Infosys Springboard Virtual Internship

"Edu2Job": Predicting Job Roles from Educational Background

Milestone 2 – Education Input & Preprocessing

Name: Adiya T V
Date: 08-12-2025

Milestone 2 – Education Input & Preprocessing

1.Introduction

Milestone 2 focused on building the foundational data pipeline necessary for the Job Prediction Engine. This involved creating the interface and establishing the robust backend logic to collect, validate, and transform raw user education details. The primary deliverable of this stage is a secure system capable of converting diverse, messy user inputs and bulk dataset records into a standardized, numerical feature vector. This milestone directly addresses the common challenges in machine learning preparation, including encoding categorical fields and ensuring data consistency across multiple sources.

2.Objective

The overarching objective of Milestone 2 was to implement a universally functional preprocessing routine that eliminates data vulnerabilities before the ML model is trained (Milestone 3). Specific objectives included designing the user input form to capture essential fields like Degree, Specialization, CGPA, and Graduation Year, and developing secure, maintainable transformation logic. Critically, the pipeline was designed to handle data from **two distinct sources** (the live user form and two external academic datasets) using a single, unified "recipe". This objective required fixing vulnerabilities such as code duplication and incomplete mapping to ensure all academic fields are correctly encoded into numeric features.

3.Workflow

The workflow proceeded in three integrated stages: Input, Validation, and Transformation.

A. Input & Validation: The front-end form was finalized to capture required details, including the critical CGPA Scale selector. Frontend validation checks were implemented for better UX, and security was enforced via robust Server-Side Validation (Joi) in the backend, ensuring data integrity against maximum CGPA limits (4.0, 5.0, or 10.0) and valid graduation years.

B. Data Transformation & Centralization: The preprocessing logic was centralized in a dedicated module, preprocessing_logic.js. This module houses the Standardized Transformation Maps (DEGREE_MAP, SPEC_MAP), which were fully expanded to correctly encode all course codes and full department names (e.g., 'Mathematics', 'BCA', 'ICE'), eliminating the previous risk of data defaulting to a code of '0'. The logic also

successfully performs Normalization, converting all CGPA scores to the universal 0-1 range, regardless of the source scale (4.0/5.0/10.0).

C. Universal Data Proof: The final achievement was proving the pipeline's versatility. The live user flow was verified by saving the raw and processed vector (education_processed) to MongoDB. Simultaneously, the batch_processor.js script demonstrated the successful processing of two external datasets, confirming the identical encoding and scaling logic works reliably on bulk data, achieving the final deliverable of a complete, verifiable preprocessing system.

4.Code Implementation

Index.js:

```
require('dotenv').config();
const express = require('express');
const path = require('path');
const cors = require('cors');
const jwt = require('jsonwebtoken');
const bcrypt = require('bcrypt');
const { OAuth2Client } = require('google-auth-library');
const Joi = require('joi');

const { preprocessEducationData } = require('./preprocessing_logic');

const connectDB = require('./config/db');
const User = require('./models/User');

const app = express();
const PORT = process.env.PORT || 3000;

const JWT_SECRET = process.env.JWT_SECRET || 'dev_jwt_secret';
const JWT_EXPIRES = process.env.EXPIRES || '7d';
const BCRYPT_SALT_ROUNDS = parseInt(process.env.BCRYPT_SALT_ROUNDS || '10', 10);

const GOOGLE_CLIENT_ID = '526435487486-
oeipknm1gvcf17be6qb9gn8c0r9i42an.apps.googleusercontent.com';
const googleClient = new OAuth2Client(GOOGLE_CLIENT_ID);

const MONGO_URI = process.env.MONGO_URI || 'mongodb://127.0.0.1:27017/jobrole';
connectDB(MONGO_URI);

app.use(cors());
app.use(express.json());

const FRONTEND_PATH = path.join(__dirname, '..', 'frontend');
app.use(express.static(FRONTEND_PATH));

function generateToken(user) {
```

```
  return jwt.sign(
    { id: user._id.toString(), username: user.username, email: user.email },
    JWT_SECRET,
    { expiresIn: JWT_EXPIRES }
  );
}

function authenticateToken(req, res, next) {
  const authHeader = req.headers['authorization'];
  const token = authHeader && authHeader.split(' ')[1];
  if (!token) return res.status(401).json({ message: 'Token required' });

  jwt.verify(token, JWT_SECRET, (err, payload) => {
    if (err) return res.status(403).json({ message: 'Invalid or expired token' });
    req.user = payload;
    next();
  });
}

app.get('/', (req, res) => {
  res.send('Job Role Predictor Backend is Running');
});

app.post('/auth/register', async (req, res) => {
  try {
    const { username, email, password } = req.body;
    if (!username || !email || !password) return res.status(400).json({ message: 'All fields required' });

    const existing = await User.findOne({ email: email.toLowerCase() });
    if (existing) return res.status(409).json({ message: 'Email already in use' });

    const passwordHash = await bcrypt.hash(password, BCRYPT_SALT_ROUNDS);

    const user = new User({
       username,
       email: email.toLowerCase(),
       passwordHash,
       education: {}
    });
    await user.save();

    const token = generateToken(user);
    res.status(201).json({ message: 'Registration successful', token, user });
  } catch (err) {
    console.error('[POST /register] ', err.message);
    res.status(500).json({ message: 'Server error' });
  }
});

app.post('/auth/login', async (req, res) => {
  try {
    const { email, password } = req.body;
    if (!email || !password) return res.status(400).json({ message: 'Email and password required' });
```

```
    const user = await User.findOne({ email: email.toLowerCase() });
    if (!user || !user.passwordHash) return res.status(401).json({ message: 'Invalid credentials' });

    const ok = await bcrypt.compare(password, user.passwordHash);
    if (!ok) return res.status(401).json({ message: 'Invalid credentials' });

    const token = generateToken(user);
    const userData = user.toObject();
    delete userData.passwordHash;

    res.json({ message: 'Login successful', token, user: userData });
  } catch (err) {
    console.error('[POST /login] ', err.message);
    res.status(500).json({ message: 'Server error' });
  }
});

app.post('/auth/google', async (req, res) => {
  const idToken = req.body.token;
  if (!idToken) return res.status(400).json({ message: 'Google ID token required' });

  try {
    const ticket = await googleClient.verifyIdToken({
      idToken,
      audience: GOOGLE_CLIENT_ID,
    });

    const payload = ticket.getPayload();
    const { sub: googleId, email, name } = payload;

    let user = await User.findOne({ $or: [{ googleId }, { email: email.toLowerCase() }] });
    if (user) {
      user.googleId = user.googleId || googleId;
      user.username = name || user.username;
      await user.save();
    } else {
      user = new User({
          username: name || email,
          email: email.toLowerCase(),
          googleId,
          education: {}
      });
      await user.save();
    }

    const token = generateToken(user);
    const userData = user.toObject();
    delete userData.passwordHash;

    res.json({ message: 'Google auth successful', token, user: userData });
  } catch (err) {
    console.error('[POST /auth/google] ', err.message);
```

```
    res.status(401).json({ message: 'Google token verification failed', error: err.message });
  }
});

app.get('/profile', authenticateToken, async (req, res) => {
  try {
    const user = await User.findById(req.user.id).lean();
    if (!user) return res.status(404).json({ message: 'User not found' });
    delete user.passwordHash;
    res.json({ message: 'Profile fetched', user });
  } catch (err) {
    console.error('[GET /profile] ', err.message);
    res.status(500).json({ message: 'Server error' });
  }
});

const educationSchema = Joi.object({
    degree: Joi.string().trim().min(2).max(100).required().messages({
        'any.required': 'Degree is required.',
        'string.empty': 'Degree cannot be empty.'
    }),
    specialization: Joi.string().trim().min(2).max(100).required().messages({
        'any.required': 'Specialization is required.',
        'string.empty': 'Specialization cannot be empty.'
    }),

    cgpa: Joi.number().precision(2).min(0).custom((value, helpers) => {
        const scale = helpers.state.ancestors[0].cgpaOutOf;
        let max = 10.0;
        if (scale === '4') max = 4.0;
        else if (scale === '5') max = 5.0;

        if (value > max) {
            return helpers.error('number.max', { limit: max });
        }
        return value;
    }).optional().allow(null, '').messages({
        'number.base': 'CGPA must be a number.',
        'number.min': 'CGPA must be 0 or higher.',
        'number.max': 'CGPA must be less than or equal to {{#limit}} based on your selected scale.'
    }),

    cgpaOutOf: Joi.string().valid('4', '10', '5').required().messages({
        'any.required': 'CGPA scale (Out of 4, 5, or 10) is required.'
    }),

    yearOfGraduation: Joi.number().integer().min(2000).max(new Date().getFullYear() +
5).required().messages({
        'any.required': 'Graduation Year is required.',
        'number.base': 'Graduation Year must be a number.',
        'number.min': 'Graduation Year must be 2000 or later.',
        'number.max': 'Graduation Year cannot be more than 5 years in the future.'
    }),
```

```
    employmentType: Joi.string().valid('full-time', 'part-time', 'internship').required(),
    certifications: Joi.alternatives().try(
        Joi.string().allow(''),
        Joi.array().items(Joi.string())
    ).optional(),
    username: Joi.string().trim().min(2).optional()
});

app.post('/education/add', authenticateToken, async (req, res) => {
    try {

        const { error, value: validatedData } = educationSchema.validate(req.body, { abortEarly: false });

        if (error) {
            const validationError = error.details[0].message;
            return res.status(400).json({ message: `Validation Error: ${validationError}` });
        }

        const {
            degree,
            specialization,
            cgpa,
            cgpaOutOf,
            yearOfGraduation,
            certifications,
            employmentType,
            username
        } = validatedData;


        const processedVector = preprocessEducationData(validatedData);

        const updates = {
            education_processed: processedVector,
            'education.degree': degree,
            'education.specialization': specialization,
            'education.yearOfGraduation': yearOfGraduation,
            'education.cgpaOutOf': cgpaOutOf,
            'education.employmentType': employmentType,
        };

        if (username) {
            updates.username = username;
        }

        if (cgpa !== undefined && cgpa !== null && cgpa !== '') {
            updates['education.cgpa'] = parseFloat(cgpa);
        } else {
            updates['education.cgpa'] = null;
        }

        if (certifications !== undefined) {
            if (typeof certifications === 'string') {
```

```
            updates['education.certifications'] = certifications.split(',').map(c => c.trim()).filter(Boolean);
        } else if (Array.isArray(certifications)) {
            updates['education.certifications'] = certifications;
        } else {
            updates['education.certifications'] = [];
        }
    }

    const user = await User.findByIdAndUpdate(
        req.user.id,
        { $set: updates },
        { new: true }
    ).lean();

    if (!user) return res.status(404).json({ message: 'User not found' });
    delete user.passwordHash;


    const responseJson = { message: 'Education data saved and processed successfully.', user };

    console.log('\n--- LIVE USER SUBMISSION SUCCESS ---');
    console.log(JSON.stringify(responseJson, null, 2));
    console.log('------------------------------------\n');

    res.json(responseJson);

  } catch (err) {
    console.error('[POST /education/add] ', err.message);
    res.status(500).json({ message: 'Server error while saving education data.' });
  }
});


app.put('/profile', authenticateToken, async (req, res) => {
 try {
  const { error, value: validatedData } = educationSchema.options({ allowUnknown: true }).validate(req.body,
{ abortEarly: false });

  if (error) {
   const validationError = error.details[0].message;
   return res.status(400).json({ message: `Validation Error: ${validationError}` });
  }

  const {
    username,
    degree,
    specialization,
    cgpa,
    cgpaOutOf,
    yearOfGraduation,
    certifications,
    employmentType
  } = validatedData;
```

```
    const updates = {};

    if (username) updates.username = username;

    if (degree && specialization && yearOfGraduation) {
      const processedVector = preprocessEducationData(validatedData);
      updates.education_processed = processedVector;

      updates['education.degree'] = degree;
      updates['education.specialization'] = specialization;
      updates['education.yearOfGraduation'] = yearOfGraduation || null;
      updates['education.cgpaOutOf'] = cgpaOutOf || '10';
      updates['education.employmentType'] = employmentType || 'full-time';

      if (cgpa !== undefined && cgpa !== null && cgpa !== '') {
        updates['education.cgpa'] = parseFloat(cgpa);
      } else {
        updates['education.cgpa'] = null;
      }

      if (certifications !== undefined) {
        if (typeof certifications === 'string') {
          updates['education.certifications'] = certifications.split(',').map(c => c.trim()).filter(Boolean);
        } else if (Array.isArray(certifications)) {
          updates['education.certifications'] = certifications;
        } else {
          updates['education.certifications'] = [];
        }
      }
    }

    const user = await User.findByIdAndUpdate(
      req.user.id,
      { $set: updates },
      { new: true }
    ).lean();

    if (!user) return res.status(404).json({ message: 'User not found' });
    delete user.passwordHash;

    res.json({ message: 'Profile updated successfully (Preprocessed)', user });
  } catch (err) {
    console.error('[PUT /profile] ', err.message);
    res.status(500).json({ message: 'Server error' });
  }
});


const passwordChangeSchema = Joi.object({
    currentPassword: Joi.string().optional().allow(''),
    newPassword: Joi.string().min(8).required(),
    confirmPassword: Joi.string().valid(Joi.ref('newPassword')).required().messages({
```

```
        'any.only': 'New passwords must match.'
    })
});

app.post('/auth/change-password', authenticateToken, async (req, res) => {
    try {
        const { error, value } = passwordChangeSchema.validate(req.body, { abortEarly: false });

        if (error) {
            return res.status(400).json({ message: `Validation Error: ${error.details[0].message}` });
        }

        const { currentPassword, newPassword } = value;
        const user = await User.findById(req.user.id);

        if (!user) {
            return res.status(404).json({ message: 'User not found' });
        }

        if (user.passwordHash) {
            if (!currentPassword) {
                return res.status(400).json({ message: 'Current password is required to change it.' });
            }
            const isMatch = await bcrypt.compare(currentPassword, user.passwordHash);
            if (!isMatch) {
                return res.status(401).json({ message: 'Invalid current password.' });
            }
        }

        const newPasswordHash = await bcrypt.hash(newPassword, BCRYPT_SALT_ROUNDS);
        user.passwordHash = newPasswordHash;
        await user.save();

        const responseMessage = user.googleId && !user.passwordHash
                        ? 'Password setup successful. You can now sign in using your email and password.'
                        : 'Password updated successfully.';

        res.json({ message: responseMessage });

    } catch (err) {
        console.error('[POST /auth/change-password] ', err.message);
        res.status(500).json({ message: 'Server error during password operation.' });
    }
});

app.use((req, res) => {
  res.sendFile(path.join(FRONTEND_PATH, 'index.html'));
});

app.listen(PORT, () => {
  console.log(`Server running on http://localhost:${PORT}`);
});
```

Index.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width,initial-scale=1" />
  <title>StudentHub Dashboard</title>

  <link href="https://fonts.googleapis.com/css2?family=Inter:wght@300;400;500;600;700;800&display=swap"
rel="stylesheet">
  <script src="https://cdn.tailwindcss.com"></script>
  <script src="https://accounts.google.com/gsi/client" async defer></script>
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.4.0/css/all.min.css">

  <style>
    :root{
      --accent:#7C3AED;
      --accent-hover:#6D28D9;
      --bg-light: #F3F4F6;
    }
    body { font-family: 'Inter', sans-serif; background-color: var(--bg-light); color: #1f2937; margin:0; overflow-
x: hidden; }


    .auth-bg { background: linear-gradient(135deg, #f5f3ff 0%, #ede9fe 100%); min-height: 100vh; display: flex;
align-items: center; justify-content: center; position: relative; overflow: hidden; }
    .wavy-blob{ position:absolute; filter: blur(50px); opacity:0.6; z-index: 0; animation: float 10s infinite ease-in-
out; }
    @keyframes float { 0%{transform:translate(0,0)} 50%{transform:translate(20px, -20px)}
100%{transform:translate(0,0)} }
    .glass-card { background: rgba(255, 255, 255, 0.9); backdrop-filter: blur(12px); border: 1px solid
rgba(255,255,255,0.5); box-shadow: 0 20px 40px -5px rgba(0,0,0,0.1); }
    .auth-tab { cursor:pointer; padding: 0.5rem 1rem; border-radius: 0.5rem; transition: all .2s; color: #6b7280;
font-weight: 500; }
    .auth-tab.active { color: var(--accent); background: #ede9fe; font-weight:600; }

    #dashboard-container { display: none; min-height: 100vh; }


    .sidebar-link.active {
      background-color: #ede9fe;
      color: #7c3aed;
      font-weight: 600;
    }
    .sidebar-link.active i {
       color: #7c3aed;
    }


    .content-section.hidden {
```

```
        display: none;
    }


    .btn-primary { background: var(--accent); color: white; transition: all 0.2s; }
    .btn-primary:hover { background: var(--accent-hover); box-shadow: 0 4px 12px rgba(124, 58, 237, 0.3); }
    .fade-in { animation: fadeIn 0.4s ease-out forwards; }
    @keyframes fadeIn { from { opacity:0; transform: translateY(10px); } to { opacity:1; transform: translateY(0); } }

    input:disabled { background-color: transparent; border-color: transparent; color: #374151; cursor: default; }


    #message-box { transition: transform 0.3s cubic-bezier(0.175, 0.885, 0.32, 1.275); transform: translateX(120%); }
    #message-box.show { transform: translateX(0); }
  </style>
</head>
<body>

  <div id="auth-container" class="auth-bg">
    <div class="wavy-blob w-96 h-96 bg-purple-300 rounded-full -top-20 -left-20"></div>
    <div class="wavy-blob w-80 h-80 bg-indigo-300 rounded-full -bottom-20 -right-20 animation-delay-2000"></div>

    <div class="glass-card relative z-10 w-full max-w-4xl rounded-2xl overflow-hidden flex shadow-2xl m-4 min-h-[600px]">

      <div class="hidden md:flex w-1/2 bg-gradient-to-br from-violet-600 to-indigo-700 p-12 flex-col justify-between text-white relative overflow-hidden">
        <div class="absolute inset-0 bg-[url('https://www.transparenttextures.com/patterns/cubes.png')] opacity-10"></div>
        <div>
          <div class="h-12 w-12 bg-white/20 rounded-xl flex items-center justify-center backdrop-blur-sm mb-6">
            <i class="fa-solid fa-graduation-cap text-2xl"></i>
          </div>
          <h1 class="text-4xl font-bold leading-tight mb-4">Track Stats & Predict Career</h1>
          <p class="text-violet-100 text-lg opacity-90">Manage your academic profile and predict job roles based on your skills.</p>
        </div>
        <div class="text-sm text-violet-200 opacity-70">© 2024 JobPredictor. Secure & Private.</div>
      </div>

      <div class="w-full md:w-1/2 p-8 md:p-12 bg-white flex flex-col justify-center">
        <div class="flex bg-gray-100 p-1 rounded-lg mb-8 w-fit mx-auto">
          <button id="show-login-btn" class="auth-tab active">Sign In</button>
          <button id="show-register-btn" class="auth-tab">Create Account</button>
        </div>

        <div id="login-view" class="space-y-6 fade-in">
          <div class="text-center mb-6">
            <h2 class="text-2xl font-bold text-gray-800">Welcome Back!</h2>
            <p class="text-gray-500 text-sm">Please enter your details to sign in.</p>
```

```
        </div>
        <form id="login-form" class="space-y-4">
          <div>
            <label class="block text-sm font-medium text-gray-700 mb-1">Email</label>
            <div class="relative">
              <i class="fa-regular fa-envelope absolute left-3 top-3.5 text-gray-400"></i>
              <input id="login-email" type="email" placeholder="name@example.com" class="w-full pl-10 pr-4
py-3 rounded-xl border border-gray-200 focus:border-violet-500 focus:ring-2 focus:ring-violet-100 outline-none
transition" />
            </div>
          </div>
          <div>
            <label class="block text-sm font-medium text-gray-700 mb-1">Password</label>
            <div class="relative">
              <i class="fa-solid fa-lock absolute left-3 top-3.5 text-gray-400"></i>
              <input id="login-password" type="password" placeholder="••••••••" class="w-full pl-10 pr-4 py-3
rounded-xl border border-gray-200 focus:border-violet-500 focus:ring-2 focus:ring-violet-100 outline-none
transition" />
            </div>
          </div>
          <button type="submit" class="btn-primary w-full py-3 rounded-xl font-semibold shadow-lg shadow-
violet-200">Sign In</button>
        </form>
      </div>

      <div id="register-view" class="hidden space-y-6 fade-in">
        <div class="text-center mb-6">
          <h2 class="text-2xl font-bold text-gray-800">Get Started</h2>
          <p class="text-gray-500 text-sm">Create an account to start predicting.</p>
        </div>
        <form id="register-form" class="space-y-4">
          <div>
            <label class="block text-sm font-medium text-gray-700 mb-1">Full Name</label>
            <input id="register-username" type="text" placeholder="John Doe" class="w-full px-4 py-3 rounded-xl
border border-gray-200 focus:border-violet-500 focus:ring-2 focus:ring-violet-100 outline-none transition" />
          </div>
          <div>
            <label class="block text-sm font-medium text-gray-700 mb-1">Email</label>
            <input id="register-email" type="email" placeholder="name@example.com" class="w-full px-4 py-3
rounded-xl border border-gray-200 focus:border-violet-500 focus:ring-2 focus:ring-violet-100 outline-none
transition" />
          </div>
          <div>
            <label class="block text-sm font-medium text-gray-700 mb-1">Password</label>
            <input id="register-password" type="password" placeholder="Create a password" class="w-full px-4
py-3 rounded-xl border border-gray-200 focus:border-violet-500 focus:ring-2 focus:ring-violet-100 outline-none
transition" />
          </div>
          <button type="submit" class="btn-primary w-full py-3 rounded-xl font-semibold shadow-lg shadow-
violet-200">Sign Up</button>
        </form>
      </div>
```

```
      <div class="relative my-6">
        <div class="absolute inset-0 flex items-center"><div class="w-full border-t border-gray-
200"></div></div>
        <div class="relative flex justify-center text-sm"><span class="px-2 bg-white text-gray-500">Or continue
with</span></div>
      </div>

      <div class="flex justify-center">
        <div id="g_id_onload" data-client_id="526435487486-
oeipknm1gvcf17be6qb9gn8c0r9i42an.apps.googleusercontent.com" data-ux_mode="popup" data-
callback="handleCredentialResponse" data-auto_prompt="false"></div>
        <div class="g_id_signin" data-type="standard" data-shape="rectangular" data-theme="outline" data-
text="signin_with" data-size="large" data-logo_alignment="left"></div>
      </div>
    </div>
  </div>
 </div>

 <div id="dashboard-container" class="flex h-screen bg-gray-100">

  <aside class="w-64 bg-white/80 backdrop-blur-md border-r border-white/50 shadow-xl p-6 flex flex-col
justify-between z-50">
    <div>
      <div class="flex items-center gap-3 mb-10">
        <div class="bg-violet-600 text-white p-2 rounded-lg"><i class="fa-solid fa-graduation-cap"></i></div>
        <span class="font-bold text-xl tracking-tight text-gray-800">StudentHub</span>
      </div>
      <nav class="space-y-2">
        <a href="#" class="sidebar-link active flex items-center gap-3 px-4 py-3 rounded-xl text-gray-700 font-
medium hover:bg-violet-50 transition-all" data-target="main-dashboard">
          <i class="fa-solid fa-house text-gray-400"></i> Main Dashboard
        </a>
        <a href="#" class="sidebar-link flex items-center gap-3 px-4 py-3 rounded-xl text-gray-700 font-medium
hover:bg-violet-50 transition-all" data-target="edit-profile">
          <i class="fa-solid fa-user-pen text-gray-400"></i> Edit Student Data
        </a>
        <a href="#" class="sidebar-link flex items-center gap-3 px-4 py-3 rounded-xl text-gray-700 font-medium
hover:bg-violet-50 transition-all" data-target="change-password">
          <i class="fa-solid fa-key text-gray-400"></i> Change Password
        </a>
      </nav>
    </div>
    <button id="logout-btn" class="flex items-center gap-3 px-4 py-3 rounded-xl text-red-600 font-medium
hover:bg-red-50 transition-all w-full">
      <i class="fa-solid fa-arrow-right-from-bracket"></i> Logout
    </button>
  </aside>

  <main class="flex-1 overflow-y-auto p-8">

    <div class="flex justify-end mb-8">
      <div class="flex items-center gap-4">
        <div class="text-right">
```

```
          <p id="nav-username" class="text-sm font-semibold text-gray-800">Loading...</p>
          <p class="text-xs text-gray-500">Student Account</p>
        </div>
        <div class="h-10 w-10 bg-violet-100 text-violet-600 rounded-full flex items-center justify-center font-bold text-lg border-2 border-white shadow-sm"><i class="fa-regular fa-user"></i></div>
      </div>
    </div>

    <div id="main-dashboard" class="content-section fade-in space-y-6">
      <h1 class="text-3xl font-bold text-gray-900 mb-2">Overview</h1>
      <p class="text-gray-500 mb-6">Welcome back to your student dashboard.</p>

      <div class="grid grid-cols-1 lg:grid-cols-3 gap-8">

        <div class="bg-white rounded-2xl p-6 shadow-sm border border-gray-100 col-span-1">
          <h3 class="font-bold text-lg text-gray-800 mb-4 flex items-center gap-2">
            <i class="fa-solid fa-briefcase text-violet-600"></i> Current Job Prediction
          </h3>

          <div class="space-y-4">
            <div class="text-center py-4 bg-violet-50 rounded-xl border border-violet-100">
              <p class="text-4xl font-extrabold text-violet-700" id="prediction-confidence">92%</p>
              <p class="text-sm text-gray-500">Confidence Score</p>
            </div>

            <p class="text-sm font-bold text-gray-700 uppercase tracking-wider mb-2 pt-2 border-t  border-gray-100">Top Predicted Roles (Dummy)</p>
            <ul class="space-y-2 text-sm">
              <li class="flex justify-between items-center text-gray-700 font-medium">
                <span><i class="fa-solid fa-code text-violet-500 mr-2"></i> Software Developer</span>
                <span class="text-xs bg-violet-100 text-violet-700 px-2 py-0.5 rounded-full">High</span>
              </li>
              <li class="flex justify-between items-center text-gray-700 font-medium">
                <span><i class="fa-solid fa-cloud text-violet-500 mr-2"></i> Cloud Engineer</span>
                <span class="text-xs bg-gray-100 text-gray-600 px-2 py-0.5 rounded-full">Medium</span>
              </li>
              <li class="flex justify-between items-center text-gray-700 font-medium">
                <span><i class="fa-solid fa-chart-line text-violet-500 mr-2"></i> Data Analyst</span>
                <span class="text-xs bg-gray-100 text-gray-600 px-2 py-0.5 rounded-full">Low</span>
              </li>
            </ul>
          </div>
        </div>

        <div class="bg-white rounded-2xl shadow-sm border border-gray-100 overflow-hidden relative transition-all duration-300 col-span-1 lg:col-span-2">
          <div class="bg-gray-50/50 p-6 border-b border-gray-100 flex justify-between items-center">
            <div>
              <h3 class="font-bold text-lg text-gray-800">Academic Profile</h3>
              <p class="text-sm text-gray-400">Your current details</p>
            </div>
            <button class="bg-violet-100 hover:bg-violet-200 text-violet-700 px-4 py-2 rounded-lg text-sm font-semibold transition" onclick="document.querySelector('[data-target=edit-profile]').click()">
```

```
                <i class="fa-solid fa-pen-to-square mr-2"></i>Edit
              </button>
            </div>
            <div class="p-6 grid grid-cols-1 md:grid-cols-2 gap-6">
              <div class="col-span-1 md:col-span-2 border-b border-gray-100 pb-4">
                <label class="text-xs font-bold text-gray-400 uppercase tracking-wider mb-1 block">Student
Name</label>
                <p id="view-name" class="text-lg font-bold text-gray-800 p-2 pl-0">Loading...</p>
                <label class="text-xs font-bold text-gray-400 uppercase tracking-wider mb-1 block">Email
Address</label>
                <p id="view-email" class="text-gray-600 p-2 pl-0">loading...</p>
              </div>

              <div>
                <label class="text-xs font-bold text-gray-400 uppercase tracking-wider mb-1
block">Degree</label>
                <p id="view-degree" class="text-gray-800 font-medium p-2 pl-0"></p>
              </div>
              <div>
                <label class="text-xs font-bold text-gray-400 uppercase tracking-wider mb-1
block">Specialization</label>
                <p id="view-spec" class="text-gray-800 font-medium p-2 pl-0"></p>
              </div>
              <div>
                <label class="text-xs font-bold text-gray-400 uppercase tracking-wider mb-1 block">CGPA /
Marks</label>
                <p id="view-cgpa" class="text-gray-800 font-medium p-2 pl-0"></p>
              </div>
              <div>
                <label class="text-xs font-bold text-gray-400 uppercase tracking-wider mb-1 block">Year of
Graduation</label>
                <p id="view-year" class="text-gray-800 font-medium p-2 pl-0"></p>
              </div>
              <div class="col-span-1 md:col-span-2">
                <label class="text-xs font-bold text-gray-400 uppercase tracking-wider mb-1
block">Certifications</label>
                <p id="view-certs" class="text-gray-800 font-medium p-2 pl-0"></p>
              </div>
            </div>
          </div>
        </div>
      </div>

      <div id="edit-profile" class="content-section hidden fade-in space-y-6">
        <h1 class="text-3xl font-bold text-gray-900 mb-6">Edit Student Data</h1>
        <div class="bg-white rounded-2xl shadow-sm border border-gray-100 overflow-hidden relative transition-
all duration-300 max-w-4xl">
          <div class="p-6">
            <form id="academic-form" class="grid grid-cols-1 md:grid-cols-2 gap-6">
              <div class="col-span-1 md:col-span-2">
                <label class="text-xs font-bold text-gray-400 uppercase tracking-wider mb-1 block">Student
Name</label>
                <input id="student-name" type="text" value="Loading..."
```

```
                    class="w-full bg-white border border-gray-200 focus:border-violet-500 focus:ring-2
focus:ring-violet-100 p-3 rounded-xl transition-all outline-none" />
                </div>

                <div class="col-span-1 md:col-span-2">
                  <label class="text-xs font-bold text-gray-400 uppercase tracking-wider mb-1 block">Email
Address</label>
                  <input id="student-email" type="email" value="loading..." disabled
                    class="w-full bg-gray-50 text-gray-500 border border-gray-200 p-3 rounded-xl outline-none
cursor-not-allowed" />
                </div>

                <div>
                  <label class="text-xs font-bold text-gray-400 uppercase tracking-wider mb-1
block">Degree</label>
                  <input id="student-degree" type="text" value="" required
                    class="w-full bg-white border border-gray-200 focus:border-violet-500 focus:ring-2
focus:ring-violet-100 p-3 rounded-xl transition-all outline-none placeholder-gray-300" placeholder="e.g.
B.Tech" />
                </div>

                <div>
                  <label class="text-xs font-bold text-gray-400 uppercase tracking-wider mb-1
block">Specialization</label>
                  <input id="student-spec" type="text" value="" required
                    class="w-full bg-white border border-gray-200 focus:border-violet-500 focus:ring-2
focus:ring-violet-100 p-3 rounded-xl transition-all outline-none placeholder-gray-300" placeholder="e.g. CSE"
/>
                </div>

                <div>
                  <label class="text-xs font-bold text-gray-400 uppercase tracking-wider mb-1 block">CGPA /
Marks</label>
                  <input id="student-cgpa" type="number" step="0.01" value="" required
                    class="w-full bg-white border border-gray-200 focus:border-violet-500 focus:ring-2
focus:ring-violet-100 p-3 rounded-xl transition-all outline-none placeholder-gray-300" placeholder="e.g. 8.5" />
                </div>

                <div>
                  <label class="text-xs font-bold text-gray-400 uppercase tracking-wider mb-1 block">CGPA
Scale (Out of)</label>
                  <select id="student-cgpa-out-of" required
                    class="w-full bg-white border border-gray-200 focus:border-violet-500 focus:ring-2
focus:ring-violet-100 p-3 rounded-xl transition-all outline-none">
                    <option value="">-- Select Scale --</option>
                    <option value="10">Out of 10.0</option>
                    <option value="4">Out of 4.0</option>
                  </select>
                </div>

                <div>
                  <label class="text-xs font-bold text-gray-400 uppercase tracking-wider mb-1 block">Year of
Graduation</label>
```

```
                    <input id="student-year" type="number" value="" required
                        class="w-full bg-white border border-gray-200 focus:border-violet-500 focus:ring-2
focus:ring-violet-100 p-3 rounded-xl transition-all outline-none placeholder-gray-300" placeholder="e.g. 2024"
/>
                </div>

                <div class="hidden">
                    <input id="student-employment-type" type="hidden" value="full-time" />
                </div>

                <div class="col-span-1 md:col-span-2">
                    <label class="text-xs font-bold text-gray-400 uppercase tracking-wider mb-1
block">Certifications (Comma separated)</label>
                    <input id="student-certs" type="text" value=""
                        class="w-full bg-white border border-gray-200 focus:border-violet-500 focus:ring-2
focus:ring-violet-100 p-3 rounded-xl transition-all outline-none placeholder-gray-300" placeholder="e.g. AWS,
Python, Google Cloud" />
                </div>

                <div class="col-span-1 md:col-span-2 mt-4 pt-4 border-t border-gray-100">
                    <div class="flex gap-3 justify-end">
                        <button type="button" class="px-5 py-3 rounded-xl text-sm font-medium text-gray-500
hover:bg-gray-100 transition" onclick="document.querySelector('[data-target=main-
dashboard]').click()">Cancel</button>
                        <button type="submit" class="px-5 py-3 rounded-xl text-sm font-bold text-white   bg-violet-
600 hover:bg-violet-700 shadow-md transition">Save Changes</button>
                    </div>
                </div>
            </form>
        </div>
    </div>
</div>

<div id="change-password" class="content-section hidden fade-in space-y-6">
    <h1 class="text-3xl font-bold text-gray-900 mb-6">Change Password</h1>
    <div class="bg-white rounded-2xl shadow-sm border border-gray-100 p-8 max-w-md">
        <p id="password-setup-note" class="hidden text-sm text-gray-600 bg-yellow-50 border-l-4 border-
yellow-400 p-3 mb-6">
            You signed in via Google. Please set a new password below if you wish to sign in with your email
and password in the future.
        </p>

        <form id="password-form" class="space-y-6">
            <div id="current-password-field">
                <label class="block text-sm font-medium text-gray-700 mb-2">Current Password</label>
                <input id="current-password" type="password" placeholder="••••••••" class="w-full px-4 py-3
rounded-xl border border-gray-200 focus:border-violet-500 focus:ring-2 focus:ring-violet-100 outline-none
transition" />
            </div>
            <div>
                <label class="block text-sm font-medium text-gray-700 mb-2">New Password (Min 8
characters)</label>
```

```
                <input id="new-password" type="password" placeholder="••••••••" class="w-full px-4 py-3
rounded-xl border border-gray-200 focus:border-violet-500 focus:ring-2 focus:ring-violet-100 outline-none
transition" />
              </div>
              <div>
                <label class="block text-sm font-medium text-gray-700 mb-4">Confirm New Password</label>
                <input id="confirm-password" type="password" placeholder="••••••••" class="w-full px-4 py-3
rounded-xl border border-gray-200 focus:border-violet-500 focus:ring-2 focus:ring-violet-100 outline-none
transition" />
              </div>
              <button type="submit" class="btn-primary w-full py-3 rounded-xl font-semibold shadow-lg shadow-
violet-200">Update Password</button>
            </form>
          </div>
        </div>

      </main>
    </div>

    <div id="message-box" class="fixed bottom-6 right-6 z-[100] bg-white shadow-2xl rounded-lg p-4 flex
items-center gap-3 border-l-4 border-violet-600 max-w-sm">
      <div id="msg-icon" class="text-violet-600"><i class="fa-solid fa-circle-info"></i></div>
      <div>
        <h4 class="font-bold text-gray-800 text-sm" id="msg-title">Notification</h4>
        <p class="text-gray-500 text-xs" id="message-text">Message goes here.</p>
      </div>
    </div>

    <script>
      const API_BASE_URL = 'http://localhost:3000';
      const $ = s => document.querySelector(s);
      const $$ = s => document.querySelectorAll(s);

      const authContainer = $('#auth-container');
      const dashboardContainer = $('#dashboard-container');
      const loginView = $('#login-view');
      const registerView = $('#register-view');
      const loginForm = $('#login-form');
      const registerForm = $('#register-form');
      const showLoginBtn = $('#show-login-btn');
      const showRegisterBtn = $('#show-register-btn');

      const sidebarLinks = $$('.sidebar-link');
      const contentSections = $$('.content-section');

      const navUsername = $('#nav-username');
      const academicForm = $('#academic-form');

      const view = {
        name: $('#view-name'),
        email: $('#view-email'),
        degree: $('#view-degree'),
        spec: $('#view-spec'),
```

```
    cgpa: $('#view-cgpa'),
    year: $('#view-year'),
    certs: $('#view-certs')
};

const inputs = {
    name: $('#student-name'),
    email: $('#student-email'),
    degree: $('#student-degree'),
    spec: $('#student-spec'),
    cgpa: $('#student-cgpa'),
    year: $('#student-year'),
    certs: $('#student-certs'),
    cgpaOutOf: $('#student-cgpa-out-of'),
    employmentType: $('#student-employment-type')
};

const passwordForm = $('#password-form');
const currentPasswordField = $('#current-password-field');
const currentPasswordInput = $('#current-password');
const newPasswordInput = $('#new-password');
const confirmPasswordInput = $('#confirm-password');
const passwordSetupNote = $('#password-setup-note');
let currentUser = null;
const messageBox = $('#message-box');
const messageText = $('#message-text');

function showMessage(msg, type='info'){
  messageText.textContent = msg;
  const title = $('#msg-title');
  const icon = $('#msg-icon');
  const box = messageBox;

  if(type==='error'){
    box.style.borderLeftColor = '#EF4444';
    icon.innerHTML = '<i class="fa-solid fa-circle-exclamation text-red-500"></i>';
    title.textContent = 'Error';
  } else if(type==='success'){
    box.style.borderLeftColor = '#10B981';
    icon.innerHTML = '<i class="fa-solid fa-circle-check text-green-500"></i>';
    title.textContent = 'Success';
  } else {
    box.style.borderLeftColor = '#7C3AED';
    icon.innerHTML = '<i class="fa-solid fa-circle-info text-violet-500"></i>';
    title.textContent = 'Info';
  }
  box.classList.add('show');
  setTimeout(()=> box.classList.remove('show'), 3500);
}

function switchAuthMode(mode) {
    if(mode === 'login') {
        loginView.classList.remove('hidden');
```

```
        registerView.classList.add('hidden');
        showLoginBtn.classList.add('active');
        showRegisterBtn.classList.remove('active');
    } else {
        loginView.classList.add('hidden');
        registerView.classList.remove('hidden');
        showLoginBtn.classList.remove('active');
        showRegisterBtn.classList.add('active');
    }
}

function toggleAppView(isLoggedIn) {
    if(isLoggedIn) {
        authContainer.style.display = 'none';
        dashboardContainer.style.display = 'flex';
    } else {
        authContainer.style.display = 'flex';
        dashboardContainer.style.display = 'none';
    }
}
sidebarLinks.forEach(link => {
    link.addEventListener('click', (e) => {
        e.preventDefault();

        sidebarLinks.forEach(l => l.classList.remove('active'));
        link.classList.add('active');

        const targetId = link.getAttribute('data-target');
        contentSections.forEach(section => {
            if (section.id === targetId) {
                section.classList.remove('hidden');
                if (targetId === 'change-password' && currentUser) {
                    const isGoogleUser = currentUser.googleId && !currentUser.passwordHash;

                    if (isGoogleUser) {
                        currentPasswordField.classList.add('hidden');
                        currentPasswordInput.removeAttribute('required');
                        passwordSetupNote.classList.remove('hidden');
                    } else {
                        currentPasswordField.classList.remove('hidden');
                        currentPasswordInput.setAttribute('required', 'required');
                        passwordSetupNote.classList.add('hidden');
                    }
                }

            } else {
                section.classList.add('hidden');
            }
        });
    });
});

academicForm.addEventListener('submit', async (e) => {
```

```
    e.preventDefault();

    const updatedData = {
        username: inputs.name.value,
        degree: inputs.degree.value,
        specialization: inputs.spec.value,
        cgpa: inputs.cgpa.value,
        yearOfGraduation: inputs.year.value,
        certifications: inputs.certs.value,

        cgpaOutOf: inputs.cgpaOutOf.value,
        employmentType: inputs.employmentType.value
    };

    try {
        const res = await fetch(`${API_BASE_URL}/education/add`, {
            method: 'POST',
            headers: { 'Content-Type': 'application/json', ...generateAuthHeader() },
            body: JSON.stringify(updatedData)
        });

        if(!res.ok) {
            const err = await res.json();

            throw new Error(err.message || 'Failed to update education profile.');
        }
        const data = await res.json();

        updateProfileUI(data.user);
        showMessage('Education details saved and processed!', 'success');
        document.querySelector('[data-target=main-dashboard]').click();

    } catch (error) {
        showMessage(error.message || 'Saving education data failed', 'error');
    }
});

function saveToken(token){ localStorage.setItem('jwtToken', token); }
function getToken(){ return localStorage.getItem('jwtToken'); }
function clearToken(){ localStorage.removeItem('jwtToken'); }
function generateAuthHeader(){ const t=getToken(); return t?{'Authorization':'Bearer '+t}:{}; }

async function handleCredentialResponse(response){
  showMessage('Verifying Google login...');
  try{
    const res = await fetch(`${API_BASE_URL}/auth/google`, {
      method:'POST',
      headers:{'Content-Type':'application/json'},
      body: JSON.stringify({ token: response.credential })
    });
    if(!res.ok){ const err=await res.json(); return showMessage(err.message,'error'); }
    const data = await res.json();
    saveToken(data.token);
```

```
    showMessage('Welcome!','success');
    initDashboard();
  }catch(e){ showMessage('Network error','error'); }
}

function updateProfileUI(user) {
    currentUser = user;
    const edu = user.education || {};
    const certsString = Array.isArray(edu.certifications) ? edu.certifications.join(', ') : (edu.certifications || "");

    navUsername.textContent = user.username || "Student";

    view.name.textContent = user.username || "Not set";
    view.email.textContent = user.email || "";
    view.degree.textContent = edu.degree || "Not set";
    view.spec.textContent = edu.specialization || "Not set";

    const cgpaScale = edu.cgpaOutOf ? ` (Out of ${edu.cgpaOutOf})` : '';
    view.cgpa.textContent = edu.cgpa ? `${edu.cgpa}${cgpaScale}` : "Not set";

    view.year.textContent = edu.yearOfGraduation || "Not set";
    view.certs.textContent = certsString || "None";

    inputs.name.value = user.username || "";
    inputs.email.value = user.email || "";
    inputs.degree.value = edu.degree || "";
    inputs.spec.value = edu.specialization || "";
    inputs.cgpa.value = edu.cgpa || "";
    inputs.year.value = edu.yearOfGraduation || "";
    inputs.certs.value = certsString;

    if (inputs.cgpaOutOf && edu.cgpaOutOf) {
        inputs.cgpaOutOf.value = edu.cgpaOutOf;
    }
}

async function fetchProfileData(){
  try{
    const res = await fetch(`${API_BASE_URL}/profile`, { headers: generateAuthHeader() });
    if(!res.ok) throw new Error('Failed');
    const data = await res.json();
    updateProfileUI(data.user);
  }catch(e){
    console.error(e);
    showMessage('Session expired, please login again','error');
    clearToken();
    toggleAppView(false);
  }
}

async function initDashboard() {
    if(!getToken()) {
        toggleAppView(false);
```

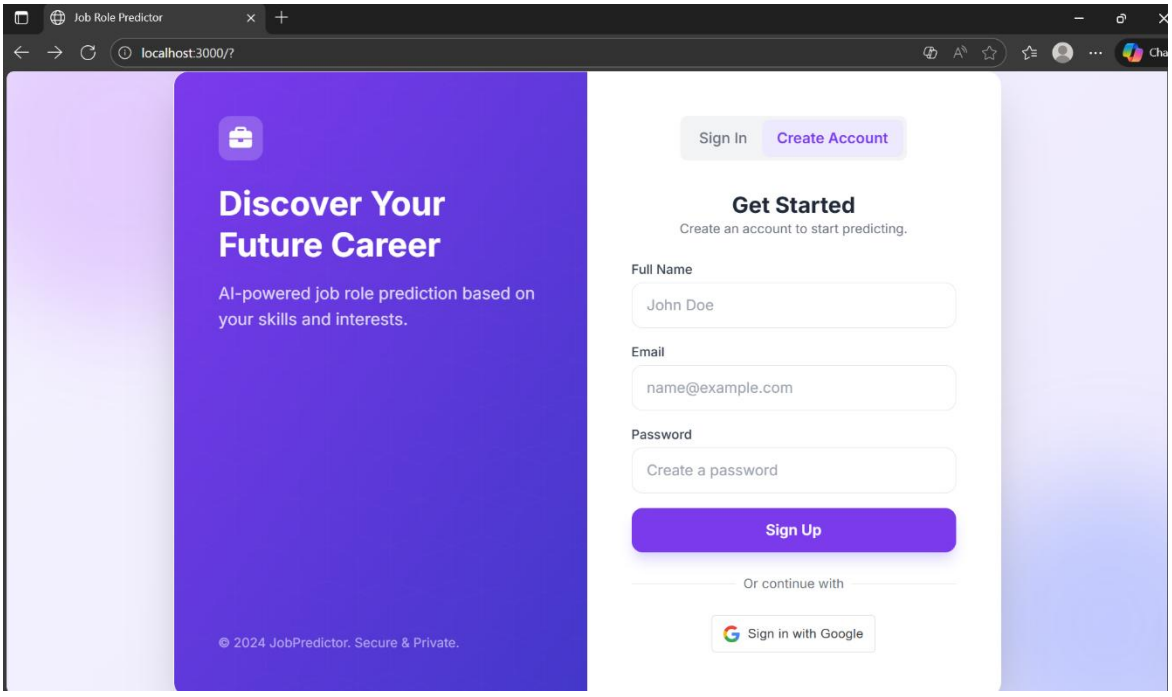
```
        return;
    }
    toggleAppView(true);
    await fetchProfileData();
}

showLoginBtn.addEventListener('click', () => switchAuthMode('login'));
showRegisterBtn.addEventListener('click', () => switchAuthMode('register'));

loginForm.addEventListener('submit', async e=>{
  e.preventDefault();
  const email=$('#login-email').value.trim();
  const password=$('#login-password').value;
  try{
    const res = await fetch(`${API_BASE_URL}/auth/login`, {
        method:'POST',
        headers:{'Content-Type':'application/json'},
        body: JSON.stringify({email,password})
    });
    if(!res.ok){ const err=await res.json(); return showMessage(err.message,'error'); }
    const data = await res.json();
    saveToken(data.token);
    showMessage('Login successful!','success');
    initDashboard();
  }catch(e){ showMessage('Login failed','error'); }
});

registerForm.addEventListener('submit', async e=>{
  e.preventDefault();
  const username=$('#register-username').value.trim();
  const email=$('#register-email').value.trim();
  const password=$('#register-password').value;
  try{
    const res = await fetch(`${API_BASE_URL}/auth/register`, {
        method:'POST',
        headers:{'Content-Type':'application/json'},
        body: JSON.stringify({username,email,password})
    });
    if(!res.ok){ const err=await res.json(); return showMessage(err.message,'error'); }
    const data = await res.json();
    saveToken(data.token);
    showMessage('Account created!','success');
    initDashboard();
  }catch(e){ showMessage('Registration failed','error'); }
});

$('#logout-btn').addEventListener('click', ()=>{
  clearToken();
  showMessage('Logged out successfully','success');
  toggleAppView(false);
  switchAuthMode('login');
});
```

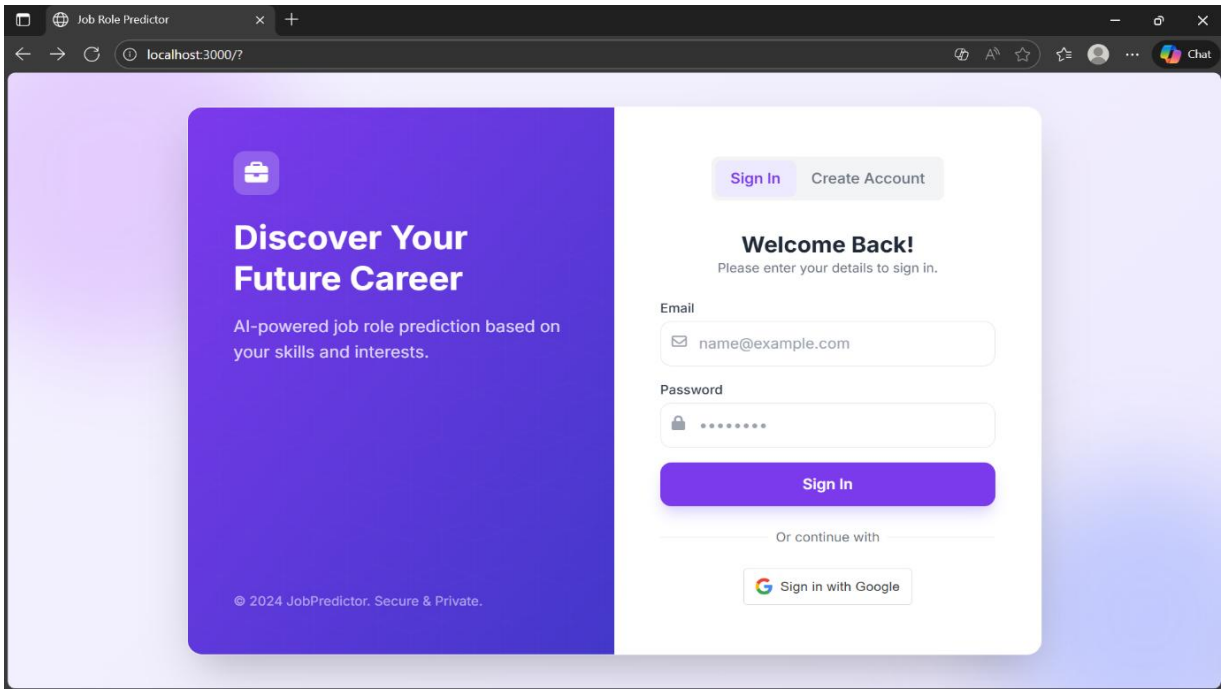
```
    passwordForm.addEventListener('submit', async (e) => {
      e.preventDefault();

      const currentPassword = currentPasswordInput.value;
      const newPassword = newPasswordInput.value;
      const confirmPassword = confirmPasswordInput.value;

      if (newPassword !== confirmPassword) {
        return showMessage('New passwords must match.', 'error');
      }

      const payload = {
        newPassword,
        confirmPassword
      };

      if (!currentPasswordField.classList.contains('hidden')) {
         payload.currentPassword = currentPassword;
      }

      try {
        const res = await fetch(`${API_BASE_URL}/auth/change-password`, {
          method: 'POST',
          headers: { 'Content-Type': 'application/json', ...generateAuthHeader() },
          body: JSON.stringify(payload)
        });

        if (!res.ok) {
          const err = await res.json();
          throw new Error(err.message || 'Failed to change password.');
        }
        const data = await res.json();

        showMessage(data.message, 'success');

        currentPasswordInput.value = '';
        newPasswordInput.value = '';
        confirmPasswordInput.value = '';

        await fetchProfileData();

      } catch (error) {
        showMessage(error.message || 'Password change failed.', 'error');
      }
    });

    initDashboard();
  </script>
</body>
</html>
```

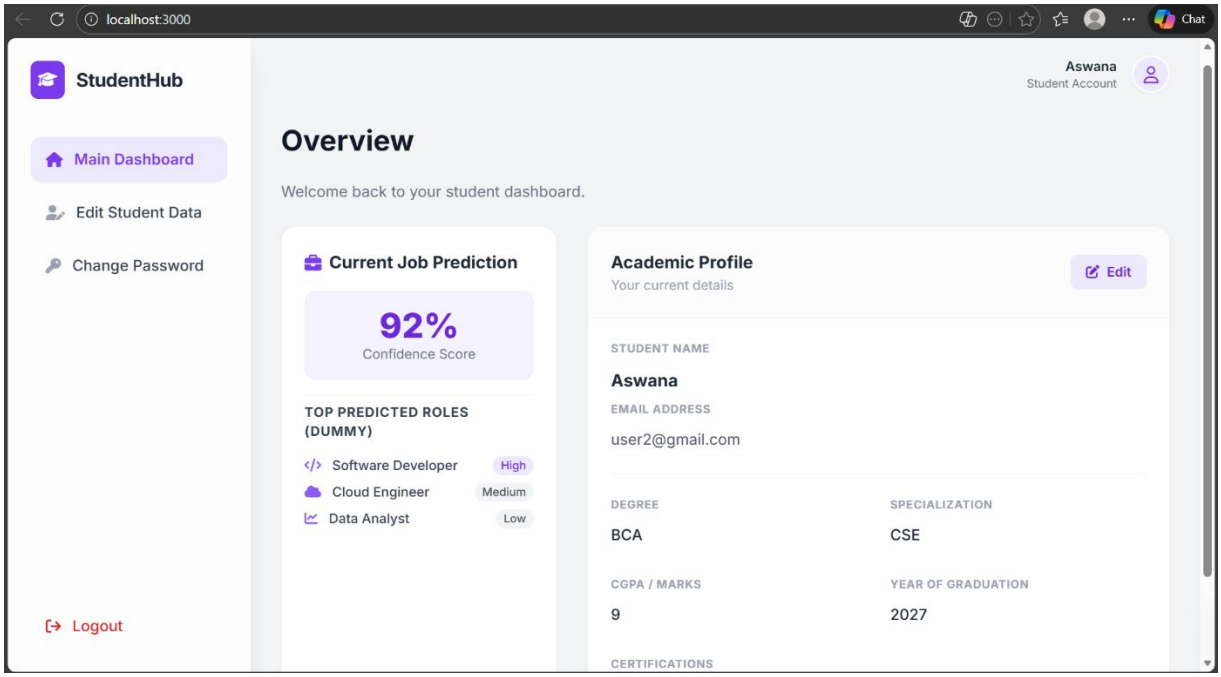
5.Explanation of code

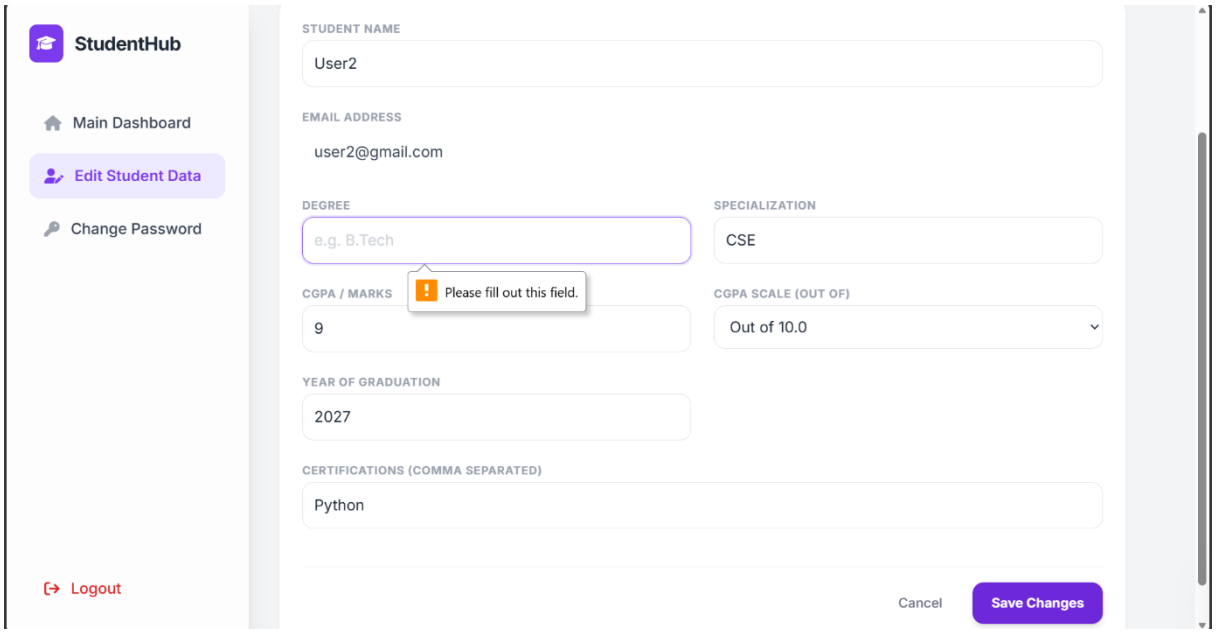
First, Data Integrity was established through finalized frontend input validation and stringent backend validation using Joi, which is active on the POST /education/add route. This ensures raw data, such as CGPA, is checked against the user-selected scale (4.0, 5.0, or 10.0) before being processed. Second, Transformation Logic was Centralized in a dedicated file. This module contains the universal "recipe" for data conversion. The logic successfully handles Categorical Encoding, converting text fields like "Computer Science," "BCA," and specific program codes from external datasets into corresponding numerical features. Crucially, it performs Normalization, scaling CGPA values to the necessary 0-1 range after applying necessary conversion fixes (e.g., scaling 5.0 CGPAs to a 10.0 equivalent).
Third, Universal Proof was generated by running the final logic across all project data paths. The successful live user submission to the index.js server verifies secure storage of the raw data alongside the generated ML feature vector in MongoDB. Furthermore, the batch_processor.js script proved the pipeline's scalability by processing two distinct external academic datasets with varying structures and scales, confirming the data is consistently clean and ready for machine learning. This holistic approach guarantees reliability for the upcoming prediction stage.

6.Output Screenshots




Job Role Predictor
localhost:3000/?
Chat
Discover Your Future Career
AI-powered job role prediction based on your skills and interests.
© 2024 JobPredictor. Secure & Private.
Sign In
Create Account
Welcome Back!
Please enter your details to sign in.
Email
name@example.com
Password
Sign In
Or continue with
Sign in with Google


localhost:3000
Chat
StudentHub
Aswana
Student Account
Main Dashboard
Edit Student Data
Change Password
Logout
Overview
Welcome back to your student dashboard.
Current Job Prediction
92%
Confidence Score
TOP PREDICTED ROLES (DUMMY)
Software Developer
High
Cloud Engineer
Medium
Data Analyst
Low
Academic Profile
Your current details
Edit
STUDENT NAME
Aswana
EMAIL ADDRESS
user2@gmail.com
DEGREE
BCA
SPECIALIZATION
CSE
CGPA / MARKS
9
YEAR OF GRADUATION
2027
CERTIFICATIONS


StudentHub
Main Dashboard
Edit Student Data
Change Password
Logout
STUDENT NAME
User2
EMAIL ADDRESS
user2@gmail.com
DEGREE
e.g. B.Tech
Please fill out this field.
SPECIALIZATION
CSE
CGPA / MARKS
9
CGPA SCALE (OUT OF)
Out of 10.0
YEAR OF GRADUATION
2027
CERTIFICATIONS (COMMA SEPARATED)
Python
Cancel
Save Changes

7.Conclusion:

In Milestone 2, the Edu2Job platform successfully moved from user authentication to establishing a secure, scalable, and robust Education Data Pipeline. The primary objective of transforming raw academic information into a standardized ML feature vector has been fully realized.

The system now features a validated Education Details Input Form that enforces data integrity using Joi Server-Side Validation Critically, the implementation guarantees data universality: the same core preprocessing "recipe" is used to save live user data to MongoDB. This holistic approach results in clean, consistent numerical feature vectors (the ML-ready input). This comprehensive data foundation is now complete and fully verified, making the Edu2Job platform ready for integrating the Machine Learning Prediction Engine and utilizing these processed feature vectors to predict job roles.