Student Number: 2307932

Student Name: Binod Bhandari

Module: 7cc012: Mobile Application Development

**Introduction:**

In this mobile app project, my goal is to help users manage their blogs without always needing the internet. Like finding blogging can also be tricky without reliable internet. So, I am creating an app that lets users write, edit, and share blog posts even offline. It's all about making blogging easier for everyone.

**Project Overview:**

In the context of the blogging app project, this application aims to revolutionize how individuals and organizations manage their blogging activities. This app serves as a comprehensive platform for creating, editing, and sharing blog content offline, catering to users with varying levels of technical expertise. By aggregating blog data and providing seamless functionality, this app streamlines the blogging process, enabling users to efficiently share their thoughts and stories with the world.

**Goals and Objectives:**

Facilitate individuals and organizations in efficiently managing their blogging activities offline.

• Establish a user-friendly platform that caters to bloggers of varying technical proficiency levels.

• Provide a seamless and transparent system for creating, editing, and sharing blog content.

• Enhance the accessibility of blogging by enabling users to work offline and synchronize data when connected to the internet.

• Foster a vibrant community of bloggers by offering features that encourage collaboration and sharing of ideas.

• Empower users to have full control over their blog content, including easy management of posts and media attachments.

• Ensure reliability and efficiency in the offline functionality, allowing users to create and edit content anytime, anywhere.

• Promote engagement and interaction by implementing search features for exploring blog archives and discovering relevant content.

• Enhance the user experience through intuitive design and smooth navigation, ensuring a pleasant blogging experience.

**Technologies Used:**

* Programming Language: Dart
* Framework: Flutter
* Database: Sqlite
* Flutter Pakages:bloc, flutter\_bloc, sqflite, Cupertino\_icons, equitable,path, dartz, get\_it, flutter\_screenutil, font\_awesome\_flutter, image\_picker, path\_provider, share\_plus, another\_flushbar, flutter\_toast, image\_cropper, flutter\_slidable, flutter\_launcher\_icons, simple\_animations, shared\_preferences, url\_launcher, Mockito.

**Functional Specification:**

* Offline Blog Creation: Allow users to create, edit, and manage blog posts without requiring an internet connection.
* Local Database Storage: Store blog data locally on the device to ensure accessibility even when offline.
* Text Input: Provide a user-friendly interface for inputting blog titles and content.
* Image Attachment: Enable users to attach photos or images to their blog posts from the device's photo gallery or camera.
* Blog Management: Allow users to view, edit, and delete individual blog posts, as well as manage the overall list of blog items.
* Search Functionality: Implements a search feature to enable users to search for specific text within their blog posts or across the entire blog archive.
* Sharing Mechanism: Enable users to share individual blog posts via email using the standard platform "Share" mechanism.
* Group Deletion: Allow users to select and delete multiple blog items simultaneously for efficient management.
* Date Stamp: Automatically record the date of blog entry for each post to maintain chronological order.
* User-Friendly Interface: Design an intuitive and easy-to-navigate interface suitable for users with varying levels of technical proficiency.

**Risks:**

* Data Loss: Risk of data loss due to local storage issues; mitigated by regular backups.
* Limited Offline Functionality: Users may experience feature limitations offline.
* Compatibility Issues: Possible inconsistencies across devices; thorough testing planned.
* User Adoption: Risk of low adoption if app doesn't meet expectations; continuous feedback loop established.
* Technical Challenges: Development may face bugs, performance issues; agile approach adopted for quick resolution.
* Regulatory Compliance: Legal risks due to non-compliance with data protection laws; ongoing monitoring ensured.
* Network Dependency: Certain features require network access; alternatives provided for offline use.

**Cost Considerations:**

* Development Costs: Investment required for development tools, software licenses,
* Operational Costs: Ongoing expenses for platform updates, customer support, and marketing efforts.
* Resource Allocation: Strategies for efficient resource allocation to optimize budget utilization.
* Cost Management: Measures to monitor and control expenses throughout the project lifecycle.
* Revenue Generation: Exploration of monetization strategies such as subscription models or in-app purchases to offset costs.
* Sustainability Plan: Long-term strategies for financial sustainability and scalability of the project.

**Project Development life cycle:**

The Blog app for Android and IOS uses a method called the Iterative Model. This means I started with basic features and keep improving them until the whole system is ready.

Instead of making a full list of what the app should do at the beginning, it starts with some features. Then, as it gets worked on, more features are added in each round.

The process has four steps: gathering what the app needs to do, designing how it will work, building and testing it, and then checking to see if it meets what's needed and making any necessary changes.

A diagram of a process

Description automatically generated

**Project planning and feasibility study:**

Before making any software, we need to understand what the system needs and plan the project based on that. We create a feasibility report to do a detailed analysis of the project idea. This report looks at three main things:

* 1. Economic feasibility: We check if the project can be built at a reasonable cost. We figure out if we need to reduce costs or not.
  2. Technical feasibility: We see if the software we're making is technically sound. This means it should work well with the system it's meant for.
  3. Organization feasibility: We check if the people in the organization can use the software. We especially look at whether employees, like receptionists, can use it easily.

**Requirement analysis:**

This phase is crucial in the Software Development Life Cycle (SDLC). It involves gathering detailed reports from various fields to find business solutions or create systems. It's essential to gather all requirements to avoid problems later. Clear understanding of how stock is currently recorded and how the owner wants it recorded in the new system is crucial in this phase.

**System design:**

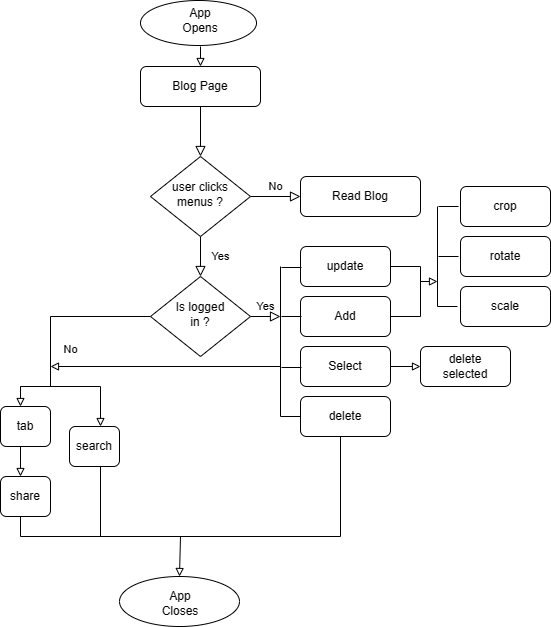
Design refers to the modules used to build the software. There are different modules such as E-R diagram, DFD’s, flowchart, etc. So, it’s very difficult to code. Thus, we have used the concept of object-oriented programming. In this program the modules in the main menu and subdivision are the functions.

E-R Diagram:

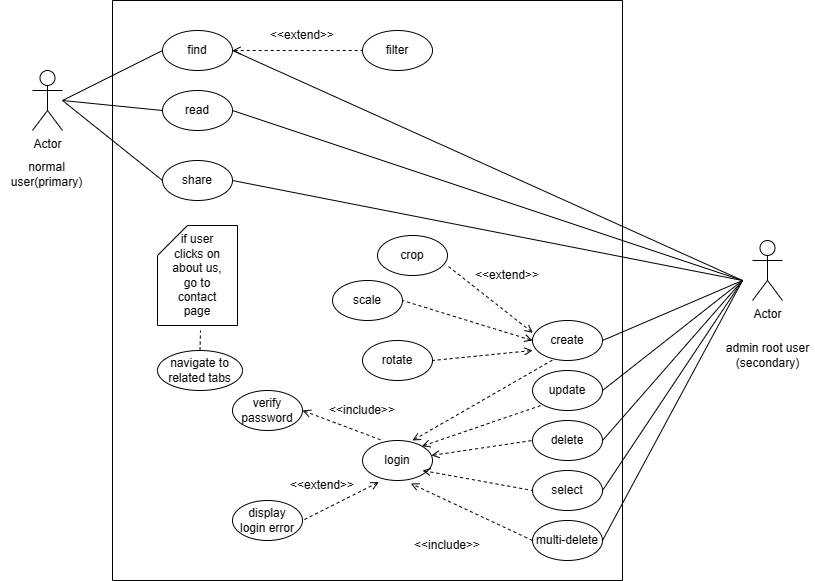
A black and white rectangular object

Description automatically generated

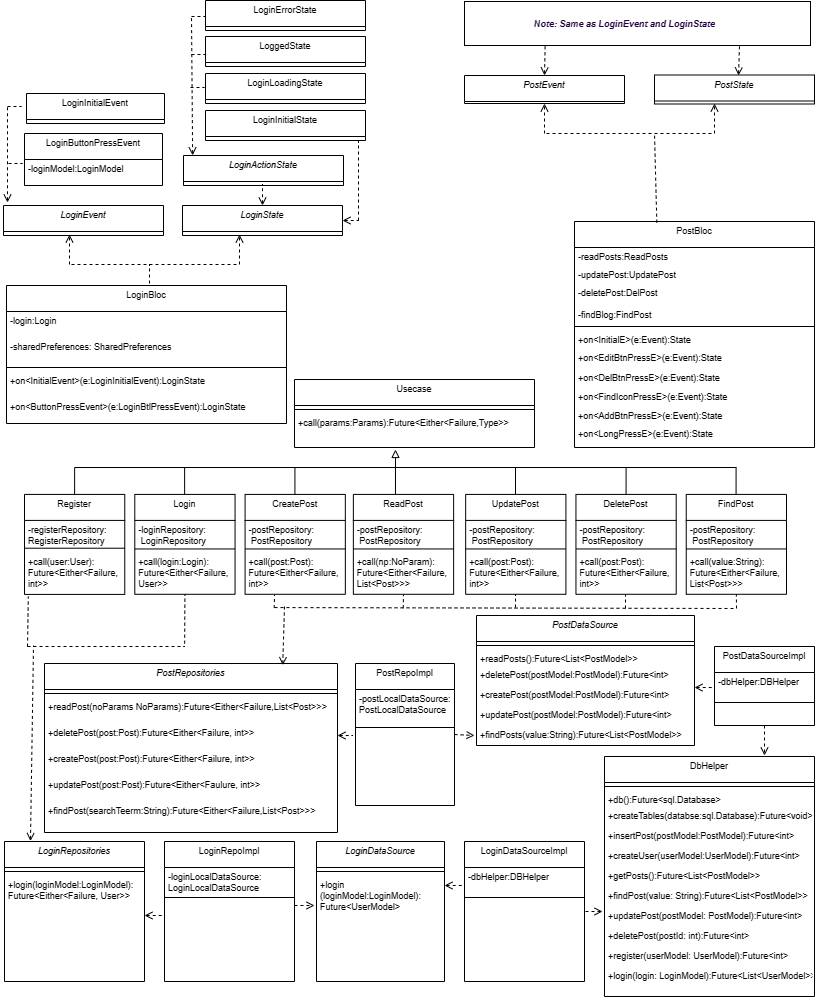
Flowchart:



Usecase Diagram:



Class Diagram:



Architecture:

A diagram of data flow

Description automatically generatedA screenshot of a computer

Description automatically generated

**Architecture Implementation:**

**A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated**

A screenshot of a computer

Description automatically generated

**Coding:** Based on the system design, I did the coding of the system. The coding is done using the dart programming language.

SOLID:

In simple terms, clean architecture and test-driven development suggest organizing code in a way that promotes independence and flexibility. The code as layers of an onion: at the core are entities, which represent the business logic and don't rely on anything else. Surrounding them are use cases, which orchestrate the flow of data and actions, but still don't depend on external frameworks or libraries. Further out are interfaces that define how different parts of your code interact, allowing for easy swapping of components without affecting the core functionality. By following these principles, I wrote code that's easier to understand, maintain, and test, even as this project grows and evolves.

Diagram of a diagram of a clean architecture

Description automatically generated

**Testing:** Testing is like finding and fixing mistakes in software. We test to make sure the program works right and to figure out why it might not. Testing helps make sure the software is good quality and works correctly. I did this by running the program with different types of data and checking if it behaves the way it's supposed to. So, when I finish testing, I know my app is error-free and works well.

Test Result for readPost(): passed

A screenshot of a computer program

Description automatically generated

Test Result for findPosts(String searchTerm): Tests failed:

A screenshot of a computer program

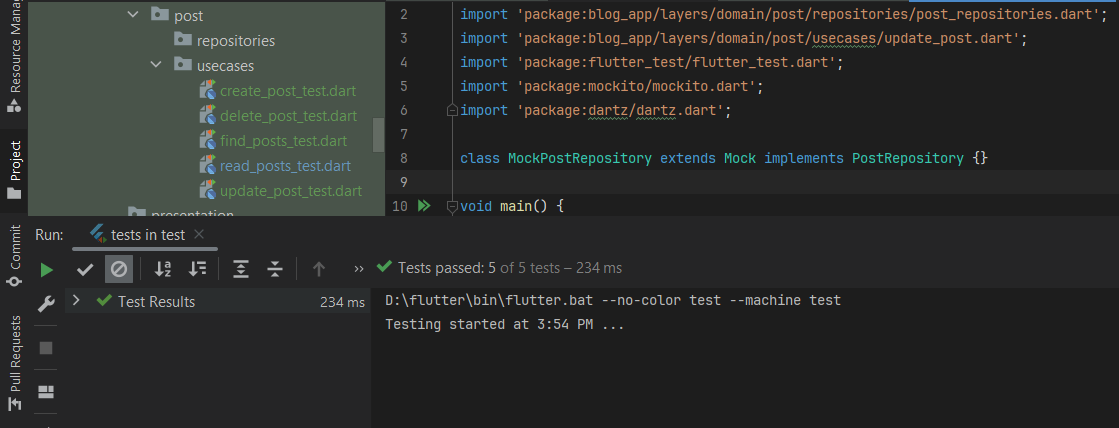
Description automatically generated

**Issue fixed:**  Issue was I forgot to pass the string parameter on verify stage of the test. I encountered test failed several times, I noticed later I have to pass the string parameter, then finally I solved this issue by pasing searchTerm for findPosts() method of PostRepositories.

**A screenshot of a computer program

Description automatically generated**

*Note: Similarly, I run every test to ensure that, my application is running and will run smoothly.*

**

**A screenshot of a computer

Description automatically generated**

**Debugging:**

I have debugged our program for finding the reducing the number of bugs, or defects, in a computer or a piece of electronic hardware thus making it behave as expected.

Although each debugging experience is unique, there are certain general principles I have applied in debugging. The basic steps that I have used in our program are: -

* Recognize that a bug exists.
* Isolate the source of the bug.
* Identify the cause of the bugs.
* Determine a fix for the bugs.
* Apply the fix and test it.

**Implementation:** System implementation generally focuses on the coding and installation of the system. My app implementation is composed of activities, which are coding, testing and installation. The purpose of these steps is to convert the physical system specification into working and reliable application.

Coding is the process whereby the physical design specification created by the analysis team(me) is turned into working computer code by the programming team(myself). Once the coding has begun the testing procedure began and proceeded in parallel.

**Documentation and Evaluation:** I have documented all the activity performed during development of this system, which will be very helpful in the future modifications or changes. As per the time being if the vendor wants to make some amendments (changes) in the existing program of his system, the developer should edit the programs per his requirements. This phase of fulfilling the demands comes up with evaluation phase.

Manpower and Limitations:

Manpower:

The required human resource to operate the software: -

* Basic English knowledge.
* Familiar with basic android/ios mobile system.
* Familiar with basic use of application.

**Limitations:**

* Currently application is available in android/ios platform only.
* Needed android mobile with at least Lollipop operating system version or higher.
* Should have good basic knowledge of Smartphone.
* User need to know privacy and policies.
* Should have basic knowledge of computer-based system.

**Cost Estimation:**

|  |  |  |
| --- | --- | --- |
| Sn | Activities | Price($) |
| 1. | Google play console | 25 |
| 2. | Hardware and Software | 20 |
| 3. | Data collection | 10 |
| 4. | System Design | 10 |
| 5. | Testing | 20 |
| 6. | Stationary related Expenses | 10 |
| 7. | Miscellaneous Expenses | 15 |
|  | Total | 110 |

**Duration:**

|  |  |
| --- | --- |
| ACTIVITIES | DURATION(DAYS) |
| PROBLEM ANALYSIS | 3 |
| REQUIREMENT ANALYSIS | 3 |
| DESIGNING | 5 |
| CODING AND TESTING | 10 |
| IMPLEMENTATION | 2 |
| DOCUMENTATION | 5 |
| TOTAL ESTIMATES (DURATION) | 29 |

**Gantt Chart:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | TASK | 2024 | | | | | | | | | | | | |
| S.N. | Feb. | | | | March. | | | | April. | | | | |
| 1st | 2nd | 3rd | 4th | 1st | 2nd | 3rd | 4th | 1st | 2nd | 3rd | 4th |
| 1. | Requirement Analysis |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |
| 2. | Designing |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |
| 3. | Coding |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
| 4. | Testing |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |
| 5. | Implementation |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |
| 6. | Documentation |  |  |  |  |  |  |  |  |  |  |  |  |
|  |

**Screenshots:**

**A blue logo on a blue background

Description automatically generatedA logo of a company

Description automatically generatedA screenshot of a phone

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a login screen

Description automatically generatedA screenshot of a login screen

Description automatically generatedA screenshot of a login screen

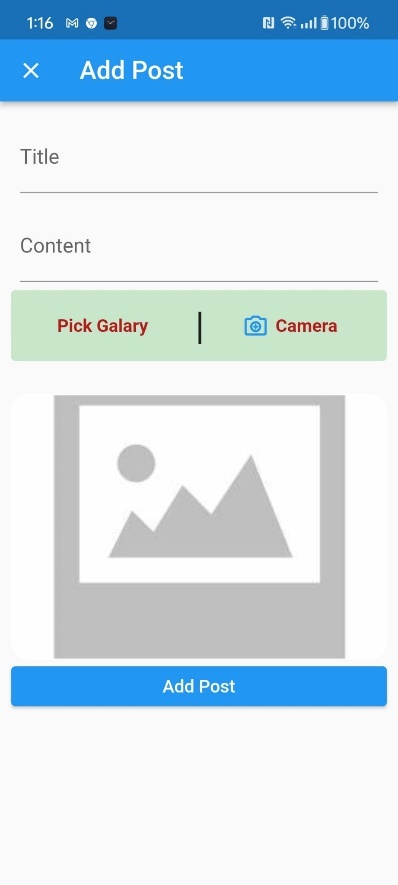
Description automatically generatedA screenshot of a login screen

Description automatically generatedA screenshot of a login screen

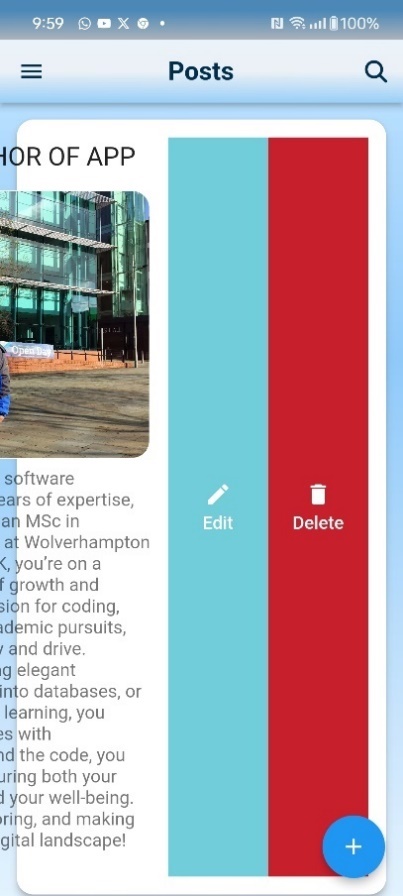
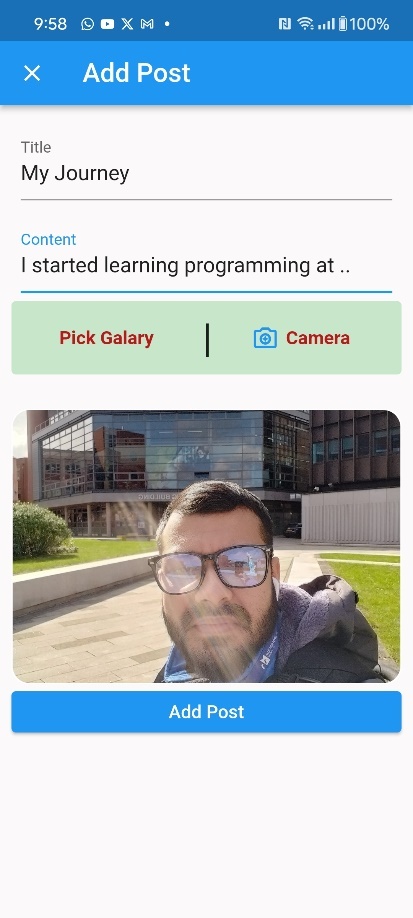
Description automatically generatedA screenshot of a login screen

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a phone

Description automatically generatedA group of people posing for a photo

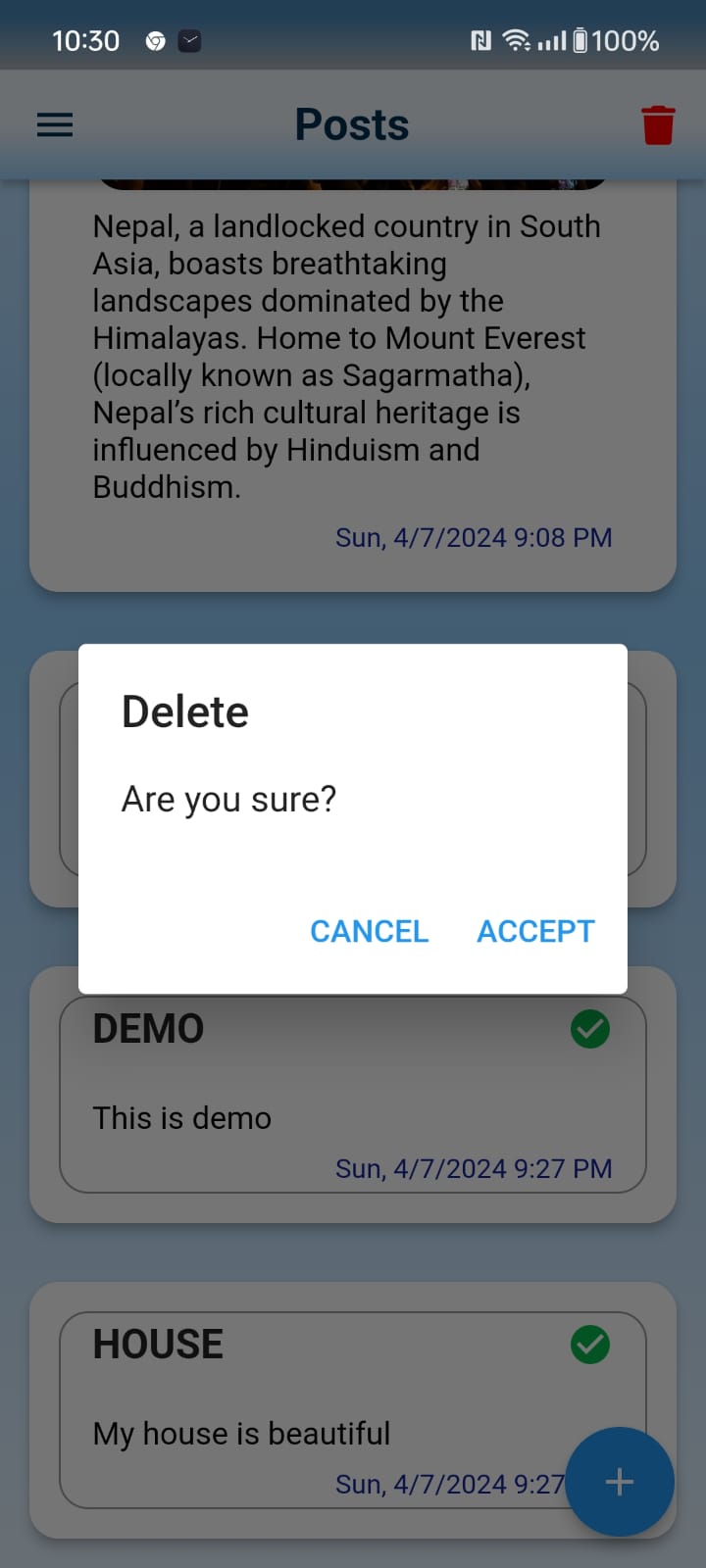
Description automatically generatedA screenshot of a football team

Description automatically generatedA screenshot of a cellphone

Description automatically generatedA screenshot of a phone

Description automatically generatedA screenshot of a phone

Description automatically generatedScreens screenshot of a test

Description automatically generatedA screenshot of a phone

Description automatically generatedA person standing in front of a building

Description automatically generatedA screenshot of a phone

Description automatically generatedA screenshot of a phone

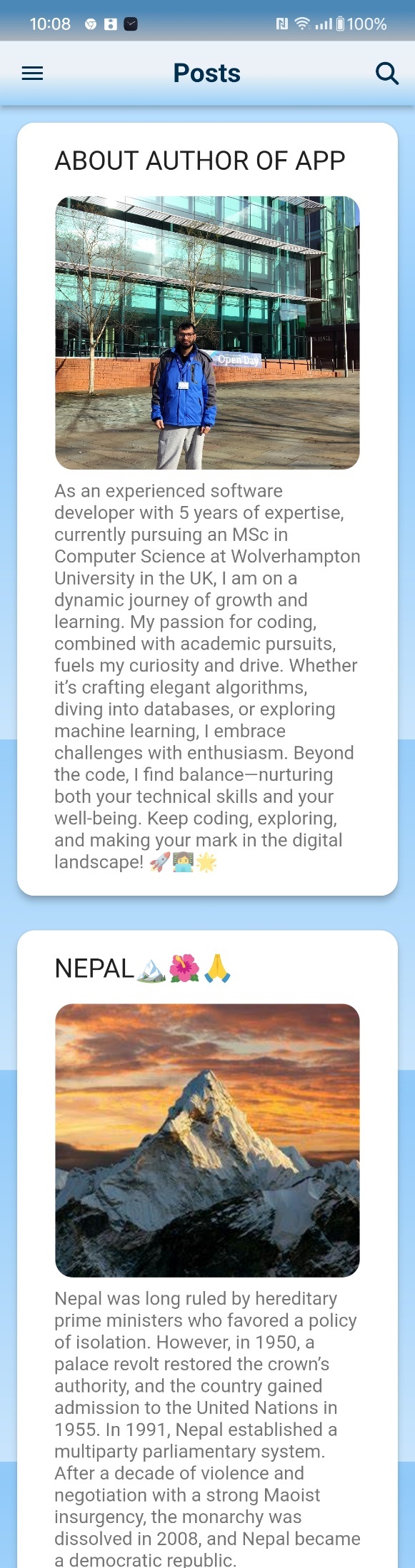
Description automatically generatedA screenshot of a phone

Description automatically generatedA screenshot of a phone

Description automatically generatedA screenshot of a phone

Description automatically generatedA white background with black text

Description automatically generated**

**A screenshot of a phone

Description automatically generated**

**GitHub Updates:**

**A computer screen shot of a computer

Description automatically generated**

**GitHub Commits with SHA:**

**git log**

Windows PowerShell

Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

commit 9ca577358d12ca8aa98748210392094931d6fbb8 (HEAD -> main, origin/main)

Author: binod <binodcoder@gmail.com>

Date: Fri Apr 5 12:24:53 2024 +0100

some bug fixed

commit 45860cf3087964fb8bd7b1bb5823af8312dfb75e

Author: binod <binodcoder@gmail.com>

Date: Fri Apr 5 09:40:58 2024 +0100

exception, failure handled for login

commit b9354691a012641efaa247a2ad93c9b416b01c17

Author: binod <binodcoder@gmail.com>

Date: Thu Apr 4 18:21:47 2024 +0100

admin role, delete all dialog

commit 73866db453501335a346d411acd7e57b6b0cf923

Author: binod <binodcoder@gmail.com>

Date: Thu Apr 4 01:55:53 2024 +0100

login and drawer added

commit bb5696de403eef3d7cd10a0e2f92bad702c81db5

Author: binod <binodcoder@gmail.com>

Date: Thu Apr 4 00:11:11 2024 +0100

re-arranged folder structure , design changes

commit 585504f7b5d5c278ff83fd7d7642fb67ad77aa1e

Author: binod <binodcoder@gmail.com>

commit 9ca577358d12ca8aa98748210392094931d6fbb8 (HEAD -> main, origin/main)

Author: binod <binodcoder@gmail.com>

Date: Fri Apr 5 12:24:53 2024 +0100

some bug fixed

commit 45860cf3087964fb8bd7b1bb5823af8312dfb75e

Author: binod <binodcoder@gmail.com>

Date: Fri Apr 5 09:40:58 2024 +0100

exception, failure handled for login

commit b9354691a012641efaa247a2ad93c9b416b01c17

Author: binod <binodcoder@gmail.com>

Date: Thu Apr 4 18:21:47 2024 +0100

admin role, delete all dialog

commit 73866db453501335a346d411acd7e57b6b0cf923

Author: binod <binodcoder@gmail.com>

Date: Thu Apr 4 01:55:53 2024 +0100

login and drawer added

commit bb5696de403eef3d7cd10a0e2f92bad702c81db5

Author: binod <binodcoder@gmail.com>

Date: Thu Apr 4 00:11:11 2024 +0100

re-arranged folder structure , design changes

commit 585504f7b5d5c278ff83fd7d7642fb67ad77aa1e

Author: binod <binodcoder@gmail.com>

Date: Wed Apr 3 15:47:12 2024 +0100

commit 9ca577358d12ca8aa98748210392094931d6fbb8 (HEAD -> main, origin/main)

Author: binod <binodcoder@gmail.com>

Date: Fri Apr 5 12:24:53 2024 +0100

some bug fixed

commit 45860cf3087964fb8bd7b1bb5823af8312dfb75e

Author: binod <binodcoder@gmail.com>

Date: Fri Apr 5 09:40:58 2024 +0100

exception, failure handled for login

commit b9354691a012641efaa247a2ad93c9b416b01c17

Author: binod <binodcoder@gmail.com>

Date: Thu Apr 4 18:21:47 2024 +0100

admin role, delete all dialog

commit 73866db453501335a346d411acd7e57b6b0cf923

Author: binod <binodcoder@gmail.com>

Date: Thu Apr 4 01:55:53 2024 +0100

login and drawer added

commit bb5696de403eef3d7cd10a0e2f92bad702c81db5

Author: binod <binodcoder@gmail.com>

Date: Thu Apr 4 00:11:11 2024 +0100

re-arranged folder structure , design changes

commit 585504f7b5d5c278ff83fd7d7642fb67ad77aa1e

Author: binod <binodcoder@gmail.com>

Date: Wed Apr 3 15:47:12 2024 +0100

commit 9ca577358d12ca8aa98748210392094931d6fbb8 (HEAD -> main, origin/main)

Author: binod <binodcoder@gmail.com>

Date: Fri Apr 5 12:24:53 2024 +0100

some bug fixed

commit 45860cf3087964fb8bd7b1bb5823af8312dfb75e

Author: binod <binodcoder@gmail.com>

Date: Fri Apr 5 09:40:58 2024 +0100

exception, failure handled for login

commit b9354691a012641efaa247a2ad93c9b416b01c17

Author: binod <binodcoder@gmail.com>

Date: Thu Apr 4 18:21:47 2024 +0100

admin role, delete all dialog

commit 73866db453501335a346d411acd7e57b6b0cf923

Author: binod <binodcoder@gmail.com>

Date: Thu Apr 4 01:55:53 2024 +0100

login and drawer added

commit bb5696de403eef3d7cd10a0e2f92bad702c81db5

Author: binod <binodcoder@gmail.com>

Date: Thu Apr 4 00:11:11 2024 +0100

re-arranged folder structure , design changes

commit 585504f7b5d5c278ff83fd7d7642fb67ad77aa1e

Author: binod <binodcoder@gmail.com>

Date: Wed Apr 3 15:47:12 2024 +0100

desing improved

Author: binod <binodcoder@gmail.com>

Date: Fri Apr 5 12:24:53 2024 +0100

some bug fixed

commit 45860cf3087964fb8bd7b1bb5823af8312dfb75e

Author: binod <binodcoder@gmail.com>

Date: Fri Apr 5 09:40:58 2024 +0100

exception, failure handled for login

commit b9354691a012641efaa247a2ad93c9b416b01c17

Author: binod <binodcoder@gmail.com>

Date: Thu Apr 4 18:21:47 2024 +0100

admin role, delete all dialog

commit 73866db453501335a346d411acd7e57b6b0cf923

Author: binod <binodcoder@gmail.com>

Date: Thu Apr 4 01:55:53 2024 +0100

login and drawer added

commit bb5696de403eef3d7cd10a0e2f92bad702c81db5

Author: binod <binodcoder@gmail.com>

Date: Thu Apr 4 00:11:11 2024 +0100

re-arranged folder structure , design changes

commit 585504f7b5d5c278ff83fd7d7642fb67ad77aa1e

Author: binod <binodcoder@gmail.com>

Date: Wed Apr 3 15:47:12 2024 +0100

desing improved

commit d53474f3ced05b891351fd913478704fa6e1c4dd

Author: binod <binodcoder@gmail.com>

Date: Wed Apr 3 15:15:40 2024 +0100

some design changs, crop image feature added

commit d39354a43282c66b0bee8f1597ebe8c0e4afbebd

Author: binod <binodcoder@gmail.com>

Date: Mon Mar 25 10:32:29 2024 +0000

search by title and content on change text on search box

commit fa9da138c2f7a5a5fd97d6c80b1066cad2c95b51

Author: binod <binodcoder@gmail.com>

Date: Mon Mar 25 00:05:49 2024 +0000

share blog and search function added

commit f4a8f4325b453820cf721b5ee4c906eb7b038eec

Author: binod <binodcoder@gmail.com>

Date: Sun Mar 24 17:59:21 2024 +0000

image handeling, Open/Close implementation of SOLID principle

commit fa7bdaa49e5a11e265c6e81ab4d14fc83c896a91

Author: binod <binodcoder@gmail.com>

Date: Sun Feb 25 19:56:49 2024 +0000

clean domain driven development archetecture implemented successfully

commit 57f77807aa28fe686109e09633f617483a014064

Author: binod <binodcoder@gmail.com>

Date: Sun Feb 25 11:55:17 2024 +0000

data, domain layer is created for add post feature

commit b1b30433c9bfcc1e8f185448360d95fb9ffd8414

Author: binod <binodcoder@gmail.com>

Date: Sat Feb 24 20:13:13 2024 +0000

data layer added and dependency injection implemented

commit 474aa01407cfa863f86b04f3b7af5c5e430255db

Author: binod <binodcoder@gmail.com>

Date: Sat Feb 24 18:16:54 2024 +0000

clean archetecture- domain layer added

commit 10d21da665606511aa3baa5f3b88c68a623ca389

Author: binod <binodcoder@gmail.com>

Date: Tue Feb 20 15:17:47 2024 +0000

bug with state for image url is fixed

commit ab8c1ed129f83ee2faf8bebc609acbbfddfe8a76

Author: binod <binodcoder@gmail.com>

Date: Tue Feb 20 12:15:47 2024 +0000

seperate bloc made for add post

commit d2ef353ca785e0e0df44a7bda893232d986b41db

Author: binod <binodcoder@gmail.com>

Date: Tue Feb 20 09:04:35 2024 +0000

post page clean archerecture implemented

commit 3beec1957102cccae34327e0c4ea0e6947b92ab1

Author: binod <binodcoder@gmail.com>

Date: Sun Feb 18 14:25:30 2024 +0000

ath to local database

commit 69ddbc42a5c604d98edd8a7a9549a92cac114497

Author: binod <binodcoder@gmail.com>

Date: Sat Feb 17 15:41:24 2024 +0000

images saved and retrived from database

commit c98af9c85887a2424a30f4ca269114e7dc05c7f8

Author: binod <binodcoder@gmail.com>

Date: Thu Feb 15 09:21:31 2024 +0000

add delete function after selection anytime in future

commit 875c1662f0b4b07337e03d6ccef48441e3a9969d

Author: binod <binodcoder@gmail.com>

Date: Wed Feb 14 16:09:01 2024 +0000

multiple selection and delete function

commit 1d797183ea0c93b7f6926041bea800cedf28ddeb

Author: binod <binodcoder@gmail.com>

Date: Wed Feb 14 11:10:28 2024 +0000

detail page added

commit 91686d9f45fd4875bcb093ef935e3141b17603b9

Author: binod <binodcoder@gmail.com>

Date: Sun Feb 4 09:31:34 2024 +0000

floating action button to navigate to add post page

commit c2caa7b61b263b2b9bd964ae3b48d0d613e0210e

Author: binod <binodcoder@gmail.com>

Date: Sat Feb 3 14:48:54 2024 +0000

resources added

commit 06b46fd70e560f0c8f3a8e81eb19e19711af5667

Author: binod <binodcoder@gmail.com>

Date: Sat Feb 3 10:25:25 2024 +0000

bloc is seperated from ui

commit 1b95c017c7d81bc4dd21c76da22a794bb03c8152

Author: binod <binodcoder@gmail.com>

Date: Sun Jan 7 19:33:25 2024 +0000

local db initialized

(END)

git format-patch < commit\_SHA>

**Major Development Decisions and Issues:**

**Major Development Decisions:**

I chose to use two important principles in my project: the SOLID principle by Robert C Martin and the BLoC method for managing state by Felix Angelov. The project also includes elements of functional programming and follows a clean architecture approach. I am using Test Driven Development (TDD) and Domain Driven Development (DDD) for structuring the project. To start, I created a login feature for the root user. To make this work smoothly, I set up the app to automatically register an admin as the root user the first time when app opens, using default settings built into the code.

**Issues:**

During the development process, I encountered some challenges, but I managed to overcome them with dedication. Thanks to adhering to Test Driven Development principles and maintaining a clean architecture, troubleshooting errors was relatively straightforward. However, I struggled with UI design, which consumed a significant amount of time compared to actual development. Despite facing some hurdles like needing additional design elements, I was able to tackle them with determination and a bit of trial and error.

**Conclusion:**

In conclusion, sticking to SOLID principles and Test-Driven Development (TDD) made my project journey smooth and efficient. Moving forward, I plan to extend and maintain this project, incorporating RESTful APIs to facilitate access for multiple users.