Project Report

SUG

ScholarShip Management System

By

MOHD MUKDDAS (23017400024)

In the partial fulfillment of requirements for the award of degree in Masters of Computer Application

(Batch from 2024-2025)

Under the Guidance of

Jitendra Saini(assistant professor)

School of computer science and engineering

CERTIFICATE OF ACCEPTANCE

This is to certify that **Mohd Mukddas**, bearing Registration No. **23017400024** of **Shobhit University Gangoh**, has successfully worked on the project entitled **"SCHOLARSHIP MANAGEMENT SYSTEM"** under the supervision of **Jitendra Saini(assistant professor)**, **Shobhit University Gangoh**.

The project was carried out from **15th January 2025 to April 2025**.

The project is hereby accepted by **Shobhit University Gangoh**, in partial fulfilment of the requirements for the award of the **Degree in Master of Computer Application**.

DECLARATION

We hereby declare that the work recorded in this project report entitled **“SCHOLARSHIP MANAGEMENT SYSTEM”**, in partial fulfilment of the requirements for the award of the **Degree in Master of Computer Application** from **Shobhit University Gangoh**, is an original and faithful work carried out under the supervision and guidance of **Jitendra saini(assistant professor)** from **15th January 2025 to 25th March 2025**.

The results and findings presented in this project report have not been submitted for the award of any other **Degree**, **Diploma**, or presented in any other **technical forum**.

The assistance and support received during the course of this project have been duly acknowledged.

BONAFIDE CERTIFICATE

Certified that this project report titled **"SCHOLARSHIP MANAGEMENT SYSTEM"** is the bonafide work of **MOHD MUKDDAS (23017400024)**, who carried out the research under my supervision.

Certified further, that to the best of my knowledge, the work reported herein is original and has not been part of any other project report or dissertation submitted for the award of any degree or diploma on an earlier occasion by this or any other candidate.

Submitted for the **viva-voce** examination held on **_____**.

Supervisor's Signature
Jitendra saini(assistant professor)
Lecturer, Shobhit University Gangoh

TABLE OF CONTENTS

Chapter - 1

Introduction

A **Scholarship Management System (SMS)** is a digital platform designed to simplify and automate the scholarship application process within an educational institution. This system supports secure student authentication via DigiCampus API, image-based document submission, real-time application tracking, and admin-side verification.

By integrating both mobile and web interfaces, it facilitates seamless communication between students and administrative staff. The system ensures organized data management, transparent workflows, and improved decision-making — all hosted on the cloud via MongoDB Atlas.

Project Objective

The primary objective of this **Scholarship Management System** is to build a secure, user-friendly, and efficient digital solution for managing scholarship applications, from student submission to admin approval or rejection. The system is built to ensure data integrity, provide real-time status updates to students, and reduce the manual workload of administrators.

Key Objectives of the Scholarship Management System

1. Secure and Verified Student Login

- **Objective:** Integrate DigiCampus API for authenticating users, ensuring only verified students can log in.
- **Benefit:** Prevents unauthorized access, maintains institutional security, and ensures valid student participation.

2. Image-Based Document Submission

- **Objective:** Allow students to scan or upload scholarship-related documents through the mobile app.
- **Benefit:** Reduces the need for physical paperwork and supports quick, remote submission.

3. Student Form Submission with Metadata

- **Objective:** Collect essential scholarship-related information (e.g., scholarship number, personal info) with the document submission.
- **Benefit:** Ensures complete application data is submitted in a structured and accessible format.

4. Cloud-Based Storage and Access

- **Objective:** Store applications and documents in MongoDB Atlas for scalable, secure, and accessible storage.
- **Benefit:** Enhances reliability and allows real-time data access for both admins and students.

5. Admin Dashboard for Application Review

- **Objective:** Provide a computer-based interface where administrators can view, verify, accept, or reject scholarship applications.
- **Benefit:** Streamlines the review process and enables quicker decisions with better oversight.

6. Student Application Status Tracking

- **Objective:** Let students track the status of their application (e.g., "Pending", "Accepted", "Rejected") through the mobile app.
- **Benefit:** Improves transparency and keeps students informed without the need for manual inquiries.

7. Student Profile Management

- **Objective:** Allow students to view and manage their profile and submission history within the app.
- **Benefit:** Promotes user engagement and provides a centralized hub for personal scholarship records.

8. System Logging and Accountability

- **Objective:** Record key actions (submissions, approvals, rejections) for audit and review purposes.
- **Benefit:** Ensures accountability for both students and administrators, enabling fair and transparent processes.

Chapter – 2

System Proposal – Scholarship Management System

Executive Summary:

The **Scholarship Management System (SMS)** proposal outlines the development and deployment of a secure, cloud-based platform to streamline the scholarship application and verification process within an academic institution. This system enables verified students to digitally submit required documents via a mobile app and allows administrators to efficiently review, approve, or reject applications from a centralized web interface.

With authentication handled through the DigiCampus API and document storage powered by MongoDB Atlas, the system ensures secure access, organized data handling, and real-time status updates. By reducing paperwork and manual workflows, this system promotes operational efficiency, transparency, and improved communication between students and scholarship administrators.

🎯 System Objectives:

The **Scholarship Management System** is proposed with the following objectives:

1. Centralized Scholarship Application Management

- **Objective:** Maintain a secure, cloud-based repository of all submitted scholarship forms and supporting documents.
- **Benefit:** Reduces paper-based processes and enables fast retrieval and tracking of applications.

2. Secure Student Authentication via DigiCampus

- **Objective:** Authenticate students through DigiCampus to ensure only enrolled users can apply.
- **Benefit:** Enhances data integrity and protects against unauthorized access.

3. Simplified Document Submission

- **Objective:** Allow students to scan or upload required documents through an intuitive mobile interface.

- **Benefit:** Offers a convenient, mobile-first user experience, increasing accessibility and adoption.

4. Admin Review and Application Processing

- **Objective:** Provide admins with a desktop dashboard to view, accept, or reject applications and manage document records.
- **Benefit:** Streamlines the verification process and reduces administrative workload.

5. Real-Time Status Notifications

- **Objective:** Notify students via the app about their application status updates (Pending, Accepted, Rejected).
- **Benefit:** Keeps applicants informed, eliminating the need for manual follow-ups.

6. Secure and Scalable Cloud Storage

- **Objective:** Use MongoDB Atlas to store submissions and metadata securely in the cloud.
- **Benefit:** Enables scalability and robust data access across devices and locations.

7. Profile and History Management

- **Objective:** Allow students to view their profile, past applications, and status history.
- **Benefit:** Empowers students with self-service features and personal record-keeping.

⚠ Disadvantages:

1. High Initial Development Effort

- **Disadvantage:** Building a custom app with secure authentication, cloud integration, and real-time features requires time and technical investment.
- **Impact:** Institutions may need external development support or dedicate resources to maintain the system.

2. Technical Complexity

- **Disadvantage:** Implementing multi-role access control, cloud database integration, and secure document uploads adds complexity.
- **Impact:** Requires skilled developers and potential ongoing maintenance for scalability and reliability.

3. Dependence on Internet Connectivity

- **Disadvantage:** Real-time features and cloud syncing rely on stable internet access for both students and admins.
- **Impact:** Limited functionality in offline environments may affect users in areas with poor connectivity.

🛠 Proposed System – Scholarship Management System

The proposed **Scholarship Management System (SMS)** is a mobile and web-based platform built to provide a seamless, secure, and scalable solution for managing scholarship applications, document submissions, and application reviews. The system aims to enhance institutional efficiency, reduce manual processing, and improve transparency between students and scholarship administrators.

---

1. System Overview:

The **Scholarship Management System** will be accessible through both **Android mobile application** (for students) and a **web-based admin panel** (for administrators). It ensures real-time updates, secure access, and role-based functionality tailored to user needs.

• User Interface:

- Intuitive and responsive UI, optimized for smartphones and desktop browsers.
- **Role-based dashboards** for:
  - **Students**: Submit documents, check application status, and manage profiles.
  - **Admins**: Review applications, approve/reject submissions, and view logs.
- Personalized user profiles with editable fields such as personal info and scholarship number.

• Student-Focused Features:

- Upload scanned or gallery-selected images of required documents directly through the app.
- Fill out a simple, guided form for scholarship application (including scholarship number and basic details).
- View submission history and current status (e.g., Pending, Approved, Rejected).

• Administrator Control:

- Manage, review, and verify student submissions via a centralized dashboard.
- View submitted documents, approve/reject applications with reasons.

- Access logs, generate summaries, and monitor workflow efficiency.

---

2. Data Security and Privacy:

Security and data protection are core pillars of the proposed system. By integrating cloud technologies and robust access controls, the system ensures institutional and student data remains private and protected.

• Encryption and Secure Access:

- End-to-end encryption for document uploads and user data.
- **DigiCampus API integration** for secure, institution-based student login.
- **Role-based access control**: Students can only access their data; admins have elevated privileges with audit tracking.

• Compliance with Regulations:

- Designed with best practices for data privacy, aligned with institutional standards and frameworks like **FERPA**, **GDPR**, and **CCPA**.
- Secure credential management, access logging, and data storage in **MongoDB Atlas**, ensuring high availability and protection against unauthorized access.

Chapter -3:

Diagram And ScreenShot

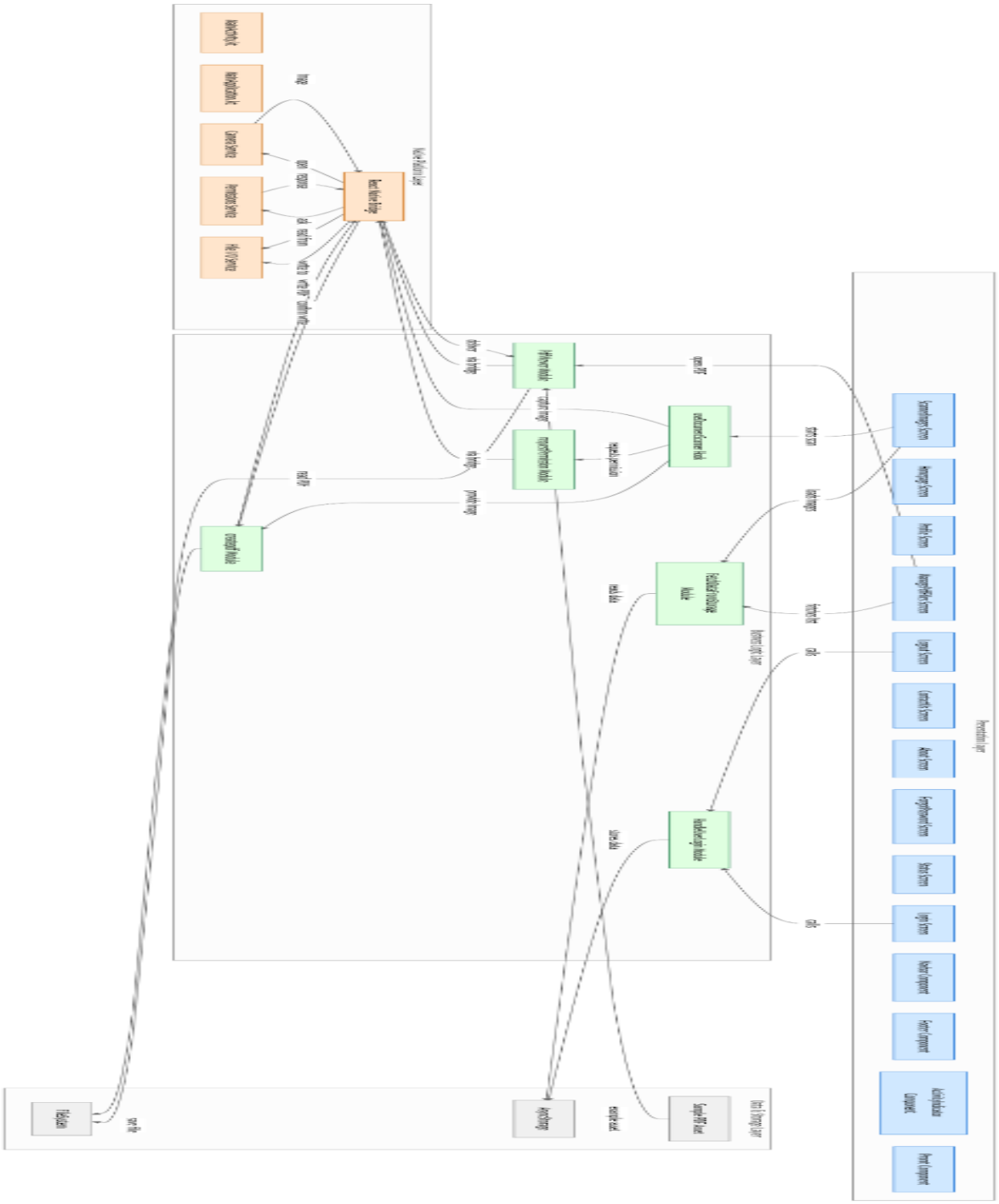
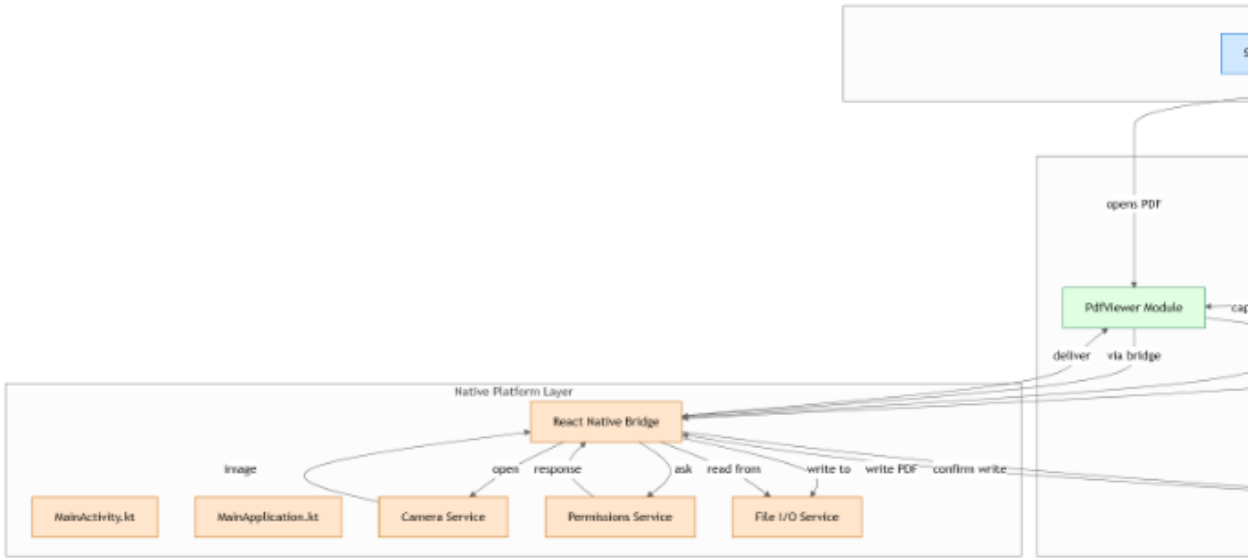
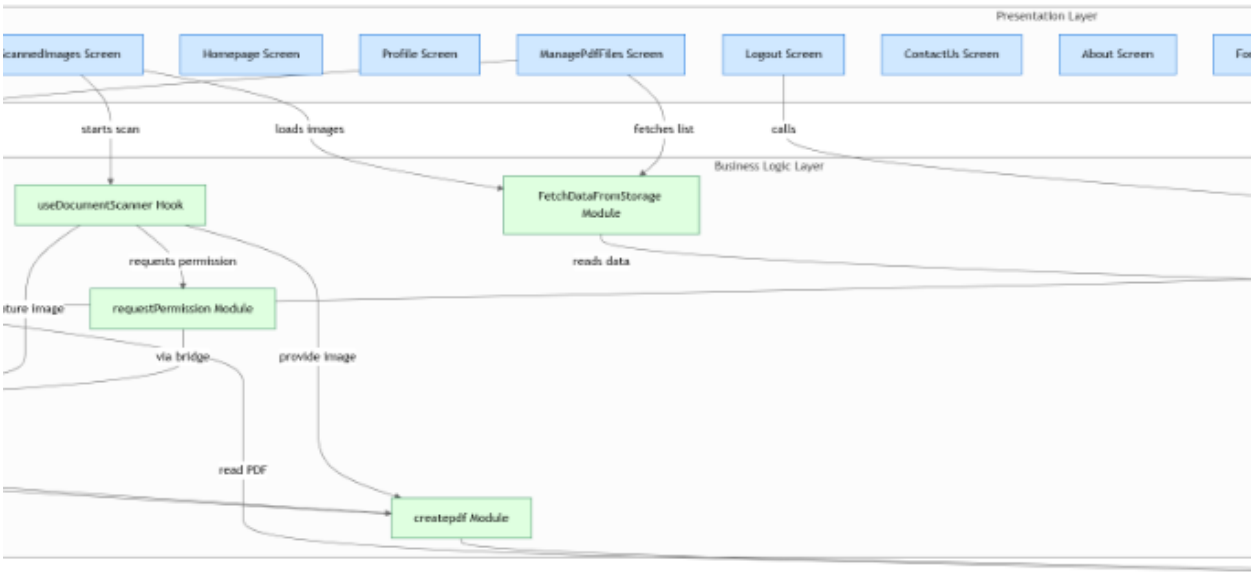
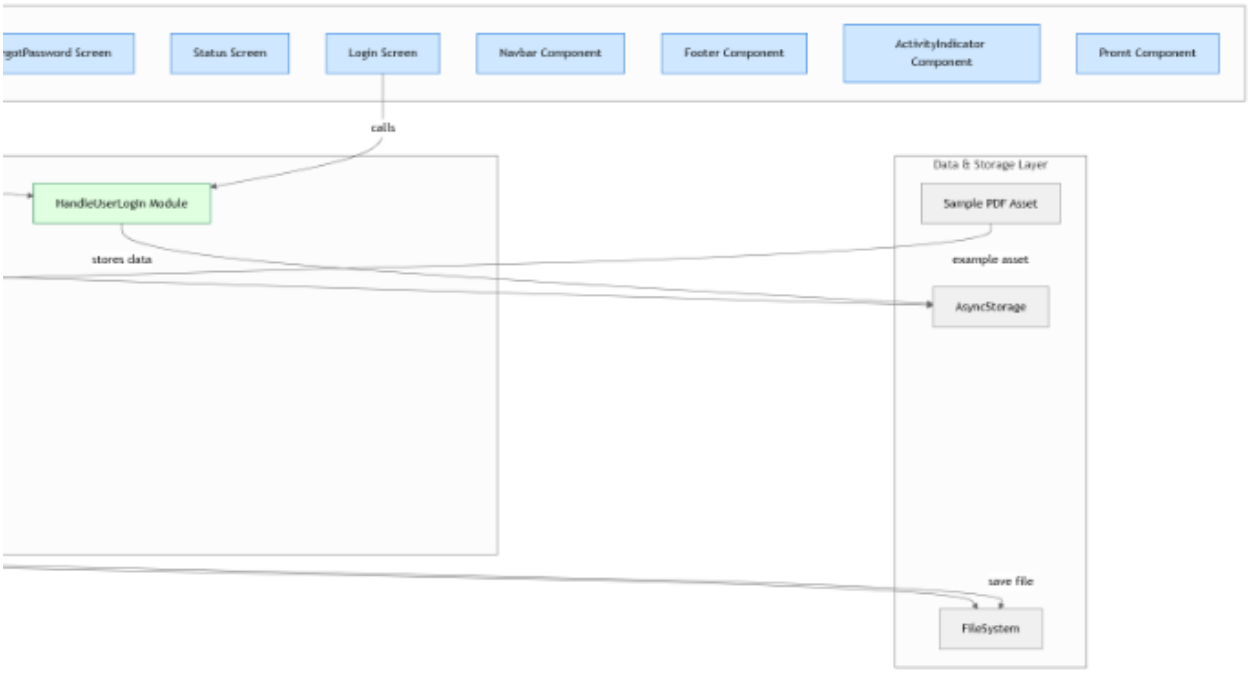
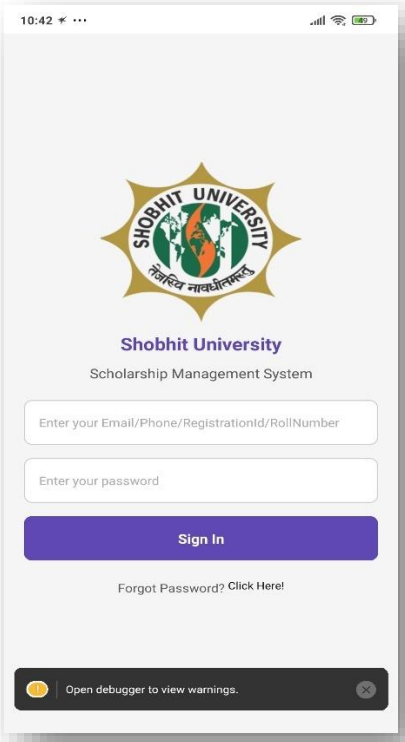


Diagram Partial View

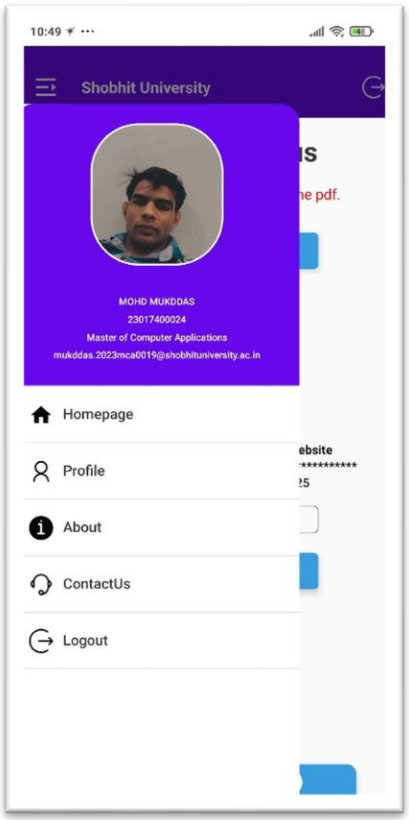


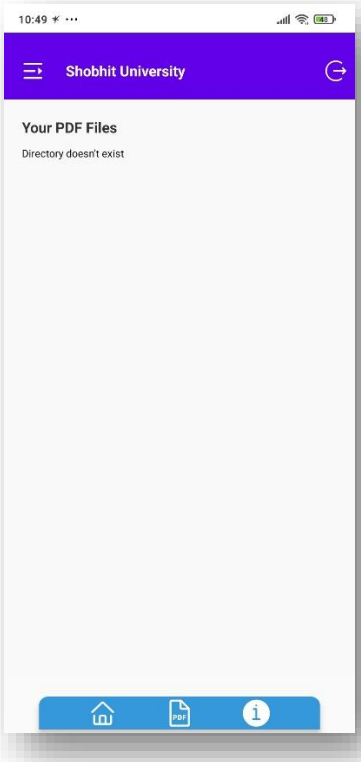

Presentation Layer
Homepage Screen
Profile Screen
ManagePdfFiles Screen
Logout Screen
ContactUs Screen
About Screen
starts scan
loads images
fetches list
calls
Business Logic Layer
useDocumentScanner Hook
FetchDataFromStorage Module
requests permission
reads data
requestPermission Module
via bridge
provide image
read PDF
createpdf Module

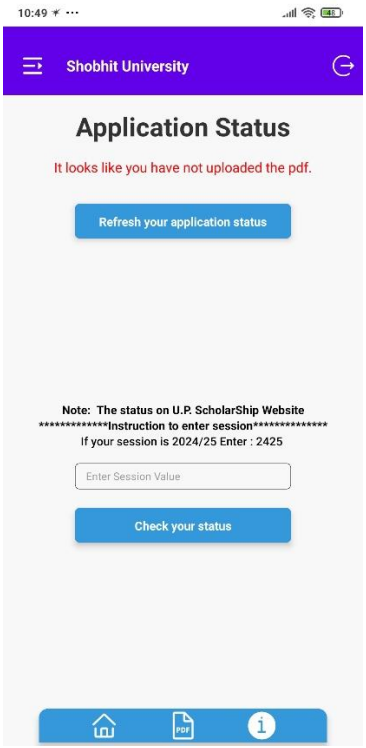

Status Screen
Login Screen
Navbar Component
Footer Component
ActivityIndicator Component
Prorat Component
calls
HandleUserLogin Module
Data & Storage Layer
Sample PDF Asset
stores data
example asset
AsyncStorage
save file
FileSystem

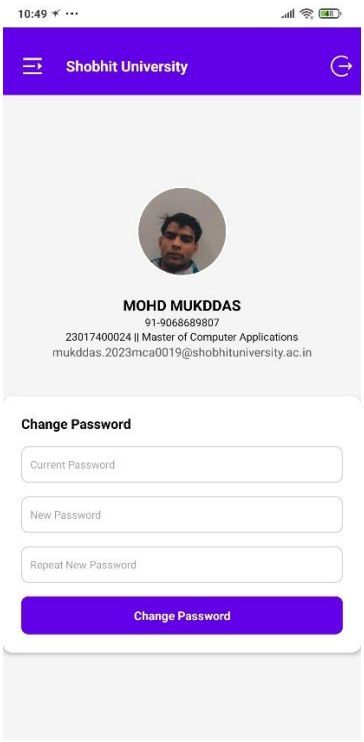

10:42
Shobhit University
Scholarship Management System
Enter your Email/Phone/RegistrationId/RollNumber
Enter your password
Sign In
Forgot Password? Click Here!
Open debugger to view warnings.

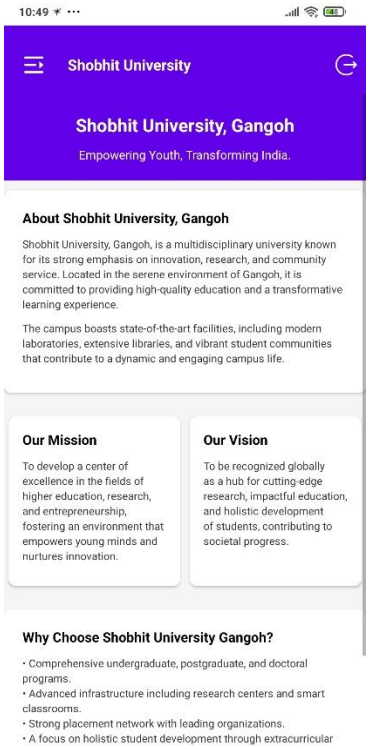

10:49
Shobhit University
Shobhit University,Gangoh
Scholarship Management System
Click Here To Scan Document

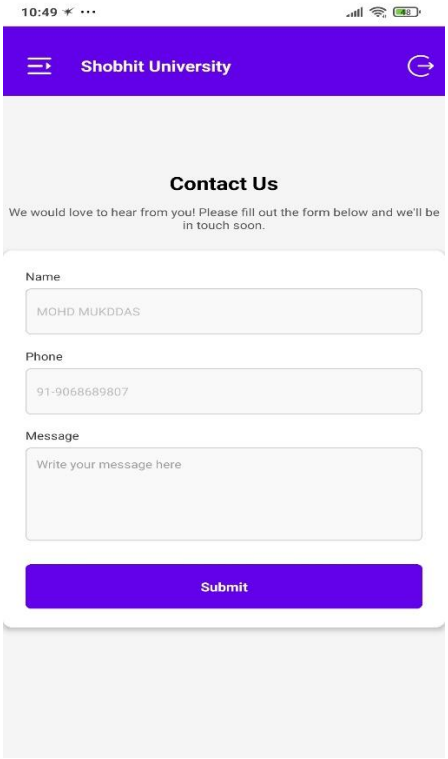

10:49
Shobhit University
MOHD MUKDDAS
23017400324
Master of Computer Applications
mukddas.2023mca0019@shobhituniversity.ac.in
Homepage
Profile
About
ContactUs
Logout

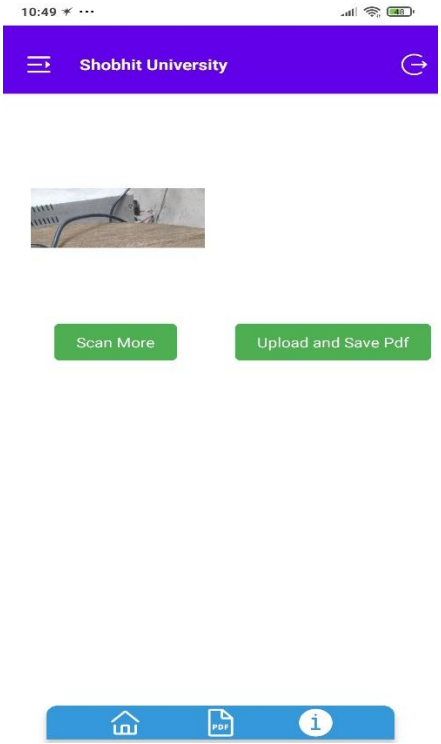

10:49
Shobhit University
Your PDF Files
Directory doesn't exist


Shobhit University
Application Status
It looks like you have not uploaded the pdf.
Refresh your application status
Note: The status on U.P. ScholarShip Website
**********Instruction to enter session**********
If your session is 2024/25 Enter : 2425
Enter Session Value
Check your status


Shobhit University
MOHD MUKDDAS
91-9068689807
23017400024 || Master of Computer Applications
mukddas.2023mca0019@shobhituniversity.ac.in
Change Password
Current Password
New Password
Repeat New Password
Change Password


Shobhit University
Shobhit University, Gangoh
Empowering Youth, Transforming India.
About Shobhit University, Gangoh
Shobhit University, Gangoh, is a multidisciplinary university known for its strong emphasis on innovation, research, and community service. Located in the serene environment of Gangoh, it is committed to providing high-quality education and a transformative learning experience.
The campus boasts state-of-the-art facilities, including modern laboratories, extensive libraries, and vibrant student communities that contribute to a dynamic and engaging campus life.
Our Mission
To develop a center of excellence in the fields of higher education, research, and entrepreneurship, fostering an environment that empowers young minds and nurtures innovation.
Our Vision
To be recognized globally as a hub for cutting-edge research, impactful education, and holistic development of students, contributing to societal progress.
Why Choose Shobhit University Gangoh?
• Comprehensive undergraduate, postgraduate, and doctoral programs.
• Advanced infrastructure including research centers and smart classrooms.
• Strong placement network with leading organizations.
• A focus on holistic student development through extracurricular


Shobhit University
Contact Us
We would love to hear from you! Please fill out the form below and we'll be in touch soon.
Name
MOHD MUKDDAS
Phone
91-9068689807
Message
Write your message here
Submit


Shobhit University
Scan More
Upload and Save Pdf

Chapter-4:

💻 1. User Interface (Frontend)

The **User Interface** of the Scholarship Management System is designed for both **students (mobile app)** and **administrators (web dashboard)**, ensuring smooth interaction, real-time updates, and secure access to relevant features.

📱 Student Portal (Android App):

• User Authentication:

- Secure login using **DigiCampus API** to ensure only verified college students access the system.
- Session handling for persistent login with optional logout functionality.

• Dashboard:

- Displays application status: **Pending**, **Accepted**, **Rejected**.
- Quick actions for **document upload**, **form completion**, and **status checks**.

• Scholarship Application Form:

- Simple UI for entering required details: **Scholarship number**, basic info, etc.
- Upload scanned or selected documents from the gallery.

• Document Preview & History:

- Preview uploaded documents before submission.
- View submission history and current status updates.

• Profile Management:

- Update personal information: name, email, course, and academic details.
- Manage password and profile photo (optional).

• Notifications:

- In-app notifications for status updates and admin decisions (e.g., rejection with reason).
- Future-ready for push notifications (Firebase integration possible).

💻 Admin Portal (Web App):

• User & Application Management:

- View list of all submitted applications with filters (e.g., date, department, status).
- Approve or reject applications with optional comments.

• Document Review:

- View submitted documents directly from the browser.
- Download documents if necessary for offline review.

• System Dashboard:

- Stats on number of submissions, acceptance/rejection rates.
- Activity logs for auditing actions by each admin.

• Role-based Access:

- Admin access is restricted to verified faculty/staff.
- Ability to manage multiple admins or reviewers if needed.

⚙️ 2. Backend / Application Logic

The backend is the core of your system — responsible for user authentication, handling requests, storing documents, and providing secure access to data. Likely built with **Node.js/Express** or similar.

🔐 User Management:

• Authentication & Sessions:

- Student authentication via DigiCampus API.
- Token-based sessions using **JWT** for secure communication.

• Role-Based Access Control:

- Role assignment: **Student** (app-only access) and **Admin** (dashboard access).
- Endpoint protection using middleware for route access.

📂 Application & Document Management:

• Scholarship Application Handling:

- CRUD operations on applications.

- MongoDB Atlas used to store metadata (student info, status, timestamps).
- Image/doc uploads stored securely (in MongoDB as Binary/Bucket or cloud storage with URLs).

• Document Validation & Storage:

- Validate allowed file formats (e.g., JPG, PNG, PDF).
- Option to resize or compress images before upload.

📡 Notifications & Communication:

• Real-time Updates:

- Status change triggers real-time in-app update or notification badge.

• Email Alerts (optional/for future):

- Admins may trigger rejection/acceptance emails using services like **SendGrid** or **SMTP**.

🗄 3. Database Layer – Scholarship Management System

The **database layer** serves as the core of the Scholarship Management System, securely storing all relevant data such as user credentials, scholarship application records, uploaded documents, and status history. The system uses **MongoDB Atlas**, a cloud-based NoSQL database, to provide scalability, high availability, and flexible schema design.

👥 Users Database:

• User Information:

- Stores profiles of students and administrators.
- Fields include:
  - `fullName`
  - `email`
  - `role` (Student/Admin)
  - `studentId` (fetched via DigiCampus)
  - `contactNumber`
  - `profilePicture`
  - `course`
  - `department`
  - `createdAt`, `updatedAt`

• Authentication Data:

- Uses **DigiCampus API** for verifying student identity; no password storage required for students.
- Admin credentials (if local) are hashed securely using algorithms like **bcrypt**.
- Stores **JWT tokens** for session validation and access control.

📄 Scholarship Applications Database:

• Application Records:

- Each document submission is linked to a student user.
- Fields include:
  - `applicationId`
  - `studentId`
  - `scholarshipNumber`
  - `submissionDate`
  - `status` (Pending, Accepted, Rejected)
  - `adminComment` (optional)
  - `isDeleted` (for soft deletes)

• Metadata:

- Tracks timestamps and versioning if a student re-submits.
- Audit logs for updates made by admins (for traceability).

🗃️ Document Storage:

• Submitted Documents:

- Images and PDFs submitted via mobile app are stored either:
  - As **binary objects in MongoDB using GridFS**, or
  - In **external storage** (e.g., AWS S3, Firebase), with only URLs saved in MongoDB.
- Metadata includes:
  - `fileName`, `fileType`, `fileSize`
  - `uploadedBy`, `uploadTime`
  - `linkedApplicationId`

🔔 Notification & Communication Logs:

• Status Notifications:

- Logs all system-generated messages regarding application status:
  - "Application submitted"
  - "Accepted by admin"
  - "Rejected with reason: [text]"
- Fields:
  - `userId`, `messageBody`, `timestamp`, `readStatus`

• Admin Comments/Feedback:

- Stored alongside the application entry if a rejection includes feedback.
- Useful for transparency and appeals (if system allows).

📝 Optional – Admin Audit Logs:

- Keeps a record of all admin actions:
  - Application approvals/rejections
  - User edits or deletions
- Fields:
  - `adminId`, `actionType`, `targetId`, `timestamp`

CHAPTER 5:

SYSTEM REQUIREMENTS

This section outlines the minimum hardware and software requirements needed to run and manage the **Scholarship Management System**, including both the **Admin Web Panel** and **Student Mobile Application**.

5.0 HARDWARE REQUIREMENTS

• For Admin (Web Portal Access):

- **Operating System:** Windows XP / Vista or later
- **Processor:** Dual-core 2.0 GHz or higher
- **RAM:** Minimum 2 GB
- **Storage:** At least 100 MB of free space (excluding browser cache and updates)
- **Display:** 1024×768 resolution or higher
- **Internet Connection:** Required (stable connection recommended for real-time sync)

• For Students (Mobile Application):

- **Device Type:** Android Smartphone
- **Operating System:** Android 6.0 (Marshmallow) or above
- **RAM:** Minimum 1 GB (2 GB recommended for smoother performance)
- **Storage:** At least 100 MB of free space for app and cached documents
- **Camera:** Required (for scanning/uploading documents)
- **Internet Connection:** Required (Wi-Fi or Mobile Data)

CHAPTER 6:

⚙ 6.1 TECHNICAL REQUIREMENTS

This section outlines the technologies used in the development of the **Scholarship Management System**, covering both frontend and backend components, database choices, security practices, and platform compatibility.

□ Frontend:

- **Technologies:**
  - `HTML`, `CSS`, `JavaScript`
  - **React Native** – for building the Android mobile application
  - **Electron.js** – for potential cross-platform desktop applications (optional)
- **UI/UX Design:**
  - Responsive design with a focus on mobile-first usability
  - Clean interface using standard web technologies

💻 Backend:

- **Technology Used:**
  - **Node.js** – server-side runtime for handling API requests and business logic
  - **Express.js** – (assumed) for building the RESTful API endpoints
- **Security Features:**
  - **SSL Encryption** for secure data transmission between client and server
  - **JWT (JSON Web Token)** for session handling and authentication
  - **Password Hashing** using libraries like `bcrypt` for admin accounts

---

🗄 Database:

- **Primary Databases:**
  - **MongoDB (Atlas)** – for storing student profiles, application data, and document metadata
  - **MySQL** – can be used optionally for relational data if needed (e.g., admin logs or reporting)

🌐 Platform & Compatibility:

- **Supported Web Browsers:**
  - Google Chrome (recommended)
  - Internet Explorer 8+ (basic compatibility)
- **Supported Operating Systems (Admin/Web Panel):**
  - Windows 7, 10, 11
- **Mobile OS (Student App):**
  - Android 6.0 and above (developed with React Native)

💻 Language Used:

- **JavaScript (JS)** – used across both frontend (React Native, Electron, Web) and backend (Node.js)

Chapter-7

Coding

Login.js

```
import React, {  useState } from "react";
import { Link } from "expo-router";
import { Text, TextInput, TouchableOpacity, Image,ActivityIndicator,View } from "react-native";
import { SafeAreaProvider, SafeAreaView } from "react-native-safe-area-context";
import { styles } from "../src/stylesheets/Authentication";
import { ActivityIndicatorLoading } from "../stylesheets/ActivityIndicator";
import HandleUserLogin from "../src/helper/HandleUserLogin";

const Index = () => {
const [bGColor,setBGColor]=useState(null)
  const [username, setUsername] = useState(null);
  const [password, setPassword] = useState(null);
  const [loading,setLoading]=useState(false)
const handlePress=async () => {
   setLoading(true)

   await HandleUserLogin(username,password)
   setLoading(false)
}


  return (
    <SafeAreaProvider>
      <SafeAreaView style={styles.container}>

       <Image
        style={styles.logo}
        source={require('../assets/logo.png')}
       />
       <Text style={styles.title}>Shobhit University</Text>
       <Text style={styles.subtitle}>Scholarship Management System</Text>

       <TextInput
       editable={loading?false:true}
        onChangeText={setUsername}
        style={styles.input}
        placeholder="Enter your Email/Phone/RegistrationId/RollNumber"
        placeholderTextColor="#aaa"
       />
       <TextInput
       editable={loading?false:true}
        onChangeText={setPassword}
        style={styles.input}
        placeholder="Enter your password"
        placeholderTextColor="#aaa"
        secureTextEntry
       />

       <TouchableOpacity
        onPress={()=>!loading&&handlePress(username,password)}
        style={styles.button}
        accessible={true}
        accessibilityLabel="Sign in button"
       >

        <Text style={styles.buttonText}>Sign In</Text>
```

```
      </TouchableOpacity>

      <Text style={styles.infoText}>
        Forgot Password?{" "}
        <TouchableOpacity accessible={true} accessibilityLabel="Forgot Password button">
          <Link href={'/forgotpassword'}>Click Here!</Link>
        </TouchableOpacity>
      </Text>
     <View  style={[ActivityIndicatorLoading.loading,bGColor]}>
     <ActivityIndicator

     size={80}
      animating={loading}/>
     </View>
    </SafeAreaView>
   </SafeAreaProvider>
  );
};

export default Index;
```

About.js

```
import React from 'react';
import {Dimensions, View, Text, StyleSheet, ScrollView } from 'react-native';
import Navbar from '../src/components/Navbar';
const {width,height}=Dimensions.get('window')
const About = () => {
  return (
    <View style={{ flex: 1 }}>
      <Navbar />
      <ScrollView contentContainerStyle={styles.container}>
        {/* Hero Section */}
        <View style={styles.heroSection}>
          <Text style={styles.heroTitle}>Shobhit University, Gangoh</Text>
          <Text style={styles.heroSubtitle}>Empowering Youth, Transforming India.</Text>
        </View>

        {/* Introduction Section */}
        <View style={styles.section}>
          <Text style={styles.sectionTitle}>About Shobhit University, Gangoh</Text>
          <Text style={styles.sectionText}>
            Shobhit University, Gangoh, is a multidisciplinary university known for its strong emphasis on innovation,
            research, and community service. Located in the serene environment of Gangoh, it is committed to providing
            high-quality education and a transformative learning experience.
          </Text>
          <Text style={styles.sectionText}>
            The campus boasts state-of-the-art facilities, including modern laboratories, extensive libraries,
            and vibrant student communities that contribute to a dynamic and engaging campus life.
          </Text>
        </View>

        {/* Mission and Vision Section */}
        <View style={styles.missionVisionContainer}>
          <View style={styles.missionVisionBox}>
            <Text style={styles.sectionTitle}>Our Mission</Text>
            <Text style={styles.sectionText}>
              To develop a center of excellence in the fields of higher education, research, and entrepreneurship,
              fostering an environment that empowers young minds and nurtures innovation.
            </Text>
```

```
        </View>
        <View style={styles.missionVisionBox}>
          <Text style={styles.sectionTitle}>Our Vision</Text>
          <Text style={styles.sectionText}>
            To be recognized globally as a hub for cutting-edge research, impactful education, and
            holistic development of students, contributing to societal progress.
          </Text>
        </View>
      </View>

      {/* Highlights Section */}
      <View style={styles.section}>
        <Text style={styles.sectionTitle}>Why Choose Shobhit University Gangoh?</Text>
        <Text style={styles.listItem}>• Comprehensive undergraduate, postgraduate, and doctoral programs.</Text>
        <Text style={styles.listItem}>• Advanced infrastructure including research centers and smart classrooms.</Text>
        <Text style={styles.listItem}>• Strong placement network with leading organizations.</Text>
        <Text style={styles.listItem}>• A focus on holistic student development through extracurricular activities.</Text>
        <Text style={styles.listItem}>• Dedicated faculty members with international exposure and expertise.</Text>
      </View>

      {/* Footer */}
      <View style={styles.footer}>
        <Text style={styles.footerText}>
          &copy; {new Date().getFullYear()} Shobhit University, Gangoh. All Rights Reserved.
        </Text>
      </View>
    </ScrollView>
  </View>
 );
};

const styles = StyleSheet.create({
 container: {
  backgroundColor: '#f5f5f5',
 },
 heroSection: {
  backgroundColor: '#6200ea',
  padding: 20,
  alignItems: 'center',
 },
 heroTitle: {
  color: '#fff',
  fontSize: 24,
  fontWeight: 'bold',
  marginBottom: 10,
 },
 heroSubtitle: {
  color: '#fff',
  fontSize: 16,
 },
 section: {
  backgroundColor: '#fff',
  padding: 20,
  marginVertical: 10,
  borderRadius: 8,
  shadowColor: '#000',
  shadowOffset: { width: 0, height: 1 },
  shadowOpacity: 0.2,
  shadowRadius: 1,
  elevation: 2,
 },
 sectionTitle: {
  fontSize: 18,
  fontWeight: 'bold',
  marginBottom: 10,
 },
```

```
  sectionText: {
    fontSize: 14,
    lineHeight: 20,
    marginBottom: 10,
    color: '#333',
  },
  missionVisionContainer: {
    flexDirection: 'row',
    justifyContent: 'space-between',
    marginVertical: 10,
  },
  missionVisionBox: {
    flex: 1,
    margin: 5,
    padding: 15,
    backgroundColor: '#fff',
    borderRadius: 8,
    shadowColor: '#000',
    shadowOffset: { width: 0, height: 1 },
    shadowOpacity: 0.2,
    shadowRadius: 1,
    elevation: 2,
  },
  listItem: {
    fontSize: 14,
    lineHeight: 20,
    color: '#333',
  },
  footer: {
    backgroundColor: '#6200ea',
    padding: 10,
    alignItems: 'center',
  },
  footerText: {
    color: '#fff',
    fontSize: 14,
  },
});

export default About;
```

ContactUs.js

```
import React, { useEffect, useState } from 'react';
import { View, Text, TextInput, TouchableOpacity, StyleSheet, Alert, ScrollView } from 'react-native';
import Navbar from '../src/components/Navbar'; // Adapted Navbar component for React Native
import AsyncStorage from '@react-native-async-storage/async-storage';

const ContactUs = () => {
  const [name,setName]=useState(null)
  const [phone,setPhone]=useState(null)
  const [formMessage,setFormMessage]=useState('')
useEffect(()=>{
  const getUserDetails=async ()=>{
    const keys=['name','phone']
    const data=await AsyncStorage.multiGet(keys)
const userdata=Object.fromEntries(data)
  setName(userdata.name)
  setPhone(userdata.phone)
   }
   getUserDetails();
},[])



  return (
```

```
   <View style={styles.container}>
    {/* Navbar */}
    <Navbar />

    {/* Heading and Instructions */}
    <View style={styles.header}>
     <Text style={styles.title}>Contact Us</Text>
     <Text style={styles.description}>
      We would love to hear from you! Please fill out the form below and we'll be in touch soon.
     </Text>
    </View>

    {/* Contact Form */}
    <View style={styles.form}>
     {/* Name Field */}
     <View style={styles.formGroup}>
      <Text style={styles.label}>Name</Text>
      <TextInput
       style={styles.input}
       value={name}
      editable={false}
      />
     </View>

     {/* Email Field */}
     <View style={styles.formGroup}>
      <Text style={styles.label}>Phone</Text>
      <TextInput
       style={styles.input}
       value={phone}
       editable={false}

      />
     </View>

     {/* Message Field */}
     <View style={styles.formGroup}>
      <Text style={styles.label}>Message</Text>
      <TextInput
       style={[styles.input, styles.textArea]}
       placeholder="Write your message here"
       multiline={true}
       numberOfLines={4}
       value={formMessage}

      />
     </View>

     {/* Submit Button */}
     <TouchableOpacity style={styles.submitButton} >
      <Text style={styles.submitButtonText}>Submit</Text>
     </TouchableOpacity>
    </View>
   </View>
  );
};

const styles = StyleSheet.create({
 container: {
  flex: 1,
  backgroundColor: '#f5f5f5',
 },
 header: {
  marginTop:80,
  marginBottom: 20,
 },
```

```
  title: {
    fontSize: 24,
    fontWeight: 'bold',
    textAlign: 'center',
    marginBottom: 10,
  },
  description: {
    fontSize: 14,
    textAlign: 'center',
    color: '#555',
  },
  form: {
    backgroundColor: '#fff',
    padding: 20,
    borderRadius: 8,
    shadowColor: '#000',
    shadowOffset: { width: 0, height: 2 },
    shadowOpacity: 0.2,
    shadowRadius: 3,
    elevation: 4,
  },
  formGroup: {
    marginBottom: 15,
  },
  label: {
    fontSize: 14,
    color: '#333',
    marginBottom: 5,
  },
  input: {
    borderWidth: 1,
    borderColor: '#ccc',
    borderRadius: 5,
    padding: 10,
    fontSize: 14,
    backgroundColor: '#f9f9f9',
  },
  textArea: {
    height: 100,
    textAlignVertical: 'top',
  },
  submitButton: {
    backgroundColor: '#6200ea',
    padding: 15,
    borderRadius: 5,
    alignItems: 'center',
    marginTop: 10,
  },
  submitButtonText: {
    color: '#fff',
    fontSize: 16,
    fontWeight: 'bold',
  },
});

export default ContactUs;
```

Homepage.js

```
      import React, { useState } from "react";
import {
 View,
 Text,
 StyleSheet,
 Alert,
 TouchableOpacity,
 Image,
 ScrollView,
 ActivityIndicator,
} from "react-native";
import Navbar from "../src/components/Navbar";

import Footer from "../src/components/Footer";
import CreatePdf from "../src/helper/createpdf";
import useDocumentScanner from "../src/helper/useDocumentScanner";

import { width, height } from "../src/wrapper/Dimensions";

const Homepage = () => {
 const { scannedImage, scanDocument, scanMoreDocument } = useDocumentScanner();
 const [loading, setLoading] = useState(false);

 const handleCreatePdf = async () => {
  setLoading(true);
  const result = await CreatePdf();
  setLoading(false);
  if (result) {
   Alert.alert("Sucess", "Pdf Created Sucessfully");
  } else {
   Alert.alert("Error ", "Error in creating pdf please try again");
  }
 };

 return (
  <View style={styles.container}>
   <Navbar />
   <ScrollView contentContainerStyle={styles.mainContainer}>
    {scannedImage.length > 0 ? (
     <View style={styles.scannedImagesContainer}>
      {scannedImage.map((image, index) => (
       <View style={styles.imageContainer} key={index}>
        <Image
         resizeMode="contain"
         style={styles.scannedImage}
         source={{ uri: image }}
        />
       </View>
      ))}
      <View style={styles.footer}>
       <TouchableOpacity
        style={styles.scanButton}
        onPress={scanMoreDocument}
        disabled={loading?true:false}
       >
        <Text style={styles.scanButtonText}>Scan More</Text>
       </TouchableOpacity>
       <TouchableOpacity
```

```
            style={styles.scanButton}
            onPress={handleCreatePdf}
            disabled={loading?true:false}
           >
            <Text style={styles.scanButtonText}>Upload and Save Pdf</Text>
           </TouchableOpacity>
          </View>
         </View>
       ) : (
         <View style={styles.welcomeContainer}>
          <Text style={styles.greeting}>Shobhit University,Gangoh</Text>
          <Text style={styles.subgreetText}>
           Scholarship Management System
          </Text>
          <View style={styles.welcomeFooter}>
       <TouchableOpacity style={styles.scanButton} onPress={scanDocument}>
         <Text style={styles.scanButtonText}>Click Here To Scan Document</Text>
       </TouchableOpacity>

     </View>
        </View>
       )}
     </ScrollView>
     <View style={styles.ActivityIndicatorContainer}>
      <ActivityIndicator
       style={styles.activityIndicatorIcon}
       animating={loading}
       size="100"
      />
     </View>


     {/** footer */}
    {loading?'': <Footer />}
   </View>

 );
};

const styles = StyleSheet.create({
 container: {
  flex: 1,
  backgroundColor: "#fff",
 },
 welcomeContainer: {
  flex: 1,
  alignItems: "center",
  height:height*.9,
 },
 mainContainer: {
  padding: 20,
  flexWrap: "wrap",
  flexDirection: "row",



 },
 scannedImagesContainer:{
flexDirection:'row',
flexWrap:'wrap'
 },
 footerContainer: {
  alignItems: "center",
  marginBottom: 20,
 },
 imageContainer: {
```

```
  margin: 5,
  width: width * 0.4,
 },
 greeting: {
  fontSize: 30,
  fontWeight: "bold",
 },
 scannedImage: {
  width: width * 0.4,

  height: height * 0.3,
 },

 scanButton: {
  backgroundColor: "#4CAF50",
  paddingVertical: 10,
  paddingHorizontal: 20,
  borderRadius: 5,
  flexDirection: "row",
 },
 scanButtonText: {
  color: "#fff",
  fontSize: 18,
 },
 subgreetText: {
  fontSize: 14,
  marginBottom: 20,
 },
 footer: {
  marginBottom: 20,
  flexDirection: "row",
  width: width,
  alignItems:'center',
  justifyContent:'space-around',


 },
 welcomeFooter:{
  marginBottom: 20,
  flexDirection: "row",
  width: width,
  alignItems:'center',
  justifyContent:'space-around',
  position:'absolute',
  bottom:0
 },
 ActivityIndicatorContainer: {
  flex: 1,
  position: "absolute",
  top: height * 0.5,
  left: width * 0.5,
  bottom: height * 0.5,
  right: width * 0.5,
 },
 alreadyText: {
  fontSize: 20,
  marginTop: 20,
 },
});

export default Homepage;
```

ManagePdfFiles.js

```
import React, { useEffect, useState } from 'react';
import * as fs from 'expo-file-system';
import { Text, View, TouchableOpacity, FlatList, StyleSheet, Alert, Image, ScrollView } from 'react-native';
import Navbar from '../src/components/Navbar';
import Footer from '../src/components/Footer';
import Icon from 'react-native-vector-icons/MaterialIcons';
import { AntDesign } from '@expo/vector-icons';
import PdfViewer from '../src/helper/PdfViewer';
import prompt from 'react-native-prompt-android';
import { useRouter } from 'expo-router';
import { width, height } from '../src/wrapper/Dimensions';
import * as DocumentPicker from 'expo-document-picker'
import * as Sharing from 'expo-sharing'




const managePdfFiles = () => {
 const imageDirectory = fs.documentDirectory + 'scannedImages/'; //for direct use in jsx
 const router = useRouter()

 const [isDirectoryExist, setIsDirectoryExist] = useState(false);
 const [pdfFiles, setPdfFiles] = useState([]);

 const [metadata,setMetadata]=useState([])
 const formatDate=(timestamp)=>{
  let date= new Date(timestamp*1000)

  let formattedDate = date.toLocaleDateString('en-GB', {
   day: 'numeric',
   month: 'long',
   year: 'numeric'
  });
  return formattedDate

 }


 useEffect(() => {
  const checkDirectories = async () => {

   try {
    // Checking the PDF directory
    const pdfDirectoryPath = fs.documentDirectory + 'scannedPdf';
    const pdfInfo = await fs.getInfoAsync(pdfDirectoryPath);
    if (pdfInfo.exists && pdfInfo.isDirectory) {
     const pdfFilesList = await fs.readDirectoryAsync(pdfDirectoryPath);
     console.log(pdfFilesList);
     setPdfFiles(pdfFilesList);  // Set the list of PDF files
     setIsDirectoryExist(true);  // Directory exists

     const info=await Promise.all(
       pdfFilesList.map(async value=>{
        const data= await fs.getInfoAsync(pdfDirectoryPath+'/'+value)
      return{
       uri:(data.uri.replace('file:///data/user/0/','sdcard/')),
       creationTime:formatDate(data.modificationTime),
       size:data.size
      }
       })
     )
```

```
      setMetadata(info)



    } else {
      setIsDirectoryExist(false);  // Directory doesn't exist
      setPdfFiles([]);  // Clear files if directory doesn't exist
    }


  } catch (error) {
    console.log('Error while checking directories', error);
    setIsDirectoryExist(false);  // In case of an error, set the state accordingly
    setPdfFiles([]);  // Clear PDF files
  }


 };

 checkDirectories();
}, []);


const viewPdfHandler = (item) => {
 console.log(item)
 const fileUri = fs.documentDirectory + 'scannedPdf/' + item
 console.log(fileUri)
 PdfViewer(fileUri)
}


return (
 <View style={styles.container}>
   <Navbar />
   <View style={styles.pdfContainer}>
     <Text style={styles.header}>Your PDF Files</Text>

          <ScrollView>
      {isDirectoryExist ? (
        metadata.map((value,index)=>(
          <View key={index}>

            <TouchableOpacity  onPress={()=>viewPdfHandler(pdfFiles[index])}>
            <View style={styles.subcontainer}>
              <View style={styles.ImageContainer}>
                <Image
                style={styles.PdfIcon}
                source={{uri:imageDirectory+(pdfFiles[index].replace('.pdf','.png'))}}/>
              </View>
              <View style={styles.pdfDetails}>
              <Text style={styles.pdfName}>{pdfFiles[index]}</Text>
             <View style={styles.pdfDAndS}>
             <Text style={styles.pdfCreationDate}>{value.creationTime}</Text>
             <Text style={styles.pdfSize}>{Math.round(value.size/1024/1024*100)/100} MB</Text>
             </View>


              </View>


            </View>
            </TouchableOpacity>
          </View>
        ))
      ) : (
        <Text>Directory doesn't exist</Text>
```

```
        )}

      </ScrollView>
    </View>


    <Footer />

  </View>
 );
};

const styles = StyleSheet.create({
 container: {
  flex: 1,
 },
 pdfContainer: {
  flex: 1,
  padding: 20,
  backgroundColor: '#f9f9f9',
  marginBottom: 20,
  borderRadius: 10,
  shadowColor: '#000',
  shadowOffset: { width: 0, height: 2 },
  shadowOpacity: 0.2,
  shadowRadius: 5,
 },
 header: {
  fontSize: 20,
  fontWeight: 'bold',
  marginBottom: 10,
  color: '#333',
 },
 pdfFileText: {

  fontSize: 16,
  marginVertical: 5,
  color: '#666',
  borderWidth: 1,
  padding: 10,
  borderRadius: 5,
  width: width * .55

 },
 createPdfContainer: {
  padding: 20,
  alignItems: 'center',
  marginTop: 10,
 },
 createButton: {
  backgroundColor: '#4CAF50',
  paddingVertical: 12,
  paddingHorizontal: 30,
  borderRadius: 8,
  elevation: 3,
 },
 buttonText: {
  color: 'white',
  fontSize: 16,
  fontWeight: 'bold',
 },
 pdfFiles: {

  flex: 1,
  flexDirection: 'row',
```

```
   justifyContent: 'space-between',
   alignItems: 'center',

   marginBottom: 10
  },
  AIBG: {
   backgroundColor: 'white',
  },
 subcontainer:{
borderBottomWidth:0.2,
flexDirection:'row',
marginBottom:10,
width:'100%',
borderTopWidth:.2,
borderRightWidth:.2,


 },
ImageContainer:{
width:'25%',

alignItems:'center',
justifyContent:'center',
borderTopWidth:.2,
borderRightWidth:.2,
borderLeftWidth:.2

},
PdfIcon:{
width:'100%',
height:100,
 objectFit:'contain',
 marginVertical:'auto',

},
pdfDetails:{
 marginLeft:10,
 width:'60%',
 justifyContent:'center'
}
});

export default managePdfFiles;
```

profile.js

```
import React, { useState, useEffect } from 'react';
import { View, Text, Image, TouchableOpacity, TextInput, Alert, ScrollView } from 'react-native';
import { styles } from '../src/stylesheets/Profile';
import Navbar from '../src/components/Navbar';
import AsyncStorage from '@react-native-async-storage/async-storage';

const ProfilePage = () => {
 const [name, setName] = useState(null);
 const [image, setImage] = useState(null);
 const [registrationId, setRegistrationId] = useState(null);
 const [programmeName, setProgrammeName] = useState(null);
 const [phone, setPhone] = useState(null);
 const [email, setEmail] = useState(null);
const [ukid,setUkid]=useState(null)
const [token,setToken]=useState(null)
```

```
const [currentPassword, setCurrentPassword] = useState('');
const [newPassword, setNewPassword] = useState('');
const [repeatPassword, setRepeatPassword] = useState('');

// Fetch user data from AsyncStorage inside useEffect
useEffect(() => {
 const getUserData = async () => {
  const keys = ['name', 'phone', 'image', 'registrationId', 'programmeName', 'email', 'Token'];
  const data = await AsyncStorage.multiGet(keys);
  const parsedData = Object.fromEntries(data);

  setName(parsedData.name);
  setImage(parsedData.image);
  setRegistrationId(parsedData.registrationId);
  setProgrammeName(parsedData.programmeName);
  setEmail(parsedData.email);
  setPhone(parsedData.phone);
 };

 getUserData();
}, []);

const handlePasswordChange = () => {
 // Simple password validation
 if (newPassword !== repeatPassword) {
  Alert.alert('Error', 'New password and repeat password do not match');
 } else if (newPassword.length < 6) {
  Alert.alert('Error', 'New password must be at least 6 characters');
 } else {
  // Handle password change logic here
  Alert.alert('Success', 'Password has been changed successfully');
 }
};


return (
 <View style={styles.container}>
  <Navbar />
  {/* Profile Picture */}
  <View style={styles.profilePictureContainer}>
   <Image source={{ uri: image }} style={styles.profilePicture} />
  </View>

  {/* User Details */}
  <View style={styles.userInfo}>
   <Text style={styles.userName}>{name}</Text>
   <Text>{phone}</Text>
   <Text>{registrationId} || {programmeName}</Text>
   <Text style={styles.userEmail}>{email}</Text>
  </View>

  {/* Password Change Form */}
  <View style={styles.passwordChangeContainer}>
   <Text style={styles.sectionTitle}>Change Password</Text>

    <TextInput
     style={styles.input}
     placeholder="Current Password"
     secureTextEntry
     value={currentPassword}
     onChangeText={setCurrentPassword}
    />
    <TextInput
     style={styles.input}
     placeholder="New Password"
     secureTextEntry
```

```
        value={newPassword}
        onChangeText={setNewPassword}
      />
      <TextInput
        style={styles.input}
        placeholder="Repeat New Password"
        secureTextEntry
        value={repeatPassword}
        onChangeText={setRepeatPassword}
      />

      <TouchableOpacity
        style={styles.button}
        onPress={handlePasswordChange}
      >
        <Text style={styles.buttonText}>Change Password</Text>
      </TouchableOpacity>
    </View>

    {/* Action Buttons */}
    <View style={styles.actionButtons}>
      {/* Add additional action buttons if needed */}
    </View>
  </View>
 );
};

export default ProfilePage;
```

Status.js

```
import React, { useEffect, useState } from "react";
import { View, Text, StyleSheet, TouchableOpacity, TextInput, Alert } from "react-native";
import Navbar from "../src/components/Navbar";
import Footer from "../src/components/Footer";
import { width, height } from "../src/wrapper/Dimensions";
import WebView from "react-native-webview";

const Status = () => {
  const [session,setSession]=useState('')
  const [showWebView,setShowWebView]=useState(false)
  const [status,setStatus]=useState(null)
  const refreshButton=()=>{
    setStatus('pending')

  }
  useEffect(()=>{

    setStatus(null)
  },[])

  const handleWebViewPress = () => {
    if (session) {
      const sessionLength = session.length;

      if (sessionLength >= 4 && sessionLength < 5) {
        let secondNumber=Number(session[1])
        const fourthNumber=Number(session[3])
        // console.log(secondNumber,fourthNumber)
        secondNumber+=1
```

```
            if(secondNumber===fourthNumber){

              setShowWebView(true)
              return
            }{
              Alert.alert('Error', 'Invalid Session');
              return
            }

          } else {
            Alert.alert('Error', 'Invalid Session');
            return
          }
        } else {
          Alert.alert('Error', 'Invalid Session');
          return
        }
      };

      return (
        <View style={styles.statusWrapper}>
          <Navbar />
          {
            showWebView?(
              <View></View>
            ):(
              <View style={styles.statusContainer}>
            <Text style={styles.textTitle}>Application Status</Text>

            {/* Status Messages */}
            {
            status && status==='approved'? <Text style={styles.textSuccess}>Your application has been
approved.</Text>:(status==='pending'?<Text style={styles.textPending}>Your application is not approved yet.</Text>:<Text
style={styles.textError}>It looks like you have not uploaded the pdf.</Text>)



            }

            {/* Refresh Button */}
            <TouchableOpacity onPress={refreshButton} style={styles.refreshButton}>
              <Text style={styles.refreshButtonText}>Refresh your application status</Text>
            </TouchableOpacity>
          </View>
            )
          }
          {
          showWebView?(

            <WebView source={{uri : `https://scholarship.up.gov.in/status${session}.aspx`}}
             style={styles.webViewContainer}
             scalesPageToFit={true}

             />

          ):(
            <View style={styles.toShowWebViewContainer} >

              <Text style={{fontSize:width*.04,fontWeight:'bold'}}>Note:  The status on U.P. ScholarShip Website</Text>
              <Text style={{fontSize:width*.04,fontWeight:'bold'}}>**********Instruction to enter session**********</Text>
              <Text style={{fontSize:width*.04,fontWeight:'500'}}>If your session is 2024/25 Enter : 2425 </Text>
              <Text></Text>
              <TextInput

              style={{borderWidth:.5,width:width*.6,borderRadius:5,paddingHorizontal:12}}
              onChangeText={setSession}
```

```
            value={session}
            keyboardType="numeric"
            placeholder="Enter Session Value"
            />

          <TouchableOpacity onPress={handleWebViewPress} style={styles.refreshButton}>
          <Text style={styles.refreshButtonText}>Check your status</Text>
        </TouchableOpacity>
        </View>
        )
       }
        <Footer />
    </View>
  );
};

const styles = StyleSheet.create({
  statusWrapper: {
    width: width,
    flex: 1,
    height: height,
    backgroundColor: "#f8f8f8",
    bottom:0,


  },
  statusContainer: {
    flex:1,
    // height:height*.3,
    alignItems:'center'
  },
  textTitle: {
    padding: 15,
    fontSize: 35,
    fontWeight: "bold",
    color: "#333", // Dark color for the title
    textAlign: "center",
  },
  textError: {
    color: "red",
    fontSize: 18,
    textAlign: "center",
    marginBottom: 10,
  },
  textSuccess: {
    color: "green",
    fontSize: 18,
    textAlign: "center",
    marginBottom: 10,
  },
  textPending:{
    color: "#F29339",
    fontSize: 18,
    textAlign: "center",
    marginBottom: 10,

  },
  refreshButton: {
    marginTop: 20,
    paddingVertical: 10,
    paddingHorizontal: width*.05,
    backgroundColor: "#3498db", // Blue color for the button
    borderRadius: 5,
    shadowColor: "#000",
    shadowOffset: { width: 0, height: 3 },
    shadowOpacity: 0.1,
```

```
    shadowRadius: 5,
    elevation: 5, // For Android shadow
    width:width*.6,

  },
  refreshButtonText: {
    width:width*.5,
    color: "white",
    fontSize: 16,
    fontWeight: "bold",
    textAlign:'center'


  },
  webViewWrapper:{
    justifyContent:'center',
    alignItems:'center',
    borderWidth:1,
  flex:1,
  },
  webViewContainer:{

  },
  toShowWebViewContainer:{
    flex:1,
    alignItems:'center'
  }
});

export default Status;
```