGN

3.1 OVERVIEW OF THE PROJECT

The main aim of this mobile application is to manage all the information of the apartment with in hand and anywhere. For that purpose, the apartment details are maintained in real-time centralized database. It helps the apartment owner to keep track of Information regarding maintenance, sales, guest, etc. It helps the security to maintain all logs, those who are present and time in, time out and apartment secretary will send group notification budget and announcement.

3.2 MODULE DESCRIPTION

The project "DIARY DELIGHTS MANAGEMENT SYSTEM" contains the following modules,

- User access Module,
- Admin Access Module

USER ACCESS MODULE

Users can create account, log in, view the products available, add the products to cart and place order from this mobile application.

ADMIN ACCESS MODULE

In this module, Admin has the separate login where they have to provide only the registered password to navigate to dashboard. Admin can manage order details, stock management ,user login operations on the product, and can view the orders placed by the customer.

3.3.SYSTEM FLOW DIAGRAM

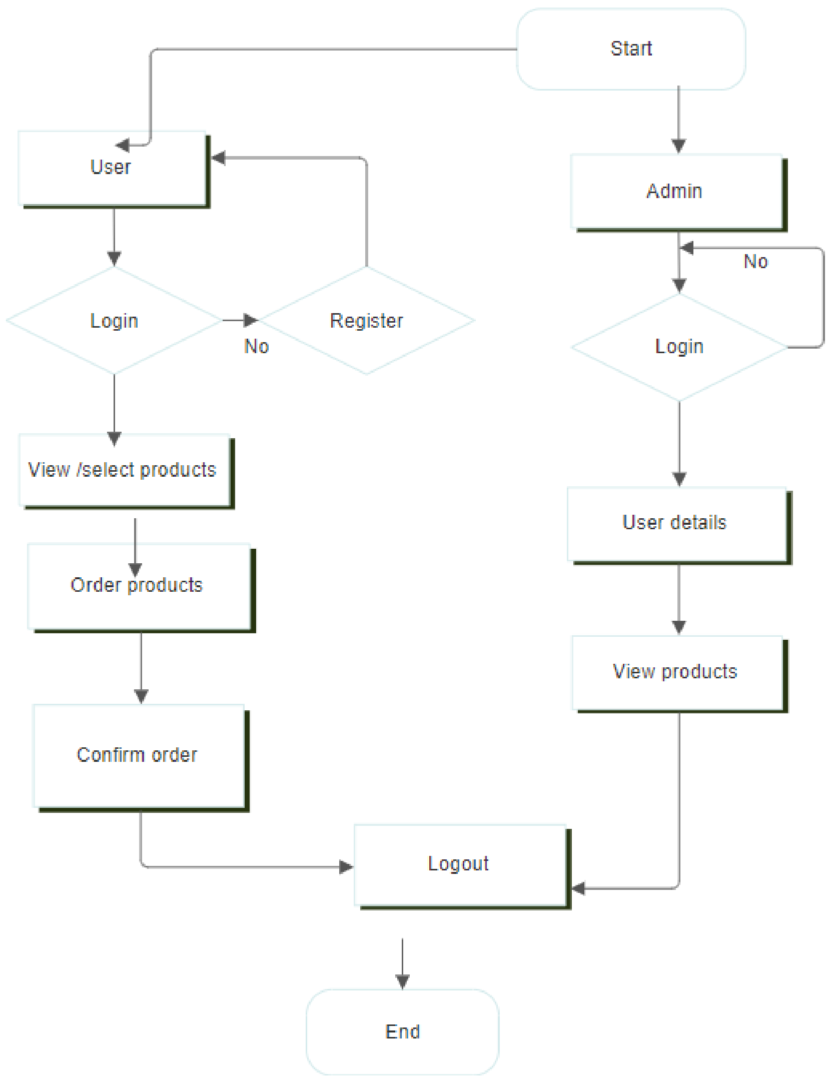


Figure 3.3 User System Flow Diagram

3.4 DATA FLOW DIAGRAM

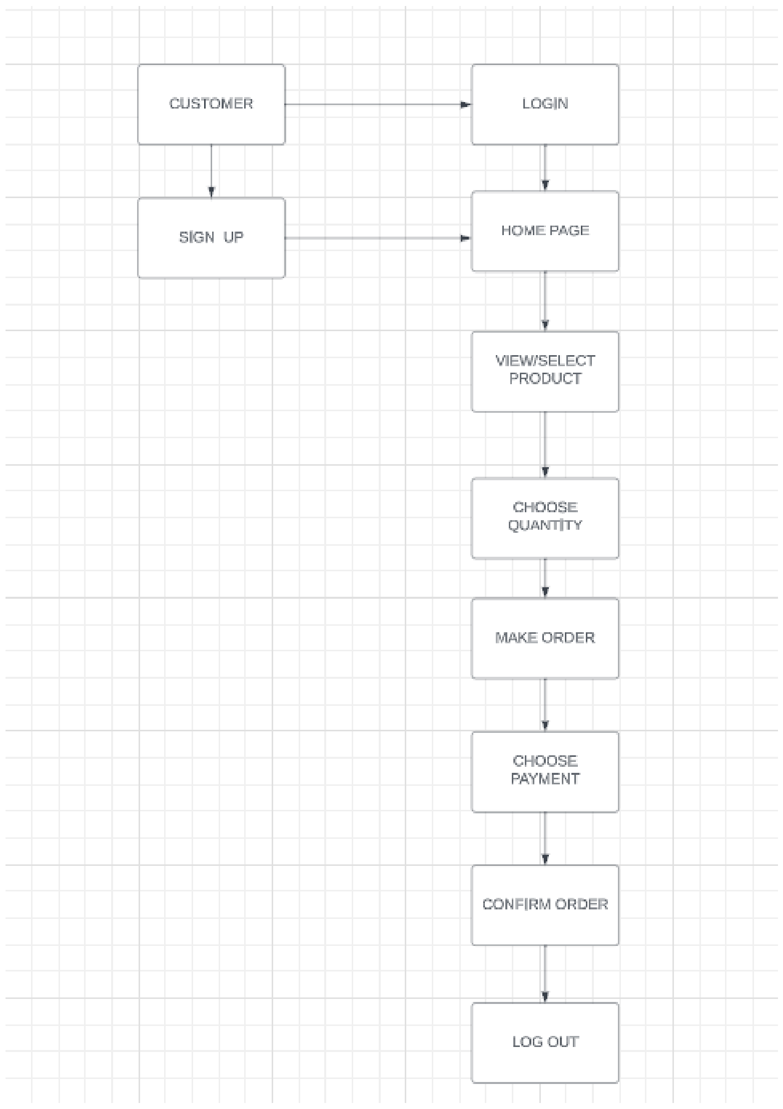


Figure 3.4 Admin Data flow diagram

3.5 ACTVITY DIAGRAM

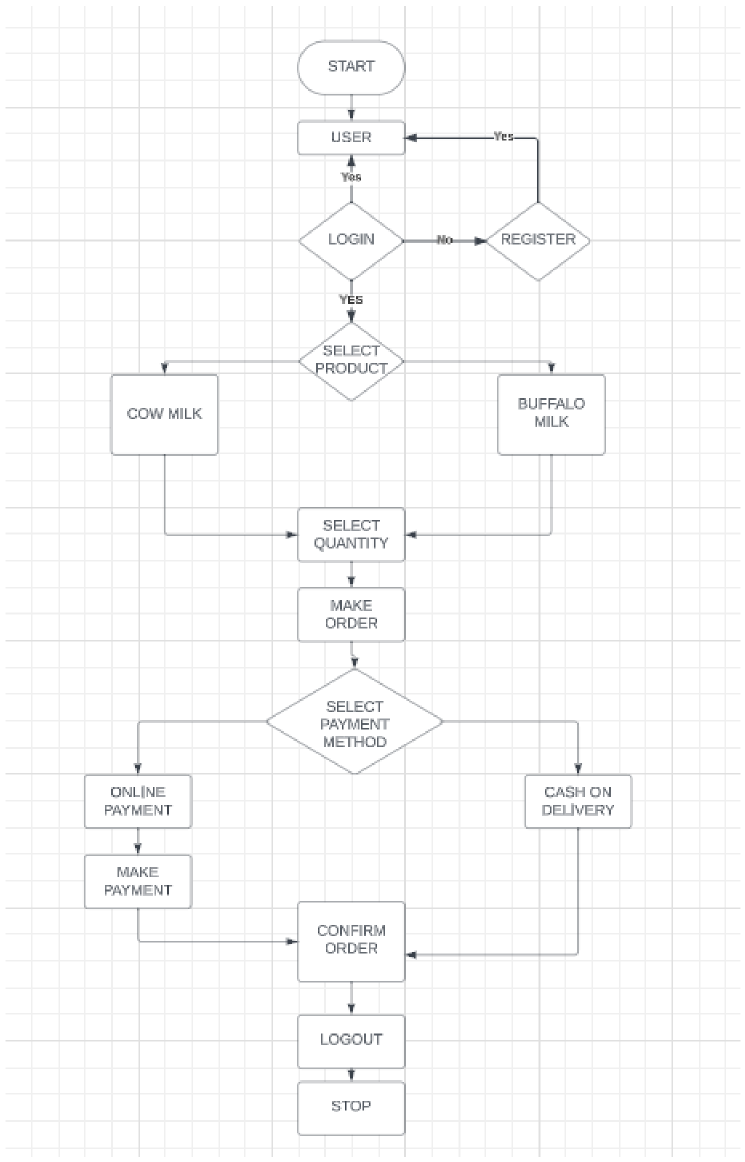


Figure 3.5 User Activity diagram

3.6 USE DIAGRAM

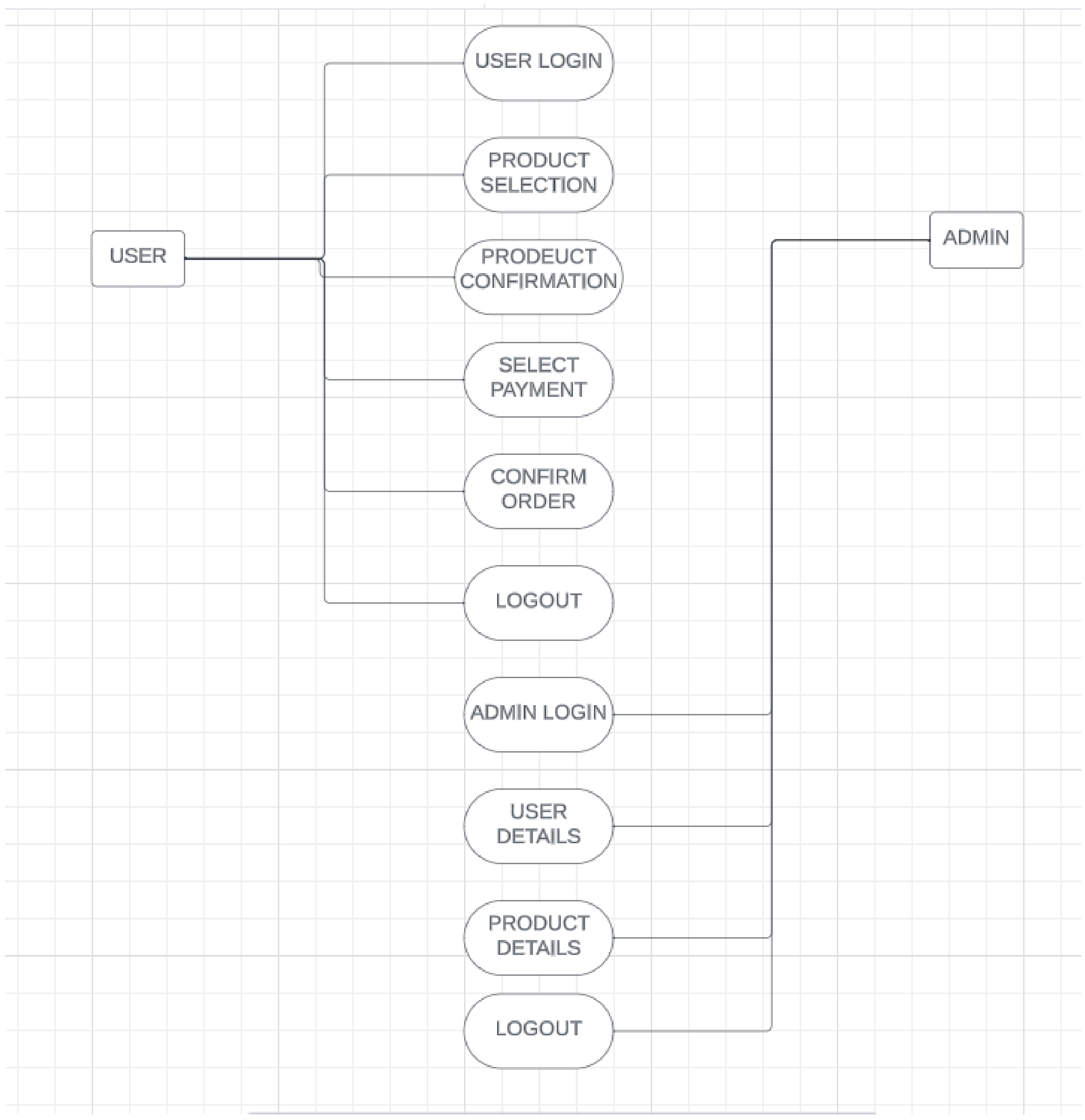
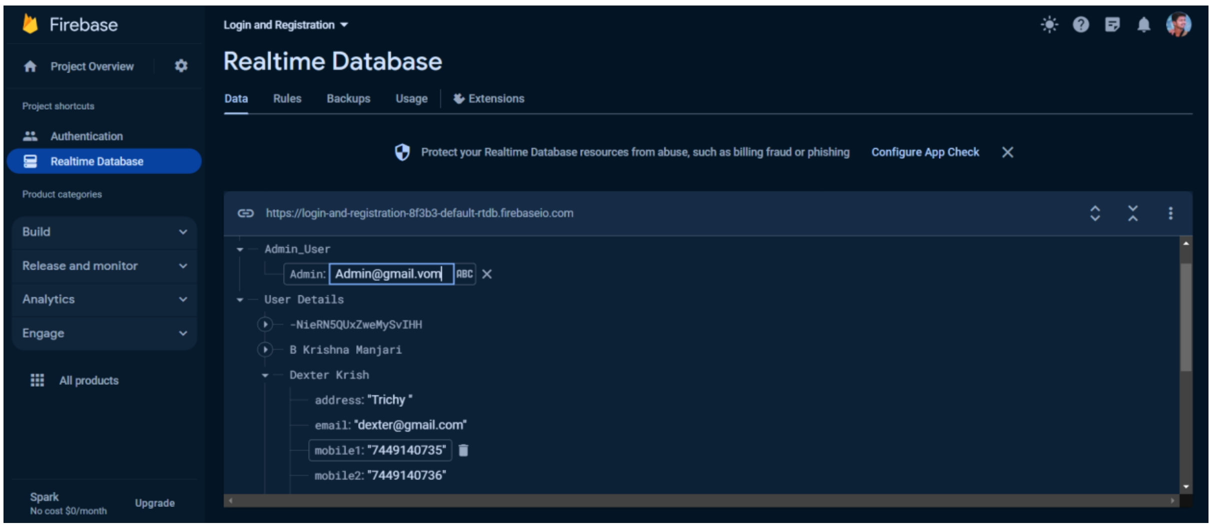


Figure 3.6 Admin /User Use case diagram

3.7 DATABASE DESIGN



```
https://login-and-registration-8f3b3-default-rtdb.firebaseio.com
Admin_User
    Admin: Admin@gmail.vom
User Details
    -NieRN5QUxZweMySvIHH
    B Krishna Manjari
    Dexter Krish
        address: "Trichy "
        email: "dexter@gmail.com"
        mobile1: "7449140735"
        mobile2: "7449140736"
```

Figure 3.7 Firebase Database diagram

3.8 INPUT DESIGN

Input design is the process of getting input from user and processing it. A large number of problems with a system can usually be tracked backs to fault input design and method. Every moment of input design should be analyzed and designed with utmost care.

The design of the input should be made the input as the over to the numerous networks in the reliable area that should be passed as the installation in the remote network. It has the following constraints in the input database.

- ✓ All the files from the disk should be acquired by data.
- ✓ It is suitable to more available data clearance and made available.
- ✓ The menu of design should be understandable and it is in the right format.

In the project, the forms are designed with easy to use options such as selecting the master records through dropdown list in transaction forms. The coding is being done such that proper validations are made to get the perfect input. No error inputs are accepted. The end users need not to give the id themselves.The details are stored with proper validation in the database. Some of the details are email id, password, time in, time out, name, proof, apartment number, etc.,

3.7 OUTPUT DESIGN

The output design is generally referring to the result and information that are generated by the system for many end-users; it should be understandable with the enhanced format. Output design is the process, how the output display. In DAIRY DELIGHTS MANAGEMENT SYSTEM all the data are formatted and displayed Products details,order,payment.

CHAPTER 4

SYSTEM TESTING

System testing is the stage of implementation, which is aimed to ensuring that the system works accurately and efficiently. The main objective of testing is to uncover errors from the system. Testing is done for each module. After testing all the modules, the modules are integrated and testing of the final system is done with the test data, specially designed to show that the system will operate successfully in all its aspects and condition.Testing is the process of exercising the software product in pre-defined ways to check if the behavior is the same as expected behavior. This project has undergone the following testing procedures to ensure it correctness.

Unit Testing

Integration Testing

4.1 UNIT TESTING

Unit Testing has been under taken when a module has been coded and successfully reviewed. Unit testing sometimes called as Program Testing. This enables to detect errors in coding and logic that are contained with the module alone. In the user login, if the user does not provide an email id or a password, an error message is shown. “Enter the email id or password”.

Test Case1:

Module	: Login
Input	: Email and Password.
Event	: Button click.
Expected output	: Logged in successfully.
Test 1:	
Input	: krish@gmail.com and krish321
Output	: Logged in successfully
Event	: Button click.
Analysis	: Email Id and password has been verified.

Test 2:

Input	: krish@gmail.com
Output	: Enter Password
Event	: Button click.
Analysis	: Password field Checked and Error Shown.

4.2 INTEGRATION TESTING

Integration testing is a systematic technique for constructing the program structure. While at the same time conducting tests to uncover errors associated with interfacing. The objective is to take unit tested modules and build a program structure that has been dictated by design. In this integration testing is done using the main module and based on the type of integration testing the subordinate stubs are replaced one at a time with actual modules.

Test Case-1:

Module	: Login
Input	: Email Id and password
Expected output	: Redirect to home page

Test-1:

Input	: krish@gmail.com and krish321
Output	: Redirected to homepage.
Analysis	:Email Id and password has been verified.

Test-2:

Input	: xyz@gmail.com and xyz
Output	: Enter the Register Email Id.
Analysis	: Email Id and password are checked and error is shown.

\

CHAPTER 5

SYSTEM IMPLEMENTATION

5.1 IMPLEMENTATION SUPPORT

When the initial design was done for the system, the client was consulted for the acceptance of the design so that further proceedings of the system development can be carried on. After the development of the system a demonstration was given to them about the working of the system. The aim of the system illustration was to identify any malfunction of the system.

After the management of the system was approved the system implemented in the concern, initially the system was run parallel with existing manual system. The system has been tested with live data and has proved to be error free and user friendly. Implementation is the process of converting a new or revised system design into an operational one when the initial design was done by the system a demonstration was given to the end user about the working system. This process is uses to verify and identify any logical working of the system by feeding various combinations of test data.

After the approval of the system by both end user and management the system was implemented. A product software implementation method is a blue print to get uses and/or organizations running with a specific software product. The method is a set of rules and views to cope with the most common issues that occur when implementing a software product business alignment from the organizational view and acceptance from the human view.

The implementation of product software, as the final link in the deployment chain of software production, is in a financial perspective of a major issue. The Implementation methodology includes four phases discovery, system development and user acceptance testing and production rollout. It's easy to be overwhelmed by slick marketing presentations, particularly when the sales force is talking about things that most people Don’t completely understand.

System implementation is made up of many activities. The major activities are as follows.

- On the mobile phone, obtain the space for deploying the app
- Install the app for the "DIARY DELIGHTS MANAGEMENT SYSTEM ".
- The system is successfully installed and is ready for usage.

CODING

Coding is the process of whereby the physical design specifications created by the analysis team turned into working computer code by the programming team.

TESTING

Once the coding process is begin and proceed in parallel, as each program module can be tested separately and a combination of modules can be tested step by step.

INSTALLATION

Installation is the process during which the current system is replaced by the new system. This includes conversion of existing data, software, and documentation and work procedures to those consistent with the new system.

DOCUMENTATION

It is result from the installation process, user guides provide the information of the system and its flow. The mobile application is tested well and end user satisfaction is found to be more. The mobile application is designed; everyone can easily access. It will reduce the time and space.A mobile application with collection of attractive forms is the best way to attract the users to use the mobile application frequently.

TRAINING AND SUPPORT

Training plan is strategy for training users, so they quickly learn about the new system. The development of the training plan probably began earlier in the project. The best-suited application package to develop the system is Android Studio with FIREBASE.

CHAPTER 6

CONCLUSION AND FUTURE ENHANCEMENTS

6.1 CONCLUSION

The project helps the administrator to maintain all orders and details. It reduces manual work. Since the application is designed as a Mobile Application, any user can be using the website. The application is tested well and end-user satisfaction is found to be more. The application is designed such that minimum computer knowledge is required for the end user.

6.2 FUTURE ENHANCEMENT

The application becomes useful if the below enhancement is made in the future.

- The consolidated complaints details can be get from the customer.
- Offers will be sent to the customer's mail.
- Report generation will be enhanced.

The application is developed such that above said enhancement can be integrated with current modules

.

APPENDIX – 1

CODING

Login page

```
package com.example.loginandregistration;

import static androidx.constraintlayout.helper.widget.MotionEffect.TAG;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentManager;
import androidx.fragment.app.FragmentTransaction;
import android.content.Intent;
import android.graphics.drawable.ColorDrawable;
import android.os.Bundle;
import android.text.InputType;
import android.text.TextUtils;
import android.util.Log;
import android.util.Patterns;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.ProgressBar;
import android.widget.TextView;
import android.widget.Toast;

import com.example.loginandregistration.MainActivity;
import com.example.loginandregistration.R;
import com.example.loginandregistration.databinding.ActivityLoginBinding;
import com.example.loginandregistration.databinding.ActivityMainBinding;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.android.material.textfield.TextInputEditText;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
public class Login extends AppCompatActivity {
    View dialogview;
    private ActivityLoginBinding binding;
    TextInputEditText editTextEmial,editTextpassword,emailText;
    Button login,admin_btn,navigation; TextView
Register_Text,Forgotpassword;
    FirebaseAuth mAuth; ImageView passwordIcon;
    ProgressBar progressBar;  FirebaseDatabase db;
    DatabaseReference reference; String useremail="";

    @Override
```

```
  protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivityLoginBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());
        mAuth= FirebaseAuth.getInstance();
        editTextEmial=findViewById(R.id.email);
        editTextpassword=findViewById(R.id.password);
        login=findViewById(R.id.btn_login);
        admin_btn=findViewById(R.id.btn_admin);
        progressBar=findViewById(R.id.progressbar);
        Forgotpassword=findViewById(R.id.forgotpass);
        navigation=findViewById(R.id.btn_bottom_nav);
        Forgotpassword.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                AlertDialog.Builder builder = new
AlertDialog.Builder(Login.this);
                View dialogView =
getLayoutInflater().inflate(R.layout.dialog_forgot, null);
                EditText emailBox =
dialogView.findViewById(R.id.forgot_email);
                builder.setView(dialogView);
                AlertDialog dialog = builder.create();

dialogView.findViewById(R.id.btnReset).setOnClickListener(new
View.OnClickListener() {
                    @Override
                    public void onClick(View view) {
                        String userEmail = emailBox.getText().toString();
                        if (TextUtils.isEmpty(userEmail) &&
!Patterns.EMAIL_ADDRESS.matcher(userEmail).matches()){
                            Toast.makeText(Login.this, "Enter your
registered email id", Toast.LENGTH_SHORT).show();
                            return;
                        }

mAuth.sendPasswordResetEmail(userEmail).addOnCompleteListener(new
OnCompleteListener<Void>() {
                            @Override
                            public void onComplete(@NonNull Task<Void>
task) {
                                if (task.isSuccessful()){
                                    Toast.makeText(Login.this, "Check your
email", Toast.LENGTH_SHORT).show();
                                    dialog.dismiss();
                                } else {
                                    Toast.makeText(Login.this, "Unable to
send, failed", Toast.LENGTH_SHORT).show();
                                }
                            }                         });
                    }                 });

dialogView.findViewById(R.id.btnCancel).setOnClickListener(new
View.OnClickListener() {
                    @Override
                    public void onClick(View view) {
```

```
  dialog.dismiss();

  }
                });
                if (dialog.getWindow() != null){
                    dialog.getWindow().setBackgroundDrawable(new
ColorDrawable(0));
                }
                dialog.show();
            }       });

        binding.textRegister.setOnClickListener(new View.OnClickListener()
{
            @Override
            public void onClick(View view) {
                Intent in=new Intent(getApplicationContext(),
Registration.class);
                startActivity(in);
                finish();
            }       });

        admin_btn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent in=new Intent(getApplicationContext(),
Admin_Page.class);
                startActivity(in);
                finish();
            }       });
        navigation.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent in=new Intent(getApplicationContext(),
Bottom_Navigation.class);
                startActivity(in);
                finish();
            }       });

        login.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                progressBar.setVisibility(View.VISIBLE);
                String email,password;
                email=String.valueOf(editTextEmial.getText());
                password=String.valueOf(editTextpassword.getText());

                if(TextUtils.isEmpty(email))
                {

                    Toast.makeText(getApplicationContext(),"Enter Mail
id!",Toast.LENGTH_LONG).show();
                    progressBar.setVisibility(View.GONE);
                    return;
                }


    else if(TextUtils.isEmpty(password))


  {
```

```
                    Toast.makeText(getApplicationContext(),"Enter
Password!",Toast.LENGTH_LONG).show();
                    progressBar.setVisibility(View.GONE);
                    return;
                }
                if(email.equals("admin@gmail.com") &&
password.equals("Admin321"))
                {
                    Intent in=new Intent(getApplicationContext(),
Admin_Page.class);
                    startActivity(in);
                    finish();
                }
                else {
                    db = FirebaseDatabase.getInstance();
                    reference = db.getReference("User Details");
                    mAuth.signInWithEmailAndPassword(email, password)
                            .addOnCompleteListener(new
OnCompleteListener<AuthResult>() {
                                @Override
                                public void onComplete(@NonNull
Task<AuthResult> task) {
                                    if (task.isSuccessful()) {
                                        Toast.makeText(Login.this, "Login
Successfully",
                                                Toast.LENGTH_SHORT).show();
                                        Intent in = new
Intent(getApplicationContext(), Bottom_Navigation.class);
                                        in.putExtra("Email",email);
                                        startActivity(in);
                                        finish();
                                    } else {

progressBar.setVisibility(View.GONE);
                                        Toast.makeText(Login.this,
"Authentication failed.",

Toast.LENGTH_SHORT).show();}
                                }
                            });
                }
            }
        });
    }
```

Registration page

```
      public class Registration extends AppCompatActivity {
          //ActivityMainBinding binding;
          //String
      user_name,user_mobile1,user_mobile2,user_email,user_addres,user_passw
      ord,user_conpassword;
          FirebaseDatabase db;
          DatabaseReference reference;



editTextEmail,editTextmobile,editTextmobile2,editTextname,editTextpassword,
editTextaddress,editTextconpassword;
    String name,email,address,mobile1,mobile2,password,conpassword,uid;
    Button register;
    TextView Login_Text;
    FirebaseAuth mAuth;
    ProgressBar progressBar;

    public void onStart() {
        super.onStart();
        FirebaseUser currentUser = mAuth.getCurrentUser();
        if(currentUser != null)
        {
            Intent in=new Intent(getApplicationContext(),
MainActivity.class);
            startActivity(in);
            finish();
        }
    }
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_registration);
        //Firebase Connectivity
        mAuth= FirebaseAuth.getInstance();

        editTextname=findViewById(R.id.res_name);
        editTextmobile=findViewById(R.id.res_mobile);
        editTextmobile2=findViewById(R.id.res_altermobile);
        editTextaddress=findViewById(R.id.res_address);
        editTextEmail=findViewById(R.id.res_email);
        editTextpassword=findViewById(R.id.res_password);
        editTextconpassword=findViewById(R.id.res_conpassword);
        register=findViewById(R.id.btn_register);
        progressBar=findViewById(R.id.progressbar);
        Login_Text=findViewById(R.id.text_login);


        Login_Text.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent in=new Intent(getApplicationContext(), Login.class);
                startActivity(in);
                finish();
            }
        });

        register.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                progressBar.setVisibility(View.VISIBLE);
                name=String.valueOf(editTextname.getText());
                mobile1=String.valueOf(editTextmobile.getText());
                mobile2=String.valueOf(editTextmobile2.getText());
                address=String.valueOf(editTextaddress.getText());
                password=String.valueOf(editTextpassword.getText());
                email=String.valueOf(editTextEmail.getText());
                conpassword=String.valueOf(editTextconpassword.getText());


    if(!name.isEmpty() && !email.isEmpty() && !mobile1.isEmpty() &&
!mobile2.isEmpty()
                   && !address.isEmpty() && !password.isEmpty() &&
!conpassword.isEmpty())
                {
                    //Toast.makeText(Registration.this,"All Fields Should
be Filled!",Toast.LENGTH_LONG).show();
                    progressBar.setVisibility(View.GONE);
                    if(conpassword.equals(password))
                    {
                        if (!mobile1.equals(mobile2)) {
                            progressBar.setVisibility(View.VISIBLE);

                            //Assigning the values to the user class
                            users user=new
users(name,mobile1,mobile2,address,email,password);
                            db = FirebaseDatabase.getInstance();
                            reference = db.getReference("User Details");
                            //String userKey = reference.push().getKey();
                            //uid =
(FirebaseAuth.getInstance().getCurrentUser().getUid());

reference.child(name).setValue(user).addOnCompleteListener(new
OnCompleteListener<Void>() {
                                @Override
                                public void onComplete(@NonNull Task<Void>
task) {
                                    name="";
                                    email="";
                                    address="";
                                    mobile1="";
                                    mobile2="";
                                    password="";

//Toast.makeText(getApplicationContext(),"Successfully
Updated!",Toast.LENGTH_LONG).show();
                                }
                            });

                            //Validate the Email and Store it in the
Firebase
                            mAuth.createUserWithEmailAndPassword(email,
password)
                                    .addOnCompleteListener(new
OnCompleteListener<AuthResult>() {
                                        @Override
                                        public void onComplete(@NonNull
Task<AuthResult> task) {

                                            if (task.isSuccessful()) {

progressBar.setVisibility(View.GONE);

Toast.makeText(Registration.this, "Account Created",

Toast.LENGTH_SHORT).show();
                                                Intent in = new
Intent(getApplicationContext(), Login.class);
                                                startActivity(in);
                                                finish();


      } else {

progressBar.setVisibility(View.GONE);

Toast.makeText(Registration.this, "Authentication failed!",
Toast.LENGTH_SHORT).show();
                                            }
                                        }
                                    });
                        }
```

Product page

```
public class ProductFragment extends Fragment {
    private ListView orderDetailsListView;
    private List<String> orderDetailsList;
    private ArrayAdapter<String> orderDetailsAdapter;
    private static final int COW_MILK_PRICE = 46;
    private static final int BUF_MILK_PRICE = 60;

    public ProductFragment() {

    }

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
    }

    private TextView welcomeTextView;
    Bundle bundle;
    private Button orderButton;
    private FragmentProductBinding binding;
    int Cow_Milk = 0, Buf_Milk = 0;

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        //View view = inflater.inflate(R.layout.fragment_product,
container, false);
        binding = FragmentProductBinding.inflate(inflater, container,
false);
        //View rootView = binding.getRoot();
        setupCheckBoxListener();
        //AccessEditText();
        orderDetailsList = new ArrayList<>();
        orderDetailsAdapter = new ArrayAdapter<>(requireContext(),
android.R.layout.simple_list_item_1, orderDetailsList);

        // Set adapter to the ListView
        binding.orderDetailsListView.setAdapter(orderDetailsAdapter);
        bundle = new Bundle();

        // Set up checkbox and edit text listeners
        setupCheckBoxListener();
        setupEditTextListeners();

        // Set up button click listener for submitting the order


        binding.submitButton.setOnClickListener(v -> submitOrder());

 return binding.getRoot();
    }
    private void setupEditTextListeners() {
        binding.cowEditText.addTextChangedListener(new TextWatcher() {
            @Override
            public void beforeTextChanged(CharSequence charSequence, int i,
int i1, int i2) {
                // Not needed
            }

            @Override
            public void onTextChanged(CharSequence charSequence, int i, int
i1, int i2) {
                //orderDetailsList.remove("Half Cow Milk: " +
getQuantity(binding.cowEditText));
                //orderDetailsList.add("Half Cow Milk: " +
getQuantity(binding.cowEditText));
                //orderDetailsAdapter.notifyDataSetChanged();
                updateCowMilkText();
            }

            @Override
            public void afterTextChanged(Editable editable) {
                // Not needed
            }
        });

        // Similar listeners can be added for other EditText fields
        binding.bufEditText.addTextChangedListener(new TextWatcher() {
            @Override
            public void beforeTextChanged(CharSequence charSequence, int i,
int i1, int i2) {
                // Not needed
            }

            @Override
            public void onTextChanged(CharSequence charSequence, int i, int
i1, int i2) {
                updateBufMilkText();
            }

            @Override
            public void afterTextChanged(Editable editable) {
                // Not needed
            }
        });
    }


    private void submitOrder() {
        if((!binding.halfCowMilk.isChecked() &&
!binding.OneCowMilk.isChecked() &&
                !binding.HalfBufMilk.isChecked() &&
!binding.OneBufMilk.isChecked()))
        {

 showToast("Please select at least one option");
            return; // Do not proceed further
        }
        OrderFragment orderFragment = new OrderFragment();
        orderFragment.setArguments(bundle);

        // Navigate to the next fragment
        requireActivity().getSupportFragmentManager().beginTransaction()
                .replace(R.id.frame_container, orderFragment)
                .addToBackStack(null)
                .commit();
    }
    private void updateSubmitButtonVisibility() {
        boolean isAnyCheckboxChecked = binding.halfCowMilk.isChecked() ||
                binding.OneCowMilk.isChecked() ||
                binding.HalfBufMilk.isChecked() ||
                binding.OneBufMilk.isChecked();

        // Set the visibility of the button based on checkbox states
        binding.submitButton.setVisibility(isAnyCheckboxChecked ?
View.VISIBLE : View.GONE);
    }


    private void showToast(String message) {
        Toast.makeText(requireContext(), message,
Toast.LENGTH_SHORT).show();
    }

    // ... Existing methods

    private void setupCheckBoxListener() {
        binding.halfCowMilk.setOnCheckedChangeListener((buttonView,
isChecked) -> {
            updateCowMilkText();
            updateSubmitButtonVisibility();
        });

        binding.OneCowMilk.setOnCheckedChangeListener((buttonView,
isChecked) -> {
            updateCowMilkText();
            updateSubmitButtonVisibility();
        });
        binding.HalfBufMilk.setOnCheckedChangeListener((buttonView,
isChecked) -> {
            updateBufMilkText();
            updateSubmitButtonVisibility();
        });

        binding.OneBufMilk.setOnCheckedChangeListener((buttonView,
isChecked) -> {
            updateBufMilkText();
            updateSubmitButtonVisibility();
        });
    }


    private void updateCowMilkText() {
        int totalCowMilkPrice = 0;

 if (binding.halfCowMilk.isChecked()) {

int oneCowMilkQuantity=23;
            //Toast.makeText(requireContext(),
String.valueOf(oneCowMilkQuantity), Toast.LENGTH_SHORT).show();
            int halfCowMilkQuantity = 23;
            totalCowMilkPrice += halfCowMilkQuantity;
            updateMilkPriceTextView(binding.CowMilkPrice, "1/2L Cow Milk",
totalCowMilkPrice);

        }
        else {
         binding.CowMilkPrice.setText("  ");
        }

        if (binding.OneCowMilk.isChecked()) {
            int oneCowMilkQuantity=46;
            //Toast.makeText(requireContext(),
String.valueOf(oneCowMilkQuantity), Toast.LENGTH_SHORT).show();
            totalCowMilkPrice+=oneCowMilkQuantity;
                String size = (totalCowMilkPrice > 46) ? "1.5L" : "1L";
                updateMilkPriceTextView(binding.CowMilkPrice, size + " Cow
Milk", totalCowMilkPrice);
                //String str = String.valueOf(totalCowMilkPrice);
                //binding.CowMilkPrice.setText("1L Cow Milk: "+str);
        }
    }
    private void updateMilkPriceTextView(TextView textView, String label,
int price) {
        String formattedText = label + ": " + price;
        textView.setText(formattedText);
        bundle.putString("formattedText", formattedText);
        bundle.putString("cowprice", String.valueOf(price));
    }
    private void updateMilkPriceTextView2(TextView textView, String label,
int price) {
        String formattedText2 = label + ": " + price;
        textView.setText(formattedText2);
        bundle.putString("formattedText2", formattedText2);
        bundle.putString("bufprice",String.valueOf(price));
    }

    private void updateBufMilkText () {
        int totalCowMilkPrice=0;
            if (binding.HalfBufMilk.isChecked()) {
                //Toast.makeText(requireContext(),
String.valueOf(oneCowMilkQuantity), Toast.LENGTH_SHORT).show();
                int halfCowMilkQuantity = 30;
                totalCowMilkPrice += halfCowMilkQuantity;
                updateMilkPriceTextView2(binding.BufMilkPrice, "1/2L
Buffalo Milk ", totalCowMilkPrice);

            }
            else {
                binding.BufMilkPrice.setText("  ");
            }

            if (binding.OneBufMilk.isChecked()) {
                //int oneCowMilkQuantity = getQuantity(binding.cowEditText)


* 2; // Assuming 1 liter = 1000 ml

  int oneCowMilkQuantity = 60;
                //Toast.makeText(requireContext(),
String.valueOf(oneCowMilkQuantity), Toast.LENGTH_SHORT).show();
                totalCowMilkPrice += oneCowMilkQuantity;
                String size = (totalCowMilkPrice > 60) ? "1.5L" : "1L";
                updateMilkPriceTextView2(binding.BufMilkPrice, size + "
Buffalo Milk ", totalCowMilkPrice);
            }
        }

}
```

Admin page

```
package com.example.loginandregistration;

import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.util.StateSet;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;

import androidx.activity.result.contract.ActivityResultContracts;
import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentManager;
import androidx.fragment.app.FragmentTransaction;

import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

import java.util.ArrayList;
import java.util.List;

public class Admin_Page extends AppCompatActivity {
    Button logout_btn, btn_addadmin, btn_Display, btn_stock, btn_order,
createCollectionButton;
    ListView listView;
    private ListView userDetailsListView;
    private List<users> userList;
    String name, mobile1, mobile2, address, email, password;
    DatabaseReference databaseReference;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

setContentView(R.layout.activity_admin_page);
        //DatabaseReference databaseReference =
FirebaseDatabase.getInstance().getReference();

//databaseReference.child("Admin_User").child("Admin").setValue("Hello,
Firebase!");
```

```
        databaseReference = FirebaseDatabase.getInstance().getReference();

        createCollectionButton = findViewById(R.id.createCollectionButton);
        createCollectionButton.setOnClickListener(v ->
createNewCollection());

        logout_btn = findViewById(R.id.logout_btn);
        btn_addadmin = findViewById(R.id.add_new_admin_btn);
        btn_Display = findViewById(R.id.view_user_details_btn);
        btn_stock = findViewById(R.id.stock_management_btn);
        btn_order = findViewById(R.id.order_details_btn);
        //listView = findViewById(R.id.user_details);

        logout_btn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent in = new Intent(getApplicationContext(),
Login.class);
                startActivity(in);
                finish();
            }
        });
        btn_Display.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent in = new Intent(getApplicationContext(),
UserList.class);
                startActivity(in);
                finish();
            }
        });
        btn_addadmin.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent in = new Intent(getApplicationContext(),
AddNewAdmin.class);
                startActivity(in);
                finish();
            }
        });

    }
 private void createNewCollection() {
        // Replace "new_collection" with your desired collection name
        databaseReference.child("Admin_User").setValue("Initial Data")
                .addOnSuccessListener(aVoid -> {
                    // Collection created successfully
                    Toast.makeText(Admin_Page.this, "New collection
created!", Toast.LENGTH_SHORT).show();
                })
                .addOnFailureListener(e -> {
                    // Handle any errors
                    Toast.makeText(Admin_Page.this, "Failed to create
collection: " + e.getMessage(), Toast.LENGTH_SHORT).show();
                });
    }
}
```

APPENDIX 2

SCREEN SHOTS

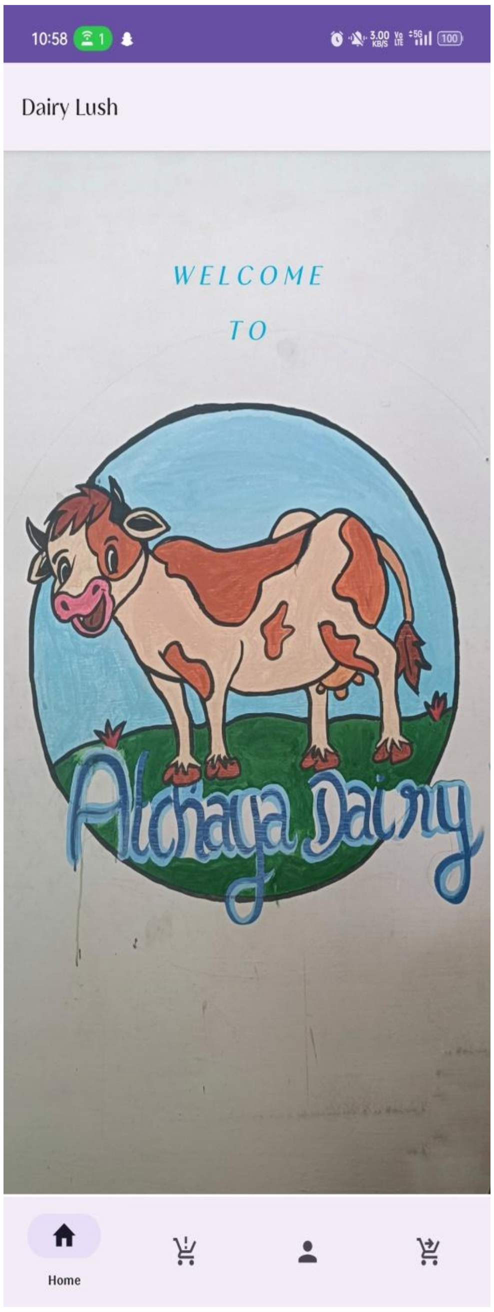


Figure.A.2.1 Home Page

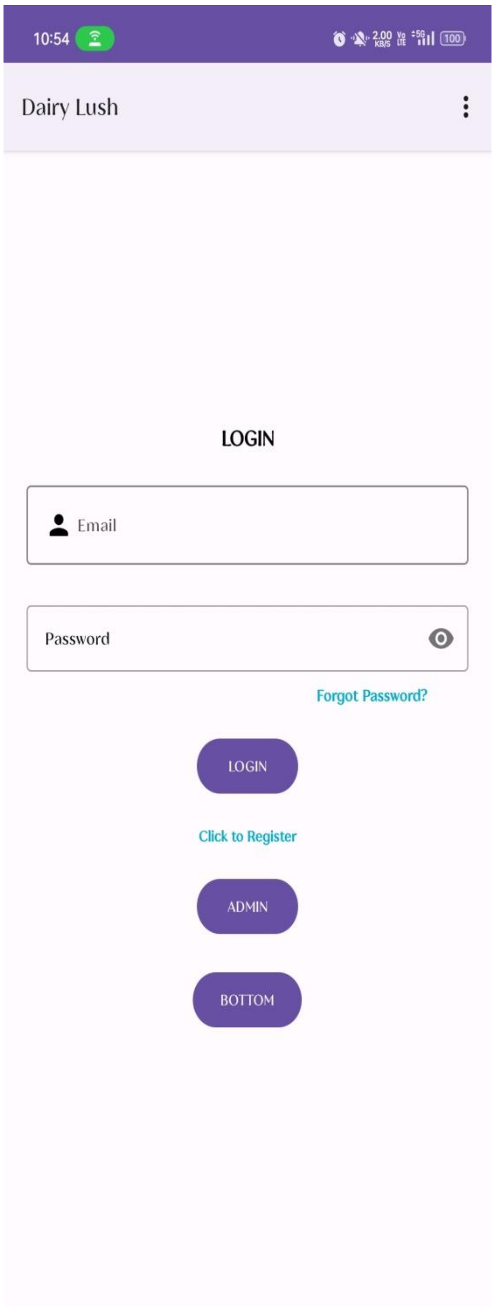


Figure A.2.2 Login page

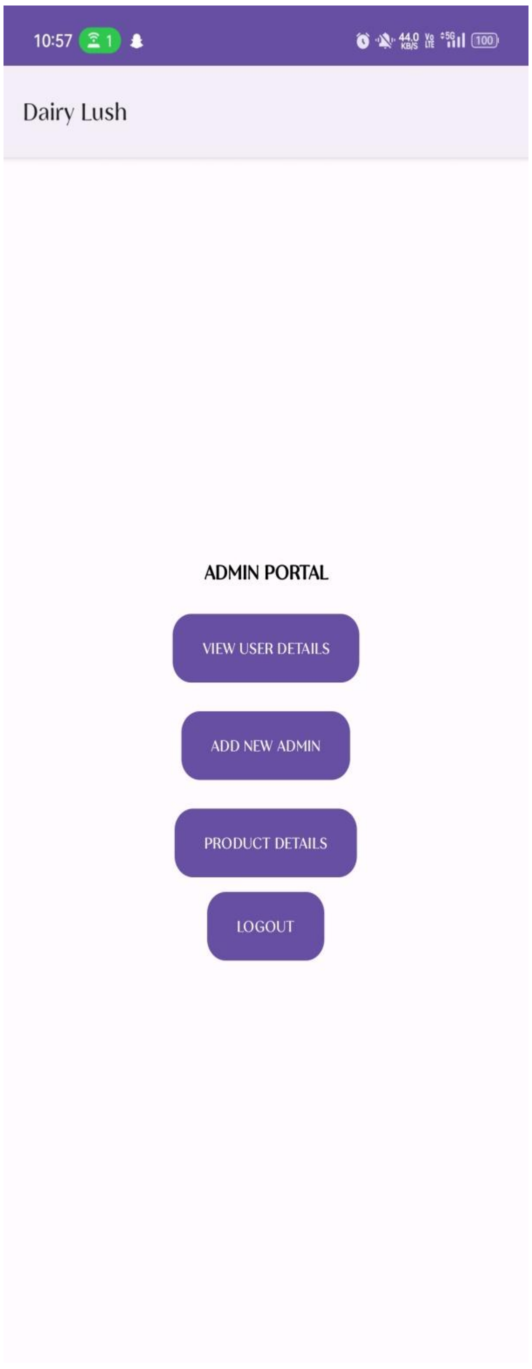


Figure A.2.3 Admin page

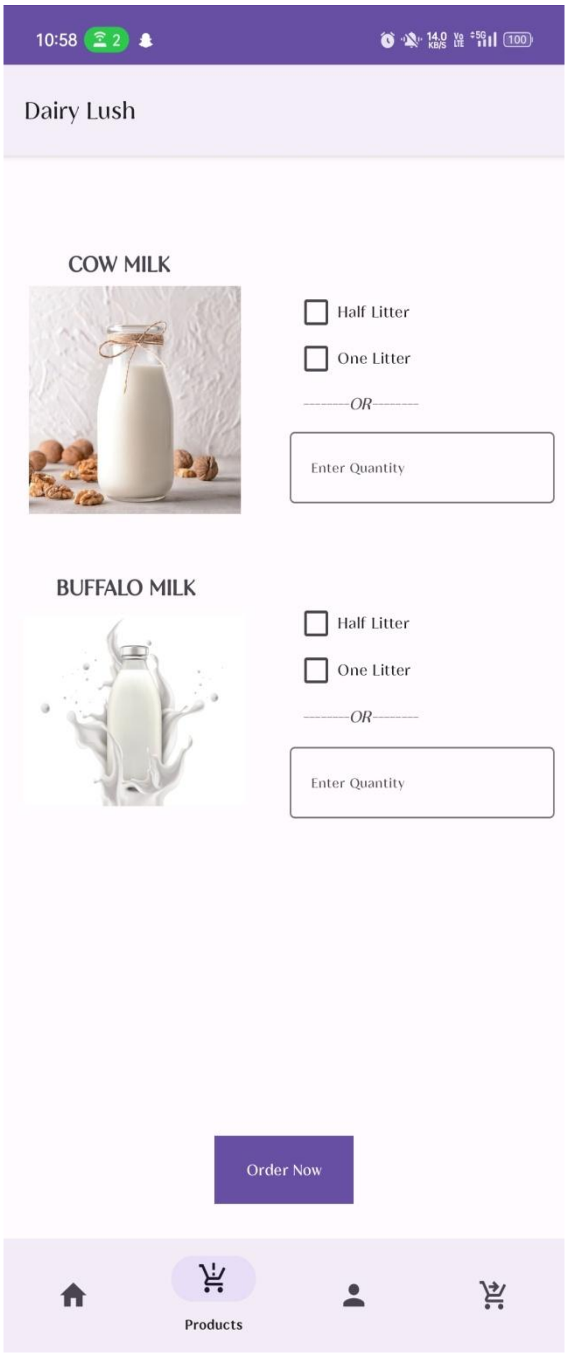
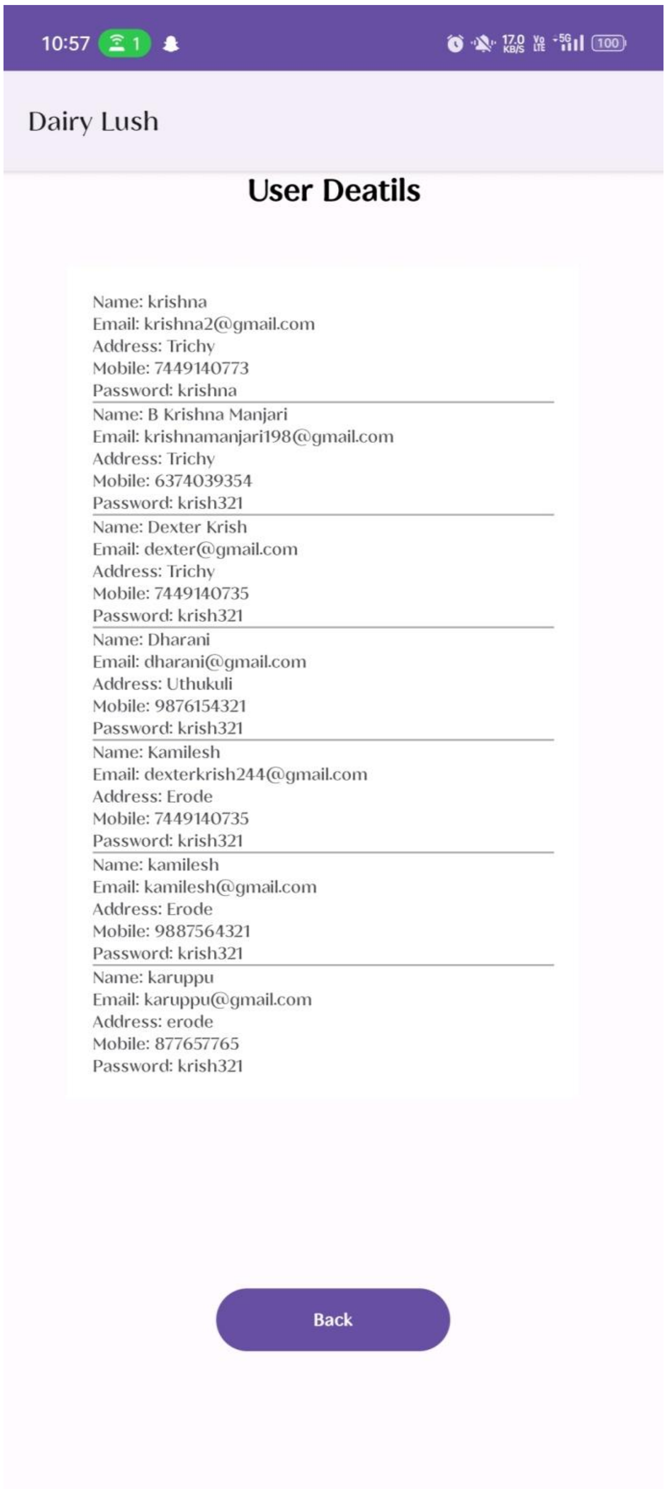


Figure A.2.4 Order page



User Deatils

Name: krishna
Email: krishna2@gmail.com
Address: Trichy
Mobile: 7449140773
Password: krishna

Name: B Krishna Manjari
Email: krishnamanjari198@gmail.com
Address: Trichy
Mobile: 6374039354
Password: krish321

Name: Dexter Krish
Email: dexter@gmail.com
Address: Trichy
Mobile: 7449140735
Password: krish321

Name: Dharani
Email: dharani@gmail.com
Address: Uthukuli
Mobile: 9876154321
Password: krish321

Name: Kamilesh
Email: dexterkrish244@gmail.com
Address: Erode
Mobile: 7449140735
Password: krish321

Name: kamilesh
Email: kamilesh@gmail.com
Address: Erode
Mobile: 9887564321
Password: krish321

Name: karuppu
Email: karuppu@gmail.com
Address: erode
Mobile: 877657765
Password: krish321

Figure A.4.5 User Management page

REFERENCES

1.sDevelopers, Android. "What is android." *Dosegljivo:http://www.academia. edu/download/30551848/andoid--tech. pdf* (2011).

2. Moroney, Laurence, and Laurence Moroney. "The firebase realtime database." *The Definitive Guide to Firebase: Build Android Apps on Google's Mobile Platform* (2017).

WEBSITES REFERRED

1. https://youtu.be/QAKq8UBv4GI?si=C6TxuY6RVNVsGREP

2. https://developer.android.com/guide/components/fundamentals/

3. https://firebase.google.com/docs/

4. https://firebase.google.com/docs/database/web/start