EDUCAM

### SOFTWARE REQUIREMENT SPECIFICATION

#### Table of Contents

1. Introduction
2. Purpose
3. Scope
4. Definitions, Acronyms, Abbreviations
5. References
6. Overall Description
7. Overall Description
8. Product Perspective
9. Product Functions
10. User Characteristics
11. User Constraints
12. Assumptions & Dependencies
13. Apportioning Requirements
14. Specific Requirements
15. Interface Requirements
16. Functional Requirements
17. Performance Requirements
18. Logical Database Requirements
19. Design Constraints
20. Software System Attributes
21. Supporting Requirements

## 

## **Introduction**

### **Purpose**

This Software Requirements Specification (SRS) aims to define the functional and non-functional requirements for developing a EduCam. It outlines the system's features, functionalities, constraints, serving as a development, testing, and deployment guide.

### **Scope**

**a)** The software product to be produced is the "EduCam."

**b)** The software will:

* Able to partially automate the teaching faculty in entering student data in Excel sheets ..
* Ensure user authentication, confidentiality, and anonymity of marks.

**c)** The EduCam aims to:

* Simplify the marks entering process for university faculties.
* Streamline marks entering tasks for teaching faculty,
* reducing most manual effort and errors.

### **Definitions. Acronyms, Abbreviations**

* SRS: Software Requirements Specification - The description of a particular product, program, or set of programs that perform a set of functions in a target environment.
* GUI/UI: (Graphical) User Interface - The visual interface through which users interact with software.
* DBMS: Database Management System - Software for managing databases and handling data operations.

### **References**

***IEEE Recommended Practice for Software Requirements Specifications***

* 1. Date: October 20, 1998
  2. Published by: Software Engineering Standards Committee of the IEEE Computer Society

### **Overall Description**

This project focuses on the development of an Android application, EduCam, designed to streamline the processing of academic mark sheets in universities. Leveraging Optical Character Recognition (OCR) technology, the application efficiently extracts student information and marks from mark sheet images, converting them into a structured, digital format or doing manual data entry in case of finding difficulty in image processing, reducing human error and saving valuable time for administrative staff.

## **Overall Description**

### **Product Perspective**

Educam is designed to operate as an independent and self-contained software product. It is not a component of a larger system but serves as a standalone solution specifically tailored for managing university-level elections.

1. **Product Functions**

User Authentication: Users can log in to the application using their credentials.

Capture Mark Sheets: Users can capture images of mark sheets using their device's camera.

OCR Processing: The application will use OCR technology to extract text from the captured images.

EnterMarks: The application will give you a option to enter the marks of students according to their ID.

1. **User Characteristics**

Many users of the application are educators who need to process mark sheets. Users should have basic knowledge of operating a smartphone or tablet

1. **User Constraints**

Users must have access to a device with a functioning camera.

1. **Assumptions & Dependencies**

* Users have access to a device with a functioning camera.
* The application requires a valid internet connection.

1. **Apportioning Requirements**

Identify requirements that can be postponed to future releases.

## **Overall Description**

This section of the SRS contains all of the software requirements to a level of detail sufficient to enable designers to design a system to satisfy those requirements, and testers to test that the system satisfies those requirements.

### **Interface Requirements**

#### **User Interface**

[APPENDIX-I]

#### **Hardware Interface**

-

#### **Software Interface**

* Dart
* Firebase
* Python
* Flask

#### **Communication Interface**

Internet

### **Functional Requirements**

#### **Use Case Model**

[Appendix - II]

#### **Use Case Specifications**

[Appendix - III]

#### **Activity Diagram**

[Appendix - IV]

#### **Analysis Classes**

[Appendix - V]

### **Performance Requirements**

It could range from a few to thousands of requests per second (Information based on Http request through Ip address).

### **Logical Database Requirements**

Firebase :A NoSQL cloud database that allows you to store and sync app data in real-time across clients, enabling collaborative and dynamic experiences.

### **Design Constraints**

The application should have a user-friendly interface and be easy to navigate.

### **Software System Attributes**

#### **Reliability**

The application should be reliable and not crash or lose data.

#### **Availability**

The application should be available to users at all times.

#### **Security**

User credentials and captured images should be stored securely.

#### **Maintainability**

The application should be easy to maintain and update.

#### **Portability**

The application should be compatible with different devices and operating systems.

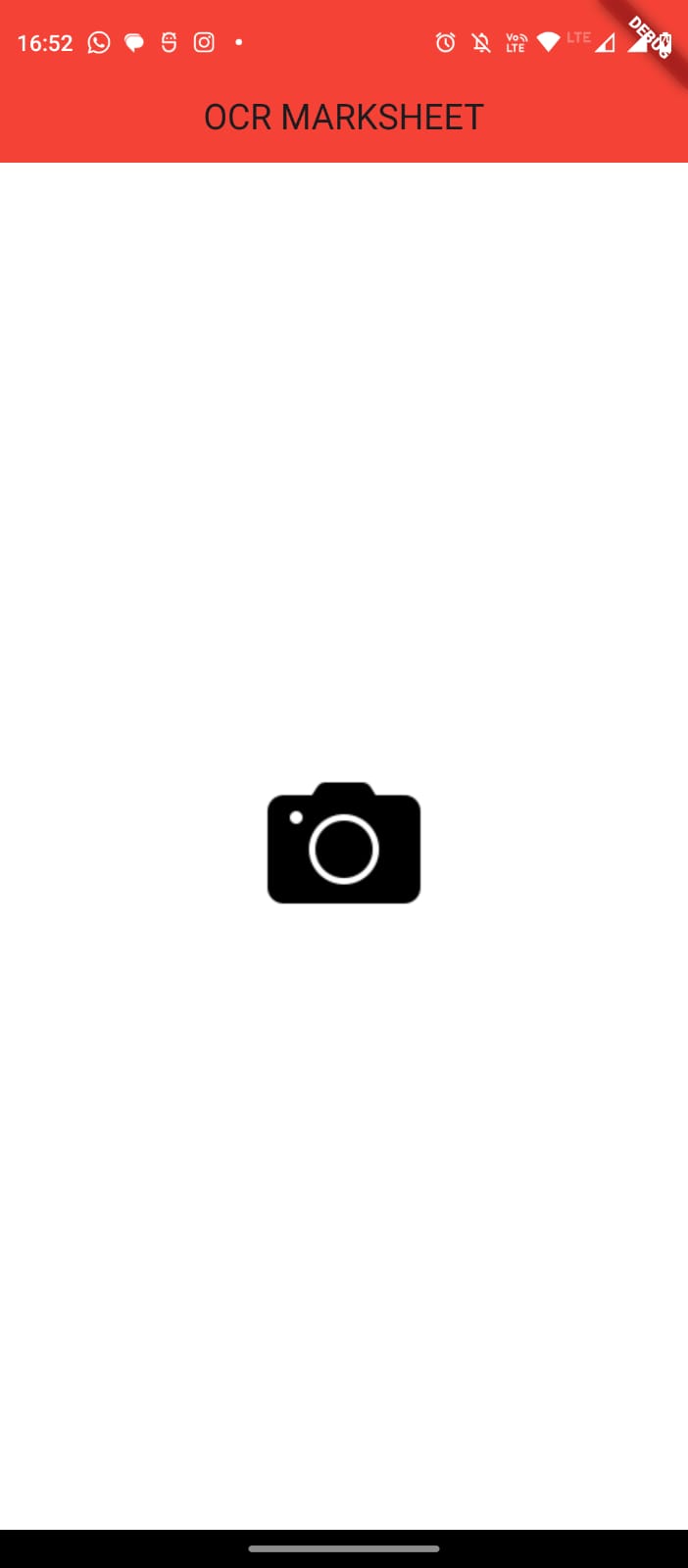
## **Supporting Information**

The supporting information includes the following:

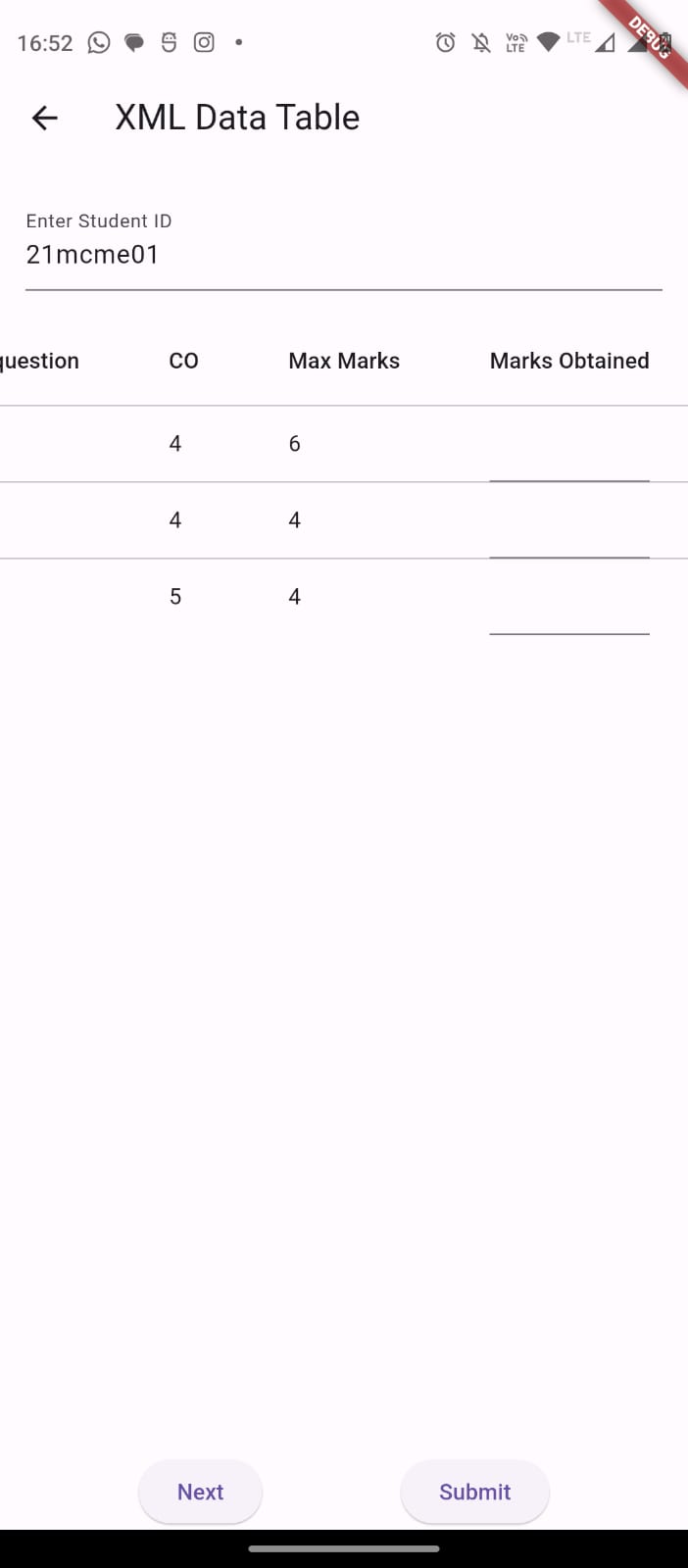
1. [Table of contents](https://docs.google.com/document/d/1kw7nkYjfNPgY9zyYYXWL3MoPfm-SWMtnlg382WT5TLo/edit#heading=h.e45rdukzaa6t) [see the beginning of this document]
2. Appendixes [see from the next page]

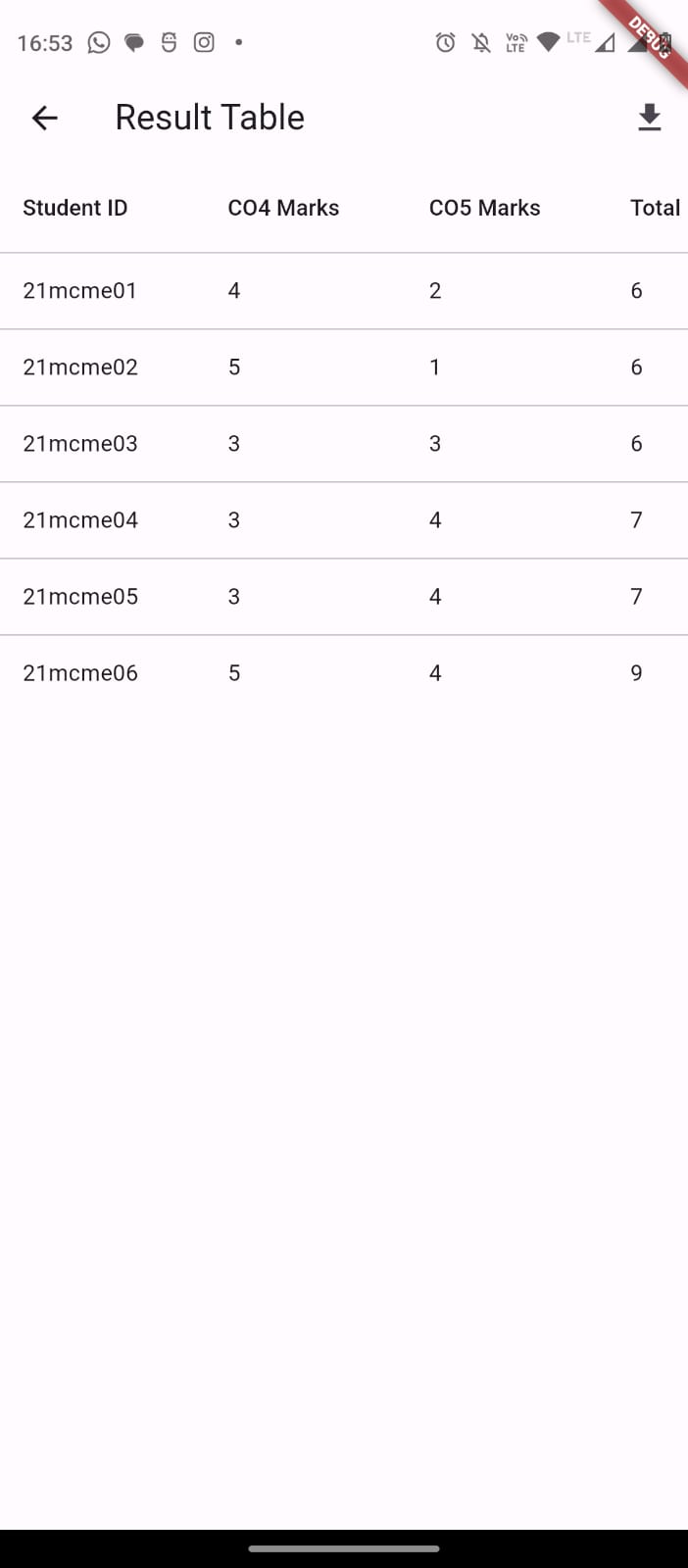
**Appendix-I:** User Interface

User Login Interface

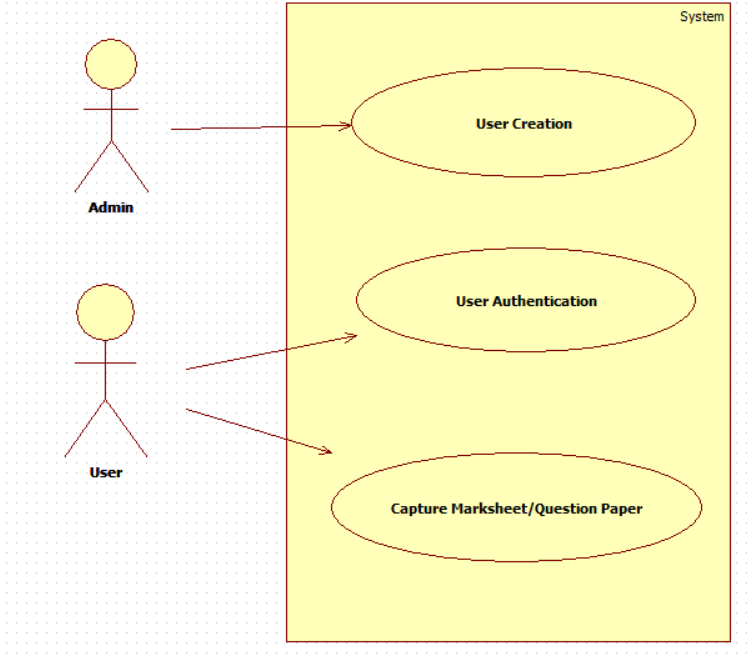





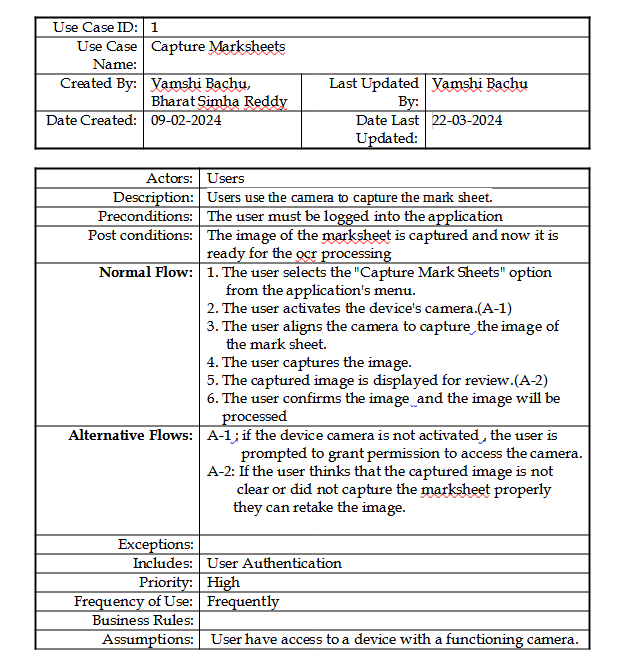
****

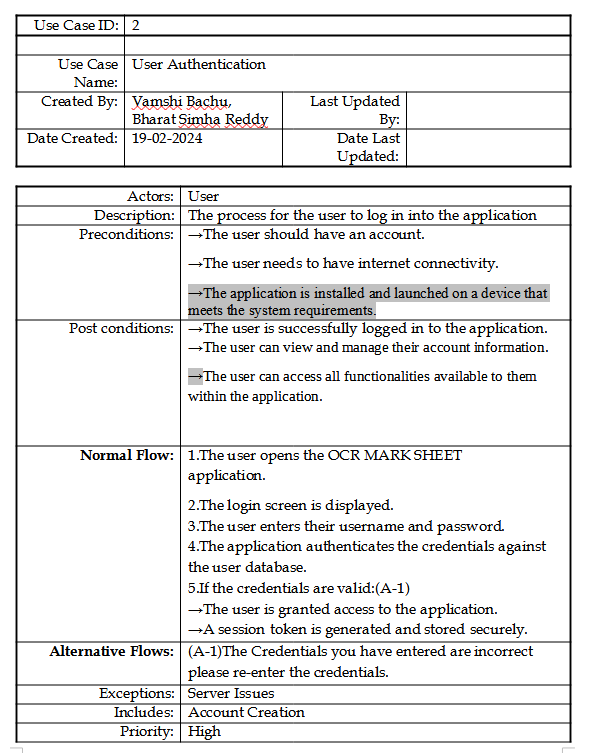
****

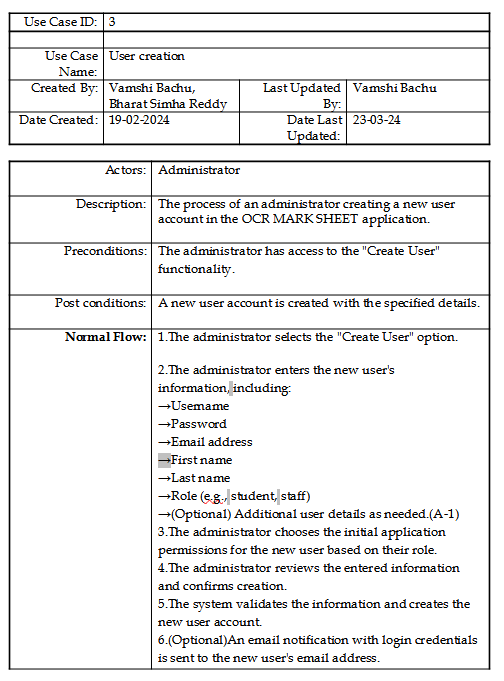
**Appendix-II:** Use Case Model

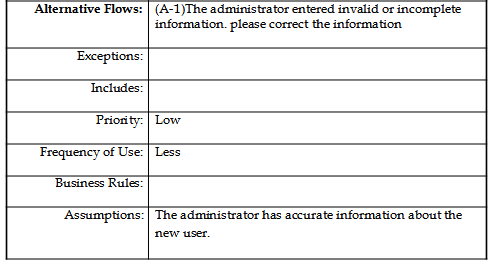


**Appendix-III:** User Case Specifications



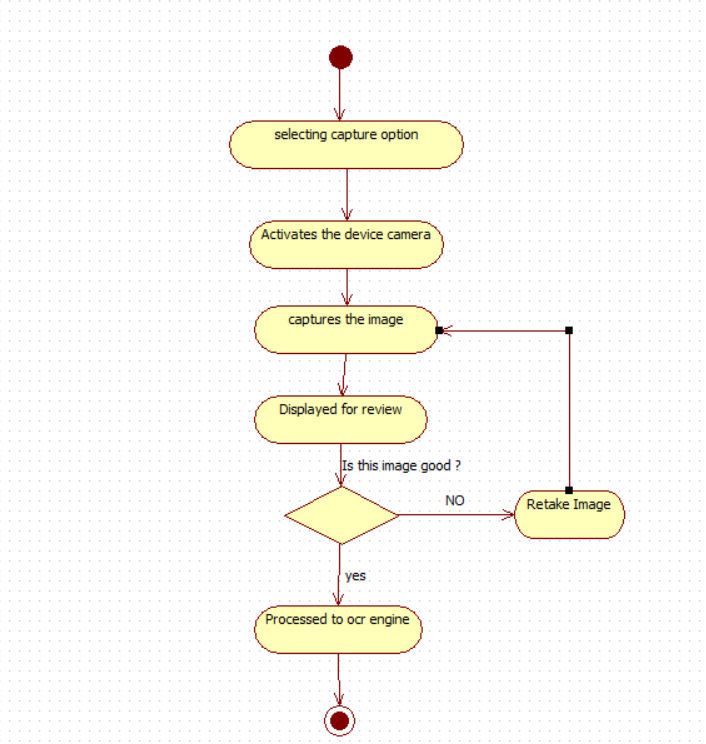
****

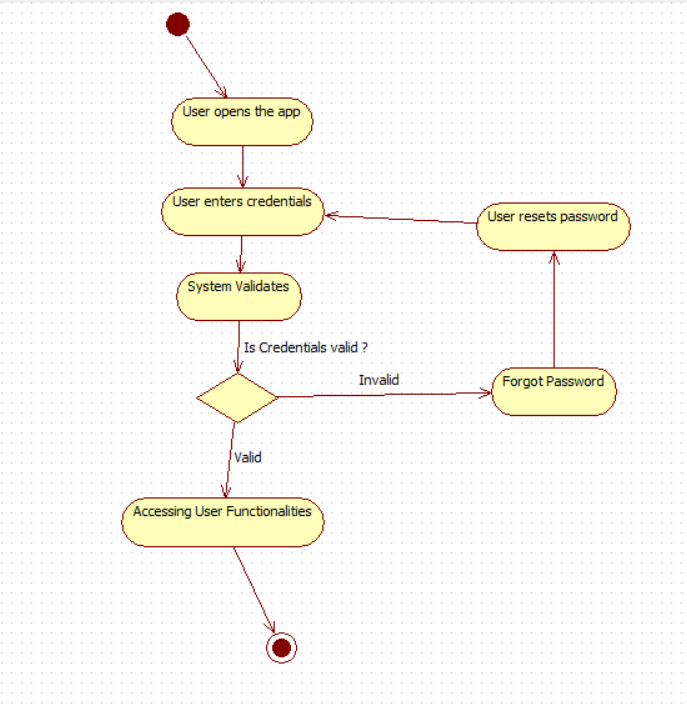
****

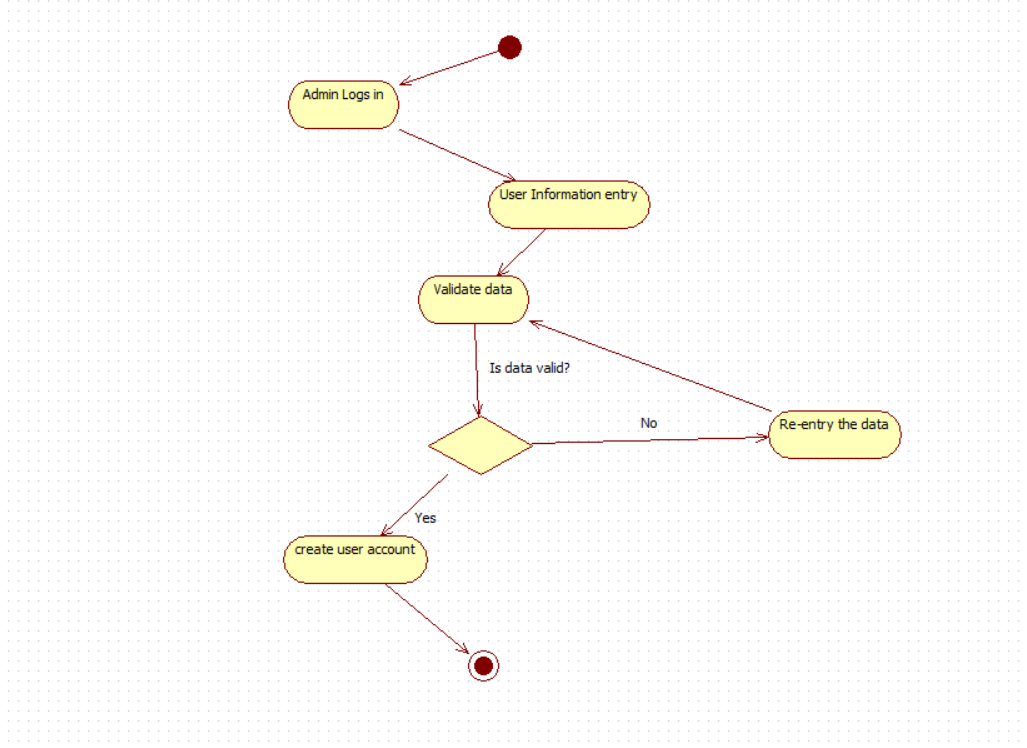
****

**Appendix-IV:** Activity Diagrams

-Capture Marksheet



* **User authentication**
* 
* User creation

****

**Appendix-V:** Analysis Classes

. 1. User

- Attributes: username, password,.

- Methods: authenticate()

2. Admin

- Attributes: username, password

- Methods: userCreation()

Data Dictionary

1. User Table\*

- Fields: id, username, password\_hash, email, role

2. Session Table

- Fields: id, user\_id, token, created\_at, expired\_at