

**Discovering Knowledge**

**FINAL YEAR PROJECT REPORT**

**2025**

**Builder Management System**

**Group Members**

|  |  |
| --- | --- |
| **Student Name** | **Enrolment#** |
| Muhammad Shoaib Akhter Qadri | 02-131212-009 |

**Supervised by**

Engr Adnan-ur-Rahman

**Department of Software Engineering**

**BAHRIA UNIVERSITY KARACHI CAMPUS**

**Intellectual Property & Submission Policy**

This is to declare that,

1. The project under the supervision of \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ having title \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ carried out in partial fulfillment of the requirements of Bachelor of Engineering in Software Engineering degree program and is the sole property of the Bahria University.
2. This report is submitted as the requirement for the project in accordance with the rules laid down by the Bahria University as part of the requirements for the award of the degree of Bachelor of Engineering in Software Engineering.
3. The work presented in this report is our own except where due reference or acknowledgement is given to the work of others.
4. We are aware of Bahira University asserts legal and beneficial ownership rights over all Intellectual Property developed as a result of support either directly from or channeled through Bahria University.
5. We are agreed to assign to Bahira University all of their rights, title and interest in and to Intellectual Property developed as a result of utilization of Bahira University Resources including copyright in any material that is teaching material, computer programs, or created at the request or direction of Bahira University.

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No.** | **Student Name** | **Enrolment #** | **Signatures** |
| **1** | Muhammad Shoaib Akhter Qadri | 02-131212-009 | \_\_\_\_\_\_\_\_\_\_\_ |

**Reference:**

**[1]** R&D policy handbook (BUORIC-P15)

**Submission Proforma**

It is acknowledged that I as a supervisor gone through this report. The report contains all the essential sections as required by the department in accordance with the rules laid down by the Bahria University.

|  |  |
| --- | --- |
| **Supervisor Name:** |  |
|  |  |
| **Supervisor Signatures:** |  |
| **Dated:** |  |

**Acknowledgments**

We would like to express our sincere gratitude to all the individuals and institutions who have contributed to the successful completion of our Final Year Project, "**Builder Management System**".

First and foremost, we would like to thank our supervisor, Engr Adnan-ur-Rahman, for his invaluable guidance, expertise, and continuous support throughout the development process. His insightful feedback, constructive criticism, and prompt assistance have been instrumental in shaping the project and ensuring its successful execution. We are truly grateful for his mentorship and encouragement.

We would also like to extend our heartfelt appreciation to the faculty members of Bahria University for their valuable insights and assistance whenever needed. Their expertise and knowledge in their respective fields have greatly enriched our understanding and helped us overcome various challenges during the project.

Furthermore, we would like to acknowledge the Head of Institute for providing us with the necessary facilities and resources that facilitated the smooth progress of our work. Their support and encouragement have been pivotal in our journey.

We are indebted to the technical support staff who have aided us in setting up the required infrastructure and resolving technical issues promptly. Their dedication and expertise have been invaluable in ensuring the smooth functioning of the project.

Lastly, we would like to express our gratitude to our friends and family for their unwavering support, understanding, and motivation throughout the project. Their encouragement and belief in our abilities have been a constant source of inspiration.

To all those mentioned above and anyone else who has contributed to our project in any way, we extend our deepest appreciation. Your assistance and support have played a significant role in the successful completion of "**Builder Management System”**

**Abstract**

The proposed project aims to develop a **Builder Management Software** tailored for the residential construction industry to address key challenges such as poor communication, lack of transparency, and ineffective project management. The platform will serve as a centralized hub for customers, builders, and tradies to collaborate and track construction projects in real-time. It will feature tools for seamless communication, project tracking, document management, and financial oversight, leveraging advanced technologies such as **React.js** with **Firebase** task automation.

Existing solutions like BuildTools and Procore offer functionalities in project management but fail to provide comprehensive real-time updates and personalized communication. The proposed system distinguishes itself through features such as AI-generated work breakdown structures (WBS), customizable project tracking, and direct communication channels, enhancing customer experience and ensuring accountability.

Adopting an **Agile Development Process**, the project will deliver iterative improvements based on feedback from stakeholders to ensure user satisfaction. By aligning with **Sustainable Development Goals** (SDGs) 9 and 11, the project promotes innovation and infrastructure development while fostering well-managed urban communities.

This software aspires to improve transparency, minimize disputes, and streamline project delivery, setting a new standard in residential construction management.

TABLE OF Contents

[1. INTRODUCTION 11](#_Toc170226041)

[1.1 Introduction 12](#_Toc170226042)

[1.2 Problem Statement 12](#_Toc170226043)

[1.3 Background / Literature Review 12](#_Toc170226044)

[1.4 Problem in existing system 13](#_Toc170226045)

[1.5 Project Solution 13](#_Toc170226046)

[1.5.1 Features of the project 13](#_Toc170226046)

[1.5.2 Methodology/Algorithm 13](#_Toc170226046)

[1.5.3 Technologies to be used 13](#_Toc170226046)

[1.5.4 Project Scope / Deliverables 13](#_Toc170226046)

[1.6 Project Scope / Deliverables 13](#_Toc170226046)

[2. Proposed model 14](#_Toc170226047)

[2.1 Features of the project 15](#_Toc170226048)

[2.2 Methodology/Algorithm 15](#_Toc170226053)

[2.3 Technologies to be used 15](#_Toc170226054)

[3. Software Project Management Plan 16](#_Toc170226055)

[3.1 Software Process Model 17](#_Toc170226056)

[3.2 Roles and Responsibilities 18](#_Toc170226057)

[3.3 Tools and Techniques 18](#_Toc170226058)

[3.3.1 Programming Languages: 18](#_Toc170226059)

[3.3.2 Techniques: 18](#_Toc170226060)

[3.3.3 Tools: 18](#_Toc170226061)

[3.3.4 Coding Standards: 19](#_Toc170226062)

[3.4 Project Management Plan 19](#_Toc170226063)

[3.4.1 Tasks 19](#_Toc170226064)

[3.4.2 Task: SPMP-T001 20](#_Toc170226065)

[3.4.3 Task: SPMP-T002 21](#_Toc170226066)

[3.4.4 Task: SPMP-T003 22](#_Toc170226067)

[3.5 Assignments 24](#_Toc170226068)

[3.6 Timetable 25](#_Toc170226069)

[4. Software Requirements Specifications 26](#_Toc170226070)

[4.1 Introduction 27](#_Toc170226071)

[4.2 Product Overview 27](#_Toc170226072)

[4.3 Specific Requirements 27](#_Toc170226073)

[4.3.1 Image Recognition System Based on Research Findings 27](#_Toc170226074)

[4.3.2 Mobile Application Development 27](#_Toc170226075)

[4.3.3 Recommendations Module 28](#_Toc170226076)

[4.3.4 Testing 28](#_Toc170226077)

[4.4 Functional and Data Requirements 28](#_Toc170226078)

[4.4.1 Image Recognition System 28](#_Toc170226079)

[4.4.2 Mobile Application Development 29](#_Toc170226080)

[4.4.3 Multiclass Classification 29](#_Toc170226081)

[4.4.4 Nutritional Information 30](#_Toc170226082)

[4.4.5 Recommendations Module 30](#_Toc170226083)

[4.5 Non-Functional Requirements 30](#_Toc170226084)

[4.5.1 Reliability 30](#_Toc170226085)

[4.5.2 Availability 30](#_Toc170226086)

[4.5.3 Maintainability 31](#_Toc170226087)

[4.5.4 Portability 31](#_Toc170226088)

[4.5.5 Performance 31](#_Toc170226089)

[4.5.6 Usability Requirements 32](#_Toc170226090)

[4.6 Proposed Solution 32](#_Toc170226091)

[4.6.1 Technology Stack: 33](#_Toc170226092)

[4.7 External Interface Requirements 34](#_Toc170226093)

[4.7.1 User Interfaces 34](#_Toc170226094)

[4.7.2 Hardware Interfaces 36](#_Toc170226095)

[4.7.3 Software Interfaces 37](#_Toc170226096)

[4.7.4 Communications Protocols 38](#_Toc170226097)

[4.8 Database Requirements 38](#_Toc170226098)

[5. Software Design Description 39](#_Toc170226099)

[5.1 Introduction 40](#_Toc170226100)

[5.2 Design Overview 40](#_Toc170226101)

[5.3 Work Flow Diagram 40](#_Toc170226102)

[5.4 Work Breakdown Structure 41](#_Toc170226103)

[5.5 Use case Diagram 42](#_Toc170226104)

[5.6 Sequence Diagram 43](#_Toc170226105)

[5.7 Requirements Traceability Matrix 44](#_Toc170226106)

[5.8 System Architectural Design 45](#_Toc170226107)

[5.9 Chosen System Architecture 45](#_Toc170226108)

[The architecture is outlined below: 46](#_Toc170226109)

[5.10 System Interface Description 47](#_Toc170226110)

[5.10.1 Operating System Interface: 47](#_Toc170226111)

[5.10.2 Files Interface: 48](#_Toc170226112)

[5.10.3 Networking Interface: 48](#_Toc170226113)

[5.10.4 Libraries: 48](#_Toc170226114)

[5.11 Detailed Description of Components 49](#_Toc170226115)

[5.11.1 Component-1: Image Recognition System Based on Research Findings 49](#_Toc170226116)

[5.11.2 Component-2: Mobile Application 49](#_Toc170226117)

[5.11.3 Component 3: Real-time Classification Component 50](#_Toc170226118)

[5.11.4 Component 4: Multiclass Classification Component 50](#_Toc170226119)

[5.11.5 Component 5: User-Friendly Interface Component 50](#_Toc170226120)

[5.11.6 Component 6: Nutritional Information Component 51](#_Toc170226121)

[5.11.7 Component 7: Recommendations Module Component 51](#_Toc170226122)

[5.12 User Interface Design 52](#_Toc170226123)

[6. Implementation 55](#_Toc170226124)

[6.1 Algorithms: 56](#_Toc170226125)

[6.1.1 Resnet-50: 56](#_Toc170226126)

[6.1.2 VGG-16 64](#_Toc170226127)

[6.2 Frontend: 74](#_Toc170226128)

[6.2.1 Classification Component: 74](#_Toc170226129)

[6.2.2 Recommendation Component 79](#_Toc170226130)

[6.2.3 Nutrition Card Component 79](#_Toc170226131)

[6.2.4 Home Screen: 80](#_Toc170226132)

[6.2.5 Calories Box: 81](#_Toc170226133)

[6.3 Backend Code: 82](#_Toc170226134)

[6.3.1 Date Image Consumer Code: 82](#_Toc170226135)

[6.3.2 DockerFile: 83](#_Toc170226136)

[6.3.3 Asgi.py: 84](#_Toc170226137)

[7. Software Test Document 85](#_Toc170226138)

[7.1 System Overview: 86](#_Toc170226139)

[7.1.1 First Model (Date Identification): 86](#_Toc170226140)

[7.1.2 Second Model (Date Classification): 86](#_Toc170226141)

[7.2 System Testing: 86](#_Toc170226142)

[7.2.1 Algorithm Testing: 86](#_Toc170226143)

[7.2.2 WebSocket Testing: 87](#_Toc170226144)

[7.2.3 Software Version: 87](#_Toc170226145)

[7.3 Test Approach 87](#_Toc170226146)

[7.3.1 Major Testing Activities and Techniques 87](#_Toc170226147)

[7.3.2 Major Feature Groups and Combinations 89](#_Toc170226148)

[7.4 Test Plan: 89](#_Toc170226149)

[7.4.1 Scope 89](#_Toc170226150)

[7.4.2 Algorithm Performance Testing: 90](#_Toc170226151)

[7.4.3 Date Identification Testing: 90](#_Toc170226152)

[7.4.4 Date Classification Testing: 90](#_Toc170226153)

[7.4.5 WebSocket Testing: 90](#_Toc170226154)

[7.4.6 Frontend Data Retrieval Testing: 90](#_Toc170226155)

[7.4.7 Resources: 90](#_Toc170226156)

[7.4.8 Items Being Tested: 91](#_Toc170226157)

[7.5 Features to be Tested 91](#_Toc170226158)

[7.5.1 Features to be Tested 91](#_Toc170226159)

[7.5.2 Combination of Features to be Tested 93](#_Toc170226160)

[7.6 Test Cases 94](#_Toc170226161)

[7.6.1 Test Case Specifications 94](#_Toc170226162)

[7.6.2 Test Case: TC-007 100](#_Toc170226163)

[8. Appendix A: Test Results 102](#_Toc170226164)

[8.1.1 Test Results 102](#_Toc170226165)

[8.1.2 VGG-16 117](#_Toc170226166)

[9. Conclusions and Further Work 121](#_Toc170226167)

[9.1 Conclusions: 121](#_Toc170226168)

[9.2 Further Work: 121](#_Toc170226169)

[10. References 122](#_Toc170226170)

# INTRODUCTION

## Introduction

In the residential construction industry, **seamless communication, transparency, and efficient project management** between customers, builders, and tradies are crucial for the success of building projects. This proposal outlines the development of a Builder Management Software aimed at solving key pain points faced by both customers and builders. The proposed solution will offer a **centralized platform** where all parties can track and manage residential construction projects in real-time, ensuring transparency, communication, and accountability at all stages.

## Problem Statement

In the residential building industry, customers face significant challenges in managing communication and expectations with builders during the construction process. After selecting a builder and signing a contract, many customers experience frustration due to a **lack of transparency, poor communication, and missing documented evidence** of important discussions and agreements made during the project. Key project details, such as selected materials, appliances, and finishes, are often discussed verbally but not properly tracked, leading to misunderstandings and disputes over what was promised versus what is delivered.

This lack of transparency and accountability leads to customer dissatisfaction, project delays, and potential cost overruns due to unforeseen variations or changes that weren’t clearly documented or agreed upon.

The proposed web application aims to solve these problems by providing a centralized platform where customers and builders can seamlessly manage their residential construction projects. This platform will allow customers to create and track project details, agree on itemized lists with builders, and ensure that every decision is properly documented and agreed upon before the contract is signed. Additionally, it will provide continuous communication and progress updates throughout the building process, helping both parties stay aligned and informed, thus reducing misunderstandings and disputes.

## Background / Literature Review

Currently, the market offers several builder management software solutions such as BuildTools, Houzz Pro, Procore, CoConstruct, and Buildertrend. While these platforms address some construction management issues, there remain gaps in transparency, real-time communication, and customer-builder collaboration that we aim to fill with our software.

The existing solutions provide functionalities like project management, budget tracking, and progress updates but lack critical features that enhance the customer experience, particularly around personalized communication and real-time documentation of decisions.

Notable Builder Software Platforms:

* BuildTools: <https://www.buildtools.com/>
* Procore: <https://www.procore.com/>
* CoConstruct: <https://www.coconstruct.com/>
* Buildertrend: <https://www.buildertrend.com/>

## Problem in existing system

The existing builder management systems lack transparency, real-time communication, and comprehensive customer-builder collaboration. Customers face challenges in tracking project progress, managing agreements on materials and finishes, and ensuring documentation of important decisions. This often leads to misunderstandings, disputes, delays, and cost overruns. Current solutions also fail to provide personalized communication and real-time updates, leaving customers disconnected and dissatisfied.

## Proposed Solution

Our Builder Management Software will provide an **all-in-one platform** that integrates project management, customer communication, document tracking, and financial oversight. Key features are designed to address the core issues outlined in the problem statement, while offering enhanced functionality not found in existing builder software solutions.

### Features of the Project:

1. **Chatting & Communication:** A real-time messaging system allowing customers, builders, and tradies to communicate seamlessly.
2. **Products Selection & Documentation**: Tracking customer selections for materials, appliances, and finishes with documented agreements.
3. **Profile Management**: Comprehensive profile pages for customers, custom builders, and tradies.
4. **Project Management:** Features for customers and builders to add and track projects, including descriptions, images, and building links.
5. **Companies Section:** A dedicated section for listing companies involved in a project, visible in the sidebar for easy navigation.
6. **Project and Task Dashboard:** Centralized project dashboards for tracking progress, deadlines, payments, and key milestones.
7. **Documentation:** Automatic generation of project-related documents, and search keywords from documents.
8. **Customer Reviews:** A system for customers to leave feedback about builders and companies.
9. **Notifications System:** Real-time alerts for project updates, task completions, and document approvals.
10. **Task Management:** A task assignment system where customers can assign tasks to builders and track progress.
11. **Verified Builders Status:** A status system for verifying builders’ credentials and project compliance with Victorian standards.

# 1.5.2 Methodology/Algorithm:

**Agile Development Process:**

* **Sprints and Iterations**: The project will be divided into **sprints**, with each sprint lasting between 2 to 4 weeks. Each sprint will focus on completing specific deliverables (e.g., frontend design, backend integration, AI functionality).
* **Daily Stand-ups**: Short daily meetings with the development team to discuss progress, roadblocks, and tasks for the day.
* **User Stories and Backlog**: User stories (specific requirements) will be created and prioritized in the product backlog to ensure that the most important features (e.g., real-time communication, AI documentation) are developed first.
* **Continuous Integration and Delivery (CI/CD)**: New code will be continuously integrated, and regular testing will ensure that new features don’t break the system. This allows for frequent, reliable releases.
* **Feedback Loops**: The system will undergo **User Acceptance Testing (UAT)** during multiple stages of the project, incorporating feedback from builders, clients, and tradies to improve the product continuously.

### Technologies to be Used:

* **Frontend:** React.js
* **Backend/Database:** Firebase
* **Real-Time Communication:** Real time Firebase Database
* **Task Scheduling & Progress Tracking:** Firebase Firestore for scheduled tasks

## Project Scope / Deliverables

1. **Chatting & Communication:** A real-time messaging system allowing customers, builders, and tradies to communicate seamlessly.
2. **Products Selection & Documentation**: Tracking customer selections for materials, appliances, and finishes with documented agreements.
3. **Profile Management**: Comprehensive profile pages for customers, custom builders, and tradies.
4. **Project Management:** Features for customers and builders to add and track projects, including descriptions, images, and building links.
5. **Companies Section:** A dedicated section for listing companies involved in a project, visible in the sidebar for easy navigation.
6. **Customer Reviews:** A system for customers to leave feedback about builders and companies.
7. **Notifications System:** Real-time alerts for project updates, task completions, and document approvals.
8. **Verified Builders Status:** A status system for verifying builders’ credentials and project compliance with Victorian standards.

# Software Project Management Plan

## Software Process Model

The **Agile Development Process** is selected for this project to ensure flexibility, collaboration, and continuous improvement. Key components include:

* **Sprints and Iterations**: Development is divided into 2-4 week sprints, focusing on specific deliverables.
* **Daily Stand-ups**: Regular meetings to address progress, challenges, and daily tasks.
* **User Stories and Backlog**: Features are prioritized in a backlog to address critical functionalities first.
* **Continuous Integration and Delivery (CI/CD)**: Regular code integration and testing for reliable updates and releases.
* **Feedback Loops**: User Acceptance Testing (UAT) at various stages to refine the product based on stakeholder input.

This model ensures that the software evolves to meet user needs while maintaining high quality and adaptability.

## Roles and Responsibilities

|  |  |
| --- | --- |
| **Team Members** | **Role** |
| **Muhammad Shoaib Akhter Qadri** | Requirement Analysis, System Design, Development, Database Management, Real-Time Communication, Testing and Debugging, Documentation, Deployment and Stakeholder Communication |

## Tools and Techniques

1. **Frontend Development**: React.js for building a dynamic and responsive user interface.
2. **Backend Development**: Firebase for backend services, including authentication, database, and hosting.
3. **Database Management**: Firebase Firestore for real-time data storage and retrieval.
4. **Version Control**: Git and GitHub for code collaboration and version management.
5. **Task Management**: Jira for organizing and tracking project progress.
6. **Communication**: Slack or Microsoft Teams for team collaboration and communication.
7. **Testing**: React Testing Library for unit and integration testing.

**Techniques:**

1. **Agile Methodology**: Iterative development with regular feedback loops for continuous improvement.
2. **Continuous Integration and Delivery (CI/CD)**: Automating code integration, testing, and deployment.
3. **Real-Time Updates**: Leveraging Firebase for instant synchronization of data across all users.
4. **Responsive Design**: Ensuring the application is accessible on various devices using React.js.
5. **Secure Development Practices**: Implementing secure coding techniques to protect user data and prevent vulnerabilities.

**2.4 Project Management Plan**

**2.4.1 Tasks**

**2.4.1.1 Requirements Analysis and Clarification**

**Description**: Analyze and clarify project requirements to ensure understanding and identify core functionalities.  
**Subtasks**:

* Review existing project documentation.
* Discuss ambiguities within the team.
* Identify necessary features and functionalities.  
  **Task Identifier**: SPMP-T001  
  **Estimated Time**: 1 Week

**2.4.1.2 Task: System Design**

**Description**: Create a comprehensive system design, including architecture, module specifications, and interaction diagrams.  
**Subtasks**:

* Develop system architecture diagrams.
* Specify modules and their functions.
* Create interaction diagrams.
* Document the design in the Software Design Description (SDD).  
  **Task Identifier**: SPMP-T002  
  **Estimated Time**: 2 Weeks

**2.4.1.3 Task: Development - Version 1**

**Description**: Implement the system's initial version based on finalized requirements and design.  
**Subtasks**:

* Set up the development environment.
* Write code for required features.
* Conduct unit testing for components.
* Integrate components and perform system testing.  
  **Task Identifier**: SPMP-T003  
  **Estimated Time**: 5 Weeks
* **2.4.2 Task: SPMP-T001**

**Description**: Conduct requirements analysis and clarification, identifying key features of the system.  
**Deliverables and Milestones**:

* Deliverables: Updated requirements documentation and list of core functionalities.
* Milestones: Completion of requirements review and feature identification.  
  **Resources Needed**: Access to project documents and reference materials.  
  **Dependencies and Constraints**: Dependent on availability of required materials and stakeholder engagement.  
  **Risks and Contingencies**:
* Risk: Lack of resources.
  + Contingency: Use online references or consult experts.
* Risk: Limited stakeholder input.
  + Contingency: Implement regular progress updates and alternative communication channels.
* **2.4.3 Task: SPMP-T002**

**Description**: Develop a detailed system design, including architecture and module specifications.  
**Deliverables and Milestones**:

* Deliverables: System architecture diagrams, module specifications, interaction diagrams, and SDD.
* Milestones: Completion of architecture, diagrams, and SDD.  
  **Resources Needed**: Tools for diagrams and reference materials for design principles.  
  **Dependencies and Constraints**: Requires completion of SPMP-T001.  
  **Risks and Contingencies**:
* Risk: Delays in resources.
  + Contingency: Explore alternative tools.
* Risk: Requirement changes.
  + Contingency: Use a change management process.
* **2.4.4 Task: SPMP-T003**

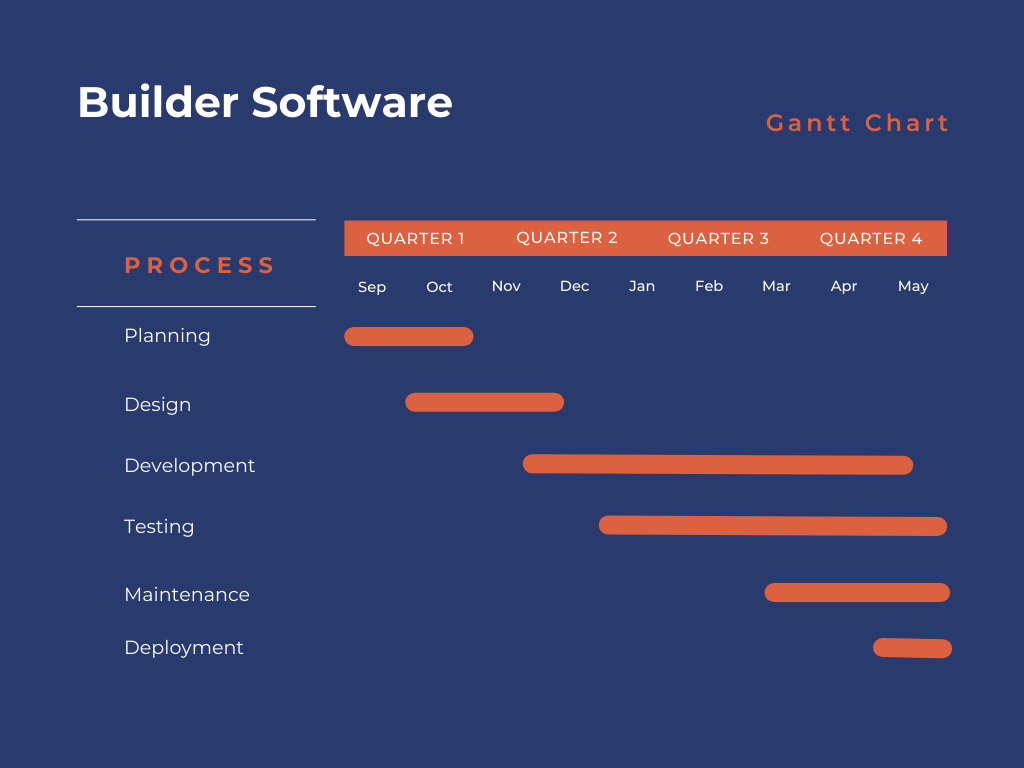
**Description**: Implement the system's functionalities and conduct testing for Version 1.  
**Deliverables and Milestones**:

* Deliverables: Functional Version 1 of the system.
* Milestones: Completion of coding, unit testing, integration, and system testing.  
  **Resources Needed**: Software Design Document (SDD), VS Code.  
  **Dependencies and Constraints**: Requires completion of SPMP-T002.  
  **Risks and Contingencies**:
* Risk: Technical challenges.
  + Contingency: Allocate additional time for debugging.
* Risk: Integration issues.
  + Contingency: Perform thorough integration testing.

## Assignments

|  |  |
| --- | --- |
| **Tasks** | **Performed by** |
| Research Work | Muhammad Shoaib Akhter Qadri |
| Data Collection | Muhammad Shoaib Akhter Qadri |
| SRS Document | Muhammad Shoaib Akhter Qadri |
| SRS Document Review | Muhammad Shoaib Akhter Qadri |
| WBS | Muhammad Shoaib Akhter Qadri |
| Use Case Diagram | Muhammad Shoaib Akhter Qadri |
| Architecture Diagram | Muhammad Shoaib Akhter Qadri |
| Sequential Diagram | Muhammad Shoaib Akhter Qadri |
| Workflow Diagram | Muhammad Shoaib Akhter Qadri |
| SDD | Muhammad Shoaib Akhter Qadri |
| Plan Document | Muhammad Shoaib Akhter Qadri |
| Gantt Chart | Muhammad Shoaib Akhter Qadri |
| Research Work | Muhammad Shoaib Akhter Qadri |
| Frontend Development | Muhammad Shoaib Akhter Qadri |
| DB Design and Creation | Muhammad Shoaib Akhter Qadri |
| API Development | Muhammad Shoaib Akhter Qadri |
| Model Training | Muhammad Shoaib Akhter Qadri |
| Model Testing | Muhammad Shoaib Akhter Qadri |
| Model Development | Muhammad Shoaib Akhter Qadri |

## Timetable



|  |  |  |
| --- | --- | --- |
| **Task** | **Start Date** | **Completion Date** |
| Planning | 2024-11-01 | 2024-11-31 |
| Design | 2024-11-01 | 2025-12-07 |
| Frontend Development | 2024-12-08 | 2025-05-01 |
| Backend Development | 2025-02-01 | 2025-05-15 |
| Testing | 2025-03-01 | 2025-04-01 |
| Maintenance | 2025-04-01 | 2025-05-01 |
| Deployment | 2025-05-01 | 2025-05-15 |

# Software Requirements Specifications

## Introduction

## The "Builder Management System" aims to address challenges in the residential construction industry by offering a centralized platform for seamless communication, transparency, and project management among customers, builders, and tradies. It ensures that all parties stay informed and aligned throughout the project lifecycle, mitigating common issues like poor communication, misunderstandings, and delays.

## 3.2 Product Overview:

## The software will integrate project management tools, customer communication features, real-time document tracking, and financial oversight. It will be equipped with key functionalities like task management, chatting systems, progress dashboards and collaboration between stakeholders.

## 3.3 Specific Requirements:

## The system requires functionalities for project tracking, communication, task assignments, document management, and customer reviews. It must support real-time interactions, secure data storage, and compatibility across various devices. Additionally, the software must be intuitive and user-friendly to cater to a wide range of users, from customers to builders and tradies.

## 3.4 Functional and Data Requirements:

## The system must allow for real-time messaging, project updates, document sharing, task assignments, and progress tracking. Data must be securely stored and easily accessible to authorized users. The platform will also manage detailed information on materials, products, and construction phases, ensuring that all decisions are documented and traceable.

## 3.5 Non-Functional Requirements:

## Reliability: The system should be stable, with minimal downtime.

## Availability: It should offer high availability to ensure users can access it anytime.

## Security: Data protection is paramount, with secure login and encryption.

## Maintainability: The system must be easy to maintain and update.

## Portability: It should be accessible on different devices and platforms.

## Performance: It must handle a large number of users and transactions efficiently.

## Usability: The interface must be intuitive and user-friendly.

## 3.6 Proposed Solution:

## The proposed Builder Management System will provide all-in-one functionality to enhance project management, communication, and documentation in the residential construction sector. Key features include real-time chat, project management tools, task tracking, verified builder status, and customer feedback.

## 3.7 Alternative Solution:

## Other existing systems like BuildTools and Procore offer similar features but lack the real-time collaboration that will distinguish your system.

## 3.8 External Interface Requirements:

## User Interfaces: Intuitive and user-friendly design.

## Hardware Interfaces: Compatible with devices such as smartphones, tablets, and desktops.

## Software Interfaces: Integrates with third-party software for financial management and task scheduling.

## Communications Protocols: Real-time communication supported by Firebase Database for instant updates.

## 3.9 Database Requirements:

## The system will rely on Firebase for real-time data syncing, secure storage, and smooth integration across devices. It will store project data, task details, messages, and financial records in a scalable and secure manner.

# Software Design Description

## Work Flow Diagram

The Builder Management System is designed with a modular architecture that separates key functionalities such as project management, communication, task tracking, and documentation. The front-end will be built using React.js for an interactive user experience, while the back-end will use Firebase for real-time data syncing and database management.

## Introduction:

This section introduces the overall design approach of the Builder Management System. It outlines the software architecture, key design principles, and the technologies that will be used to develop the platform. The design aims to provide a user-friendly interface for customers, builders, and tradies, while ensuring seamless communication, real-time updates, and data security.

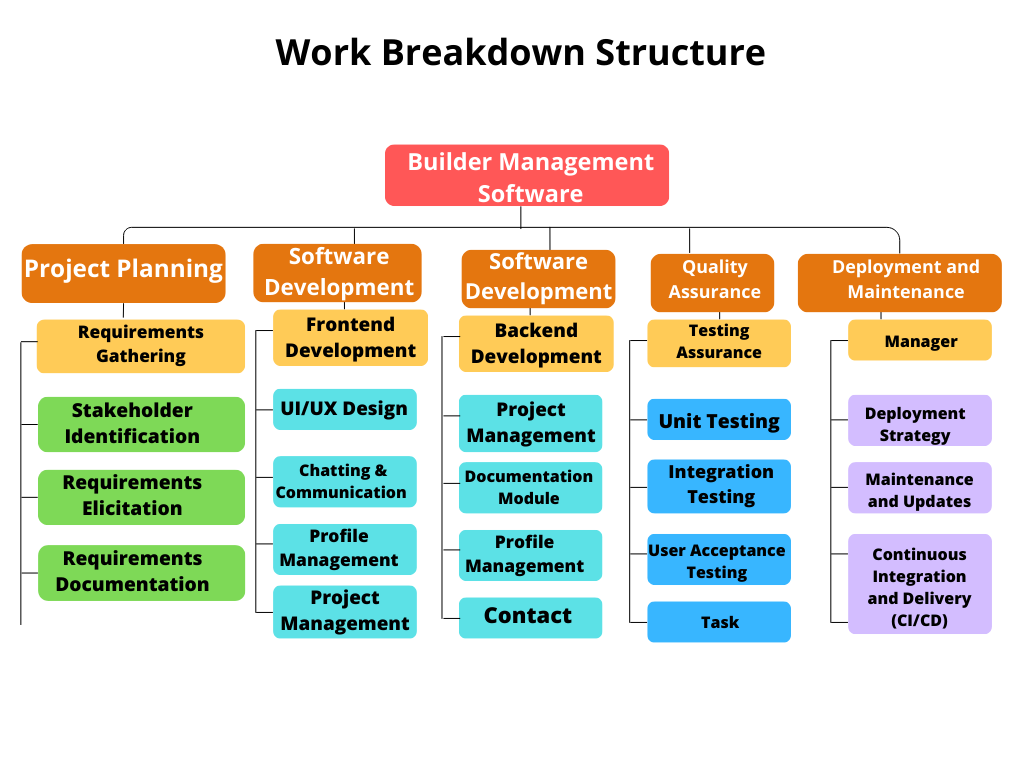
## Work Flow Diagram

A diagram of a software company

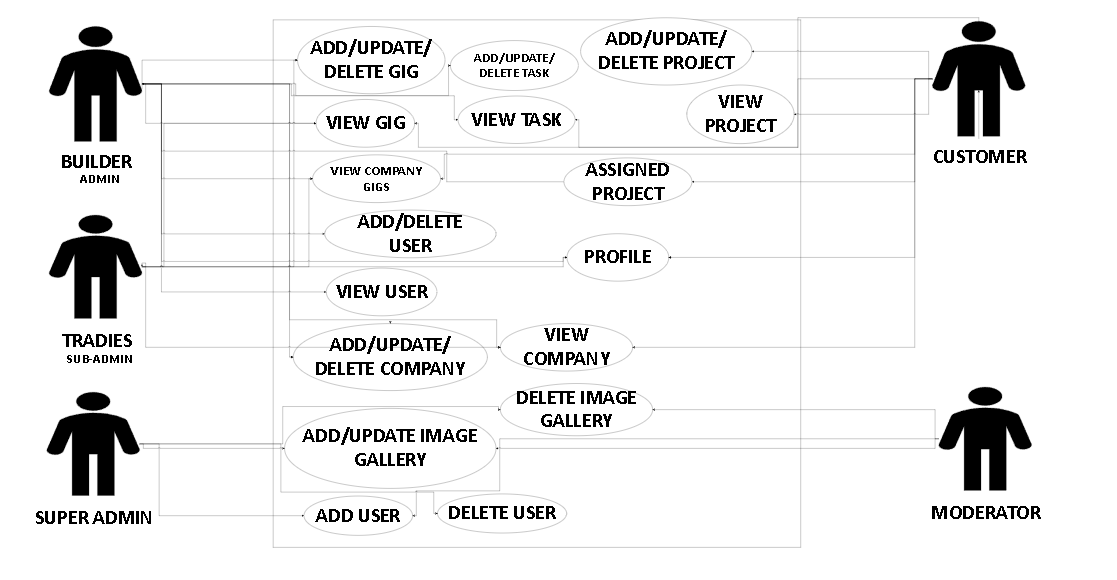
Description automatically generated

Workflow Diagram

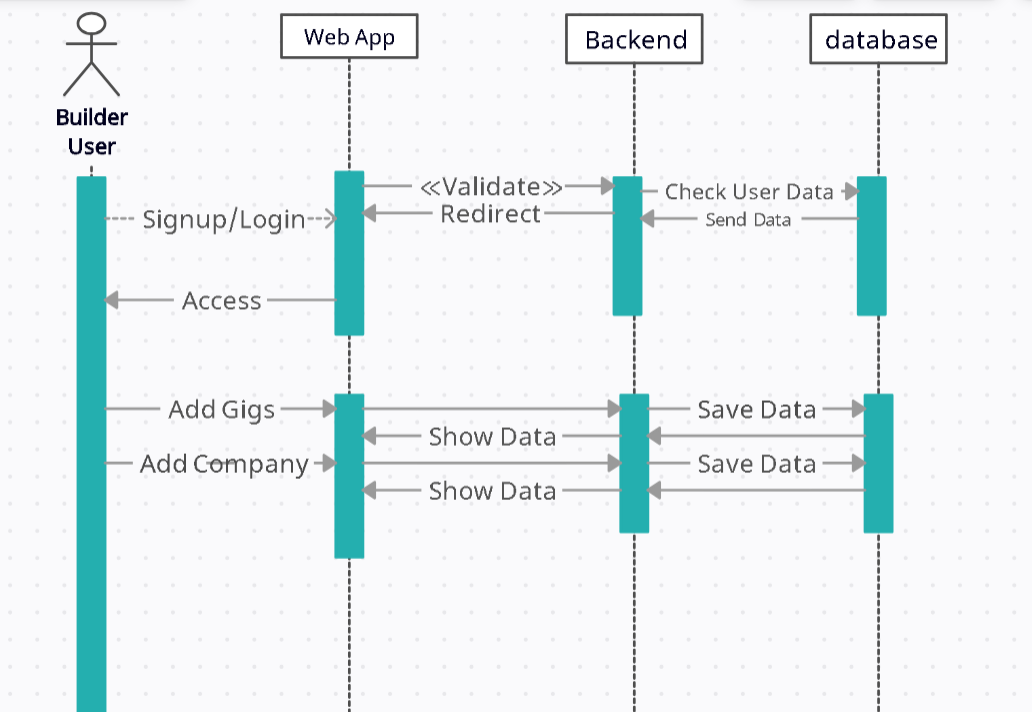
## Work Breakdown Structure



## Use case Diagram

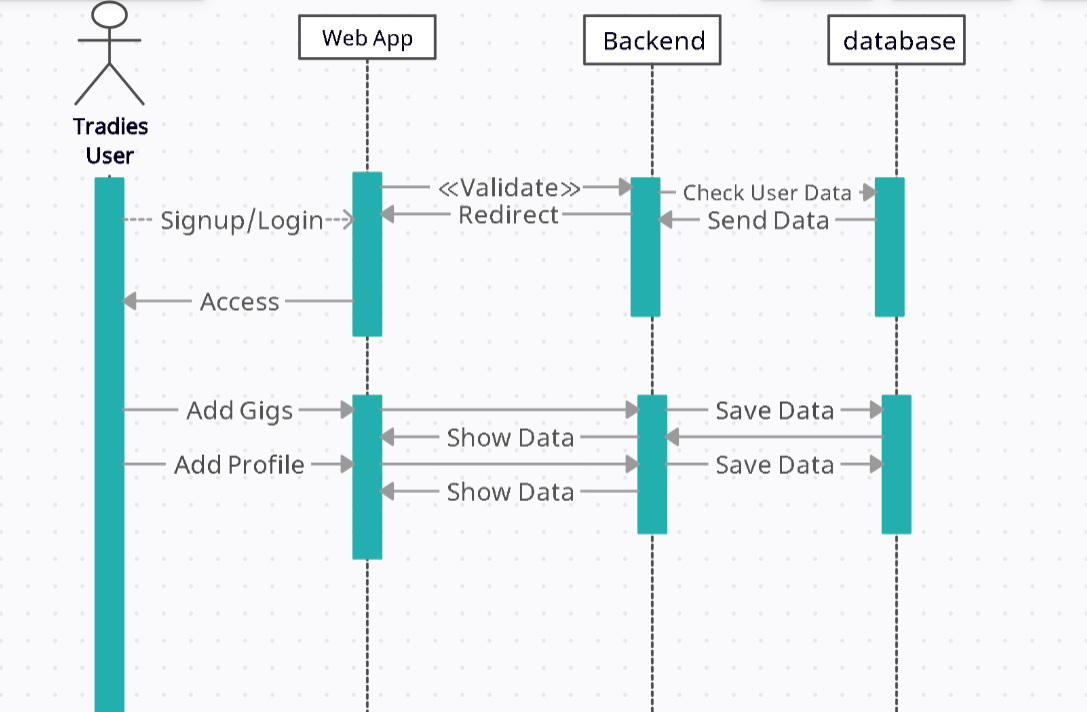


## Sequence Diagram



A diagram of a data flow

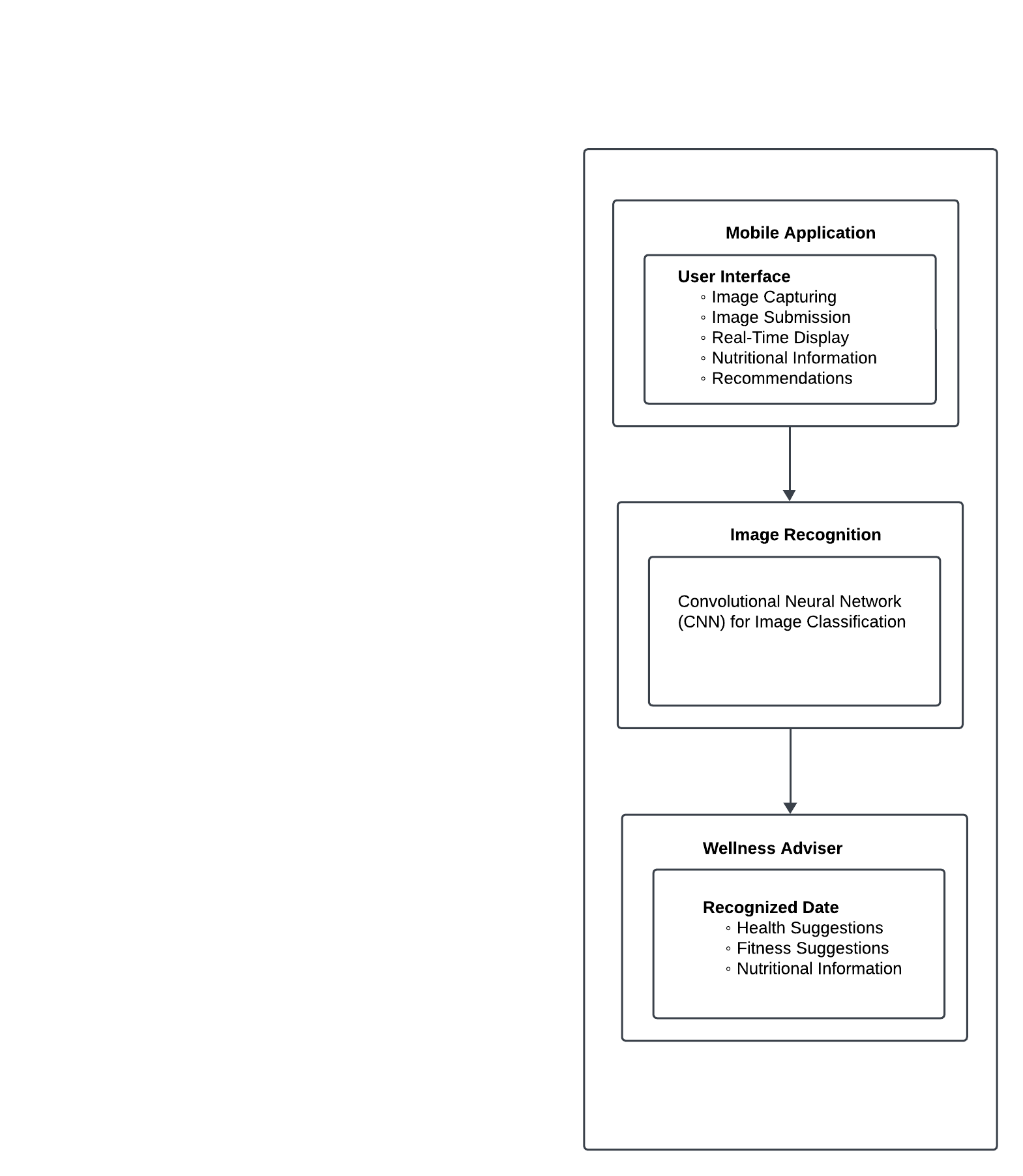
AI-generated content may be incorrect.



## Requirements Traceability Matrix

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Components/Requirements | **1**  **(Select Builder)** | **2**  **(Create Project)** | **3**  **(Track Progress)** | **4**  **(Messaging System)** | **5**  **(Task Management)** |
| 1. Real-time Messaging |  |  |  | ✓ |  |
| 2. Project Management |  | ✓ | ✓ |  |  |
| 3. Task Tracking |  |  | ✓ |  | ✓ |
| 4. Document Management |  |  | ✓ |  |  |
| 5. Builder Verification |  |  |  |  |  |
| 6. Customer Feedback |  |  |  |  |  |

## System Architectural Design



**4.9 Chosen System Architecture**

The Builder Management System follows a modular, client-server architecture, designed to separate core functionalities and ensure scalability, security, and real-time interactions.

* **Major Component Groupings:**

1. **Frontend (User Interface):**
   * Technology: React.js
   * Function: Provides an interactive and responsive interface for customers, builders, and tradies.
   * Key Features: Project dashboard, real-time messaging, task management, feedback system.
   * Internal Interface: Connects to the backend via APIs to fetch project data, messages, and tasks.
2. **Backend (Server and Database):**
   * Technology: Firebase (Real-time Database, Firestore for task scheduling, Firebase Authentication)
   * Function: Manages the logic and data flow of the system. Ensures secure data storage, real-time updates, and authentication.
   * Key Features: User authentication, real-time updates, data storage, task management.
   * Internal Interface: Communicates with the frontend to provide updates and process requests.
3. **Notification System:**
   * Technology: Firebase RealTime Database
   * Function: Sends real-time push notifications for project updates, task completions, and system alerts.
   * Internal Interface: Connects to the backend and frontend to trigger notifications.
4. **Interfaces:**

* **Internal Interfaces:** 
  + Frontend ↔ Backend: RESTful APIs using Firebase SDKs for data exchange.
  + Backend ↔ Notification System: Firebase RealTime Database for sending real-time updates to users.
* **External Interfaces:** 
  + Firebase: Handles user authentication, real-time database, and notification management.

1. **Technical Risks & Contingency Plans:**
2. **Real-time Data Syncing:** Delays or issues in real-time updates.
   * Contingency: Implement caching mechanisms and fallback strategies for offline usage.
3. **Scalability:** The system might struggle under heavy traffic or large projects.
   * Contingency: Utilize Firebase's auto-scaling capabilities and ensure optimized database structure.

**4.10 Discussion of Alternative Designs**

Several alternative designs were considered before choosing the current system architecture:

1. **Monolithic Architecture:**
   * **Description:** A single codebase with tightly coupled components for frontend, backend, and database.
   * **Reason for Rejection:** This design can lead to performance bottlenecks, difficult scalability, and harder maintenance. Decoupling components in a modular system enhances maintainability and scalability.
2. **Microservices Architecture:**
   * **Description:** The system is broken down into independent services (e.g., messaging, task management, user authentication).
   * **Reason for Rejection:** Microservices are complex and require significant overhead to manage inter-service communication, especially for a relatively small-scale application. Given the project scope, a modular architecture with integrated services is a more practical solution.

## System Interface Description

**1. Operating System Interface:**

The system will be platform-independent, utilizing modern web technologies such as HTML5, CSS3, and JavaScript (React.js). The backend relies on Firebase, which is cloud-based and does not require specific OS dependencies.

**2. Files Interface:**

The system will handle various file formats (e.g., PDFs, images for project designs, and Word documents for contracts). Files will be stored securely in Firebase's cloud storage and accessed through the frontend for download or viewing.

**3. Networking Interface:**

The system operates over HTTPS to ensure secure communication between the client (frontend) and the server (backend). Firebase real-time databases will be used for data synchronization between the client and server. All API calls between the frontend, backend, and external services (like OpenAI API) will follow RESTful principles.

**4. Libraries:**

* **Frontend Libraries:**
  + React.js for building dynamic user interfaces.
  + Axios for making API requests to the backend.
  + Firebase Realtime Database for real-time messaging features.
* **Backend Libraries:**
  + Firebase SDK for interacting with Firebase's real-time database, Firestore, and Firebase Authentication.
  + Express.js (optional) for managing API routes and backend logic if needed.

## Detailed Description of Components

### Component-1: Image Recognition System Based on Research Findings

**Responsibilities:**

* Develop a robust image recognition system using Convolutional Neural Networks (CNNs).
* Identify the best algorithm among MobileNet v, ResNet v, and VGG16 to classify date fruits accurately. The objective is to precisely classify various date fruit varieties based on their visual characteristics.

**Constraints:**

* Must meet accuracy benchmarks for date fruit classification.
* Utilizes computational resources for training and inference.

**Composition:**

* **CNN Model:** Implements the Convolutional Neural Network for image classification.
* **Training Module:** Manages the training process with labeled data.
* **Inference Module:** Handles real-time image classification based on the trained model.

**Interactions:**

* Interacts with the Mobile Application Component for receiving image data.

**Resources:**

* Computational resources for training and inference.
* Dataset for model training.

### Component-2: Mobile Application

**Responsibilities:**

* Create a user-friendly mobile application (DatePal) for Android devices.
* Enable users to capture and submit images of date fruits for classification.

**Constraints:**

* Adheres to Android OS guidelines (version 6.0 and above).
* Utilizes mobile device resources efficiently.

**Composition:**

* **User Interface Module:** Designs the visual aspects of the DatePal app.
* **Image Capture Module:** Manages the process of capturing and submitting images.
* **Communication Module:** Facilitates communication with the Image Recognition System.

**Interactions:**

* Communicates with the Image Recognition System for image classification.
* Captures and submits images based on user interactions.

**Resources:**

* Mobile device resources (CPU, memory, camera).
* Network resources for communication.

### Component 3: Real-time Classification Component

**Responsibilities:**

* Implement real-time classification for instant results upon image submission.

**Constraints:**

* Must provide immediate response to user requests.

**Composition:**

* **Real-time Classification Module:** Executes the classification algorithm for immediate results.

**Interactions:**

* Interacts with the Image Recognition System to perform real-time classification.
* Sends classification results to the Mobile Application Component.

**Resources:**

* Computational resources for real-time processing.

### Component 4: Multiclass Classification Component

**Responsibilities:**

* Train the model to classify date fruits into multiple categories, including specific varieties.

**Constraints:**

* Requires labeled data for training.
* Must update the system's knowledge base with new classifications.

**Composition:**

* **Training Module:** Manages the process of training the model.

**Interactions:**

* Interacts with the Image Recognition System for training.

**Resources:**

* Dataset for model training.

### Component 5: User-Friendly Interface Component

**Responsibilities:**

* Design an intuitive and aesthetically pleasing user interface for the DatePal app.

**Constraints:**

* Must follow UI/UX design principles.
* Ensures easy image capture and submission.

**Composition:**

* **UI Design Module:** Creates visually appealing and user-friendly interfaces.
* **Interaction Module:** Manages user interactions within the app.

**Interactions:**

* Interacts with the Mobile Application Component to render user interfaces.

**Resources:**

* Design assets for UI elements.

### Component 6: Nutritional Information Component

**Responsibilities:**

* Provide accurate nutritional details for each classified date fruit.

**Constraints:**

* Requires access to nutritional databases or data sources.

**Composition:**

* **Nutritional Information Module:** Retrieves and presents nutritional details for classified date fruits.

**Interactions:**

* Retrieves classified date fruit information from the Image Recognition System.

**Resources:**

* Nutritional databases or sources.

### Component 7: Recommendations Module Component

**Responsibilities:**

* Implement a recommendation feature suggesting date fruits based on user preferences and nutritional needs.

**Constraints:**

* Must consider user-specific preferences and nutritional criteria.

**Composition:**

* **Recommendation Engine:** Implements algorithms for generating personalized suggestions.
* **User Preferences Module:** Manages user-specific preferences.

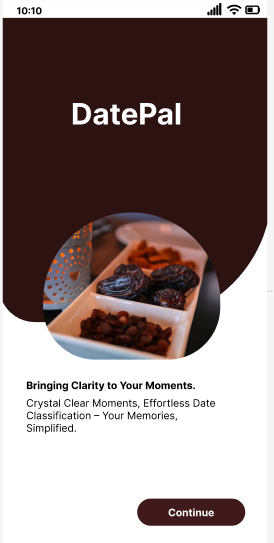
**Interactions:**

* Interacts with the Image Recognition System to receive classified date fruit information.
* Retrieves user preferences for personalized recommendations.

**Resources:**

* Computational resources for recommendation algorithms.
* User preferences data.

## User Interface Design

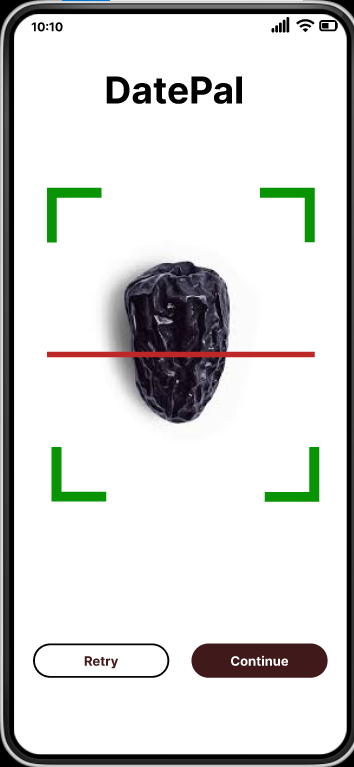
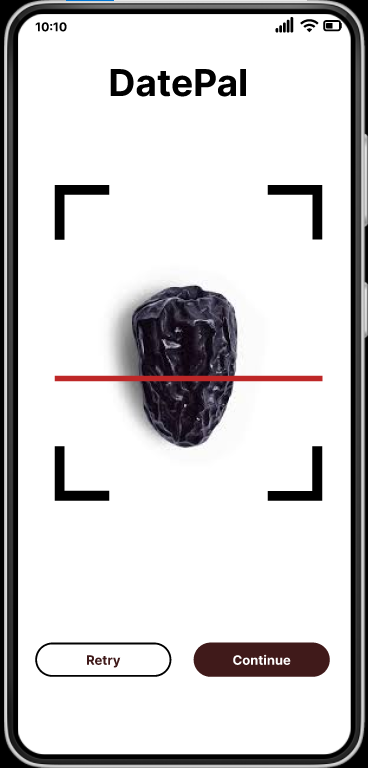


**Objects:**

* **Continue Button**

**Actions:**

**Continue Button:** Clicking this button navigates to the main scanning screen



**Objects:**

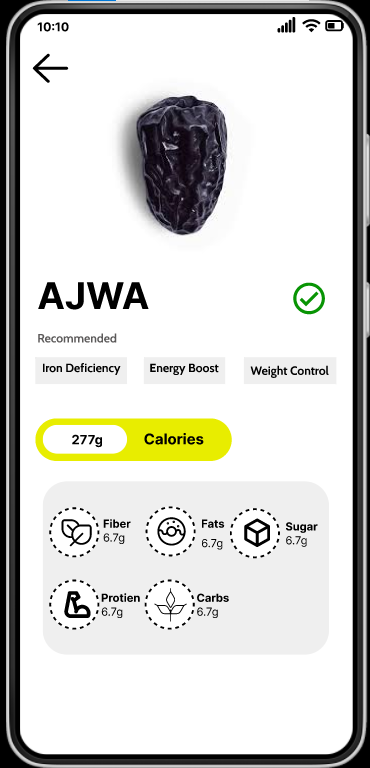
* Continue Button
* Retry Button
* Scan Box

**Actions:**

**Continue Button:** Clicking this button navigates to the result screen.

**Retry Button:** Clicking this button allows you to retry scanning.

**Scan Box:** When the scan box turns green, it indicates successful identification of the object. Click 'Continue' for results. If it turns red, it signals unsuccessful scanning. Click 'Retry' to scan again.



**Description:** This screen displays scan results, nutrition information and recommendations; no objects or actions are required.

# Implementation

## Algorithms:

### Resnet-50:

{

"cells": [

{

"cell\_type": "code",

"execution\_count": null,

"id": "demanding-trade",

"metadata": {},

"outputs": [],

"source": [

"# python librairies installation\n",

"!pip install split-folders matplotlib opencv-python spicy"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "35b34f17",

"metadata": {},

"outputs": [],

"source": [

"!pip install split-folders"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "hawaiian-bracket",

"metadata": {},

"outputs": [],

"source": [

"# display, transform, read, split ...\n",

"import numpy as np\n",

"import cv2 as cv\n",

"import os\n",

"import splitfolders\n",

"import matplotlib.pyplot as plt\n",

"\n",

"# tensorflow\n",

"import tensorflow.keras as keras\n",

"import tensorflow as tf\n",

"\n",

"# image processing\n",

"from tensorflow.keras.preprocessing import image\n",

"from tensorflow.keras.preprocessing.image import ImageDataGenerator, load\_img\n",

"\n",

"# model / neural network\n",

"from tensorflow.keras import layers\n",

"from tensorflow.keras.models import Sequential, Model\n",

"from tensorflow.keras.applications import ResNet50\n",

"from tensorflow.keras.applications.resnet50 import preprocess\_input"

]

},

{

"cell\_type": "markdown",

"id": "emotional-mention",

"metadata": {},

"source": [

"### Step 2 - Data preprocessing\n",

"\n"

]

},

{

"cell\_type": "markdown",

"id": "e72befee-de1a-4fdd-b161-1eea3eb242c3",

"metadata": {},

"source": [

"#### 2. Split data to use a ResNet model\n",

"\n"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "c0644803-a69a-46e9-8bfe-47583407a135",

"metadata": {},

"outputs": [],

"source": [

"# split data in a new folder named data-split\n",

"splitfolders.ratio(\"DateFruit Dataset 2\", output=\"data-split\", seed=1337, ratio=(0.7, 0.2, 0.1), group\_prefix=None, move=False)"

]

},

{

"cell\_type": "markdown",

"id": "6a28f282-87ab-42a0-8eb8-5ae077847947",

"metadata": {},

"source": [

"#### 3. Create Keras data generators"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "gentle-dress",

"metadata": {},

"outputs": [],

"source": [

"# Define data augmentation parameters\n",

"datagen = ImageDataGenerator(\n",

" rotation\_range=20, # Randomly rotate images by up to 20 degrees\n",

" width\_shift\_range=0.1, # Randomly shift images horizontally by up to 10% of the width\n",

" height\_shift\_range=0.1, # Randomly shift images vertically by up to 10% of the height\n",

" shear\_range=0.2, # Shear intensity (shear angle in radians)\n",

" zoom\_range=0.2, # Randomly zoom images by up to 20%\n",

" horizontal\_flip=True, # Randomly flip images horizontally\n",

" fill\_mode='nearest' # How to fill in newly created pixels after rotation or shifting\n",

")"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "361b295b",

"metadata": {},

"outputs": [],

"source": [

"class\_names = ['Ajwa', 'Aseel', 'Galaxy','Khorma', 'Nabtat Ali', 'Rutab', 'Shaishe', 'Sokari', 'Sugaey']\n"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "needed-spending",

"metadata": {},

"outputs": [],

"source": [

"# training data\n",

"train\_generator = datagen.flow\_from\_directory( \n",

" directory=\"data-split/train/\", \n",

" classes = class\_names,\n",

" target\_size=(224, 224), \n",

" batch\_size=32, \n",

" class\_mode=\"binary\", \n",

")"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "right-auditor",

"metadata": {},

"outputs": [],

"source": [

"# validation data\n",

"valid\_generator = datagen.flow\_from\_directory( \n",

" directory=\"data-split/val/\", \n",

" classes = class\_names,\n",

" target\_size=(224, 224), \n",

" batch\_size=32, \n",

" class\_mode=\"binary\", \n",

")"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "c42dd58e-7c93-4a07-a560-1b74aa341f7f",

"metadata": {},

"outputs": [],

"source": [

"# test data\n",

"test\_generator = datagen.flow\_from\_directory( \n",

" directory=\"data-split/test/\", \n",

" classes = class\_names,\n",

" target\_size=(224, 224), \n",

" batch\_size=32, \n",

" class\_mode=\"binary\", \n",

")"

]

},

{

"cell\_type": "markdown",

"id": "clear-morrison",

"metadata": {},

"source": [

"### Step 3 - Build the model\n",

"\n",

"The first step is to build the model, using \*\*ResNet50\*\*."

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "aware-numbers",

"metadata": {},

"outputs": [],

"source": [

"# ResNet50 model\n",

"resnet\_50 = ResNet50(include\_top=False, weights='imagenet', input\_shape=(224,224,3))\n",

"for layer in resnet\_50.layers:\n",

" layer.trainable = False"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "mounted-winner",

"metadata": {},

"outputs": [],

"source": [

"# build the entire model\n",

"x = resnet\_50.output\n",

"x = layers.GlobalAveragePooling2D()(x)\n",

"x = layers.Dense(512, activation='relu')(x) \n",

"x = layers.Dropout(0.5)(x)\n",

"x = layers.Dense(256, activation='relu')(x) \n",

"x = layers.Dropout(0.5)(x)\n",

"x = layers.Dense(128, activation='relu')(x) \n",

"x = layers.Dropout(0.5)(x)\n",

"x = layers.Dense(64, activation='relu')(x) \n",

"x = layers.Dropout(0.5)(x)\n",

"predictions = layers.Dense(9, activation='softmax')(x)#yaha 5 ki jagah 9 kiya hay\n",

"model = Model(inputs = resnet\_50.input, outputs = predictions)"

]

},

{

"cell\_type": "markdown",

"id": "da4962db-8a4e-434a-9d44-b8341d689c4f",

"metadata": {},

"source": [

"### Step 4 - Train the model\n",

"\n"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "protective-collect",

"metadata": {},

"outputs": [],

"source": [

"# define training function\n",

"def trainModel(model, epochs, optimizer):\n",

" batch\_size = 32\n",

" model.compile(optimizer=optimizer, loss=\"sparse\_categorical\_crossentropy\", metrics=[\"accuracy\"])\n",

" return model.fit(train\_generator, validation\_data=valid\_generator, epochs=epochs, batch\_size=batch\_size)"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "smoking-waterproof",

"metadata": {},

"outputs": [],

"source": [

"# launch the training\n",

"model\_history = trainModel(model = model, epochs = 300, optimizer = \"Adam\")"

]

},

{

"cell\_type": "markdown",

"id": "d9c56399-04cd-4e7b-9b17-f270f9d42314",

"metadata": {},

"source": [

"- Display \*\*loss\*\* curves:"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "descending-fountain",

"metadata": {},

"outputs": [],

"source": [

"loss\_train\_curve = model\_history.history[\"loss\"]\n",

"loss\_val\_curve = model\_history.history[\"val\_loss\"]\n",

"plt.plot(loss\_train\_curve, label = \"Train\")\n",

"plt.plot(loss\_val\_curve, label = \"Validation\")\n",

"plt.legend(loc = 'upper right')\n",

"plt.title(\"Loss\")\n",

"plt.show()"

]

},

{

"cell\_type": "markdown",

"id": "844fdd1f-2960-4064-97e8-767b09ab52dd",

"metadata": {},

"source": [

"- Display \*\*accuracy\*\* curves:"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "0dd67b04-29fe-4d06-9fbe-a6106e72d398",

"metadata": {},

"outputs": [],

"source": [

"acc\_train\_curve = model\_history.history[\"accuracy\"]\n",

"acc\_val\_curve = model\_history.history[\"val\_accuracy\"]\n",

"plt.plot(acc\_train\_curve, label = \"Train\")\n",

"plt.plot(acc\_val\_curve, label = \"Validation\")\n",

"plt.legend(loc = 'lower right')\n",

"plt.title(\"Accuracy\")\n",

"plt.show()"

]

},

{

"cell\_type": "markdown",

"id": "b74d7004-dba5-489c-b1a0-5c6c949f900a",

"metadata": {},

"source": [

"### Step 5 - Evaluate the model"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "2b69296b-5bc6-46ac-8b28-e3aa1e9a45b0",

"metadata": {},

"outputs": [],

"source": [

"test\_loss, test\_acc = model.evaluate(test\_generator)\n",

"print(\"The test loss is: \", test\_loss)\n",

"print(\"The best accuracy is: \", test\_acc\*100)"

]

},

{

"cell\_type": "markdown",

"id": "3476ad32-4496-4e51-b42f-a130cb35e5e8",

"metadata": {},

"source": [

"### Step 6 - Test the model on a new image"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "6626b98f-5085-4d49-9a08-e0aade03bcd7",

"metadata": {},

"outputs": [],

"source": [

"img = tf.keras.preprocessing.image.load\_img('Rhutab-Medjool\_1621x.jpg', target\_size=(224, 224))\n",

"img\_array = tf.keras.preprocessing.image.img\_to\_array(img)\n",

"img\_array = np.array([img\_array]) \n",

"img"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "e8fe14b6",

"metadata": {},

"outputs": [],

"source": [

"class\_names[3]\n"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "fca3ead9-d465-43f1-a989-2b725fecb858",

"metadata": {},

"outputs": [],

"source": [

"# generate predictions for samples\n",

"predictions = model.predict(img\_array)\n",

"print(predictions)"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "34ef2679-81de-4681-be27-d59653ab7ecd",

"metadata": {},

"outputs": [],

"source": [

"# generate argmax for predictions\n",

"class\_id = np.argmax(predictions, axis = 1)\n",

"print(class\_id)"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "3d8aa473-00b4-4dec-9f26-0c1241718ae8",

"metadata": {},

"outputs": [],

"source": [

"# transform classes number into classes name\n",

"class\_names[class\_id.item()]"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "f407379c",

"metadata": {},

"outputs": [],

"source": [

"img = tf.keras.preprocessing.image.load\_img(r'Rhutab-Medjool\_1621x.jpg', target\_size=(224, 224))\n",

"img\_array = tf.keras.preprocessing.image.img\_to\_array(img)\n",

"img\_array = np.array([img\_array]) \n",

"img"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "42d3b255",

"metadata": {},

"outputs": [],

"source": [

"\n",

"# generate predictions for samples\n",

"predictions = model.predict(img\_array)\n",

"print(predictions)\n",

"\n",

"# generate argmax for predictions\n",

"class\_id = np.argmax(predictions, axis = 1)\n",

"print(class\_id)\n",

"\n",

"# transform classes number into classes name\n",

"class\_names[class\_id.item()]"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "62d84f21",

"metadata": {},

"outputs": [],

"source": [

"def show\_image\_with\_prediction(image\_path, model, class\_names):\n",

" # Load the image\n",

" img = tf.keras.preprocessing.image.load\_img(image\_path, target\_size=(224, 224))\n",

" img\_array = tf.keras.preprocessing.image.img\_to\_array(img)\n",

" img\_array = np.array([img\_array])\n",

" \n",

" # Display the image\n",

" plt.imshow(img\_array[0].astype(\"uint8\")) # Convert to uint8 for display\n",

" plt.axis('off') # Hide axis\n",

" plt.show()\n",

" \n",

" # Generate predictions for the image\n",

" predictions = model.predict(img\_array)\n",

" \n",

" # Get the predicted class name\n",

" predicted\_class = class\_names[np.argmax(predictions)]\n",

" \n",

" # Plot the prediction probabilities\n",

" plt.figure(figsize=(8, 6))\n",

" plt.bar(class\_names, predictions[0], color='blue')\n",

" plt.xlabel('Classes')\n",

" plt.ylabel('Probability')\n",

" plt.title('Prediction Probabilities')\n",

" plt.xticks(rotation=45)\n",

" plt.show()\n",

"\n",

" print(\"Predicted class:\", predicted\_class)"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "1d12fbd4",

"metadata": {},

"outputs": [],

"source": [

"import os\n",

"\n",

"# Define the directory containing images\n",

"image\_dir = r'Rhutab-Medjool\_1621x.jpg'\n",

"\n",

"# Loop through images labeled 'Aj1' to 'Aj5'\n",

"for i in range(1, 8):\n",

" image\_name = f'Sug{i}.JPG'\n",

" image\_path = os.path.join(image\_dir, image\_name)\n",

" print(\"Processing image:\", image\_path)\n",

" show\_image\_with\_prediction(image\_path, model, class\_names)\n",

" print() # Add an empty line between each image\n"

]

},

{

"cell\_type": "markdown",

"id": "7f941624-584a-47e6-9f3a-bd4f68756459",

"metadata": {},

"source": [

"### Step 7 - Save and export the model"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "6fdafa90-8526-415e-9344-da08d5e5ed3a",

"metadata": {},

"outputs": [],

"source": [

"model.save('saved\_model/my\_model\_350\_Epoch')"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "543d75b8",

"metadata": {},

"outputs": [],

"source": [

"model.save('saved\_model/new\_model\_300\_Epoch.h5')\n"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "735b5de7-6a63-4571-aa15-acbbd0fce6dd",

"metadata": {},

"outputs": [],

"source": [

"model = tf.keras.models.load\_model('saved\_model/new\_model\_300\_Epoch.h5')\n",

"model.summary()"

]

}

],

"metadata": {

"kernelspec": {

"display\_name": "Tensorflow 2.8",

"language": "python",

"name": "python3"

},

"language\_info": {

"codemirror\_mode": {

"name": "ipython",

"version": 3

},

"file\_extension": ".py",

"mimetype": "text/x-python",

"name": "python",

"nbconvert\_exporter": "python",

"pygments\_lexer": "ipython3",

"version": "3.11.4"

}

},

"nbformat": 4,

"nbformat\_minor": 5

}

### VGG-16

{

"cells": [

{

"cell\_type": "markdown",

"metadata": {

"colab\_type": "text",

"id": "view-in-github"

},

"source": [

"<a href=\"https://colab.research.google.com/github/ShreyanshuShekhar/Flower-Recognition/blob/master/Image\_Classifier\_Project.ipynb\" target=\"\_parent\"><img src=\"https://colab.research.google.com/assets/colab-badge.svg\" alt=\"Open In Colab\"/></a>"

]

},

{

"cell\_type": "markdown",

"metadata": {

"colab\_type": "text",

"id": "5c5a\_ec87B7G"

},

"source": [

"# Developing an AI application\n",

"\n",

"Going forward, AI algorithms will be incorporated into more and more everyday applications. For example, you might want to include an image classifier in a smart phone app. To do this, you'd use a deep learning model trained on hundreds of thousands of images as part of the overall application architecture. A large part of software development in the future will be using these types of models as common parts of applications. \n",

"\n",

"In this project, you'll train an image classifier to recognize different species of flowers. You can imagine using something like this in a phone app that tells you the name of the flower your camera is looking at. In practice you'd train this classifier, then export it for use in your application. We'll be using [this dataset](http://www.robots.ox.ac.uk/~vgg/data/flowers/102/index.html) of 102 flower categories, you can see a few examples below. \n",

"\n",

"<img src='assets/Flowers.png' width=500px>\n",

"\n",

"The project is broken down into multiple steps:\n",

"\n",

"\* Load and preprocess the image dataset\n",

"\* Train the image classifier on your dataset\n",

"\* Use the trained classifier to predict image content\n",

"\n",

"We'll lead you through each part which you'll implement in Python.\n",

"\n",

"When you've completed this project, you'll have an application that can be trained on any set of labeled images. Here your network will be learning about flowers and end up as a command line application. But, what you do with your new skills depends on your imagination and effort in building a dataset. For example, imagine an app where you take a picture of a car, it tells you what the make and model is, then looks up information about it. Go build your own dataset and make something new.\n",

"\n",

"First up is importing the packages you'll need. It's good practice to keep all the imports at the beginning of your code. As you work through this notebook and find you need to import a package, make sure to add the import up here.\n",

"\n",

"Please make sure if you are running this notebook in the workspace that you have chosen GPU rather than CPU mode."

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {

"colab": {

"base\_uri": "https://localhost:8080/",

"height": 34

},

"colab\_type": "code",

"id": "Wnj-UXqkguTM",

"outputId": "886d4f05-d9f5-49b5-f902-edb83cfae777"

},

"outputs": [],

"source": [

"!git clone https://github.com/ShreyanshuShekhar/Flower-Recognition"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {

"colab": {},

"colab\_type": "code",

"id": "o60pZGC12wvo"

},

"outputs": [],

"source": [

"# installing torch on colab\n",

"# http://pytorch.org/\n",

"#from os.path import exists\n",

"#from wheel.pep425tags import get\_abbr\_impl, get\_impl\_ver, get\_abi\_tag\n",

"#platform = '{}{}-{}'.format(get\_abbr\_impl(), get\_impl\_ver(), get\_abi\_tag())\n",

"#cuda\_output = !ldconfig -p|grep cudart.so|sed -e 's/.\*\\.\\([0-9]\*\\)\\.\\([0-9]\*\\)$/cu\\1\\2/'\n",

"#accelerator = cuda\_output[0] if exists('/dev/nvidia0') else 'cpu'\n",

"\n",

"#!pip install -q http://download.pytorch.org/whl/{accelerator}/torch-0.4.1-{platform}-linux\_x86\_64.whl torchvision\n",

"#import torch"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {

"colab": {

"base\_uri": "https://localhost:8080/",

"height": 51

},

"colab\_type": "code",

"id": "--ilAuZ07B7I",

"outputId": "c1b12d45-5757-4923-f59a-e8158da3faef"

},

"outputs": [],

"source": [

"# Imports here\n",

"import torch\n",

"import PIL\n",

"import matplotlib.pyplot as plt\n",

"import matplotlib.image as mpimg\n",

"import numpy as np\n",

"import seaborn as sns\n",

"from torchvision import datasets, transforms, models\n",

"from torch import nn\n",

"from torch import optim\n",

"from collections import OrderedDict\n",

"from PIL import Image\n",

"import pathlib\n",

"import cv2\n",

"import os\n",

"%matplotlib inline\n",

"%config InlineBackend.figure\_format = 'retina'\n"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"import tensorflow as tf\n",

"from tensorflow import keras\n",

"from tensorflow.keras.models import Sequential\n",

"from tensorflow.keras.layers import Dense\n",

"from tensorflow.keras.utils import image\_dataset\_from\_directory\n",

"from tensorflow.keras import layers\n",

"from tensorflow.keras.preprocessing.image import ImageDataGenerator\n",

"import tensorflow.keras as keras\n",

"from tensorflow.keras import regularizers\n",

"from tensorflow.keras.initializers import GlorotNormal\n",

"from tensorflow.keras.metrics import TruePositives, FalsePositives, TrueNegatives, FalseNegatives, BinaryAccuracy, Precision, Recall, AUC\n"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"img\_width = 448\n",

"img\_height = 448\n",

"batch\_size = 64\n",

"color = 3\n",

"dataset\_folder = \"data\_dataset\_fyp\"\n",

"classes\_name = ['Ajwa', 'Galaxy', 'jujube', 'Nabtat Ali', 'Rutab', 'Sabzo', 'Shaishe', 'Sokari', 'Sugaey']\n"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"dataset = pathlib.Path(dataset\_folder)"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"def directory\_dataset(dataset):\n",

" folders = []\n",

" for i in dataset.iterdir():\n",

" if i.is\_dir():\n",

" folders.append(i)\n",

" return folders\n",

"\n",

"folders = directory\_dataset(dataset)\n",

"\n",

"print(folders)"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"def number\_of\_images\_in\_dataset(dataset):\n",

" images = list(dataset.glob(\"\*/\*.\*\"))\n",

" return len(images)\n",

"print(\"number of all images in dataset: {}\".format(number\_of\_images\_in\_dataset(dataset)))\n",

"\n",

"def number\_of\_images\_in\_each\_folder(folders):\n",

" for i in folders:\n",

" str\_ = \"{}: {}\".format(i, len(list(pathlib.Path(i).glob(\"\*.\*\"))))\n",

" print(str\_)\n",

"\n",

"number\_of\_images\_in\_each\_folder(folders)\n",

"\n",

"def show\_single\_image\_in\_each\_directory(folders):\n",

" for i in folders:\n",

" images = list(pathlib.Path(i).glob(\"\*.\*\"))\n",

" print(images[0])\n",

" img = cv2.imread(os.path.join(images[0]))\n",

" img = cv2.resize(img, (img\_width, img\_height))\n",

" img = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)\n",

" plt.imshow(img)\n",

" plt.show()\n",

"show\_single\_image\_in\_each\_directory(folders)"

]

},

{

"cell\_type": "markdown",

"metadata": {

"colab\_type": "text",

"id": "h2rii-Qu7B7P"

},

"source": [

"\*\*2x plots with matplotlib and the IPython Notebook.\*\* [Reference](https://gist.github.com/minrk/3301035)"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"def Blurring(image):\n",

" image = cv2.medianBlur(image,5)\n",

" return image\n",

"def image\_generator(dataset\_folder):\n",

" datagen = ImageDataGenerator( rescale=1/255, validation\_split=0.1,vertical\_flip=True ,horizontal\_flip=True,width\_shift\_range=0.2,height\_shift\_range=0.2,\n",

" rotation\_range = 5, shear\_range = 0.02,zoom\_range = 0.02, preprocessing\_function = Blurring)\n",

" train\_generator = datagen.flow\_from\_directory(\n",

" dataset\_folder,\n",

" target\_size=(img\_height, img\_width),\n",

" batch\_size=batch\_size,\n",

" class\_mode='categorical',\n",

" shuffle=True,\n",

" subset='training')\n",

" validation\_generator = datagen.flow\_from\_directory(\n",

" dataset\_folder,\n",

" target\_size=(img\_height, img\_width),\n",

" batch\_size= batch\_size,\n",

" shuffle=False,\n",

" class\_mode='categorical',\n",

" subset='validation')\n",

" \n",

" return [train\_generator, validation\_generator]\n",

"\n",

"[training\_dataset, validation\_dataset] = image\_generator(dataset)"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"# Define transforms for the training data and testing data\n",

"train\_transforms = transforms.Compose([transforms.RandomRotation(30),\n",

" transforms.RandomResizedCrop(224),\n",

" transforms.RandomHorizontalFlip(),\n",

" transforms.ToTensor(),\n",

" transforms.Normalize([0.485, 0.456, 0.406], \n",

" [0.229, 0.224, 0.225])])\n",

"\n",

"valid\_transforms = transforms.Compose([transforms.Resize(256),\n",

" transforms.CenterCrop(224),\n",

" transforms.ToTensor(),\n",

" transforms.Normalize([0.485, 0.456, 0.406], \n",

" [0.229, 0.224, 0.225])])\n",

"\n",

"# Call image\_generator to get the generators\n",

"[train\_generator, validation\_generator] = image\_generator(dataset)\n",

"\n",

"# Obtain directory paths from the generators\n",

"training\_dataset\_dir = train\_generator.directory\n",

"validation\_dataset\_dir = validation\_generator.directory\n",

"\n",

"# Define dataset objects using ImageFolder\n",

"train\_data = datasets.ImageFolder(training\_dataset\_dir, transform=train\_transforms)\n",

"valid\_data = datasets.ImageFolder(validation\_dataset\_dir, transform=valid\_transforms)\n",

"\n",

"# Create data loaders\n",

"trainloader = torch.utils.data.DataLoader(train\_data, batch\_size=50, shuffle=True)\n",

"validloader = torch.utils.data.DataLoader(valid\_data, batch\_size=50)"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"# Load a pre-trained network \n",

"model = models.vgg16(pretrained=True)\n",

"model.name = \"vgg16\"\n",

"model"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"# Freeze parameters so we don't backprop through them\n",

"for param in model.parameters():\n",

" param.requires\_grad = False"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"in\_features = 25088\n",

"mid1\_features = 4096\n",

"mid2\_features = 512\n",

"out\_features = 102\n",

"dropout = 0.2\n",

"# Define a new, untrainted feed-forward network as a classifier, using ReLU activations and dropout\n",

"classifier = nn.Sequential(OrderedDict([\n",

" ('fc1', nn.Linear(in\_features, mid1\_features, bias=True)),\n",

" ('relu1', nn.ReLU()),\n",

" ('dropout1', nn.Dropout(p=0.2)),\n",

" ('fc2', nn.Linear(mid1\_features, mid2\_features, bias=True)),\n",

" ('relu2', nn.ReLU()),\n",

" ('dropout2', nn.Dropout(p=0.2)),\n",

" ('fc3', nn.Linear(mid2\_features, out\_features, bias=True)),\n",

" ('output', nn.LogSoftmax(dim=1))\n",

" ]))\n",

" \n",

"model.classifier = classifier"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"# Device agnostic code, automatically uses CUDA if it's enabled\n",

"device = torch.device(\"cuda:0\" if torch.cuda.is\_available() else \"cpu\")\n",

"device"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"# change to device\n",

"model.to(device);"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"# Define loss and optimizer\n",

"criterion = nn.NLLLoss()\n",

"optimizer = optim.Adam(model.classifier.parameters(), lr=0.001)\n",

"\n",

"# Define deep learning method\n",

"epochs = 15\n",

"print\_every = 30 # Prints every 30 images out of batch of 50 images\n",

"steps = 0"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"def validation(model, validloader, criterion):\n",

" valid\_loss = 0\n",

" accuracy = 0\n",

" \n",

" # Set model to evaluation mode\n",

" model.eval()\n",

" \n",

" # Turn off gradients for validation\n",

" with torch.no\_grad():\n",

" for inputs, labels in validloader:\n",

" # Move inputs and labels to the appropriate device\n",

" inputs, labels = inputs.to(device), labels.to(device)\n",

" \n",

" # Forward pass\n",

" output = model(inputs)\n",

" # Calculate loss\n",

" valid\_loss += criterion(output, labels).item()\n",

" \n",

" # Calculate accuracy\n",

" ps = torch.exp(output)\n",

" equality = (labels.data == ps.max(dim=1)[1])\n",

" accuracy += equality.type(torch.FloatTensor).mean().item()\n",

" \n",

" return valid\_loss, accuracy"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"# Train the classifier layers using backpropagation using the pre-trained network to get features\n",

"print(\"Training process initializing .....\\n\")\n",

"\n",

"valid\_loss\_min = np.Inf\n",

"for e in range(epochs):\n",

" running\_loss = 0\n",

" model.train() # Technically not necessary, setting this for good measure\n",

"\n",

" for ii, (inputs, labels) in enumerate(trainloader, 1): # Start enumeration from 1 instead of 0\n",

"\n",

" inputs, labels = inputs.to(device), labels.to(device)\n",

"\n",

" optimizer.zero\_grad()\n",

"\n",

" # Forward and backward passes\n",

" outputs = model.forward(inputs)\n",

" loss = criterion(outputs, labels)\n",

" loss.backward()\n",

" optimizer.step()\n",

"\n",

" # Update running training loss\n",

" running\_loss += loss.item()\n",

"\n",

" # Print training statistics after each iteration\n",

" print(\"Epoch: {}/{} \\tIteration: {}/{} \\tLoss: {:.6f}\".format(\n",

" e + 1,\n",

" epochs,\n",

" ii,\n",

" len(trainloader),\n",

" loss.item()\n",

" ), end='\\r') # Print without newline\n",

" \n",

" # After each epoch, print validation statistics\n",

" model.eval()\n",

" with torch.no\_grad():\n",

" valid\_loss, accuracy = validation(model, validloader, criterion)\n",

" valid\_loss = valid\_loss / len(validloader.dataset)\n",

" print(\"\\nEpoch: {}/{} \\tValidation Loss: {:.6f}\".format(e + 1, epochs, valid\_loss))\n",

"\n",

" # Save model if validation loss has decreased\n",

" if valid\_loss <= valid\_loss\_min:\n",

" print('Validation loss decreased ({:.6f} --> {:.6f}). Saving model ...'.format(\n",

" valid\_loss\_min,\n",

" valid\_loss))\n",

" torch.save({'model\_state\_dict': model.state\_dict()}, 'model.pt')\n",

" valid\_loss\_min = valid\_loss\n",

"\n",

"print(\"Training complete!\")\n"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"# TODO: Do validation on the test set\n",

"# Do validation on the test set\n",

"correct = 0\n",

"total = 0\n",

"with torch.no\_grad():\n",

" model.eval()\n",

" for data in trainloader:\n",

" images, labels = data\n",

" images, labels = images.to(device), labels.to(device)\n",

" outputs = model(images)\n",

" \_, predicted = torch.max(outputs.data, 1)\n",

" total += labels.size(0)\n",

" correct += (predicted == labels).sum().item()\n",

"\n",

"print('Accuracy achieved by the network on test images is: %d%%' % (100 \* correct / total))"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"# TODO: Save the checkpoint \n",

"# Create this `class\_to\_idx` attribute quickly\n",

"model.class\_to\_idx = train\_data.class\_to\_idx"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"checkpointprev = torch.load('model.pt')\n",

"checkpoint = {'architecture': model.name,\n",

" 'classifier': model.classifier,\n",

" 'class\_to\_idx': model.class\_to\_idx,\n",

" 'state\_dict':model.state\_dict()}\n",

" \n",

"# print(checkpoint['state\_dict'])\n",

"torch.save(model.state\_dict(), 'my\_checkpoint.pt')"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"for param\_tensor in model.state\_dict():\n",

" print(param\_tensor, \"\\t\", model.state\_dict()[param\_tensor].size())"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"# TODO: Write a function that loads a checkpoint and rebuilds the model\n",

"# Write a function that loads a checkpoint and rebuilds the model\n",

"def load\_checkpoint():\n",

" \"\"\"\n",

" Loads deep learning model checkpoint.\n",

" \"\"\"\n",

" \n",

" # Load the saved file\n",

" checkpoint = torch.load(\"'my\_checkpoint.pth\")\n",

" \n",

" # Download pretrained model\n",

" model = models.vgg16(pretrained=True);\n",

" \n",

" # Freeze parameters so we don't backprop through them\n",

" for param in model.parameters(): param.requires\_grad = False\n",

" \n",

" # Load stuff from checkpoint\n",

" model.class\_to\_idx = checkpoint['class\_to\_idx']\n",

" model.classifier = checkpoint['classifier']\n",

" model.load\_state\_dict(checkpoint['state\_dict'])\n",

"\n",

" \n",

" return model"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"def process\_image(image):\n",

" ''' Scales, crops, and normalizes a PIL image for a PyTorch model,\n",

" returns a NumPy array\n",

" '''\n",

"\n",

" test\_image = PIL.Image.open(image)\n",

"\n",

" # Resize the image while preserving aspect ratio\n",

" size = (256, 256)\n",

" test\_image.thumbnail(size)\n",

"\n",

" # Find pixels to crop on to create a 224x224 image\n",

" left = (256 - 224) / 2\n",

" top = (256 - 224) / 2\n",

" right = (256 + 224) / 2\n",

" bottom = (256 + 224) / 2\n",

"\n",

" # Crop the image\n",

" test\_image = test\_image.crop((left, top, right, bottom))\n",

"\n",

" # Convert to NumPy array and normalize\n",

" np\_image = np.array(test\_image) / 255.0\n",

" mean = np.array([0.485, 0.456, 0.406])\n",

" std = np.array([0.229, 0.224, 0.225])\n",

" np\_image = (np\_image - mean) / std\n",

"\n",

" # Transpose the color channels\n",

" np\_image = np\_image.transpose((2, 0, 1))\n",

"\n",

" return np\_image"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"def imshow(image, ax=None, title=None):\n",

" if ax is None:\n",

" fig, ax = plt.subplots()\n",

" \n",

" # PyTorch tensors assume the color channel is the first dimension\n",

" # but matplotlib assumes is the third dimension\n",

" image = image.transpose((1, 2, 0))\n",

" \n",

" # Undo preprocessing\n",

" mean = np.array([0.485, 0.456, 0.406])\n",

" std = np.array([0.229, 0.224, 0.225])\n",

" image = std \* image + mean\n",

" \n",

" # Image needs to be clipped between 0 and 1 or it looks like noise when displayed\n",

" image = np.clip(image, 0, 1)\n",

" \n",

" ax.imshow(image)\n",

" \n",

" return ax"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"# Test functions on an example\n",

"imshow(process\_image(\"AD.JPG\"));"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"def predict(image\_path, model, top\_k=5):\n",

" ''' Predict the class (or classes) of an image using a trained deep learning model.\n",

" \n",

" image\_path: string. Path to image, directly to image and not to folder.\n",

" model: pytorch neural network.\n",

" top\_k: integer. The top K classes to be calculated\n",

" \n",

" returns top\_probabilities(k), top\_labels\n",

" '''\n",

" \n",

" # No need for GPU on this part (just causes problems)\n",

" model.to(\"cpu\")\n",

" \n",

" # Set model to evaluate\n",

" model.eval()\n",

"\n",

" # Preprocess the image\n",

" processed\_image = process\_image(image\_path)\n",

" \n",

" # Convert image from numpy to torch\n",

" torch\_image = torch.from\_numpy(np.expand\_dims(processed\_image, axis=0)).type(torch.FloatTensor)\n",

"\n",

" # Find probabilities (results) by passing through the function (note the log softmax means that its on a log scale)\n",

" log\_probs = model.forward(torch\_image)\n",

"\n",

" # Convert to linear scale\n",

" linear\_probs = torch.exp(log\_probs)\n",

"\n",

" # Find the top K results\n",

" top\_probs, top\_labels = linear\_probs.topk(top\_k)\n",

" \n",

" # Detach all of the details\n",

" top\_probs = np.array(top\_probs.detach())[0]\n",

" top\_labels = np.array(top\_labels.detach())[0]\n",

" \n",

" # Convert labels to classes\n",

" idx\_to\_class = {val: key for key, val in model.class\_to\_idx.items()}\n",

" top\_labels = [idx\_to\_class[lab] for lab in top\_labels]\n",

" \n",

" return top\_probs, top\_labels"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"# TODO: Display an image along with the top 5 classes\n",

"# Define image path\n",

"image\_path = \"Testing\\Sug\\Sug7.JPG\"\n",

"\n",

"# Set up plot\n",

"plt.figure(figsize=(6, 10))\n",

"ax = plt.subplot(2, 1, 1)\n",

"\n",

"# Plot flower\n",

"img = process\_image(image\_path)\n",

"imshow(img, ax)\n",

"\n",

"# Make prediction\n",

"probs, labs = predict(image\_path, model)\n",

"\n",

"# Plot bar chart\n",

"plt.subplot(2, 1, 2)\n",

"sns.barplot(x=probs, y=labs, color=sns.color\_palette()[0])\n",

"plt.show()"

]

}

],

"metadata": {

"accelerator": "GPU",

"colab": {

"collapsed\_sections": [],

"include\_colab\_link": true,

"name": "Image Classifier Project.ipynb",

"provenance": [],

"toc\_visible": true,

"version": "0.3.2"

},

"kernelspec": {

"display\_name": "Python 3",

"name": "python3"

},

"language\_info": {

"codemirror\_mode": {

"name": "ipython",

"version": 3

},

"file\_extension": ".py",

"mimetype": "text/x-python",

"name": "python",

"nbconvert\_exporter": "python",

"pygments\_lexer": "ipython3",

"version": "3.11.4"

}

},

"nbformat": 4,

"nbformat\_minor": 0

}

## Frontend:

### Classification Component:

import React, { useState, useEffect } from "react";

import { View, Text, Button, StyleSheet, Dimensions } from "react-native";

import { Camera, CameraView } from "expo-camera";

import { StatusBar } from "expo-status-bar";

import CustomButton from "../components/button";

import \* as FileSystem from "expo-file-system";

import \* as MediaLibrary from "expo-media-library";

import { CameraType } from "expo-camera/build/legacy/Camera.types";

import Toast from 'react-native-toast-message';

var ws;

const Classification = ({ navigation }) => {

const [hasCameraPermission, setHasCameraPermission] = useState(null);

const [camera, setCamera] = useState(null);

const [imagePath, setImagePath] = useState(null);

const [hasMediaLibraryPermission, setHasMediaLibraryPermission] =

useState(true);

const [isButtonDisabled, setIsButtonDisabled] = useState(false);

useEffect(() => {

connectWebSocket();

const requestPermissions = async () => {

try {

const { status: cameraPermission } =

await Camera.requestCameraPermissionsAsync();

setHasCameraPermission(cameraPermission === "granted");

console.log("Camera permission granted:", cameraPermission);

const { status: mediaLibraryPermission } =

await MediaLibrary.requestPermissionsAsync();

setHasMediaLibraryPermission(mediaLibraryPermission === "granted");

} catch (error) {

console.error("Error asking for permissions:", error);

}

};

requestPermissions();

}, []);

// let data = '';

function connectWebSocket() {

// ws = new WebSocket("ws://192.168.43.61:8000/ws/dates\_image/");

ws = new WebSocket("ws://35.175.126.2:8001/ws/dates\_image/");

ws.onopen = function(event) {

console.log("WebSocket connection opened.");

};

ws.onmessage = function(event) {

console.log("Received message from server:", event.data);

var data = JSON.parse(event.data);

if (data && data.name) {

//yaha se aap ne navigate krna hay

console.log("Record details:", data);

Toast.show({

type: "success" ,

text1: "Date classified. Loading results..." ,

position: "bottom",

bottomOffset: 150, // Adjust this value to move the toast slightly above the bottom

});

setTimeout(() => {

navigation.navigate("Result", { data });

}, 0);

}

else if(data && data.error){

Toast.show({

type: 'error',

text1: 'Date not found. Try again.',

position: "bottom",

bottomOffset: 150, // Adjust this value to move the toast slightly above the bottom

});

console.log("No record found.");

}

setIsButtonDisabled(false); // Enable the button after response

};

ws.onclose = function(event) {

Toast.show({

type: 'error',

text1: 'No connection found.',

position: "bottom",

bottomOffset: 60, // Adjust this value to move the toast slightly above the bottom

});

console.log("WebSocket connection closed.");

setIsButtonDisabled(false); // Enable the button after response

};

};

const sendImage = async (base64Image) => {

try {

if (ws.readyState === WebSocket.OPEN) {

// Send the base64 image data to the WebSocket server

ws.send(base64Image);

console.log("Image sent to WebSocket server");

} else {

console.log("WebSocket connection not open");

}

} catch (error) {

console.error("Error sending image to WebSocket server:", error);

}

};

if (hasCameraPermission === null) {

return (

<View>

<Text>Requesting camera permission...</Text>

</View>

);

}

if (hasCameraPermission === false) {

return (

<View>

<Text>No access to camera</Text>

</View>

);

}

// Function to make a permanent copy of the image

const makePermanentCopy = async (temporaryUri) => {

try {

// Generate a new permanent file path

const permanentUri = FileSystem.documentDirectory + "permanent\_photo.jpg";

// Copy the image from the temporary location to the permanent location

await FileSystem.copyAsync({

from: temporaryUri,

to: permanentUri,

});

console.log("Permanent copy of the image created at:", permanentUri);

return permanentUri; // Return the permanent file path

} catch (error) {

console.error("Error making permanent copy of the image:", error);

return null;

}

};

const takePicture = async () => {

if (!hasCameraPermission) {

console.log("No camera permission");

return;

}

if (camera) {

try {

const photo = await camera.takePictureAsync({

quality: 0.5,

base64: true,

});

setIsButtonDisabled(true);

sendImage(photo.base64);

const temporaryUri = FileSystem.documentDirectory + "photo.jpg";

await FileSystem.writeAsStringAsync(temporaryUri, photo.base64, {

encoding: FileSystem.EncodingType.Base64,

});

setImagePath(temporaryUri);

console.log("Image saved at temporary location:", temporaryUri);

if (!hasMediaLibraryPermission) {

console.log("No media library permission");

return;

}

// Save image to the device's gallery

// await MediaLibrary.saveToLibraryAsync(temporaryUri);

// console.log("Image saved to gallery");

} catch (error) {

console.error("Error taking or saving picture:", error);

}

}

};

const windowWidth = Dimensions.get("window").width;

const windowHeight = Dimensions.get("window").height;

const cameraSize = Math.min(windowWidth, windowHeight) \* 0.8; // Set camera size to 80% of the minimum dimension

// const data = {

// \_id: {

// $oid: "662c0ad5bcb34a024b3caced",

// },

// name: "Ajwa",

// calories: 27,

// recommended\_for: [

// "Energy booster",

// "Natural sweetener",

// "Dietary fiber source",

// ],

// nutrition\_info: {

// fats: 0,

// fiber: 0.3,

// carbohydrates: 6.5,

// sugar: 5.5,

// protein: 0.3,

// },

// };

return (

<View style={styles.container}>

<StatusBar style="dark" />

<Text style={{ fontWeight: "bold", fontSize: 40, alignSelf: "center" }}>

DatePal

</Text>

{hasCameraPermission ? (

<View

style={{

height: cameraSize + 50,

width: cameraSize,

alignSelf: "center",

}}

>

<CameraView

style={{ flex: 1 }}

ref={(ref) => {

if (ref) {

setCamera(ref); // Set camera ref when available

console.log("Camera ready");

} else {

console.log("Camera ref is null"); // Log when camera ref is null

}

}}

type={CameraType.back}

ratio="4:3"

/>

</View>

) : (

<Text style={{ alignSelf: "center", marginTop: 20 }}>

Waiting for camera to be ready...

</Text>

)}

<View

style={{

flexDirection: "row",

alignItems: "flex-end",

justifyContent: "space-evenly",

marginBottom: 40,

}}

>

<Toast ref={(ref) => Toast.setRef(ref)} />

{/\* <CustomButton

title={"Retry"}

color={"white"}

textColor={"black"}

onPress={() => {

navigation.navigate("result");

}}

/> \*/}

<CustomButton

title={isButtonDisabled?"Waiting...":"Take Picture"}

color={"#2D1212"}

textColor={"white"}

onPress={takePicture}

disabled={isButtonDisabled}

/>

</View>

</View>

);

};

const styles = StyleSheet.create({

container: {

flex: 1,

paddingTop: 50, // Add padding to the top to prevent text from starting at the top

backgroundColor: "#fff",

gap: 80,

flexDirection: "column",

justifyContent: "space-between",

},

});

export default Classification;

### Recommendation Component

import React from "react";

import { StyleSheet, Text, View, Dimensions } from "react-native";

import Carousel from 'react-native-snap-carousel';

export default function RecSlider({recomendation}) {

const screenWidth = Dimensions.get('window').width;

const data = recomendation.map((item, index) => ({ id: index, recommendation: item }));

const renderItem = ({ item }) => (

<View style={styles.slide}>

<Text style={styles.recomendation}>{item.recommendation}</Text>

</View>

);

return (

<View style={styles.container}>

<Carousel

data={data}

renderItem={renderItem}

sliderWidth={screenWidth}

itemWidth={screenWidth \* 0.5}

layout={'default'}

firstItem={1} // Start from the leftmost item

/>

</View>

);

}

const styles = StyleSheet.create({

container: {

alignItems: 'flex-start',

justifyContent: 'flex-start',

paddingHorizontal:2,

},

slide: {

backgroundColor: '#ebedec',

borderRadius: 5,

padding: 4,

alignItems: 'center',

justifyContent: 'center',

width:'auto'

},

recomendation: {

fontSize: 16,

},

});

### Nutrition Card Component

import React from "react";

import { StyleSheet, View, Text, Dimensions } from "react-native";

import { Ionicons } from "@expo/vector-icons";

export default function NutritionCard({icon, name, calorie}) {

return (

<View style={styles.container}>

<View style={{ flexDirection: "row", gap: 5 }}>

<View style={styles.iconContainer}>

{icon}

</View>

<View style={{ flexDirection: "column", justifyContent: "center" }}>

<View>

<Text>{name}</Text>

</View>

<View>

<Text>{calorie}</Text>

</View>

</View>

</View>

</View>

);

}

const styles = StyleSheet.create({

container: {

flexDirection: "row",

justifyContent: "flex-start",

alignContent: "center",

gap: 10,

},

iconContainer: {

borderWidth: 2,

borderStyle: "dashed",

borderRadius: 40,

padding: 5,

},

});

### Home Screen:

import { StatusBar } from 'expo-status-bar';

import { StyleSheet, Text, View, Dimensions, Button, Image } from 'react-native';

import CustomButton from '../components/button';

import {

responsiveHeight,

responsiveWidth,

responsiveFontSize

} from "react-native-responsive-dimensions";

export default function HomeScreen({ navigation }) {

const windowHeight = Dimensions.get('window').height;

const topContainerHeight = windowHeight \* 0.6;

const bottomContainerHeight = windowHeight \* 0.4;

return (

<View style={{ flex: 1, backgroundColor: '#fffafa' }}>

<StatusBar style="light" />

<View style={[styles.container, { height: topContainerHeight }]}>

<Text style={styles.heading}>DatePal</Text>

<View style={styles.imageContainer}>

<Image source={require('../assets/homeImg.png')} style={styles.image} />

</View>

</View>

<View style={[styles.bottomContainer, { height: bottomContainerHeight }]}>

<View style={{alignSelf: "flex-start"}}>

<Text style={{ color: 'black', fontWeight: 'bold', paddingLeft: 30 , fontSize: responsiveFontSize(2)}}>Bringing Clarity to Your Moments</Text>

<Text style={{ color: 'black', paddingLeft: 30, paddingRight: 20,fontSize: responsiveFontSize(2) }}>Crystal Clear Moments, Effortless Date Classification – Your Memories, Simplified.</Text>

</View>

<View style={{alignSelf:'flex-end',marginRight:responsiveWidth(5), marginBottom:responsiveHeight(2)}}>

<CustomButton title={"Continue"} color={"#2D1212"} textColor={"white"} onPress={() => navigation.navigate('classification')} />

</View>

</View>

</View>

);

}

const styles = StyleSheet.create({

container: {

backgroundColor: '#2D1212',

alignItems: 'center',

justifyContent: 'flex-start',

borderBottomRightRadius: 150,

borderBottomLeftRadius: 50,

},

heading: {

color: "white",

paddingTop: responsiveHeight(10),

fontSize: responsiveFontSize(5),

fontWeight: 'bold'

},

imageContainer: {

alignItems: 'flex-start',

justifyContent: 'center',

flex: 1, // Takes remaining vertical space

},

image: {

alignSelf: 'center', // Center the image horizontally

},

bottomContainer: {

backgroundColor: '#fffafa',

alignItems: 'center',

justifyContent: 'space-between',

paddingTop: 20

},

buttonContainer: {

alignSelf: 'flex-end',

marginVertical: 100,

marginRight: 20,

backgroundColor: "#2D1212",

borderRadius: 30,

paddingHorizontal: 20,

paddingVertical: 5,

}

});

### Calories Box:

import React from 'react';

import { StyleSheet, View ,Text, Dimensions} from 'react-native';

const screenWidth = Dimensions.get('window').width;

const containerWidth = screenWidth \* 0.5; // Adjust the percentage as needed

export default function CalorieBox({calories}) {

return (

<View style={[styles.container, { backgroundColor: "yellow" }]}>

<View style={styles.textContainer}>

<Text style={styles.text}>{calories}g</Text>

</View>

<View style={styles.numberContainer}>

<Text style={styles.number}>Calories</Text>

</View>

</View>

);

}

const styles = StyleSheet.create({

container: {

marginTop:10,

width: containerWidth,

flexDirection: 'row',

backgroundColor: 'yellow',

borderRadius: 20,

borderWidth: 1,

borderColor: 'white',

paddingHorizontal: 10,

paddingVertical: 5,

},

textContainer: {

flex: 1,

marginRight: 5,

borderWidth: 1,

borderRadius: 20,

borderColor: 'white',

backgroundColor:'white',

padding: 5,

alignItems:'center'

},

text: {

fontSize: 16,

fontWeight:'bold'

},

numberContainer: {

padding: 5,

backgroundColor: 'yellow',

},

number: {

fontSize: 16,

fontWeight:"bold",

},

});

## Backend Code:

### Date Image Consumer Code:

from channels.generic.websocket import AsyncWebsocketConsumer

from PIL import Image

import numpy as np

import io

import json

import tensorflow as tf

import base64

from pymongo import MongoClient

import tensorflow as tf

class\_names = ['Ajwa', 'Aseel', 'Galaxy','Khorma', 'Nabtat Ali', 'Rutab', 'Shaishe', 'Sokari', 'Sugaey']

class1\_names = ['Dates', 'None Dates']

class DateImageConsumer(AsyncWebsocketConsumer):

async def connect(self):

await self.accept()

await self.send(text\_data=json.dumps({'status': 'connected'}))

async def receive(self, text\_data=None, bytes\_data=None):

try:

decoded\_data = base64.b64decode(text\_data) # Extract base64 data part

image = Image.open(io.BytesIO(decoded\_data))

resized\_image = image.resize((224, 224))

array = np.array(resized\_image)

if array.ndim == 2:

array = np.stack((array,) \* 3, axis=-1)

input\_data = np.expand\_dims(array, axis=0)

model1 = tf.keras.models.load\_model('BinaryModel.h5')

# model = tf.keras.layers.TFSMLayer("my\_model\_300\_Epoch", call\_endpoint='serving\_default')

predictions = model1.predict(input\_data)

predicted\_class\_index = np.argmax(predictions)

predicted\_class\_name = class1\_names[predicted\_class\_index]

if predicted\_class\_name=='Dates':

model = tf.keras.models.load\_model('new\_model.h5')

# model = tf.keras.layers.TFSMLayer("my\_model\_300\_Epoch", call\_endpoint='serving\_default')

predictions2 = model.predict(input\_data)

predicted2\_class\_index = np.argmax(predictions2)

predicted\_class2\_name = class\_names[predicted2\_class\_index]

uri = "mongodb+srv://amjad5azx:amjad1290@cluster0.imlsmty.mongodb.net/"

client = MongoClient(uri)

if client is not None:

print("Connected to MongoDB Atlas successfully.")

else:

print("Failed to connect to MongoDB Atlas.")

db = client["DateDB"]

collection\_names = db.list\_collection\_names()

print("Available collections:", collection\_names)

collection=db[collection\_names[0]]

record = collection.find\_one({"name": predicted\_class2\_name}, {"\_id": 0})

# record = {"name": predicted\_class\_name}

if record:

await self.send(text\_data=json.dumps(record))

else:

await self.send(text\_data=json.dumps({'prediction': predicted\_class\_name, 'record': None}))

else:

print(f"Object is not a date")

await self.send(text\_data=json.dumps({'error': "Object is not a date"}))

except Exception as e:

print(f"Error processing image data: {e}")

await self.send(text\_data=json.dumps({'error': str(e)}))

async def disconnect(self, close\_code):

print("WebSocket connection closed.")

### DockerFile:

FROM python:3

RUN pip3 install django daphne channels pillow numpy tensorflow pymongo django-cors-headers channels\_redis

COPY . .

RUN python manage.py migrate

CMD ["daphne", "-b", "0.0.0.0", "-p", "8001", "fyp\_backend.asgi:application"]

### Asgi.py:

"""

ASGI config for fyp\_backend project.

It exposes the ASGI callable as a module-level variable named ``application``.

For more information on this file, see

https://docs.djangoproject.com/en/4.2/howto/deployment/asgi/

"""

import os

from channels.routing import ProtocolTypeRouter,URLRouter

from django.urls import path

from myapp.consumers.date\_image\_consumers import \*

from myapp.consumers.date\_name\_consumers import \*

from myapp.consumers.all\_dates import \*

from myapp.consumers.test\_consumer import \*

from django.core.asgi import get\_asgi\_application

os.environ.setdefault('DJANGO\_SETTINGS\_MODULE', 'fyp\_backend.settings')

application = get\_asgi\_application()

ws\_patterns=[

path('ws/dates\_image/',DateImageConsumer.as\_asgi()),

]

application=ProtocolTypeRouter({

'websocket':URLRouter(ws\_patterns)

})

# Software Test Document

## System Overview:

The software system to be tested is a mobile application named "Date Fruit Classification". This application utilizes convolutional neural network (CNN) algorithms to classify images of dates into specific categories. The app employs three CNN algorithms for analysis: MobileNetV2, ResNet50, and VGG16. After evaluating the performance of these algorithms, ResNet50 was selected for the final implementation due to its superior results.

The system is divided into two main models:

### First Model (Date Identification):

**Algorithm:** ResNet50

**Dataset:** Contains two files, one with images of dates and the other with non-date images.

**Functionality:** This model is integrated with the camera. When a user captures an image, the model activates to determine if the image is of a date. If the image is identified as a non-date, the user is notified accordingly. If the image is recognized as a date, it is forwarded to the second model for further classification.

### Second Model (Date Classification):

**Algorithm:** ResNet50

**Dataset:** Comprises nine different types of dates.

**Functionality:** This model classifies the date into one of the nine categories. Once classified, the application displays the name of the date type, nutritional information, and recommendations regarding which groups of people the date type is suitable for.

## System Testing:

The system testing encompasses two main phases:

### Algorithm Testing:

* Evaluate the accuracy of all three algorithms (MobileNetV2, ResNet50, VGG16) in date identification and classification.
* Determine the most effective algorithm, which was found to be ResNet50.

### WebSocket Testing:

* Verify that WebSockets successfully retrieve data from the database and transmit it to the frontend.
* Ensure real-time communication and data transfer integrity between the backend and frontend components.

### Software Version:

The version(s) of the software to be tested will be specified as follows:

* Initial development version: v1.0
* Post-algorithm evaluation version: v1.1
* Final implementation with ResNet50: v2.0

## Test Approach

The overall approach to testing the Date Fruit Classification mobile application includes a structured plan to ensure comprehensive coverage of all major features and functionalities. The testing approach is divided into several phases to address different components of the system. The major activities, techniques, and tools involved in the testing process are outlined below:

### Major Testing Activities and Techniques

#### Model Accuracy Testing:

**Objective:** Verify the accuracy of each model (MobileNetV2, ResNet50, VGG16) in classifying date images.

**Approach:**

* Input a diverse set of date images into each model.
* Compare the classification results against the ground truth labels.
* Measure accuracy, precision, recall, and F1-score for each model.

**Tools:** Python scripts, TensorFlow, Keras, and evaluation metrics libraries.

**Estimated Time:** 2 weeks

#### First Model (Date Identification) Testing:

**Objective:** Test the first model's ability to correctly identify whether an input image is a date or not.

**Approach:**

* Input a mix of date and non-date images into the model.
* Observe the model's predictions and verify correctness.
* Record false positives and false negatives for further analysis.

**Tools:** Python scripts, TensorFlow, Keras, image datasets.

**Estimated Time:** 1 week

#### WebSocket Functionality Testing:

**Objective:** Ensure that WebSockets correctly retrieve data from the database and transmit it to the frontend.

**Approach:**

* Simulate data requests from the frontend.
* Monitor the WebSocket communication to verify data integrity and real-time updates.
* Validate error handling and reconnection mechanisms.

**Tools:** For testing web socket, web socket king client is used. But the problem is that this tool has capability to test text but not image data. Therefore, we create simple html page for this and test web socket for image classification.

**Estimated Time:** 1 week

#### Frontend Data Retrieval Testing:

**Objective:** Test the frontend to ensure data is successfully retrieved from the database and displayed correctly.

**Approach:**

* Perform end-to-end testing from image capture to date classification display.
* Validate the nutritional information and recommendations displayed for classified dates.
* Test various user scenarios to ensure robustness.

**Tools:** manual testing.

**Estimated Time:** 1 week

### Major Feature Groups and Combinations

#### Algorithm Performance:

**Feature Group:** MobileNetV2, ResNet50, VGG16

**Approach:** Focus on evaluating each model's performance with the same dataset for consistency.

**Techniques:** Cross-validation, confusion matrix analysis.

#### Date Identification:

**Feature Group:** First Model (ResNet50 for Date vs. Non-date)

**Approach:** Test with varied images to ensure high sensitivity and specificity.

**Techniques:** Binary classification metrics, robustness tests with different lighting and angles.

#### Date Classification:

**Feature Group:** Second Model (ResNet50 for 9 Types of Dates)

**Approach:** Ensure accurate classification into the correct date type and validate display of information.

**Techniques:** Multi-class classification metrics, user interface testing.

#### Real-time Data Handling:

**Feature Group:** Web Socket Communication

**Approach:** Validate seamless real-time data flow from backend to frontend.

**Techniques:** Stress testing, latency measurement, error scenario testing.

#### User Interface:

**Feature Group:** Frontend Data Display

**Approach:** Test the user interface for correct data presentation and usability.

**Techniques:** functional testing.

## Test Plan:

The Test Plan for the Date Fruit Classification mobile application outlines the scope, approach, resources, and schedule for the testing activities. It details the items being tested, the features to be tested, the specific testing tasks, and the personnel responsible for each task.

### Scope

The scope of testing includes:

* Evaluating the accuracy and performance of three CNN algorithms (MobileNetV2, ResNet50, VGG16) for date classification.
* Testing the functionality of the first model to identify whether an image is of a date or not.
* Verifying the second model’s ability to classify dates into nine specific types.
* Ensuring that WebSockets correctly retrieve and transmit data between the backend and frontend.
* Validating that the frontend displays the retrieved data accurately and user interfaces are functional and user-friendly.
* Approach
* The overall approach to testing is structured and methodical, focusing on manual testing techniques to ensure comprehensive coverage and reliable results.

### Algorithm Performance Testing:

* Use a controlled dataset to evaluate the accuracy, precision, recall, and F1-score of each CNN algorithm.
* Perform cross-validation and confusion matrix analysis.

### Date Identification Testing:

* Input a mix of date and non-date images to the first model.
* Monitor and log predictions to calculate false positive and false negative rates.

### Date Classification Testing:

* Test the second model with images of the nine date types.
* Validate the accuracy of classifications and the correctness of displayed nutritional information and recommendations.

### WebSocket Testing:

* Simulate real-time data requests from the frontend.
* Monitor WebSocket communication for data integrity and performance under different scenarios.

### Frontend Data Retrieval Testing:

* Perform end-to-end testing from image capture to data display.
* Ensure the frontend accurately retrieves and displays data from the backend.
* Conduct usability testing to ensure the interface is user-friendly.

### Resources:

**Testing Tools:** Python, TensorFlow, Keras.

|  |  |  |
| --- | --- | --- |
| **Task** | **Duration** | **Responsible Person** |
| Algorithm Performance Testing | 2 Weeks | Jamshed Ali and Muhammad Amjad |
| Date Identification Testing | 1 Week | Jamshed Ali and Muhammad Amjad |
| Date Classification Testing | 1 Week | Jamshed Ali and Muhammad Amjad |
| Web Socket Testing | 1 Week | Muhammad Amjad |
| Frontend Data Retrieval Testing | 1 Week | Muhammad Amjad |

Test Plan

### Items Being Tested:

* MobileNetV2, ResNet50, VGG16: Performance in classifying date images.
* **First Model (Date Identification):** Accuracy in distinguishing date vs. non-date images.
* **Second Model (Date Classification):** Accuracy in classifying nine types of dates.
* **WebSockets:** Real-time data retrieval and transmission functionality.
* **Frontend:** Data display accuracy and user interface functionality.
* **Model Accuracy:** Performance metrics of MobileNetV2, ResNet50, and VGG16.
* **Date Identification:** Accuracy of the first model in identifying date images.
* **Date Classification:** Accuracy of the second model in classifying different types of dates.
* **Real-time Data Handling:** Functionality of WebSockets for data retrieval.
* **User Interface:** Correctness of data display and usability of the frontend interface.

## Features to be Tested

### Features to be Tested

#### Algorithm Performance:

**Feature Description:** Evaluate the accuracy and performance of MobileNetV2, ResNet50, and VGG16 algorithms.

**Test Cases:**

* TC-001: Test accuracy of MobileNetV2 on the dataset.
* TC-002: Test accuracy of ResNet50 on the dataset.
* TC-003: Test accuracy of VGG16 on the dataset.
* TC-004: Compare precision, recall, and F1-score across all three algorithms.

**Software Version:** v1.0, v1.1

#### Date Identification (First Model):

**Feature Description:** Identify whether an input image is of a date or not using ResNet50.

**Test Cases:**

* **TC-005:** Test model accuracy with date images.
* **TC-006:** Test model accuracy with non-date images.
* **TC-007:** Measure false positive rate.
* **TC-008:** Measure false negative rate.

**Software Version:** v1.1, v2.0

#### Date Classification (Second Model):

**Feature Description:** Classify images of dates into nine specific types using ResNet50.

**Test Cases:**

* **TC-009:** Test classification accuracy for type 1 dates.
* **TC-010:** Test classification accuracy for type 2 dates.
* **TC-011:** Test classification accuracy for type 3 dates.
* **TC-012:** Test classification accuracy for type 4 dates.
* **TC-013:** Test classification accuracy for type 5 dates.
* **TC-014:** Test classification accuracy for type 6 dates.
* **TC-015:** Test classification accuracy for type 7 dates.
* **TC-016:** Test classification accuracy for type 8 dates.
* **TC-017:** Test classification accuracy for type 9 dates.
* **TC-018:** Validate nutritional information display for classified dates.
* **TC-019:** Validate recommendations for classified dates.

**Software Version:** v2.0

#### WebSocket Functionality:

**Feature Description:** Verify WebSockets retrieve data from the database and transmit it to the frontend.

**Test Cases**:

**TC-020:** Test real-time data retrieval via WebSockets.

**TC-021:** Test data transmission to frontend via WebSockets.

**TC-022:** Validate WebSocket error handling and reconnection.

**Software Version:** v2.0

#### Frontend Data Retrieval and Display:

**Feature Description:** Ensure the frontend accurately retrieves and displays data from the database.

**Test Cases:**

* **TC-023:** Validate end-to-end data flow from image capture to display.
* **TC-024:** Test frontend display of classified date type.
* **TC-025:** Test frontend display of nutritional information.
* **TC-026:** Test frontend display of recommendations.
* **TC-027:** Validate usability of the user interface.

**Software Version:** v2.0

### Combination of Features to be Tested

#### Algorithm Performance and Date Identification:

**Feature Combination:** Ensure the best-performing algorithm (ResNet50) is used for date identification.

**Test Cases:**

* **TC-028:** Confirm accuracy of ResNet50 in date identification.
* **TC-029:** Compare performance metrics of date identification across all algorithms.

**Software Version:** v1.1, v2.0

#### Date Identification and Date Classification:

**Feature Combination:** Verify seamless integration between the first model (date identification) and the second model (date classification).

**Test Cases:**

* **TC-030:** Test transition from date identification to date classification.
* **TC-031:** Validate accuracy of combined identification and classification process.

**Software Version:** v2.0

#### WebSocket Functionality and Frontend Display:

**Feature Combination:** Ensure WebSocket communication correctly updates the frontend with classification results.

**Test Cases:**

* **TC-032:** Validate real-time update of classification results on the frontend.

## Test Cases

The following test case specifications refine the test approach, identify the features to be tested, and detail the procedures required to accomplish the testing. Each test case includes the feature pass/fail criteria, documents the actual values used for input, and specifies the anticipated outputs.

### Test Case Specifications

#### TC-001: Test Accuracy of MobileNetV2 on the Dataset

**Feature:** Algorithm Performance

**Procedure:**

* Load the MobileNetV2 model.
* Input a set of date images into the model.
* Compare the model's predictions with the ground truth labels.
* Calculate accuracy, precision, recall, and F1-score.

**Input Values:** A dataset of labeled date images.

**Anticipated Output:** Accuracy > 80%, precision, recall, and F1-score > 75%.

**Pass/Fail Criteria:** The model's performance metrics meet or exceed the anticipated thresholds.

#### TC-002: Test Accuracy of ResNet50 on the Dataset

**Feature:** Algorithm Performance

**Procedure:**

* Load the ResNet50 model.
* Input a set of date images into the model.
* Compare the model's predictions with the ground truth labels.
* Calculate accuracy, precision, recall, and F1-score.

**Input Values:** A dataset of labeled date images.

**Anticipated Output:** Accuracy > 90%, precision, recall, and F1-score > 85%.

**Pass/Fail Criteria:** The model's performance metrics meet or exceed the anticipated thresholds.

#### TC-003: Test Accuracy of VGG16 on the Dataset

**Feature:** Algorithm Performance

**Procedure:**

* Load the VGG16 model.
* Input a set of date images into the model.
* Compare the model's predictions with the ground truth labels.
* Calculate accuracy, precision, recall, and F1-score.

**Input Values:** A dataset of labeled date images.

**Anticipated Output:** Accuracy > 85%, precision, recall, and F1-score > 80%.

**Pass/Fail Criteria:** The model's performance metrics meet or exceed the anticipated thresholds.

#### TC-004: Compare Performance Metrics Across All Algorithms

**Feature:** Algorithm Performance

**Procedure:**

* Load MobileNetV2, ResNet50, and VGG16 models.
* Input the same dataset of date images into each model.
* Compare accuracy, precision, recall, and F1-score for each model.

**Input Values:** A dataset of labeled date images.

**Anticipated Output:** Performance metrics for each model, with ResNet50 showing the highest accuracy.

**Pass/Fail Criteria:** Accurate comparison data showing ResNet50 outperforms the other models.

#### TC-005: Test Model Accuracy with Date Images (First Model)

**Feature:** Date Identification

**Procedure:**

* Load the first ResNet50 model.
* Input a set of images containing dates.
* Verify the model correctly identifies each image as a date.

**Input Values:** A dataset of images of dates.

**Anticipated Output:** Model correctly identifies at least 95% of the date images.

**Pass/Fail Criteria:** The model's identification accuracy meets or exceeds 95%.

#### TC-006: Test Model Accuracy with Non-Date Images (First Model)

**Feature:** Date Identification

**Procedure:**

* Load the first ResNet50 model.
* Input a set of images not containing dates.
* Verify the model correctly identifies each image as a non-date.

**Input Values:** A dataset of non-date images.

**Anticipated Output:** Model correctly identifies at least 90% of the non-date images.

**Pass/Fail Criteria:** The model's identification accuracy meets or exceeds 90%.

#### TC-007: Measure False Positive Rate (First Model)

**Feature:** Date Identification

**Procedure:**

* Load the first ResNet50 model.
* Input a mix of date and non-date images.
* Count the number of non-date images incorrectly identified as dates.

**Input Values:** A mixed dataset of date and non-date images.

**Anticipated Output:** False positive rate < 5%.

**Pass/Fail Criteria:** The false positive rate is below the specified threshold.

#### TC-008: Measure False Negative Rate (First Model)

**Feature:** Date Identification

**Procedure:**

* Load the first ResNet50 model.
* Input a mix of date and non-date images.
* Count the number of date images incorrectly identified as non-dates.

**Input Values:** A mixed dataset of date and non-date images.

**Anticipated Output:** False negative rate < 10%.

**Pass/Fail Criteria:** The false negative rate is below the specified threshold.

#### TC-009: Test Classification Accuracy for Type 1 Dates (Second Model)

**Feature:** Date Classification

**Procedure:**

* Load the second ResNet50 model.
* Input a set of Type 1 date images.
* Verify the model correctly classifies each image as Type 1.

**Input Values:** A dataset of Type 1 date images.

**Anticipated Output:** Model correctly classifies at least 90% of the Type 1 date images.

**Pass/Fail Criteria:** The model's classification accuracy meets or exceeds 90%.

#### TC-010 to TC-017: Test Classification Accuracy for Other Date Types (Second Model)

**Feature:** Date Classification

**Procedure:** Similar to TC-009, but for each of the remaining date types (Type 2 to Type 9).

**Input Values:** Separate datasets for each date type.

**Anticipated Output:** Model correctly classifies at least 90% of images for each date type.

**Pass/Fail Criteria:** The model's classification accuracy meets or exceeds 90% for each type.

#### TC-018: Validate Nutritional Information Display for Classified Dates

**Feature:** Date Classification

**Procedure:**

* Classify an image of a date.
* Verify that the nutritional information displayed is correct and relevant to the classified date type.

**Input Values:** Images of various date types.

**Anticipated Output:** Correct nutritional information displayed for each date type.

**Pass/Fail Criteria:** Nutritional information is accurate and relevant.

#### TC-019: Validate Recommendations for Classified Dates

**Feature:** Date Classification

**Procedure:**

* Classify an image of a date.
* Verify that the recommendations displayed are correct and relevant to the classified date type.

**Input Values:** Images of various date types.

**Anticipated Output:** Correct recommendations displayed for each date type.

**Pass/Fail Criteria:** Recommendations are accurate and relevant.

#### TC-020: Test Real-Time Data Retrieval via WebSockets

**Feature:** WebSocket Functionality

**Procedure:**

* Simulate data requests from the frontend.
* Verify that the WebSocket retrieves data in real-time.

**Input Values:** Simulated data requests.

**Anticipated Output:** Real-time data retrieval without significant delay.

**Pass/Fail Criteria:** Data is retrieved in real-time.

#### TC-021: Test Data Transmission to Frontend via WebSockets

**Feature:** WebSocket Functionality

**Procedure:**

* Simulate data updates from the backend.
* Verify that the frontend receives the updates via WebSockets.

**Input Values:** Simulated data updates.

**Anticipated Output:** Data updates transmitted successfully to the frontend.

**Pass/Fail Criteria:** Data is transmitted accurately and promptly.

#### TC-022: Validate WebSocket Error Handling and Reconnection

**Feature:** WebSocket Functionality

**Procedure:**

* Simulate WebSocket connection interruptions.
* Verify error handling and automatic reconnection.

**Input Values:** Simulated connection interruptions.

**Anticipated Output:** Error handling works correctly, and reconnection is automatic.

**Pass/Fail Criteria:** WebSocket handles errors and reconnects as expected.

#### TC-023: Validate End-to-End Data Flow from Image Capture to Display

**Feature:** Frontend Data Retrieval and Display

**Procedure:**

* Capture an image using the mobile app.
* Follow the data flow from capture to display of classification results.

**Input Values:** Various date images captured using the mobile app.

**Anticipated Output:** Accurate and timely display of classification results.

**Pass/Fail Criteria:** End-to-end data flow is seamless and accurate.

#### TC-024 to TC-026: Test Frontend Display of Classification Results, Nutritional Information, and Recommendations

**Feature:** Frontend Data Retrieval and Display

**Procedure:**

* Trigger data retrieval via WebSockets.
* Verify display of classification results, nutritional information, and recommendations.
* Input Values: Classification results, nutritional information, and recommendations for various date types.

**Anticipated Output:** Accurate display of all information.

**Pass/Fail Criteria:** Information is displayed correctly and is user-friendly.

#### TC-027: Validate Usability of the User Interface

**Feature:** Frontend Data Retrieval and Display

**Procedure:**

* Conduct usability testing with a sample of users.
* Gather feedback on the interface design, navigation, and user experience.

**Input Values:** User interactions and feedback.

**Anticipated Output:** Positive user feedback and high usability scores.

**Pass/Fail Criteria:** Usability meets the specified standards.

### Test Case: TC-007

#### Purpose

* **Version:** v1.1
* **Test Items:** First Model (Date Identification)
* **Features:**
  + Identify whether an input image is of a date or not using ResNet50.
  + **Associated Requirements:**
    - SRS Section 3.1.1: Image Classification
    - SRS Section 3.1.2: Date Identification

#### Inputs

* **Version:** v1.1
* **Test Items:** First Model (Date Identification)
* **Features:**
  + Identify whether an input image is of a date or not using ResNet50.
  + **Associated Requirements:**
    - SRS Section 3.1.1: Image Classification
    - SRS Section 3.1.2: Date Identification

#### Expected Outputs & Pass/Fail Criteria

**Expected Outputs:**

* Predictions from the ResNet50 model for each input image.

**Pass/Fail Criteria:**

* The false positive rate should be less than 5%.
* The false negative rate should be less than 10%.

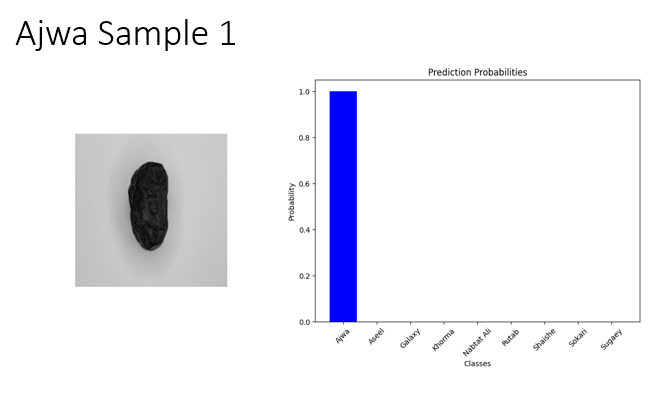
#### Test Procedure

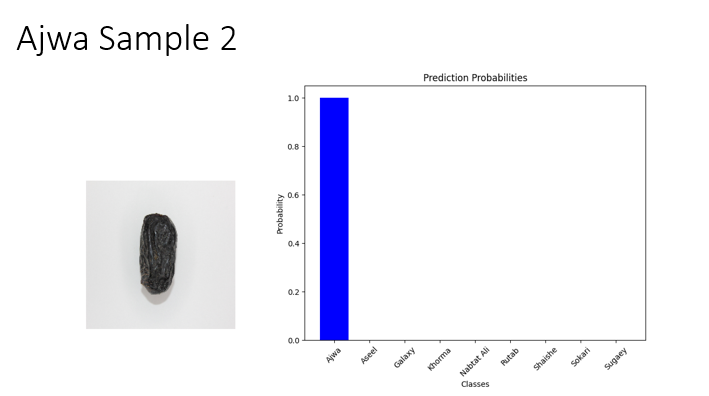
1. Load the ResNet50 model.
2. Input the image dataset into the model.
3. Record the model's predictions for each image.
4. Compare the predictions with the ground truth labels.
5. Calculate the false positive rate (FP) and false negative rate (FN).
6. Determine if the FP rate is less than 5% and the FN rate is less than 10%.
7. If both criteria are met, the test case passes; otherwise, it fails.

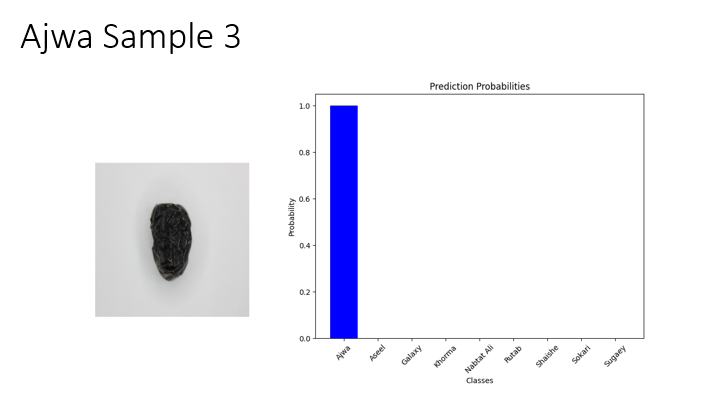
# Appendix A: Test Results

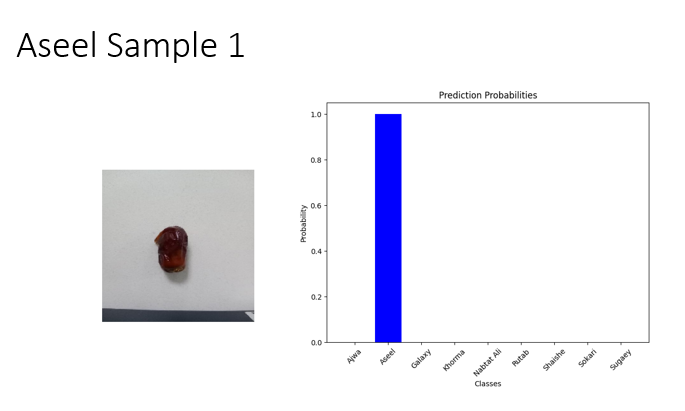
### Test Results

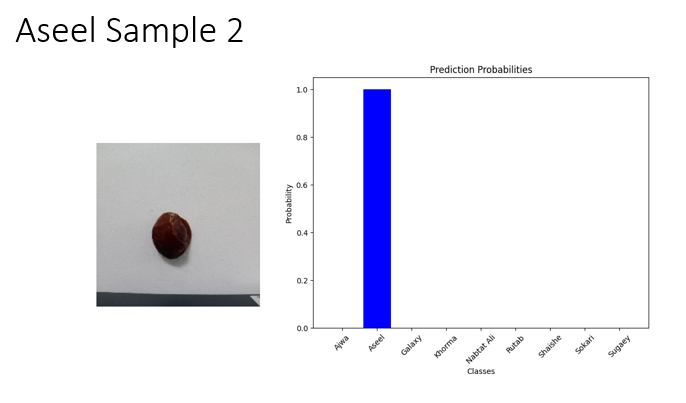
#### Resnet-50 Test Results:

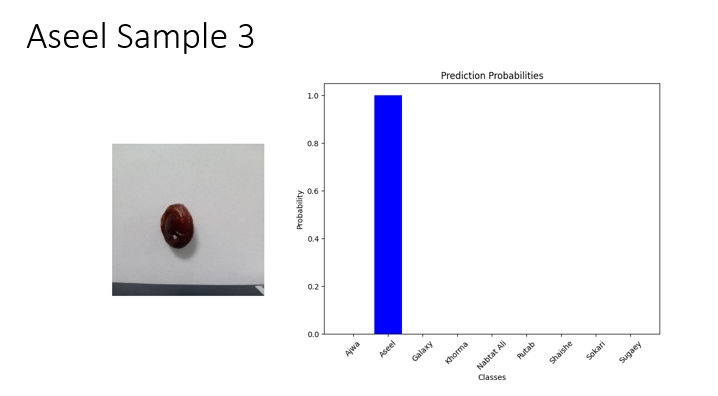


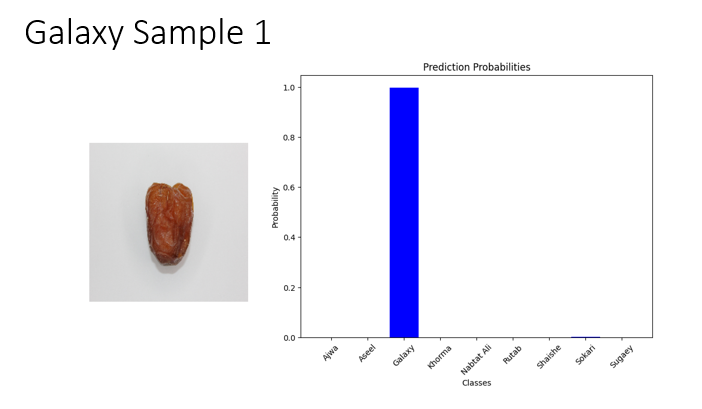


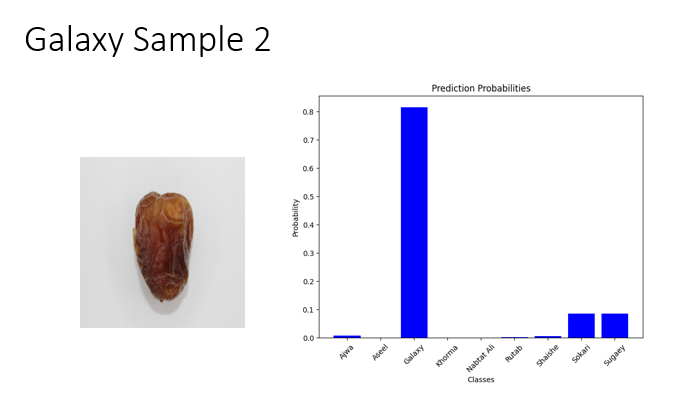


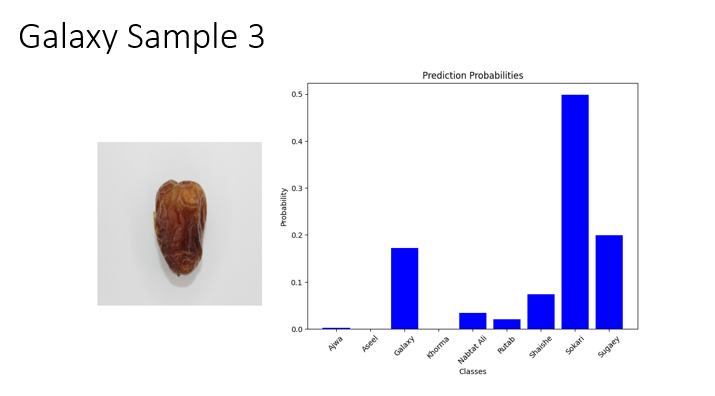


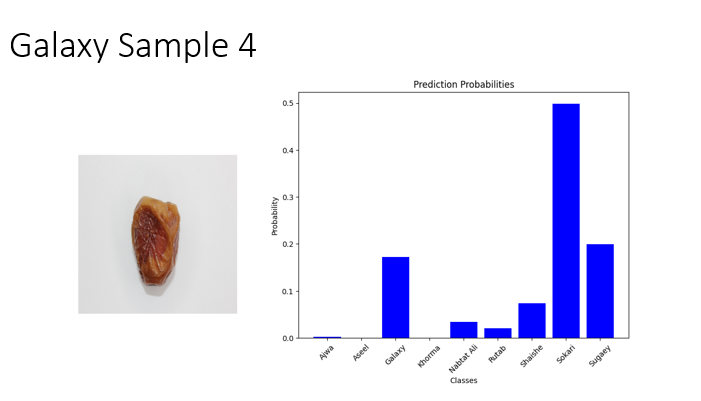


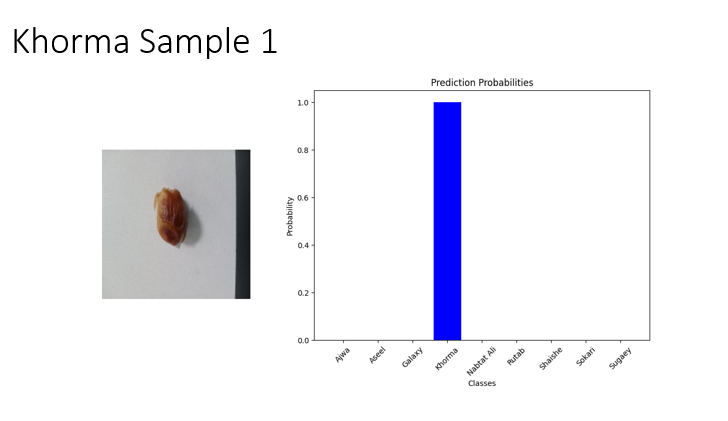


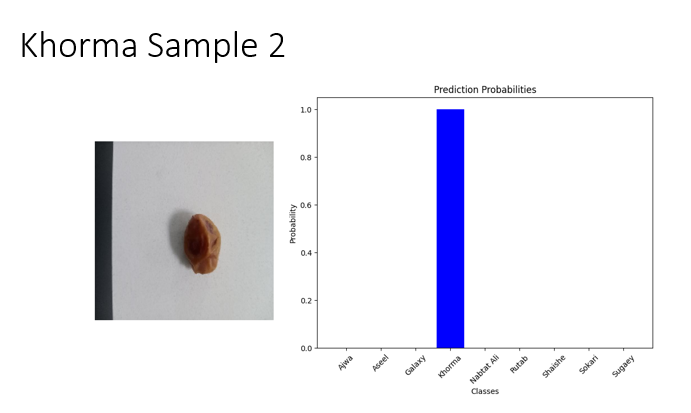


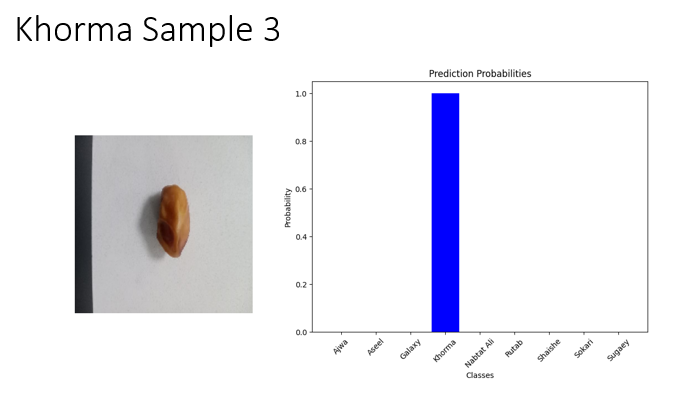


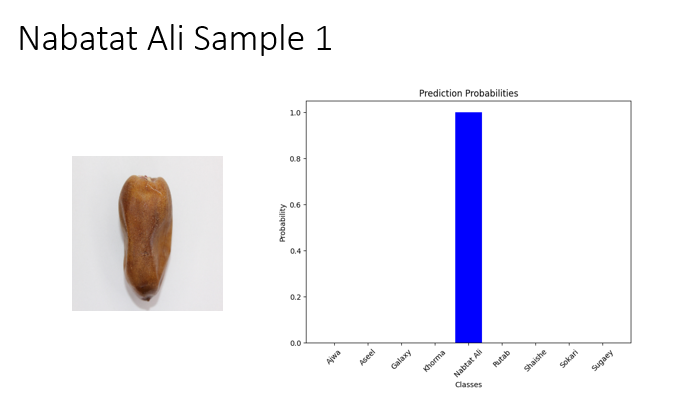


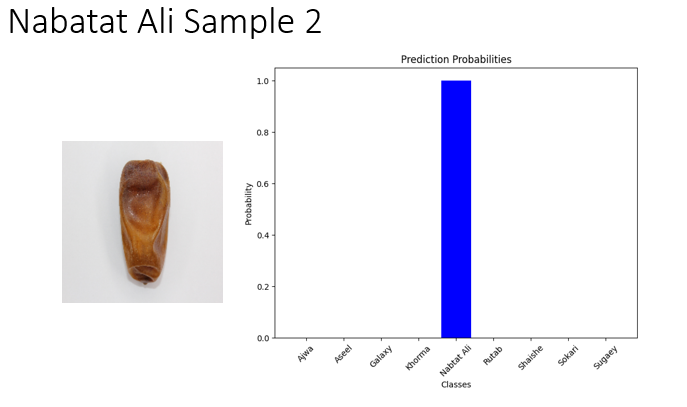


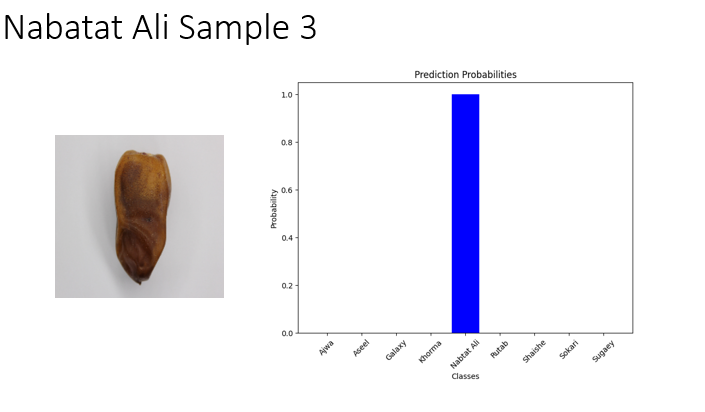


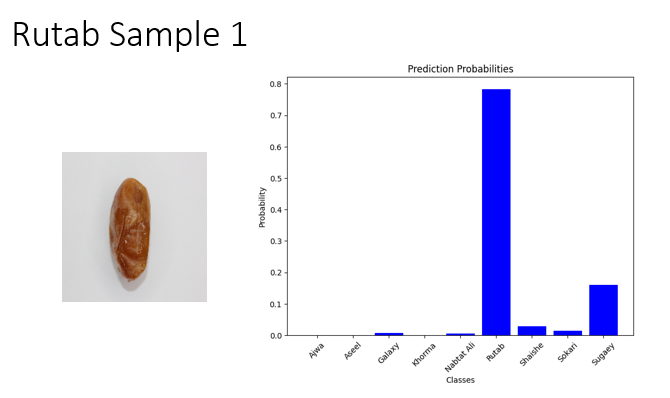


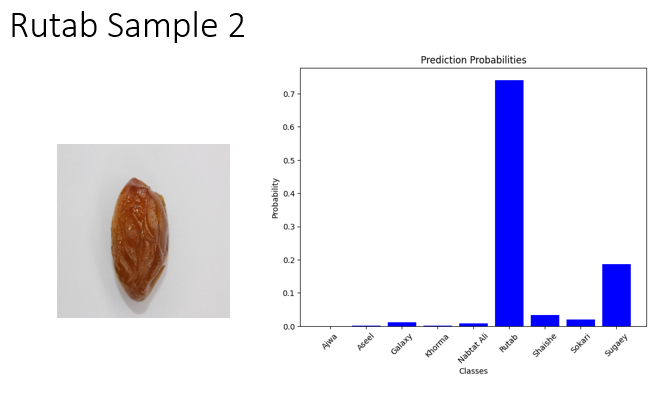


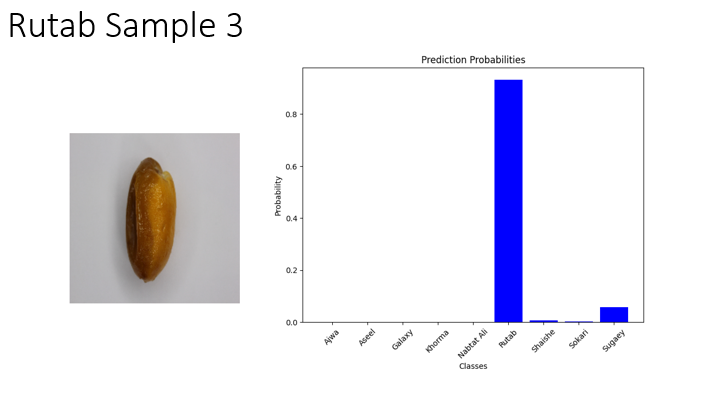


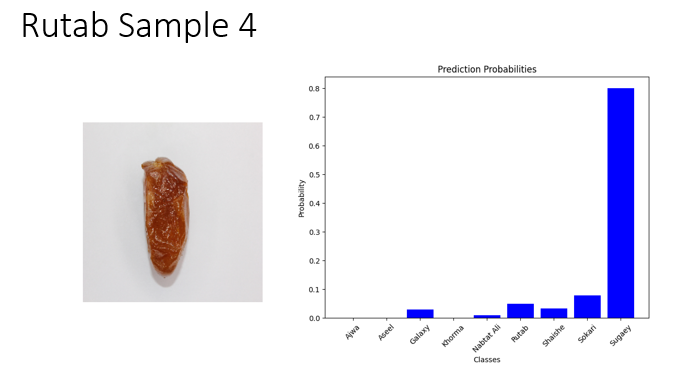


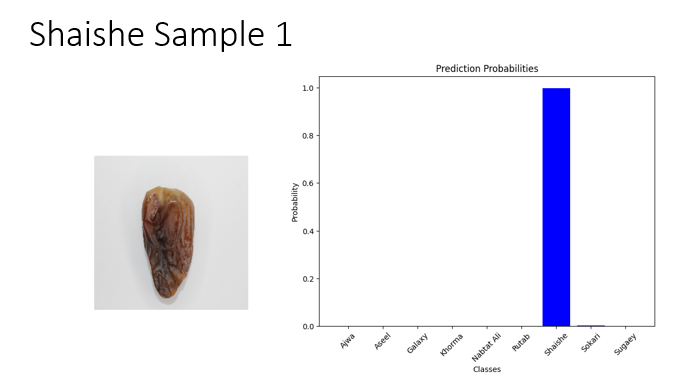


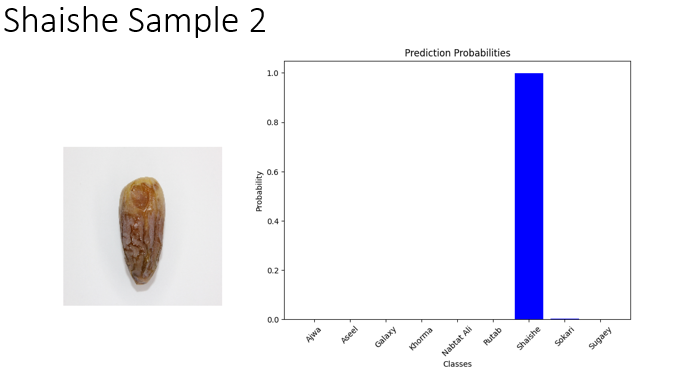


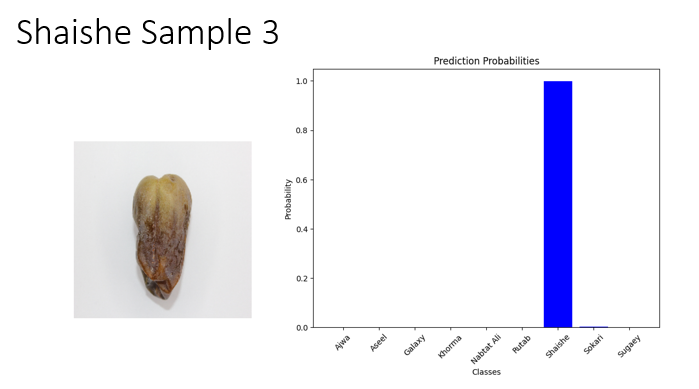


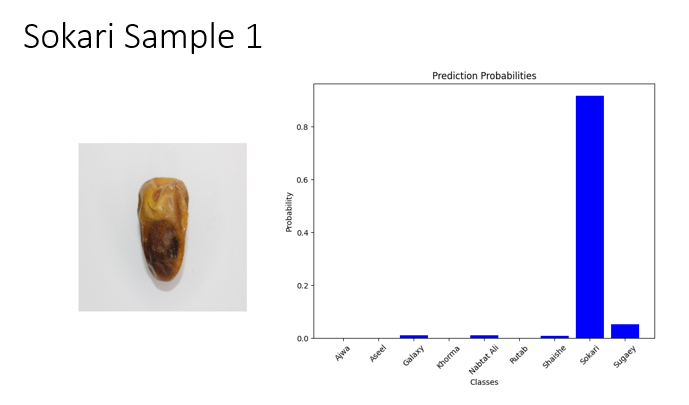


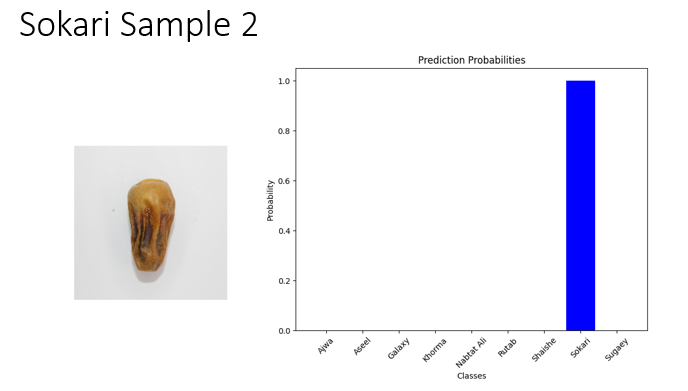


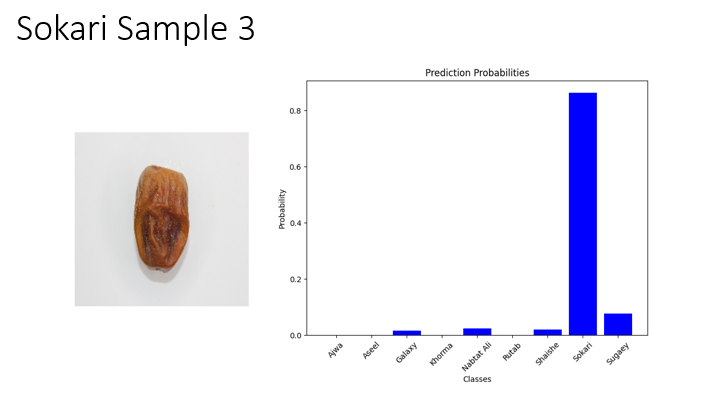


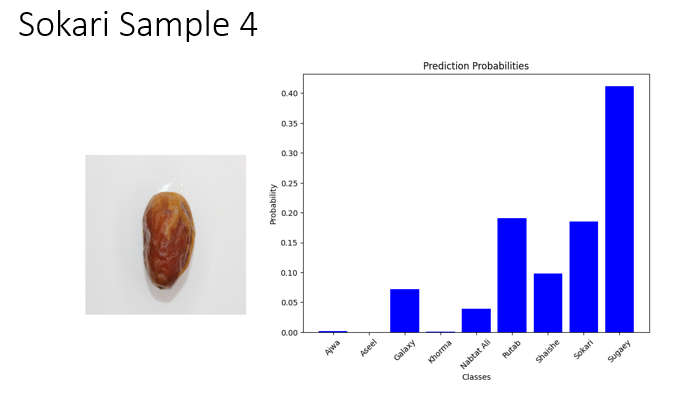


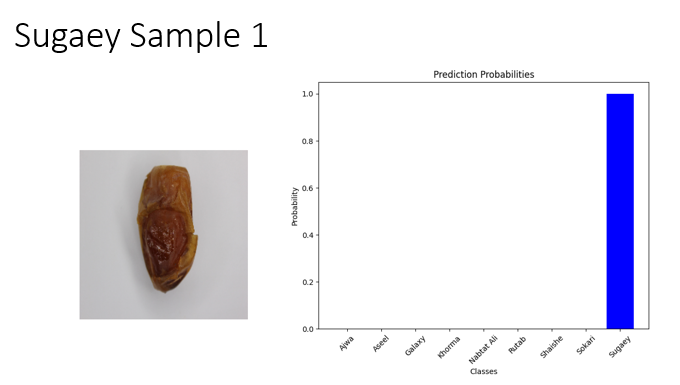


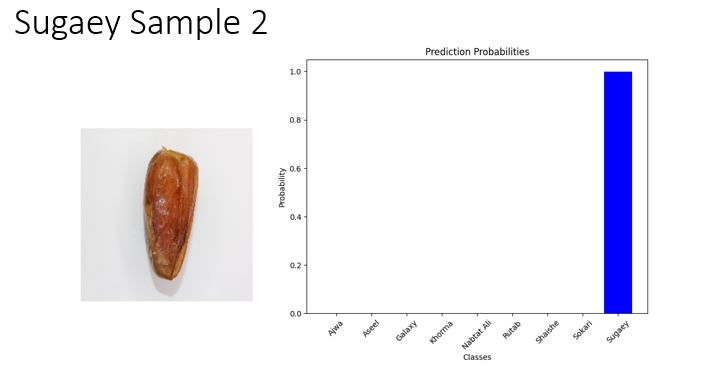


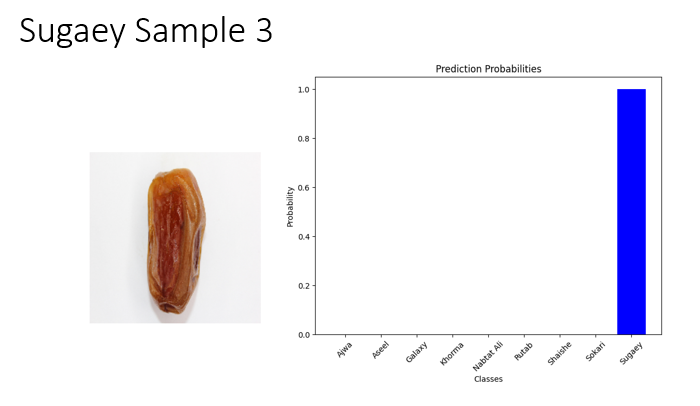




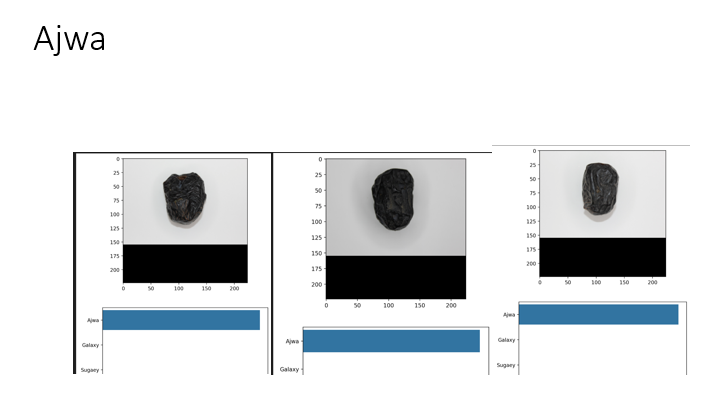


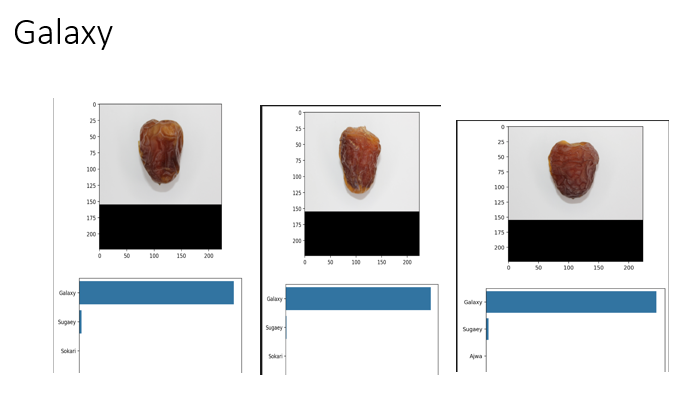


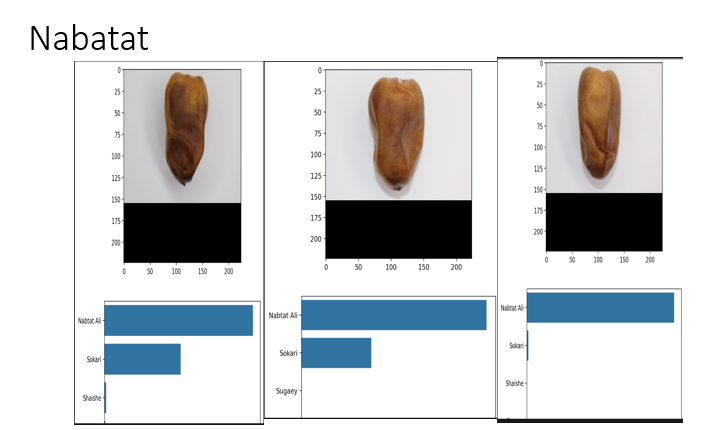


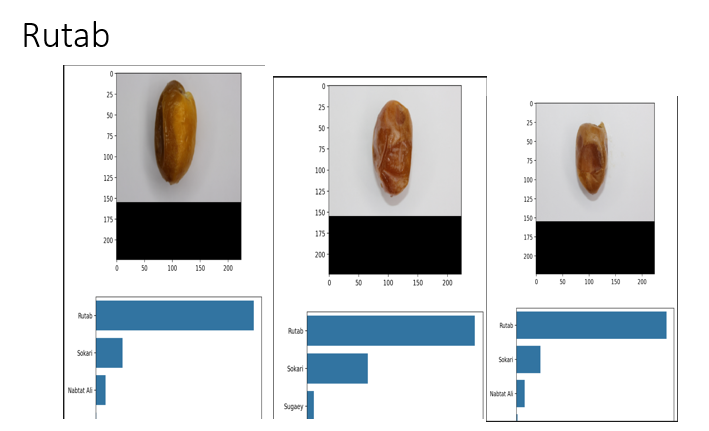


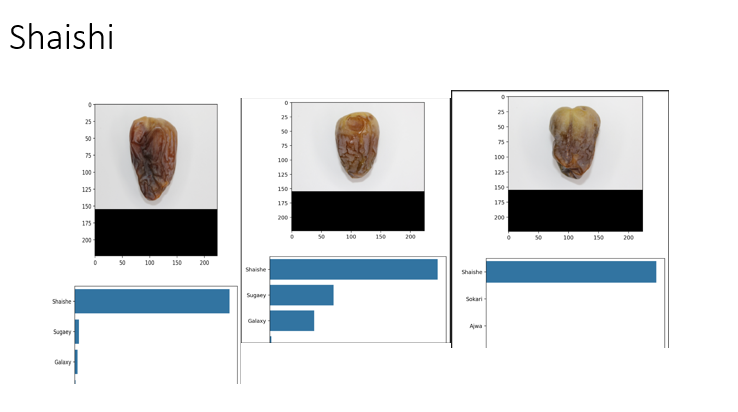
### VGG-16

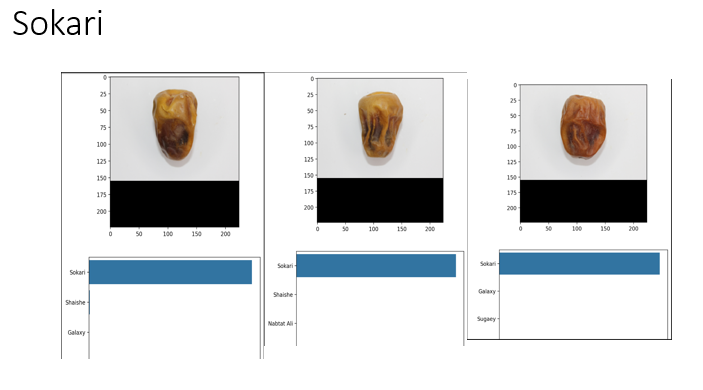


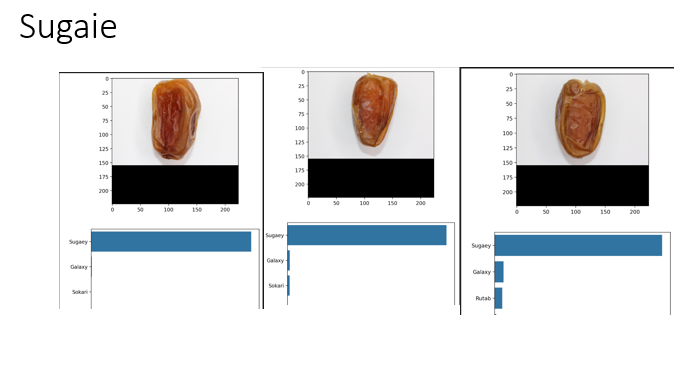












# Conclusions and Further Work

## Conclusions:

The Date Fruit Classification project has successfully developed a robust model for accurately identifying and classifying different varieties of date fruits. Leveraging advanced techniques in image processing and deep learning, we have achieved a significant milestone in automating the classification process. The integration of the ResNet50 model, coupled with extensive data preprocessing and augmentation, has enabled precise recognition of date fruit types based on visual characteristics.

Throughout the project lifecycle, rigorous testing and validation procedures have been employed to ensure the reliability and accuracy of our classification system. We have validated our model across various datasets, confirming its ability to generalize well to unseen data and maintain high classification accuracy.

By implementing this project, we aimed to facilitate easy identification and classification of date fruits, catering to agricultural and commercial sectors alike. The user-friendly interface and efficient classification capabilities enhance decision-making processes, such as quality control in production or inventory management.

## Further Work:

Despite the successful implementation of the Date Fruit Classification project, there are several avenues for further enhancement and future development:

* **Expansion of Dataset:** Increase the diversity and size of the dataset by incorporating more varieties and instances of date fruits. This expansion will improve the model's accuracy and robustness, ensuring it can handle a broader spectrum of real-world scenarios.
* **Model Refinement:** Continuously refine the classification model through advanced techniques in deep learning and computer vision. This includes exploring state-of-the-art architectures, fine-tuning hyperparameters, and optimizing model performance to achieve even higher accuracy levels.
* **Integration of User Feedback Module:** Implement a user feedback mechanism within the application to gather insights from users regarding classification accuracy and usability. This feedback will be instrumental in iteratively improving the model and enhancing user satisfaction.
* **Continuous Learning Model:** Develop a framework for continuous learning where the model adapts and improves over time based on new data inputs. This adaptive capability ensures that the classification system remains up-to-date with evolving patterns and variations in date fruit characteristics.

# References

* Sehar Tosif Jamal. "Date Fruit Classification." GitHub Repository. 2022. [Online]. Available: <https://github.com/Sahar-TJ/Date-Fruit-Classification/blob/main/README.md?plain=1>
* Sehar Tosif Jamal. "GitHub: Sahar-TJ/Date-Fruit-Classification." LinkedIn Post. 2022. [Online]. Available: <https://www.linkedin.com/posts/sahar-tosif-jamal_github-sahar-tjdate-fruit-classification-activity-7008055324976812032-lzr9?utm_source=share&utm_medium=member_desktop>
* Sehar Tosif Jamal. "Kaggle Notebook: Date Fruit Classification." Kaggle. 2022. [Online]. Available: <https://www.kaggle.com/code/sahartj/date-fruit-classification?scriptVersionId=113540986>
* Klasifikasi Jenis Buah Kurma Menggunakan Algoritma CNN Berdasarkan Jenisnya. Institut Riset dan Publikasi Indonesia (IRPI), MALCOM: Indonesian Journal of Machine Learning and Computer Science. Accessed June 13, 2024. [https://journal.irpi.or.id/index.php/malcom/article/download/724/324](file:///C:\Users\Shoaib%20Akhter\AppData\Local\Microsoft\Windows\INetCache\IE\P95904XA\Classification%20of%20Date%20Fruit%20Types%20Using%20CNN%20Algorithm%20Based%20on%20Type)