Automated Platform For Undergraduate Admissions

FYP- II

Waleed Ahmed (1734-2021)

Dua Rahim (1597-2021)

Zoya Sayeed (1696-2021)

Supervisor

Iqbal Uddin Khan

Hamdard University

Outline

Problem Statement

"The entirely manual undergraduate admissions process at Hamdard University results in significant inefficiencies, errors, and delays, highlighting the urgent need for an automated solution to streamline operations and improve overall efficiency."

- The process of collecting and entering student information into Excel sheets is prone to errors and duplications, with no automated verification or redundancy elimination.
- Admission fee payments are manually verified, leading to time-consuming processes,errors, and delays in scheduling admission tests.
- Manual scheduling of admission tests results in frequent conflicts and miscommunication with applicants.
- Communication about test dates, eligibility, and admission offers is handled manually, causing significant delays and a poor user experience.

Project Objective

The proposed project aims to develop an automated platform for undergraduate admissions, addressing inefficiencies and errors in the current manual process. It will enhance accuracy by minimizing data entry errors and ensure fairness by uniformly applying admission criteria. The system will automate tasks like application processing, payment verification, and communication, improving efficiency and reducing delays. A user-friendly interface will allow applicants to submit and track their applications online easily. The platform will ensure robust data management, scalability, and compliance with relevant policies. Additionally, it will provide real-time updates, improving the overall experience for applicants and administrative staff.

Project **Milestones**

- Database Design
- Back-end
- Testing Phase
- Deployment
- Documentation
- Reporting
- Continuous Improvement
- Maintenance

Project **Deliverables**

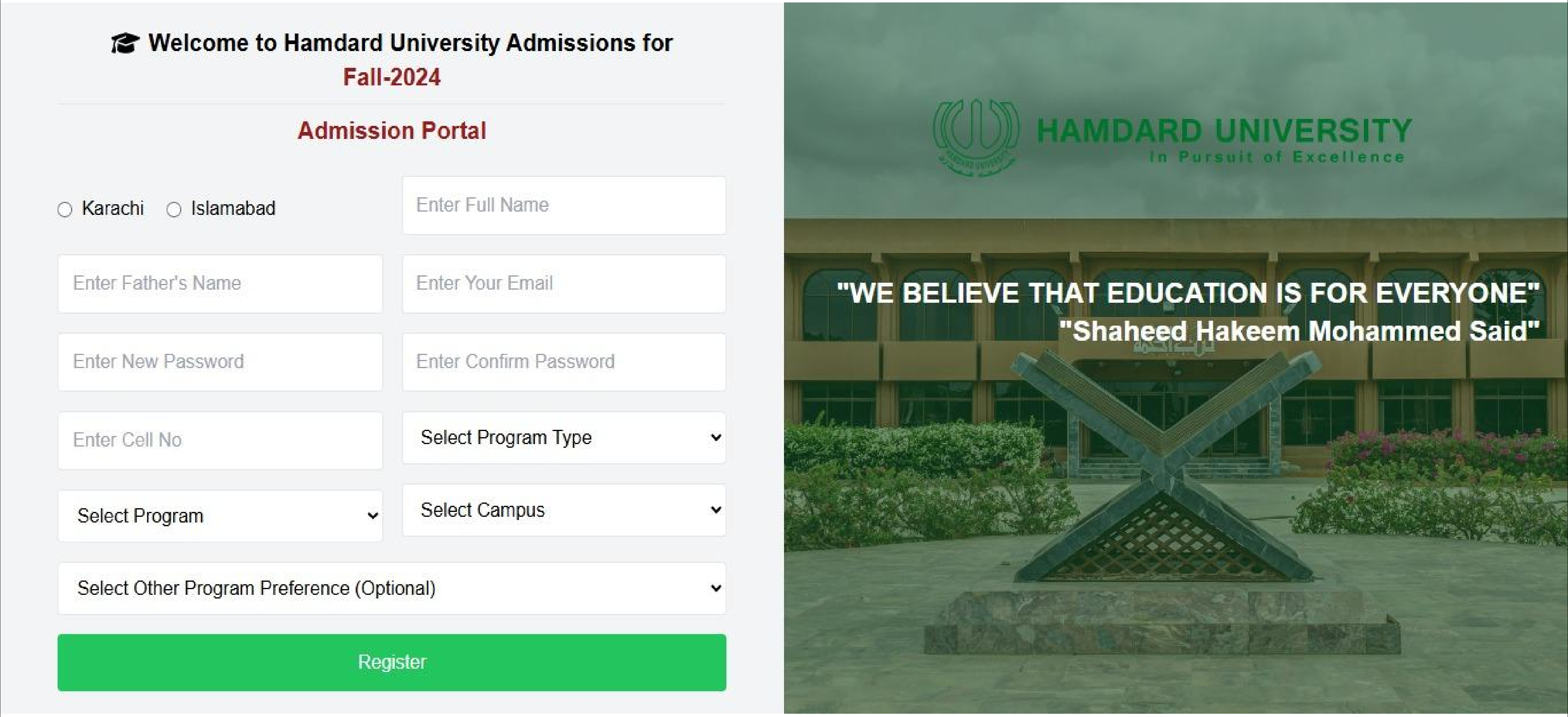
FYP-I Evaluation

- SRS Document
- Project plan
- UI Design
- Front-End
- Project Report-I

FYP-II Evaluation

- Back-End
- Database design
- Role base data shown on web App
- Hosting and Deployement
- Project Report – II
- Reporting
- Emailing

Project Front-end



Welcome to Hamdard University Admissions for **Fall-2024**

Admission Portal

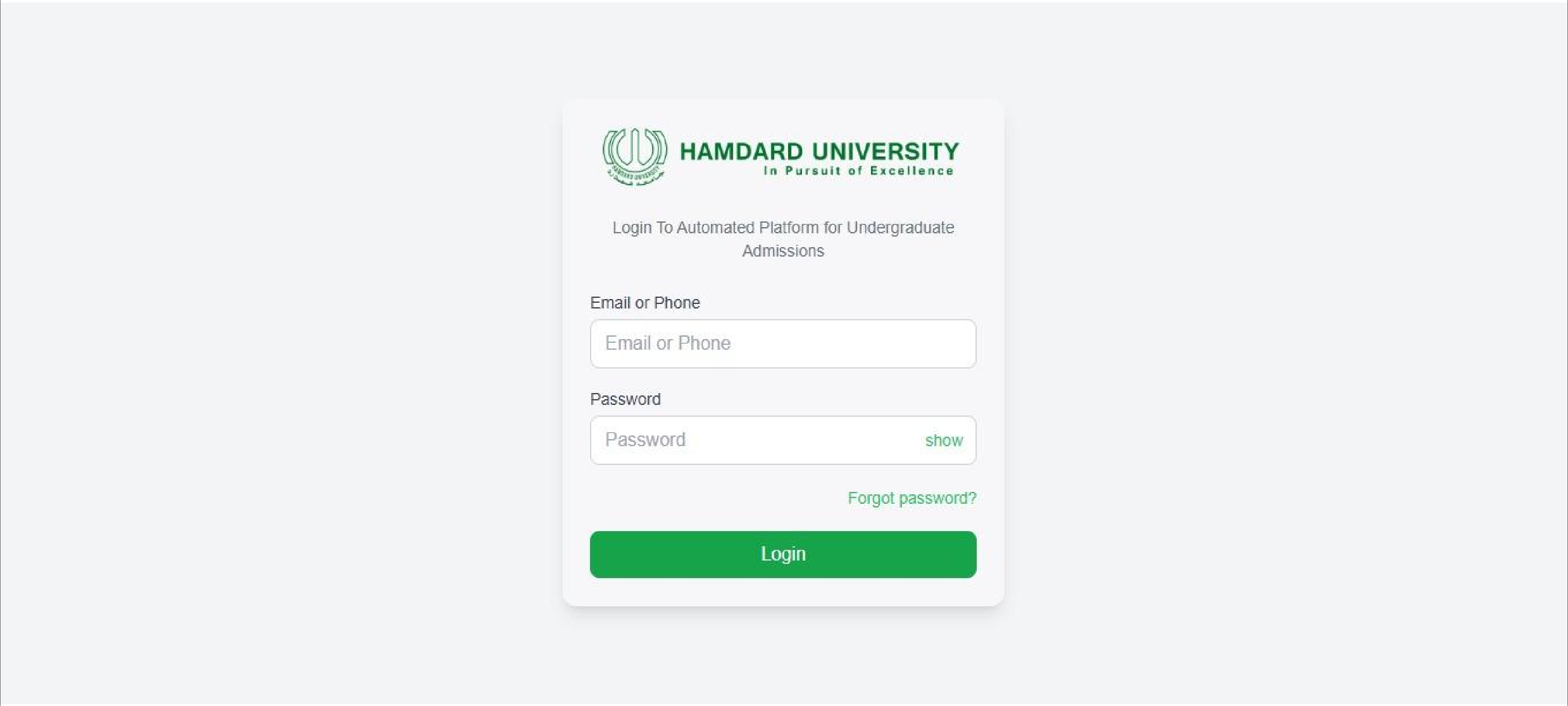
○ Karachi ○ Islamabad

	Enter Full Name
Enter Father's Name	Enter Your Email
Enter New Password	Enter Confirm Password
Enter Cell No	Select Program Type
Select Program	Select Campus

Select Other Program Preference (Optional)

Register
HAMDARD UNIVERSITY
In Pursuit of Excellence
"WE BELIEVE THAT EDUCATION IS FOR EVERYONE"
"Shaheed Hakeem Mohammed Said"

Project **Front-end**



HAMDARD UNIVERSITY
In Pursuit of Excellence

Login To Automated Platform for Undergraduate Admissions

Email or Phone

Email or Phone

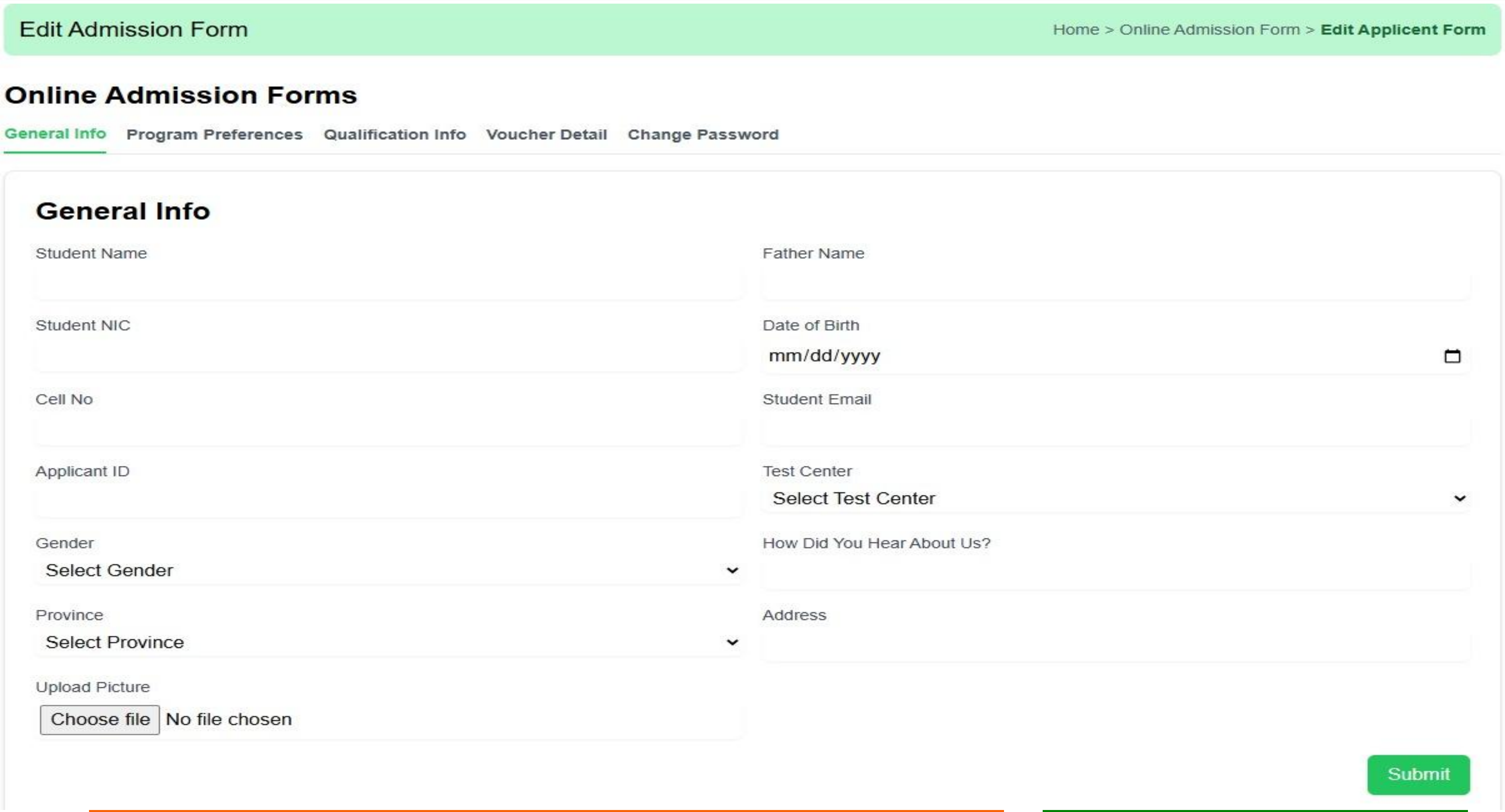
Password

Password show

Forgot password?

Login

Project Front-end


Edit Admission Form

Home > Online Admission Form > **Edit Applicent Form**

Online Admission Forms

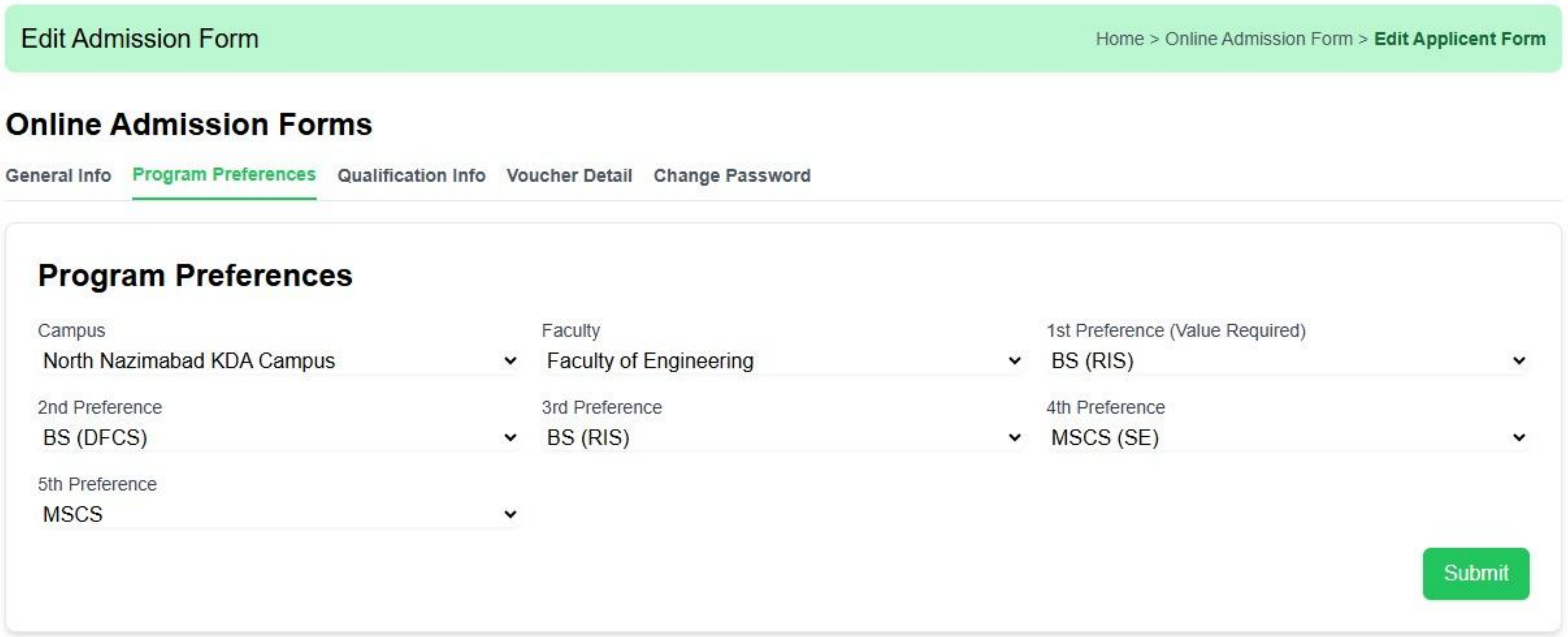
General Info | Program Preferences | Qualification Info | Voucher Detail | Change Password

General Info

Student Name	Father Name
Student NIC	Date of Birth: mm/dd/yyyy
Cell No	Student Email
Applicant ID	Test Center: Select Test Center
Gender: Select Gender	How Did You Hear About Us?
Province: Select Province	Address
Upload Picture: Choose file No file chosen	

Submit

Project Front-end



Edit Admission Form

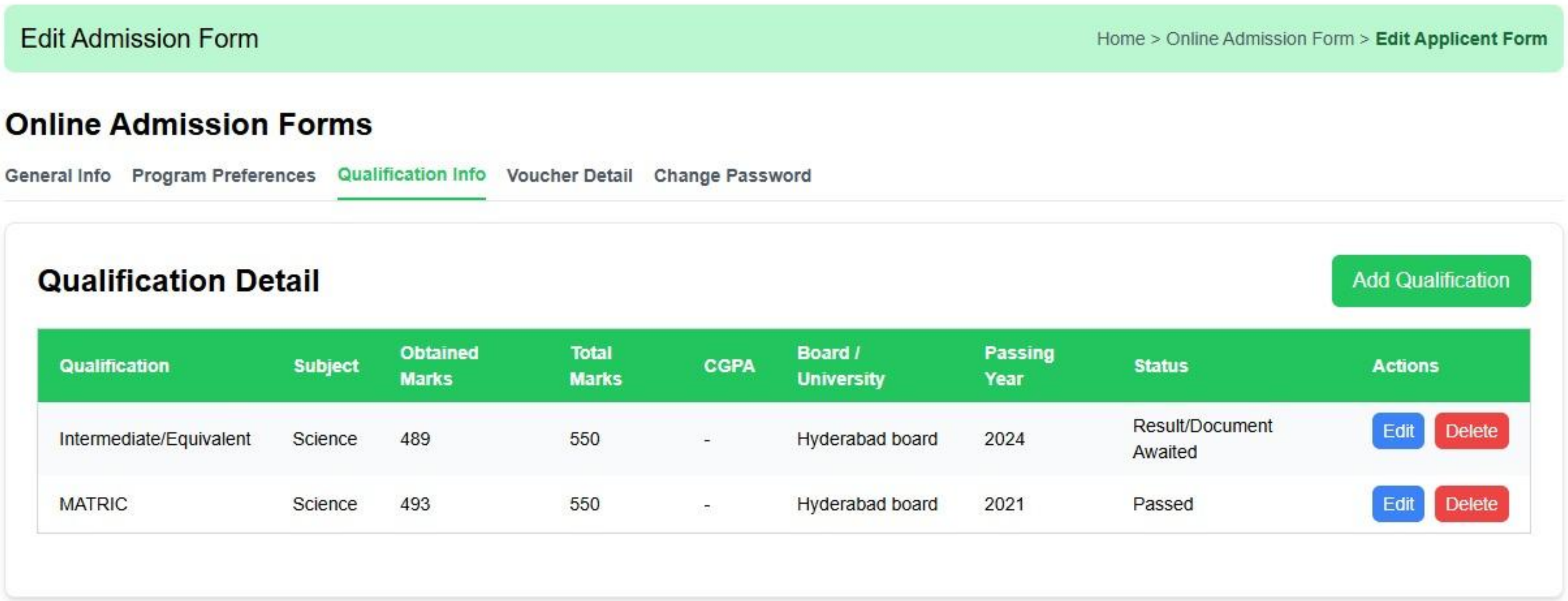
Home > Online Admission Form > **Edit Applicent Form**

Online Admission Forms

Program Preferences

Campus North Nazimabad KDA Campus	Faculty Faculty of Engineering	1st Preference (Value Required) BS (RIS)
2nd Preference BS (DFCS)	3rd Preference BS (RIS)	4th Preference MSCS (SE)
5th Preference MSCS		

Project Front-end



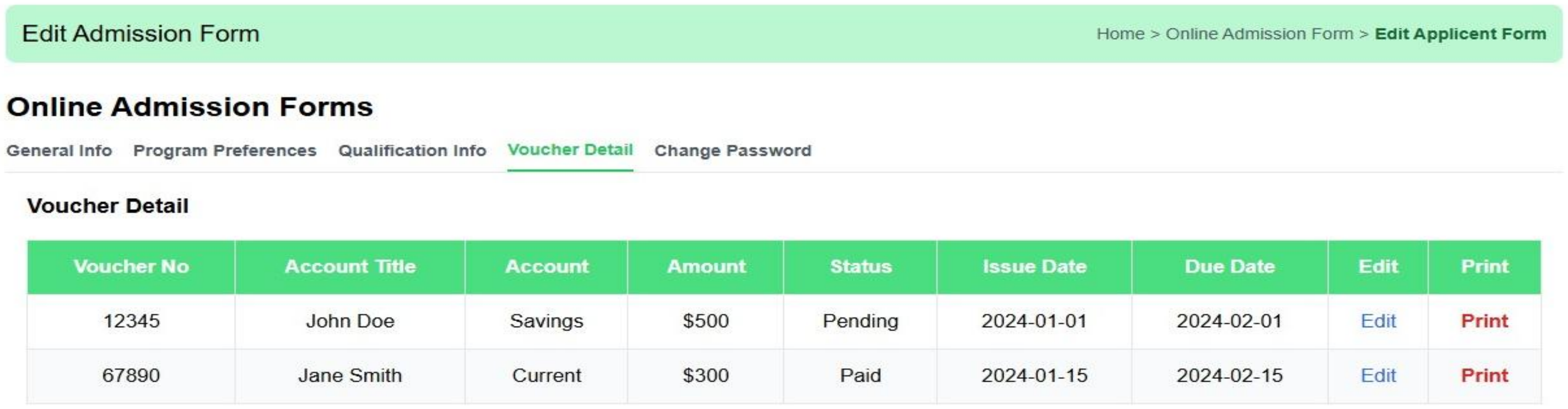
Edit Admission Form

Online Admission Forms

Qualification Detail

Qualification	Subject	Obtained Marks	Total Marks	CGPA	Board / University	Passing Year	Status	Actions
Intermediate/Equivalent	Science	489	550	-	Hyderabad board	2024	Result/Document Awaited	Edit Delete
MATRIC	Science	493	550	-	Hyderabad board	2021	Passed	Edit Delete

Project Front-end

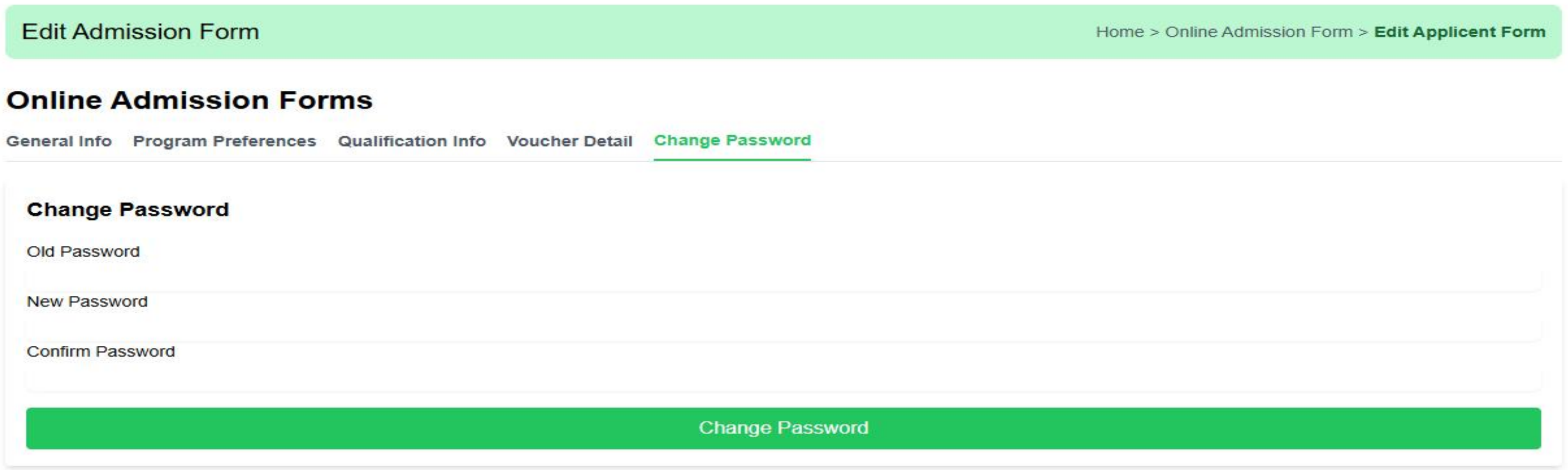


Edit Admission Form

Online Admission Forms

Voucher Detail

Voucher No	Account Title	Account	Amount	Status	Issue Date	Due Date	Edit	Print
12345	John Doe	Savings	$500	Pending	2024-01-01	2024-02-01	Edit	Print
67890	Jane Smith	Current	$300	Paid	2024-01-15	2024-02-15	Edit	Print



Edit Admission Form

Online Admission Forms

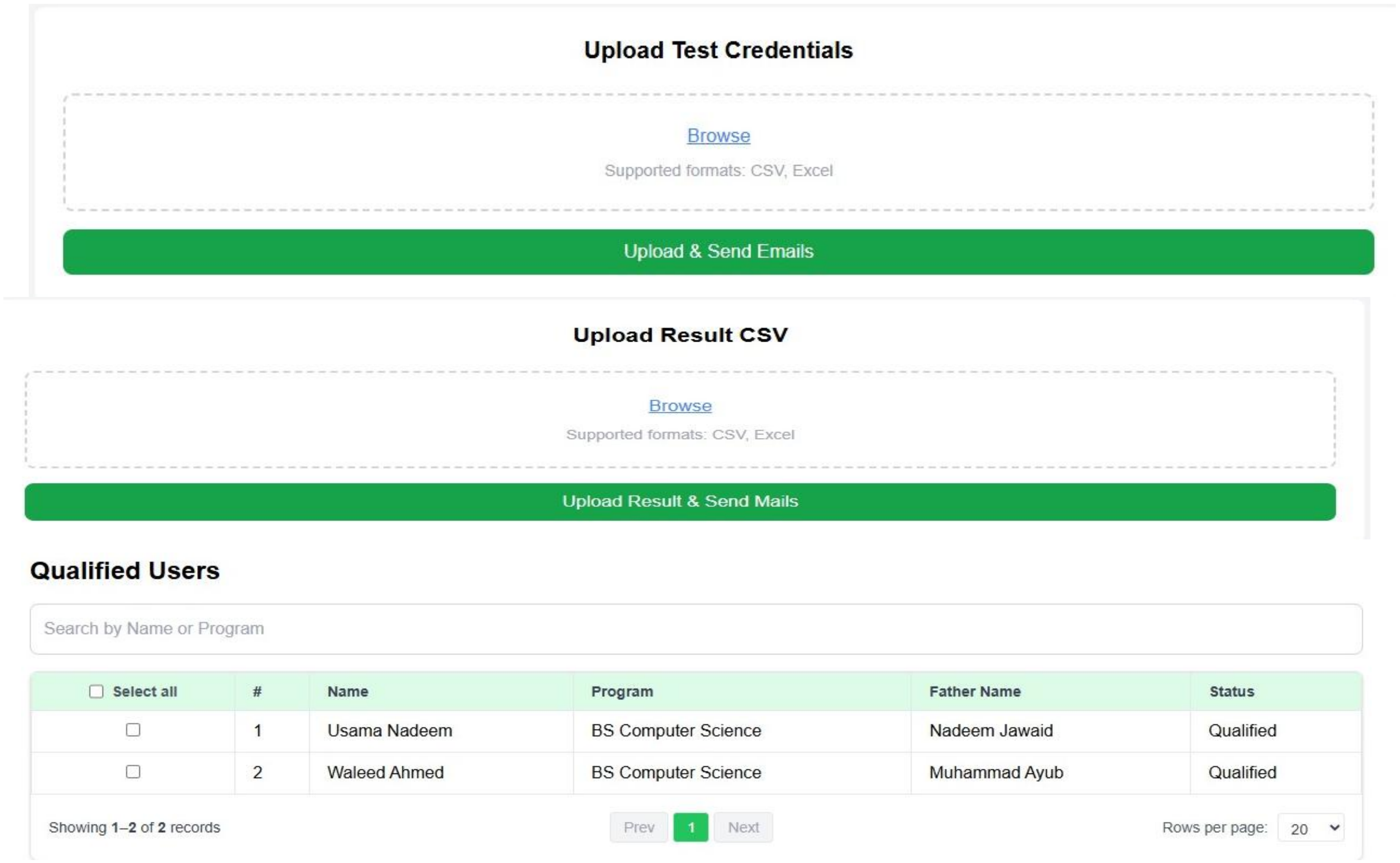
Change Password

Old Password

New Password

Confirm Password

Project Front-end



Upload Test Credentials
Browse
Supported formats: CSV, Excel
Upload & Send Emails
Upload Result CSV
Browse
Supported formats: CSV, Excel
Upload Result & Send Mails

Qualified Users

Search by Name or Program

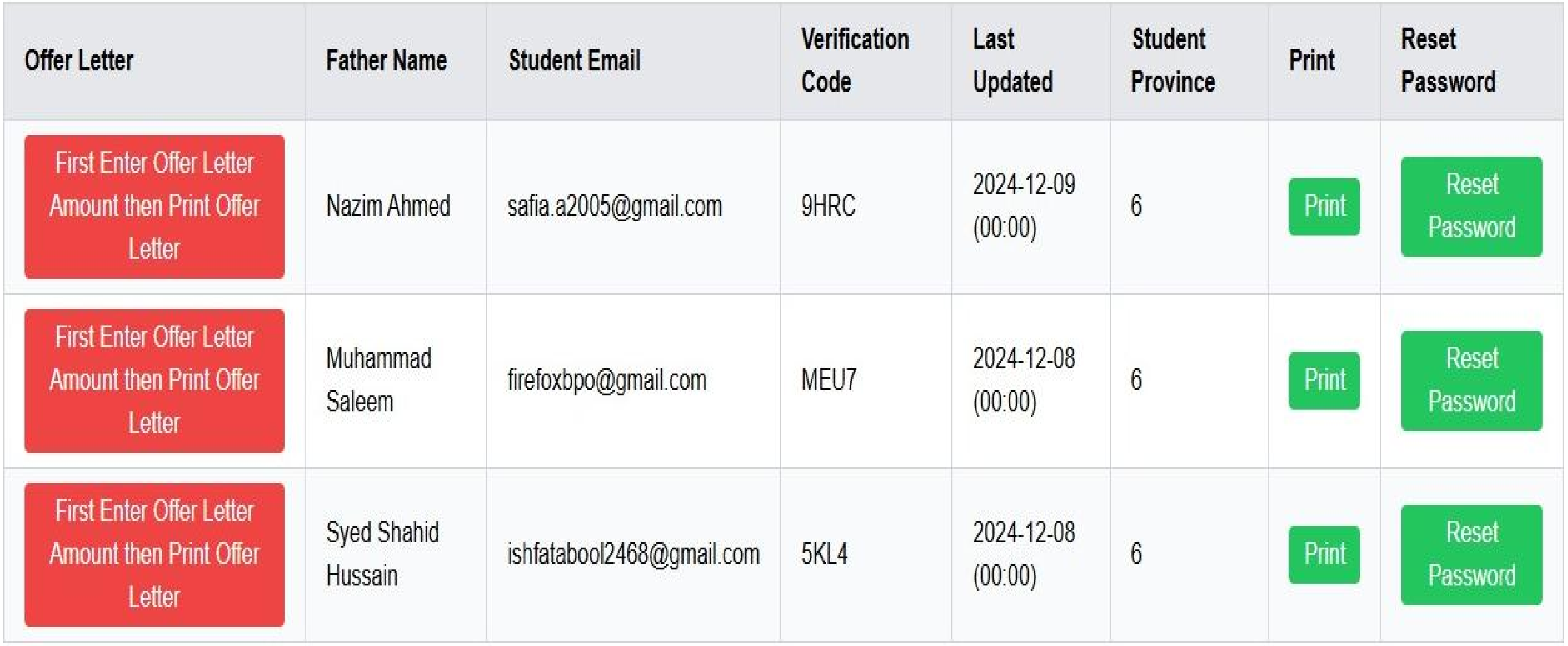
Select all	#	Name	Program	Father Name	Status
☐	1	Usama Nadeem	BS Computer Science	Nadeem Jawaid	Qualified
☐	2	Waleed Ahmed	BS Computer Science	Muhammad Ayub	Qualified

Showing **1–2** of **2** records

Prev 1 Next

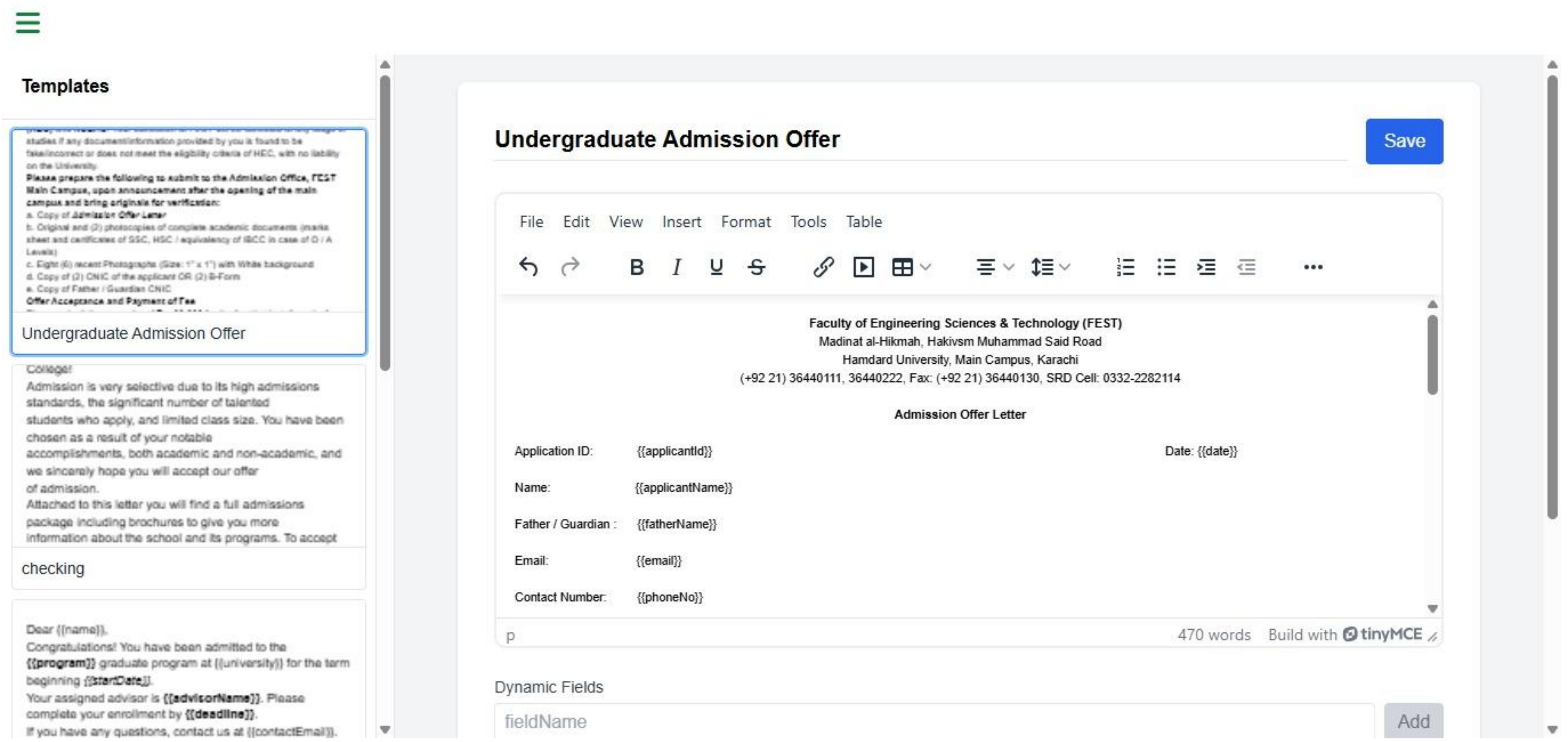
Rows per page: 20

Project Front-end



Offer Letter	Father Name	Student Email	Verification Code	Last Updated	Student Province	Print	Reset Password
First Enter Offer Letter Amount then Print Offer Letter	Nazim Ahmed	safia.a2005@gmail.com	9HRC	2024-12-09 (00:00)	6	Print	Reset Password
First Enter Offer Letter Amount then Print Offer Letter	Muhammad Saleem	firefoxbpo@gmail.com	MEU7	2024-12-08 (00:00)	6	Print	Reset Password
First Enter Offer Letter Amount then Print Offer Letter	Syed Shahid Hussain	ishfatabool2468@gmail.com	5KL4	2024-12-08 (00:00)	6	Print	Reset Password

Project Front-end



Templates

studies if any document/information provided by you is found to be fake/incorrect or does not meet the eligibility criteria of HEC, with no liability on the University.
Please prepare the following to submit to the Admission Office, FEST Main Campus, upon announcement after the opening of the main campus and bring originals for verification:
a. Copy of **Admission Offer Letter**
b. Original and (2) photocopies of complete academic documents (marks sheet and certificates of SSC, HSC / equivalency of IBCC in case of O / A Levels)
c. Eight (8) recent Photographs (Size: 1" x 1") with White background
d. Copy of (2) CNIC of the applicant OR (2) B-Form
e. Copy of Father / Guardian CNIC
Offer Acceptance and Payment of Fee

Undergraduate Admission Offer

College!
Admission is very selective due to its high admissions standards, the significant number of talented students who apply, and limited class size. You have been chosen as a result of your notable accomplishments, both academic and non-academic, and we sincerely hope you will accept our offer of admission.
Attached to this letter you will find a full admissions package including brochures to give you more information about the school and its programs. To accept

checking

Dear {{name}},
Congratulations! You have been admitted to the **{{program}}** graduate program at {{university}} for the term beginning **{{startDate}}**.
Your assigned advisor is **{{advisorName}}**. Please complete your enrollment by **{{deadline}}**.
If you have any questions, contact us at {{contactEmail}}.

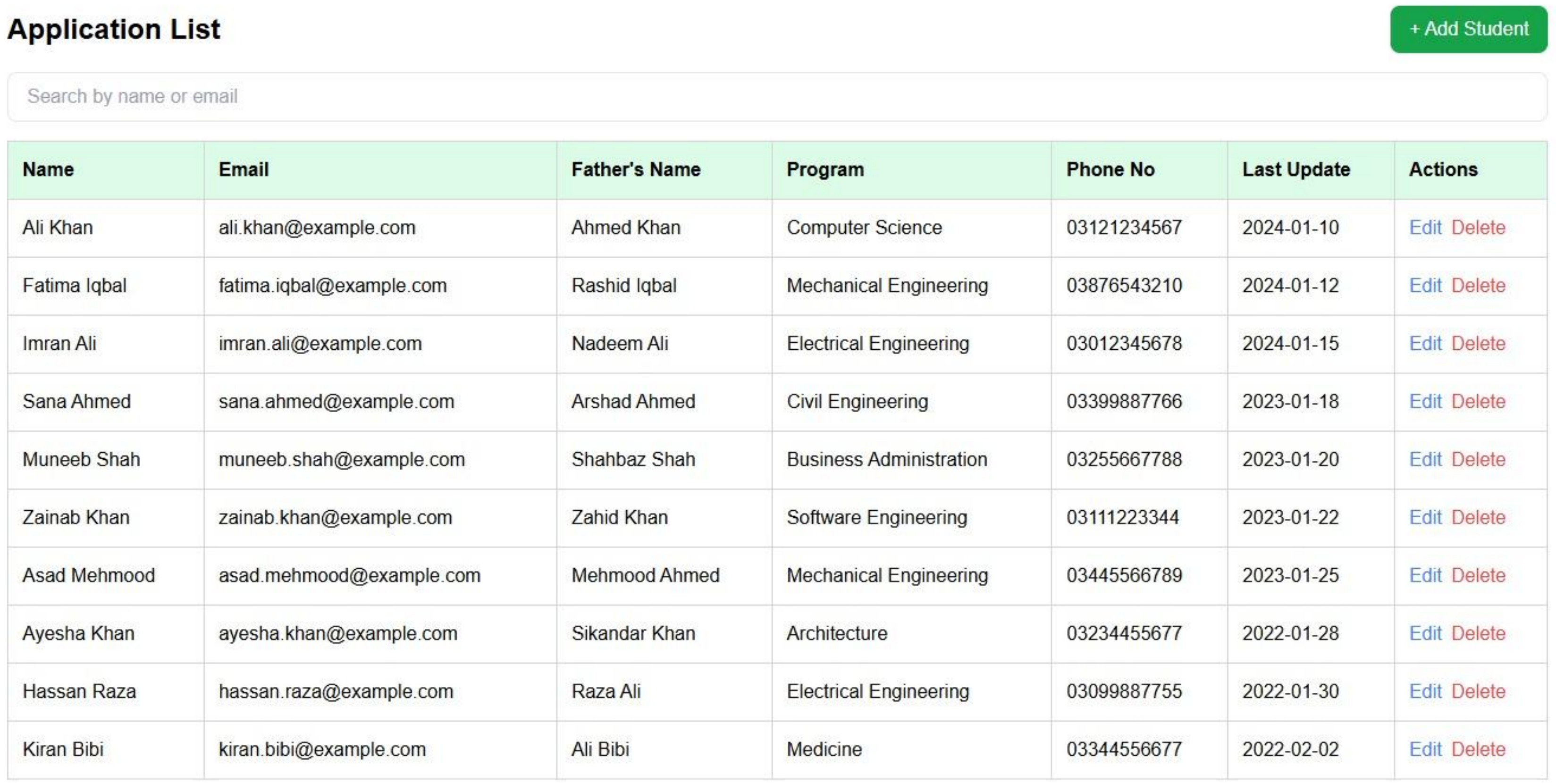
Undergraduate Admission Offer

Faculty of Engineering Sciences & Technology (FEST)
Madinat al-Hikmah, Hakivsm Muhammad Said Road
Hamdard University, Main Campus, Karachi
(+92 21) 36440111, 36440222, Fax: (+92 21) 36440130, SRD Cell: 0332-2282114

Admission Offer Letter

Application ID:	{{applicantId}}	Date: {{date}}
Name:	{{applicantName}}	
Father / Guardian :	{{fatherName}}	
Email:	{{email}}	
Contact Number:	{{phoneNo}}	

Project Front-end



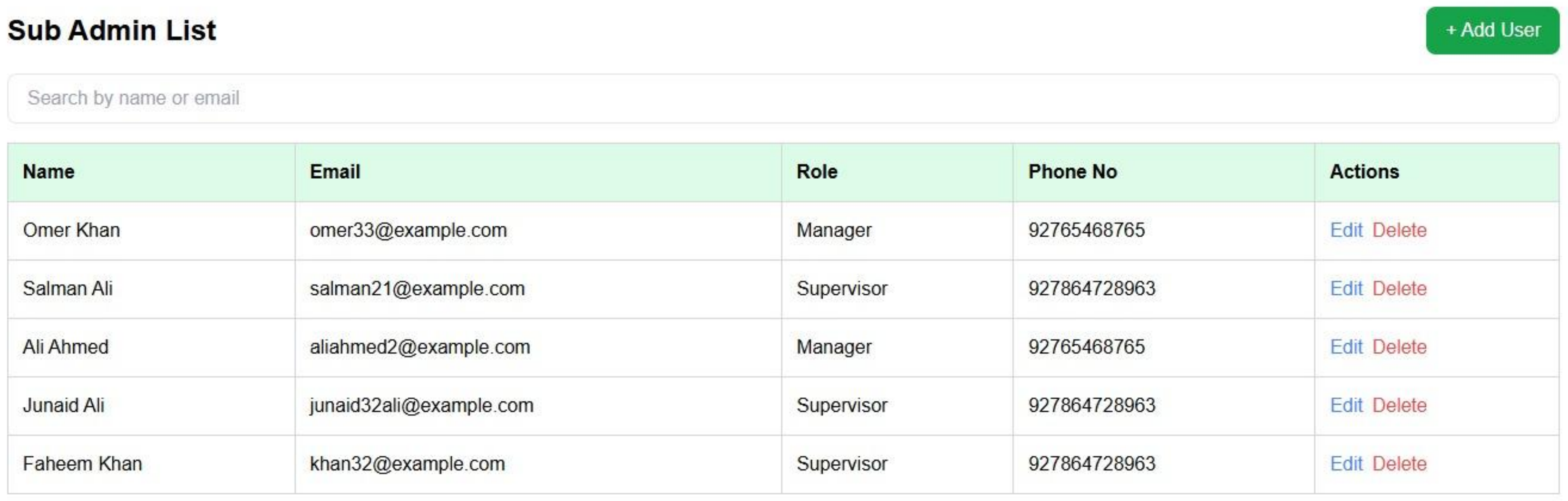
Application List

+ Add Student

Search by name or email

Name	Email	Father's Name	Program	Phone No	Last Update	Actions
Ali Khan	ali.khan@example.com	Ahmed Khan	Computer Science	03121234567	2024-01-10	Edit Delete
Fatima Iqbal	fatima.iqbal@example.com	Rashid Iqbal	Mechanical Engineering	03876543210	2024-01-12	Edit Delete
Imran Ali	imran.ali@example.com	Nadeem Ali	Electrical Engineering	03012345678	2024-01-15	Edit Delete
Sana Ahmed	sana.ahmed@example.com	Arshad Ahmed	Civil Engineering	03399887766	2023-01-18	Edit Delete
Muneeb Shah	muneeb.shah@example.com	Shahbaz Shah	Business Administration	03255667788	2023-01-20	Edit Delete
Zainab Khan	zainab.khan@example.com	Zahid Khan	Software Engineering	03111223344	2023-01-22	Edit Delete
Asad Mehmood	asad.mehmood@example.com	Mehmood Ahmed	Mechanical Engineering	03445566789	2023-01-25	Edit Delete
Ayesha Khan	ayesha.khan@example.com	Sikandar Khan	Architecture	03234455677	2022-01-28	Edit Delete
Hassan Raza	hassan.raza@example.com	Raza Ali	Electrical Engineering	03099887755	2022-01-30	Edit Delete
Kiran Bibi	kiran.bibi@example.com	Ali Bibi	Medicine	03344556677	2022-02-02	Edit Delete

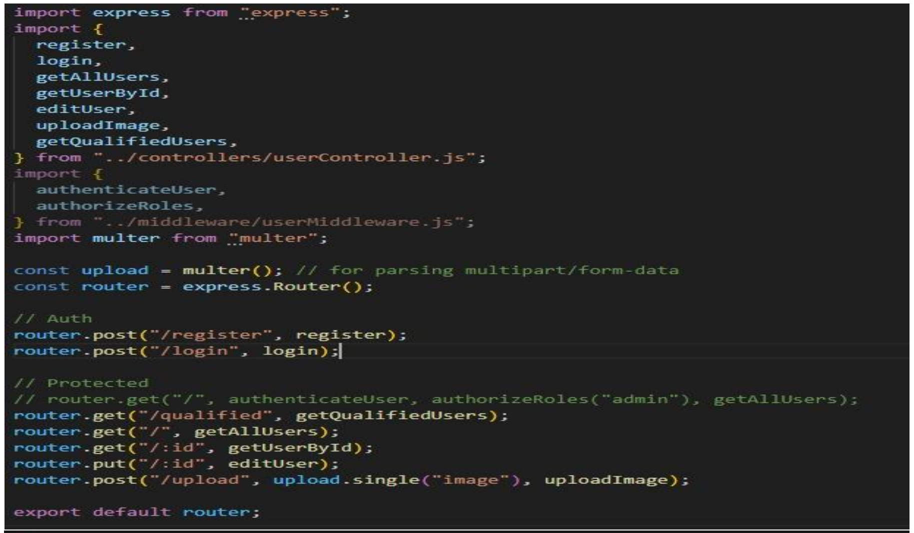
Project Front-end



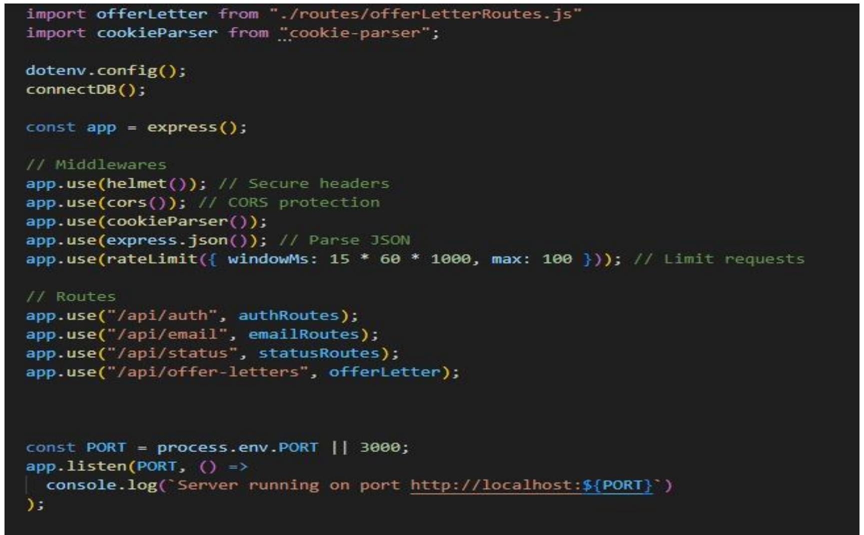
Sub Admin List

Name	Email	Role	Phone No	Actions
Omer Khan	omer33@example.com	Manager	92765468765	Edit Delete
Salman Ali	salman21@example.com	Supervisor	927864728963	Edit Delete
Ali Ahmed	aliahmed2@example.com	Manager	92765468765	Edit Delete
Junaid Ali	junaid32ali@example.com	Supervisor	927864728963	Edit Delete
Faheem Khan	khan32@example.com	Supervisor	927864728963	Edit Delete

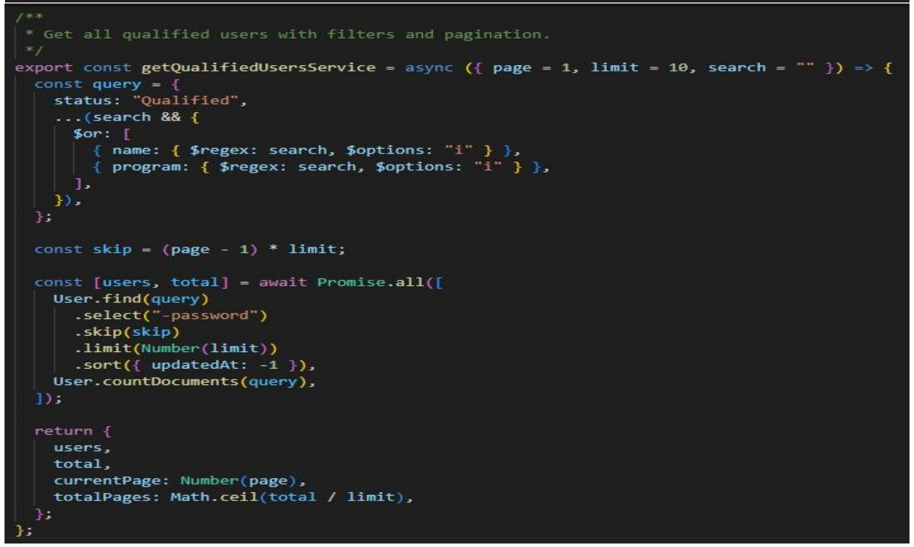
Project Back-end



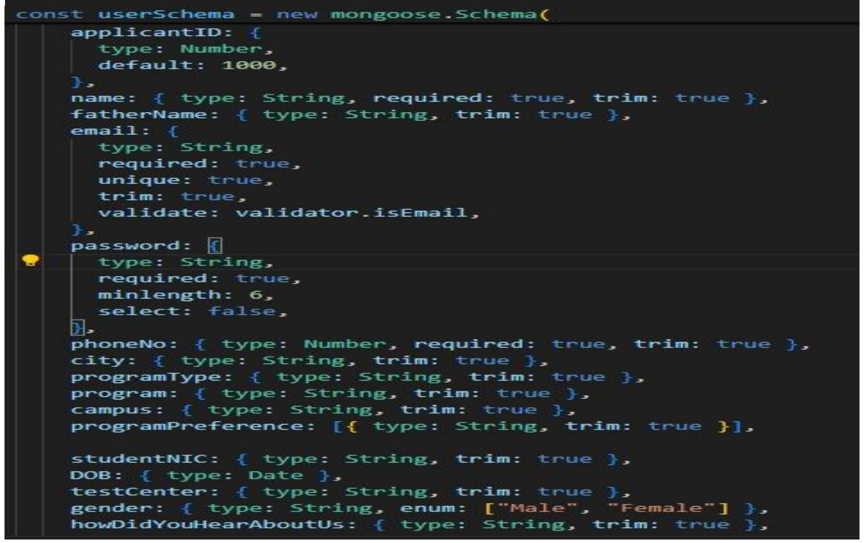
```
import express from "express";
import {
  register,
  login,
  getAllUsers,
  getUserById,
  editUser,
  uploadImage,
  getQualifiedUsers,
} from "../controllers/userController.js";
import {
  authenticateUser,
  authorizeRoles,
} from "../middleware/userMiddleware.js";
import multer from "multer";

const upload = multer(); // for parsing multipart/form-data
const router = express.Router();

// Auth
router.post("/register", register);
router.post("/login", login);

// Protected
// router.get("/", authenticateUser, authorizeRoles("admin"), getAllUsers);
router.get("/qualified", getQualifiedUsers);
router.get("/", getAllUsers);
router.get("/:id", getUserById);
router.put("/:id", editUser);
router.post("/upload", upload.single("image"), uploadImage);

export default router;
```



```
import offerLetter from "./routes/offerLetterRoutes.js"
import cookieParser from "cookie-parser";

dotenv.config();
connectDB();

const app = express();

// Middlewares
app.use(helmet()); // Secure headers
app.use(cors()); // CORS protection
app.use(cookieParser());
app.use(express.json()); // Parse JSON
app.use(rateLimit({ windowMs: 15 * 60 * 1000, max: 100 })); // Limit requests

// Routes
app.use("/api/auth", authRoutes);
app.use("/api/email", emailRoutes);
app.use("/api/status", statusRoutes);
app.use("/api/offer-letters", offerLetter);



const PORT = process.env.PORT || 3000;
app.listen(PORT, () =>
  console.log(`Server running on port http://localhost:${PORT}`)
);
```

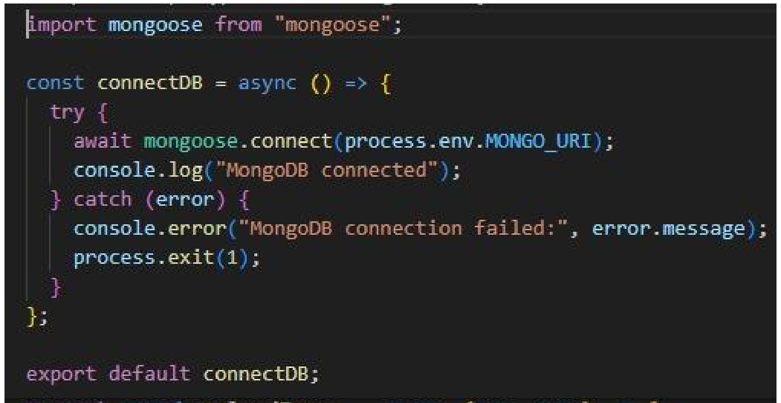


```
/**
 * Get all qualified users with filters and pagination.
 */
export const getQualifiedUsersService = async ({ page = 1, limit = 10, search = "" }) => {
  const query = {
    status: "Qualified",
    ...(search && {
      $or: [
        { name: { $regex: search, $options: "i" } },
        { program: { $regex: search, $options: "i" } },
      ],
    }),
  };

  const skip = (page - 1) * limit;

  const [users, total] = await Promise.all([
    User.find(query)
      .select("-password")
      .skip(skip)
      .limit(Number(limit))
      .sort({ updatedAt: -1 }),
    User.countDocuments(query),
  ]);

  return {
    users,
    total,
    currentPage: Number(page),
    totalPages: Math.ceil(total / limit),
  };
};
```

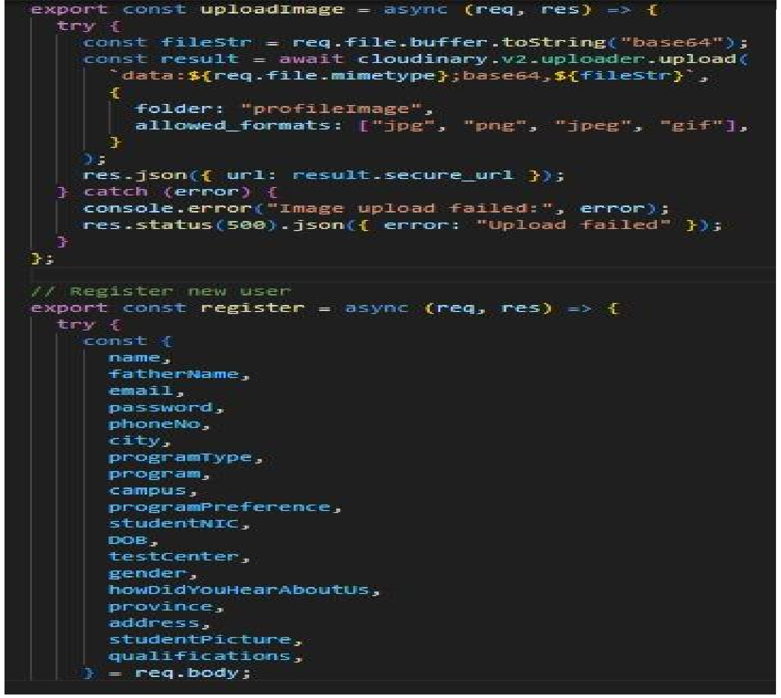


```
const userSchema = new mongoose.Schema(
    applicantID: {
      type: Number,
      default: 1000,
    },
    name: { type: String, required: true, trim: true },
    fatherName: { type: String, trim: true },
    email: {
      type: String,
      required: true,
      unique: true,
      trim: true,
      validate: validator.isEmail,
    },
    password: {
      type: String,
      required: true,
      minlength: 6,
      select: false,
    },
    phoneNo: { type: Number, required: true, trim: true },
    city: { type: String, trim: true },
    programType: { type: String, trim: true },
    program: { type: String, trim: true },
    campus: { type: String, trim: true },
    programPreference: [{ type: String, trim: true }],

    studentNIC: { type: String, trim: true },
    DOB: { type: Date },
    testCenter: { type: String, trim: true },
    gender: { type: String, enum: ["Male", "Female"] },
    howDidYouHearAboutUs: { type: String, trim: true },
```

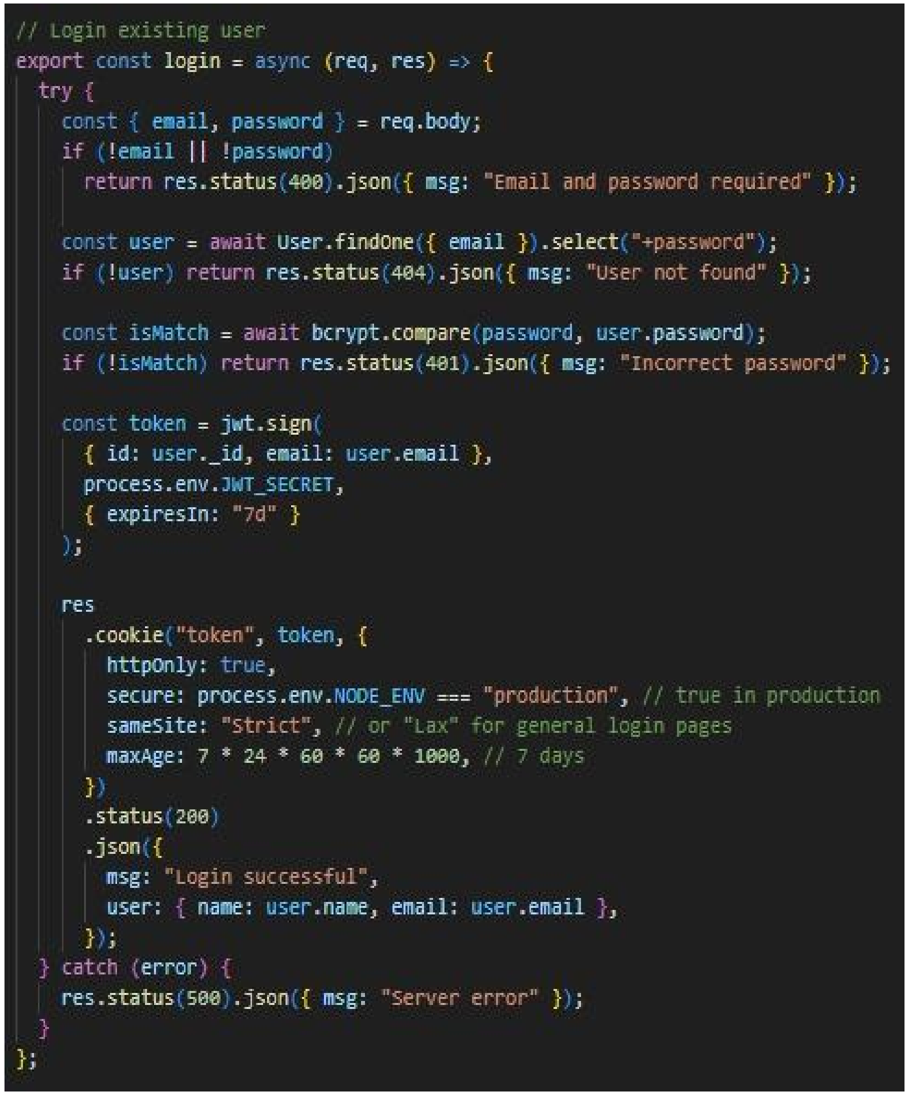
Project Back-end



```
import mongoose from "mongoose";

const connectDB = async () => {
  try {
    await mongoose.connect(process.env.MONGO_URI);
    console.log("MongoDB connected");
  } catch (error) {
    console.error("MongoDB connection failed:", error.message);
    process.exit(1);
  }
};

export default connectDB;
```

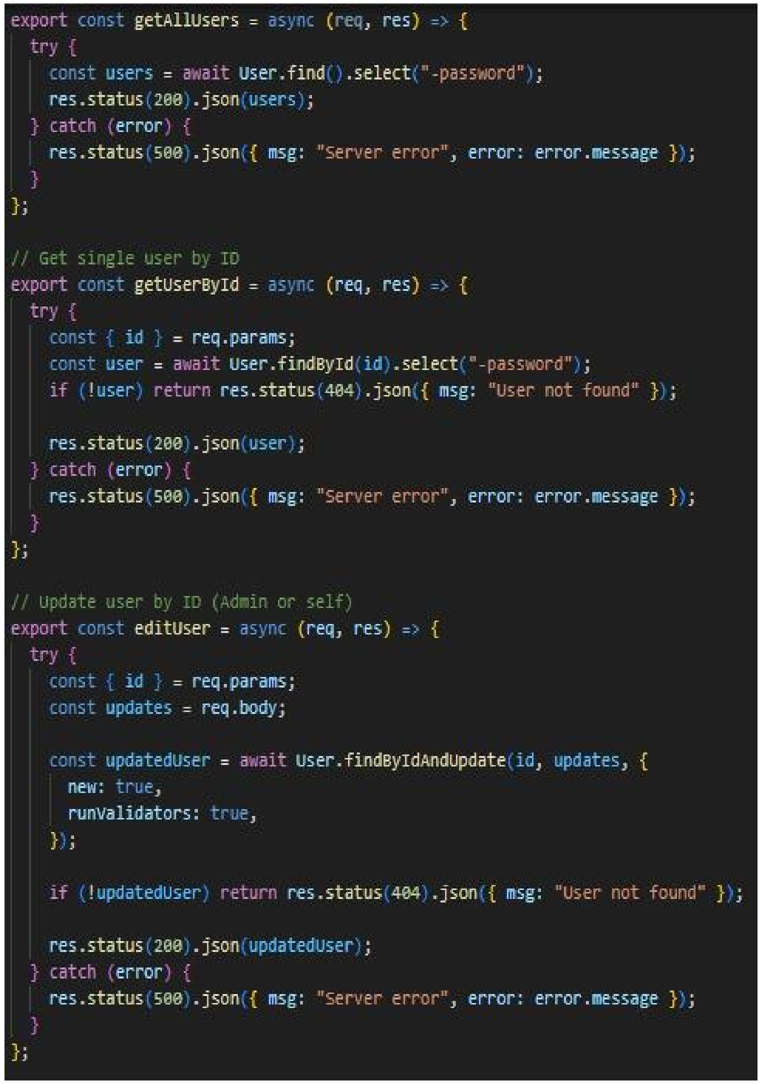


```
export const uploadImage = async (req, res) => {
  try {
    const fileStr = req.file.buffer.toString("base64");
    const result = await cloudinary.v2.uploader.upload(
      `data:${req.file.mimetype};base64,${fileStr}`,
      {
        folder: "profileImage",
        allowed_formats: ["jpg", "png", "jpeg", "gif"],
      }
    );
    res.json({ url: result.secure_url });
  } catch (error) {
    console.error("Image upload failed:", error);
    res.status(500).json({ error: "Upload failed" });
  }
};

// Register new user
export const register = async (req, res) => {
  try {
    const {
      name,
      fatherName,
      email,
      password,
      phoneNo,
      city,
      programType,
      program,
      campus,
      programPreference,
      studentNIC,
      DOB,
      testCenter,
      gender,
      howDidYouHearAboutUs,
      province,
      address,
      studentPicture,
      qualifications,
    } = req.body;
```

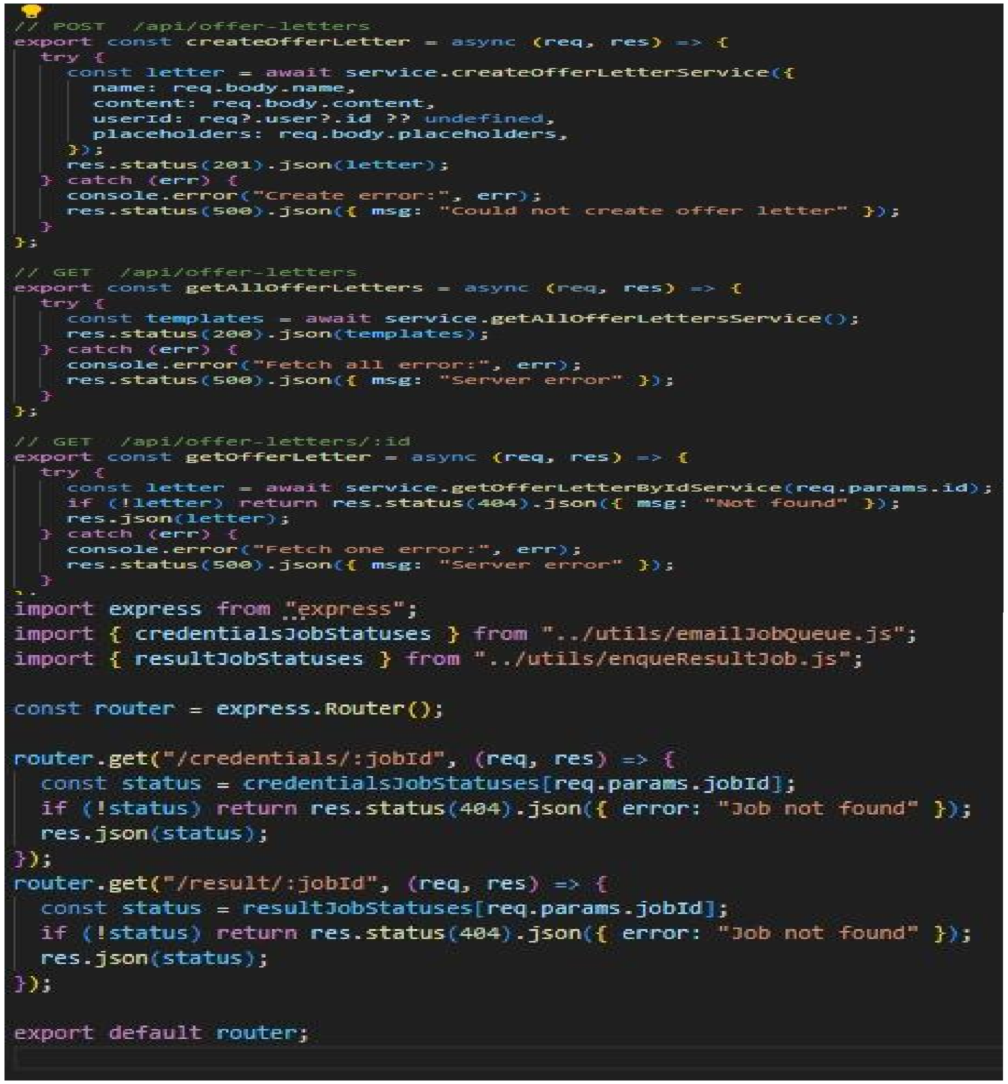


```
// Login existing user
export const login = async (req, res) => {
  try {
    const { email, password } = req.body;
    if (!email || !password)
      return res.status(400).json({ msg: "Email and password required" });

    const user = await User.findOne({ email }).select("+password");
    if (!user) return res.status(404).json({ msg: "User not found" });

    const isMatch = await bcrypt.compare(password, user.password);
    if (!isMatch) return res.status(401).json({ msg: "Incorrect password" });

    const token = jwt.sign(
      { id: user._id, email: user.email },
      process.env.JWT_SECRET,
      { expiresIn: "7d" }
    );

    res
      .cookie("token", token, {
        httpOnly: true,
        secure: process.env.NODE_ENV === "production", // true in production
        sameSite: "Strict", // or "Lax" for general login pages
        maxAge: 7 * 24 * 60 * 60 * 1000, // 7 days
      })
      .status(200)
      .json({
        msg: "Login successful",
        user: { name: user.name, email: user.email },
      });
  } catch (error) {
    res.status(500).json({ msg: "Server error" });
  }
};
```

Project Back-end



```javascript
export const getAllUsers = async (req, res) => {
  try {
    const users = await User.find().select("-password");
    res.status(200).json(users);
  } catch (error) {
    res.status(500).json({ msg: "Server error", error: error.message });
  }
};

// Get single user by ID
export const getUserById = async (req, res) => {
  try {
    const { id } = req.params;
    const user = await User.findById(id).select("-password");
    if (!user) return res.status(404).json({ msg: "User not found" });

    res.status(200).json(user);
  } catch (error) {
    res.status(500).json({ msg: "Server error", error: error.message });
  }
};

// Update user by ID (Admin or self)
export const editUser = async (req, res) => {
  try {
    const { id } = req.params;
    const updates = req.body;

    const updatedUser = await User.findByIdAndUpdate(id, updates, {
      new: true,
      runValidators: true,
    });

    if (!updatedUser) return res.status(404).json({ msg: "User not found" });

    res.status(200).json(updatedUser);
  } catch (error) {
    res.status(500).json({ msg: "Server error", error: error.message });
  }
};
```



```javascript
// POST  /api/offer-letters
export const createOfferLetter = async (req, res) => {
  try {
    const letter = await service.createOfferLetterService({
      name: req.body.name,
      content: req.body.content,
      userId: req?.user?.id ?? undefined,
      placeholders: req.body.placeholders,
    });
    res.status(201).json(letter);
  } catch (err) {
    console.error("Create error:", err);
    res.status(500).json({ msg: "Could not create offer letter" });
  }
};

// GET  /api/offer-letters
export const getAllOfferLetters = async (req, res) => {
  try {
    const templates = await service.getAllOfferLettersService();
    res.status(200).json(templates);
  } catch (err) {
    console.error("Fetch all error:", err);
    res.status(500).json({ msg: "Server error" });
  }
};

// GET  /api/offer-letters/:id
export const getOfferLetter = async (req, res) => {
  try {
    const letter = await service.getOfferLetterByIdService(req.params.id);
    if (!letter) return res.status(404).json({ msg: "Not found" });
    res.json(letter);
  } catch (err) {
    console.error("Fetch one error:", err);
    res.status(500).json({ msg: "Server error" });
  }
import express from "express";
import { credentialsJobStatuses } from "../utils/emailJobQueue.js";
import { resultJobStatuses } from "../utils/enqueResultJob.js";

const router = express.Router();

router.get("/credentials/:jobId", (req, res) => {
  const status = credentialsJobStatuses[req.params.jobId];
  if (!status) return res.status(404).json({ error: "Job not found" });
  res.json(status);
});
router.get("/result/:jobId", (req, res) => {
  const status = resultJobStatuses[req.params.jobId];
  if (!status) return res.status(404).json({ error: "Job not found" });
  res.json(status);
});

export default router;
```

Project **Demonstration**

Demo

THANK YOU!