****

**RAMAIAH INSTITUTE OF TECHNOLOGY, BANGALORE – 560054**

**(Autonomous Institute, Affiliated to VTU)**

**Department of Computer Science & Engineering**

**Internship Report**

**on**

Mobile Application Development

INT410: Intra Institutional Internship

**STUDENT NAME: MANDAR STUDENT NAME: SHIVAM S**

**BHAT KAMALAPURKAR**

**USN: 1MS24AD038 USN: 1MS24AD056**

**Ramaiah Institute of Technology**

**(Autonomous Institute, Affiliated to VTU)**

**MSR Nagar, MSRIT Post, Bangalore-560054**

**August - 2025**

****

**RAMAIAH INSTITUTE OF TECHNOLOGY, BANGALORE – 560054**

**(Autonomous Institute, Affiliated to VTU)**

**Department of Computer Science & Engineering**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**DEPARTMENT OF**

**CERTIFICATE**

This is to certify that **Mr. Mandar Bhat (USN: 1MS24AD038)** and **Mr. Shivam S Kamalapurkar (USN: 1MS24AD056)**, a student of Bachelor of Engineering, have successfully completed, 24 Hours: from 05.08.2025 to 12.08.2025 Intra Institutional Internship in Mobile Application Development from the Department of Computer Science & Engineering, M S Ramaiah Institute of Technology, Bangalore.

|  |  |  |  |
| --- | --- | --- | --- |
| **SL**  **No.** | **Component** | **Maximum Marks** | **Marks Obtained** |
| **1** | **Continuous Evaluation** | **50** |  |
| **2** | **Presentation** | **20** |  |
| **3** | **Report** | **30** |  |
| **Total Marks** | | **100** |  |

|  |  |
| --- | --- |
| **Signature of the Student with Date** | **Signature of the Faculty Co-Ordinator** |
| **Signature of Head of the Department** | |

**OVERVIEW OF INTERNSHIP ACTIVITIES**

|  |  |  |
| --- | --- | --- |
| **DATE** | **DAY** | **NAME OF THE TOPIC COMPLETED** |
| 05/08/25 | Tuesday | Introduction to Flutter, Android SDK, VS code, Dart Basics – programming Fundamentals |
| 06/08/25 | Wednesday | Building User Interfaces with Flutter Widgets and Layouts, Styling and Theming in Flutter, Dart advanced Topics, Quiz |
| 07/08/25 | Thursday | Handling User Input and Navigation, User Input and Gesture Handling Quiz |
| 09/08/25 | Saturday | Navigation and Routing, Quiz, Exercises in dart and FLUTTER |
| 11/08/25 | Monday | DART, FLUTTER Assessment, Applying Firebase and Backend |
| 12/08/25 | Tuesday | Project Demo and Viva Voce |

Contents

[1. Introduction 5](#_Toc205837207)

[2. Objectives 5](#_Toc205837208)

[3. System Analysis 5](#_Toc205837209)

[4. Code Structure 6](#_Toc205837210)

[5. Module Description 6](#_Toc205837211)

[User Interface and User Experience 6](#_Toc205837212)

[Data Handling 6](#_Toc205837213)

[Feedback Integration 7](#_Toc205837214)

[Strengths and Limitations 7](#_Toc205837215)

[Future Enhancements 7](#_Toc205837216)

[Detailed Code Analysis 8](#_Toc205837217)

[Workflow and Navigation 12](#_Toc205837218)

[UI/UX Considerations 12](#_Toc205837219)

[Testing and Validation 12](#_Toc205837220)

[Output 13](#_Toc205837221)

[Conclusion and Recommendations 13](#_Toc205837222)

# Introduction

The Campus Hub application is a comprehensive mobile solution designed to enhance navigation and resource discovery within a campus environment. The Room Finder module, a core feature of the app, empowers students, faculty, and visitors to quickly locate specific rooms based on purpose, capacity, or category. Implemented in **Flutter**, the module ensures a **cross-platform** experience on both Android and iOS without code duplication. Its design philosophy prioritizes **efficiency**, **user engagement**, and **future scalability**. The embedded feedback system ensures that the application evolves in line with user needs, bridging the gap between physical campus navigation challenges and digital convenience.

# Objectives

The primary objectives of the Room Finder module are as follows:

1. **Search Functionality:** Provide a robust search mechanism that allows users to filter rooms by category, capacity, or real-time availability.
2. **Detailed Information Access:** Ensure that each room listing contains vital details, including seating capacity, equipment availability, and accessibility features.
3. **User Feedback Integration:** Facilitate direct feedback submissions to continuously refine and improve the service.
4. **Optimized Mobile UI/UX:** Create an intuitive interface with minimal navigation steps, ensuring that users can find relevant rooms in seconds.

By achieving these objectives, the Room Finder module contributes to overall campus efficiency and user satisfaction.

# System Analysis

The Room Finder module adopts a **modular architecture** to ensure maintainability and scalability. Responsibilities are clearly separated into models, data files, UI screens, and reusable widgets. This design enables:

* **Code Reusability:** Widgets and functions can be reused across multiple screens.
* **Scalability:** Future modules (e.g., event booking, campus map) can integrate seamlessly.
* **Maintainability:** Updates to one module do not disrupt the entire application.

# 

# Code Structure

* **main.dart:** Entry point of the application, handling routing and theme configuration.
* **data/room\_data.dart:** Holds either static or dynamically fetched room data.
* **models/room.dart:** Defines the Room class with attributes for identification, categorization, and description.
* **screens/**: Houses screens such as home\_screen.dart, category\_screen.dart, and feedback\_screen.dart.
* **widgets/room\_list\_tile.dart:** A custom, reusable UI component for displaying concise room details.
* **feedback\_widget/**: Web assets (HTML, CSS, JavaScript) for embedding the feedback interface.

# Module Description

The module begins at the **Home Screen**, where users see a categorized grid of available room types. Selecting a category filters available rooms and displays them on the **Category Screen**. The **Room List Tile** widget ensures a consistent and clean presentation of essential details. The design minimizes the number of clicks required, making the navigation path efficient and predictable.

# User Interface and User Experience

The Room Finder UI focuses on **clarity**, **consistency**, and **responsiveness**:

* **Clean Layout:** Uses Flutter’s Scaffold, ListView, and GridView for logical content arrangement.
* **Responsive Design:** Adapts to multiple screen sizes without compromising usability.
* **Visual Cues:** Icons and color codes help users quickly identify categories.
* **Feedback Channel:** A dedicated screen for submitting suggestions ensures two-way communication.

# Data Handling

Currently, all room details are stored in room\_data.dart for rapid prototyping. The structure is intentionally designed for smooth migration to an API or cloud database. Each Room object contains core properties (ID, name, category, capacity, description) and can be extended to include booking status, floor number, or accessibility options.

# Feedback Integration

Feedback is handled by embedding a web-based form using HTML, CSS, and JavaScript files stored in the feedback\_widget directory. This form can later be connected to a backend service for real-time submissions. It allows users to:

* Suggest improvements.
* Report incorrect or outdated room information.
* Share usability concerns.

# Strengths and Limitations

**Strengths:**

* Cross-platform compatibility with Flutter.
* Modular and extensible code structure.
* Built-in feedback loop for continuous improvement.

**Limitations:**

* Static data source requiring manual updates.
* Absence of user authentication or personalization features.
* No real-time availability tracking yet.

# Future Enhancements

Planned improvements include:

* Connecting to a **cloud-hosted database** for live data.
* Implementing **authentication** for personalized room recommendations.
* Adding **map-based navigation** for visual guidance to rooms.
* Enhancing search with **autocomplete** and **AI-powered suggestions**.

# 

# Detailed Code Analysis

This section examines each code file and its responsibilities:

* **main.dart:** Initializes the application, sets themes, defines named routes, and manages navigation between screens.
* import 'package:flutter/material.dart';
* // To enable Firebase initialize uncomment these lines and add firebase\_core to pubspec.yaml
* // import 'package:firebase\_core/firebase\_core.dart';
* import 'screens/home\_screen.dart';
* void main() async {
* WidgetsFlutterBinding.ensureInitialized();
* // await Firebase.initializeApp(); // initialize Firebase if configured
* runApp(RoomFinderApp());
* }
* class RoomFinderApp extends StatelessWidget {
* @override
* Widget build(BuildContext context) {
* return MaterialApp(
* title: 'Room Finder',
* theme: ThemeData(
* brightness: Brightness.light,
* fontFamily: 'Montserrat',
* colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
* appBarTheme: AppBarTheme(
* backgroundColor: Colors.deepPurple.shade400,
* elevation: 6,
* titleTextStyle: TextStyle(fontSize: 24, color: Colors.white, fontWeight: FontWeight.bold),
* ),
* scaffoldBackgroundColor: Colors.grey.shade100,
* ),
* home: HomeScreen(),
* debugShowCheckedModeBanner: false,
* );
* }
* }
* **models/room.dart:** Dart class defining all Room properties and methods for easier manipulation of room data.
* class Room {
* final String name;
* final String number;
* final String floor;
* final String block;
* final String category; // e.g. Staff, Lab, Office, Department, Others
* Room({
* required this.name,
* required this.number,
* required this.floor,
* required this.block,
* required this.category,
* });
* }
* **data/room\_data.dart:** Temporary storage for room objects, acting as a placeholder for future API integration.
* **screens/home\_screen.dart:** Displays all room categories in a visually appealing grid.
* **screens/category\_screen.dart:** Lists all rooms within a selected category using RoomListTile.
* import 'package:flutter/material.dart';
* import 'feedback\_screen.dart';
* import 'category\_screen.dart';
* import '../widgets/room\_list\_tile.dart';
* import '../data/room\_data.dart';
* import '../models/room.dart';
* final List<Map<String, dynamic>> categoryInfo = [
* {'label': 'Staff', 'icon': Icons.person},
* {'label': 'Office', 'icon': Icons.business\_center},
* {'label': 'Lab', 'icon': Icons.science},
* {'label': 'Department', 'icon': Icons.apartment},
* {'label': 'Others', 'icon': Icons.more\_horiz},
* ];
* class HomeScreen extends StatefulWidget {
* @override
* State<HomeScreen> createState() => \_HomeScreenState();
* }
* class \_HomeScreenState extends State<HomeScreen> {
* String globalQuery = '';
* List<Room> get globalResults => rooms.where((room) =>
* room.name.toLowerCase().contains(globalQuery.toLowerCase()) ||
* room.number.toLowerCase().contains(globalQuery.toLowerCase()) ||
* room.block.toLowerCase().contains(globalQuery.toLowerCase()) ||
* room.category.toLowerCase().contains(globalQuery.toLowerCase())
* ).toList();
* void openCategory(String category) {
* Navigator.push(
* context,
* MaterialPageRoute(
* builder: (\_) => CategoryScreen(category: category),
* ),
* );
* }
* @override
* Widget build(BuildContext context) {
* return Scaffold(
* floatingActionButton: FloatingActionButton.extended(
* onPressed: (){ Navigator.of(context).push(MaterialPageRoute(builder: (\_) => const FeedbackScreen())); },
* label: Text('Review'),
* icon: Icon(Icons.rate\_review),
* backgroundColor: Colors.deepPurple.shade400,
* ),
* appBar: AppBar(
* title: const Text('CAMPUS HUB'),
* ),
* drawer: Drawer(
* backgroundColor: Colors.white,
* shape: const RoundedRectangleBorder(
* borderRadius: BorderRadius.only(
* topRight: Radius.circular(40),
* bottomRight: Radius.circular(40),
* )
* ),
* child: SafeArea(
* child: Column(
* children: [
* Container(
* height: 120,
* width: double.infinity,
* decoration: BoxDecoration(
* gradient: LinearGradient(
* colors: [Colors.deepPurple.shade300, Colors.deepPurple.shade100],
* ),
* borderRadius: const BorderRadius.only(
* topRight: Radius.circular(40)
* ),
* ),
* alignment: Alignment.centerLeft,
* padding: const EdgeInsets.only(left: 24),
* child: Text(
* "Navigation",
* style: TextStyle(fontSize: 28, color: Colors.deepPurple.shade900, fontWeight: FontWeight.bold),
* ),
* ),
* const SizedBox(height: 10),
* ...categoryInfo.map((cat) =>
* ListTile(
* leading: Icon(cat['icon'], color: Colors.deepPurple),
* title: Text(cat['label'], style: TextStyle(fontSize: 18)),
* onTap: () {
* Navigator.of(context).pop(); // close drawer
* openCategory(cat['label']);
* },
* )
* ).toList(),
* const Expanded(child: SizedBox())
* **screens/feedback\_screen.dart:** Loads the feedback form interface inside a WebView for cross-platform compatibility.
* **widgets/room\_list\_tile.dart:** A reusable widget that formats room data into a compact and readable card/tile layout.

# Workflow and Navigation

The app follows a **linear and intuitive workflow**:

1. **Launch App → Home Screen**
2. **Select Category → Category Screen**
3. **Select Room → View Room Details (planned)**
4. **Provide Feedback → Feedback Screen**

Navigation uses **Flutter’s Navigator API** with push and pop methods, but is structured to allow easy migration to advanced navigation frameworks.

# UI/UX Considerations

The design emphasizes:

* **Visual consistency** across screens.
* **Readable font sizes** and **sufficient contrast** for accessibility.
* **Touch-friendly components** to accommodate smaller devices.
* **Adaptive layouts** to ensure proper rendering on tablets and large displays.

# Testing and Validation

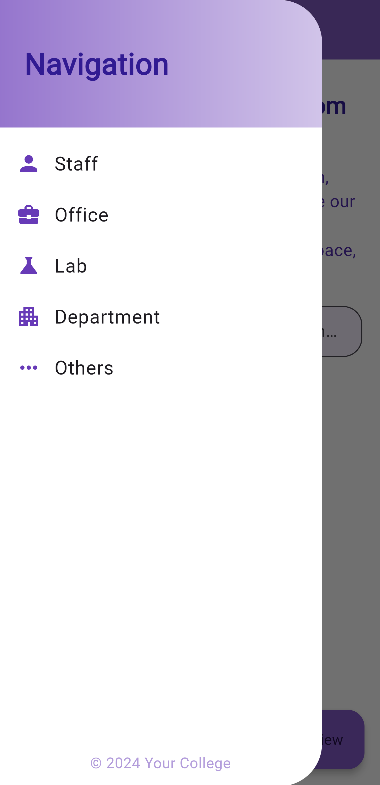
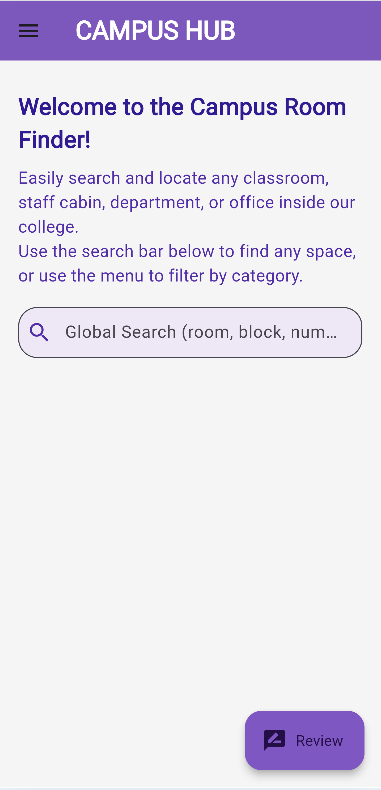
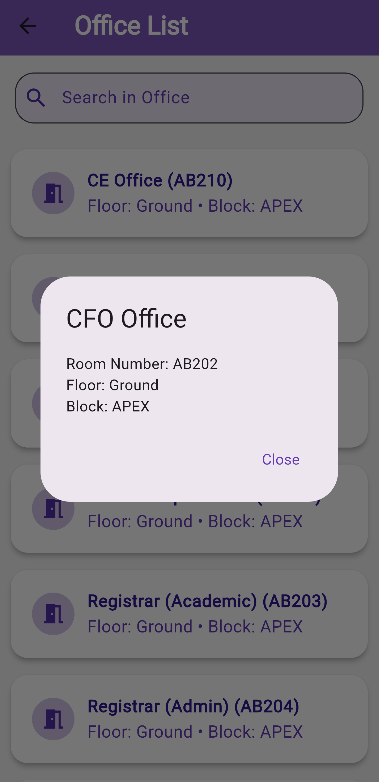
The module has been tested on both Android and iOS platforms.

**Manual Testing Process:**

* Verify that each category loads the correct set of rooms.
* Test search functionality and filter accuracy.
* Confirm the feedback form’s display and basic submission.
* Check UI responsiveness on multiple device resolutions.

# Output

# 

# Conclusion and Recommendations

The Campus Hub Room Finder module is a functional and well-structured mobile application segment. Its modular approach ensures scalability, and the integration of a feedback system highlights its focus on continuous improvement.

Recommendations:  
- Implement a database-backed API for dynamic data.  
- Enhance search capabilities with autocomplete and advanced filters.  
- Introduce authentication for personalized features.