Placement Management System

A Project Report

Submitted by

Masrita Mangalarapu (A039)

Priyanshi Mehta (A045)

Jash Mehta (A047)

Under the Guidance of

Prof. Vijayetha Thoday

for the course "Advanced Web Programming"

in partial fulfillment for the award of the degree of

MCA

At

MPSTME, NMIMS.

APRIL 2024

CERTIFICATE

This is to certify that the project entitled "**Placement Management System**" is the bonafide work carried out by **Jash Mehta** of MCA, MPSTME (NMIMS), Mumbai, during the VI semester of the academic year **2023-2024**, in partial fulfillment of the requirements for the Course Advanced Web Programming.

Prof. Vijayetha Thoday

Internal Mentor

Examiner 1

Examiner 2

DECLARATION

We, **Masrita Mangalarapu, Priyanshi Mehta, and Jash Mehta**, Roll No. **A039, A045, A047** MCA, II semester understand that plagiarism is defined as anyone or combination of the following:

1. Un-credited verbatim copying of individual sentences, paragraphs or illustration (such as graphs, diagrams, etc.) from any source, published or unpublished, including the internet.
2. Un-credited improper paraphrasing of pages paragraphs (changing a few words phrases, or rearranging the original sentence order)
3. Credited verbatim copying of a major portion of a paper (or thesis chapter) without clear delineation of who did wrote what. (Source: IEEE, The institute, Dec. 2004)
4. We have made sure that all the ideas, expressions, graphs, diagrams, etc., that are not a result of my work, are properly credited. Long phrases or sentences that had to be used verbatim from published literature have been clearly identified using quotation marks.
5. We affirm that no portion of my work can be considered plagiarism and I take full responsibility if such a complaint occurs. I understand fully well that the guide of the seminar/ project report may not be able to check for the possibility of such incidences of plagiarism in this body of work.

Names: **Masrita Mangalarapu**, **Priyanshi Mehta**, **Jash Mehta**

Roll Nos: **A039, A045, A047**

Place: Mumbai

Date: 3rd April 2024

Table of contents

INTRODUCTION

In the contemporary educational landscape, the role of technology in streamlining administrative processes within academic institutions has become increasingly indispensable. Recognizing this necessity, we present PlaceX, an innovative Placement Management System meticulously crafted using the MERN stack. Aimed at revolutionizing the conventional practices of placement management within colleges, PlaceX serves as a comprehensive solution designed specifically for the exigencies of Placement Departments. Overview PlaceX embodies two integral modules: the User Module and the Admin Module, each catering to distinct user personas while harmoniously facilitating the seamless orchestration of placement-related activities. Through the User Module, students gain access to a sophisticated platform where they can register their academic credentials, explore opportunities presented by recruiting companies, track their application progress, and even share insights through an interview experience repository. Conversely, the Admin Module empowers placement officers with robust tools for managing student data, orchestrating recruitment drives, and overseeing the placement process with precision. At the heart of PlaceX lies a comprehensive array of technologies and libraries meticulously integrated to ensure optimal performance and functionality. The backend API is developed using Node.js, while data storage and management are handled by MongoDB, a versatile NoSQL database. Authentication and authorization are implemented using JSON Web Tokens (JWT), providing a secure mechanism for user validation. To enhance security further, bcrypt is employed for password hashing, ensuring the protection of sensitive user data. Node mailer facilitates seamless communication by enabling the system to send emails to users, keeping them informed about important updates and notifications. Additionally, Express.js serves as a middleware, facilitating the handling of HTTP requests and routes, while cookie-parser helps maintain user sessions effectively.

Key Features User Module:

- Seamless Registration Process
- Comprehensive Company Listing
- Scheduled Interview Management
- FAQ Repository
- Interview Experience Sharing
- Personalized Placement Notifications

Admin Module:

- Robust Student Data Management
- Dynamic Company Management Interface
- Interview Report Generation
- Placement Status
- Tracking Exportable Reports

SCOPE

1. Placement Process Automation:

PlaceX aims to automate the entire placement process within educational institutions, reducing manual intervention and enhancing efficiency.

By providing a centralized platform for managing student data, company details, and interview scheduling, PlaceX streamlines the workflow of Placement Departments.

2. User Registration and Authentication:

PlaceX allows students to register on the platform, providing the academic details necessary for the placement process.

Authentication mechanisms, including JWT tokens and bcrypt password hashing, ensure secure access to the platform and protect sensitive user information.

3. Company Listing and Application Management:

Students can explore job opportunities from various recruiting companies listed on PlaceX.

The platform provides detailed information about each company, including job descriptions, eligibility criteria, and compensation packages.

Students can apply for companies directly through the platform and track the status of their applications.

4. Interview Scheduling and Management:

PlaceX facilitates the scheduling and management of interviews between students and recruiting companies.

Students can view their scheduled interviews, along with relevant details such as date, time, and company information.

5. Interview Experience Sharing:

A dedicated section allows students to share their interview experiences, including difficulty level, outcomes, and insights.

This feature fosters a collaborative environment where students can learn from each other's experiences and prepare more effectively for future interviews.

6. Administrator Tools:

Placement officers have access to comprehensive tools for managing student data, company details, and placement status.

Admins can generate reports based on various criteria such as branch, academic performance, and placement status, aiding in decision-making processes.

7. Scalability and Flexibility:

PlaceX is designed to be highly scalable, capable of accommodating the evolving needs of educational institutions of all sizes.

The modular architecture allows for easy integration of new features and customization based on specific requirements.

8. Enhancing Transparency and Accessibility:

PlaceX promotes transparency in the placement process by providing students and administrators with real-time access to relevant information.

The platform enhances accessibility by offering a user-friendly interface accessible from any device with an internet connection.

9. Continuous Improvement and Support:

PlaceX is a dynamic project that will undergo continuous improvement based on user feedback and emerging trends in the field of placement management.

Dedicated support and maintenance ensure the smooth functioning of the platform and timely resolution of any issues that may arise.

PURPOSE

1. Streamlining Placement Processes:

The primary purpose of PlaceX is to streamline the often complex and time-consuming processes involved in managing placements within educational institutions. By automating tasks such as student registration, company listing, interview scheduling, and data management, PlaceX reduces administrative burden and increases operational efficiency.

2. Promoting Transparency and Accountability:

PlaceX promotes transparency by providing real-time access to information about job opportunities, interview schedules, and placement status for both students and administrators. By centralizing data and communication channels, PlaceX fosters greater accountability among stakeholders, ensuring that all parties are informed and accountable for their respective roles in the placement process.

3. Empowering Students:

One of the core purposes of PlaceX is to empower students to take control of their career aspirations and professional development. Through features such as interview experience sharing and access to resources for interview preparation, PlaceX equips students with the knowledge and tools they need to succeed in the competitive job market.

4. Improving Decision Making:

PlaceX provides placement officers with valuable insights and data-driven analytics that can inform strategic decision making.

By generating reports on student performance, company engagement, and placement outcomes, PlaceX enables administrators to identify trends, evaluate program effectiveness, and make data-driven decisions to optimize the placement process.

DESCRIPTION OF MODULES

1. User Module:

The User Module of PlaceX is designed to cater to the needs of students seeking placement opportunities within the educational institution. This module offers a range of features to streamline the job search process and provide valuable resources for interview preparation. Key functionalities of the User Module include:

Registration and Authentication:

Students can register on the platform by providing their academic details and creating an account. Authentication mechanisms such as JWT tokens ensure secure access to the platform.

Company Listing:

Students can explore a comprehensive listing of companies that are recruiting through the platform. Each listing includes details such as job descriptions, eligibility criteria, and compensation packages.

Application Management:

Students can apply for job opportunities directly through the platform. They can track the status of their applications and receive notifications about interview schedules and outcomes.

Scheduled Interview Management:

A dedicated section allows students to view their scheduled interviews, including details such as date, time, and company information. This feature helps students stay organized and prepared for upcoming interviews.

FAQ Repository:

A repository of frequently asked questions related to coding rounds, technical interviews, and behavioral interviews provides valuable insights and tips for interview preparation.

Interview Experience Sharing:

Students can share their interview experiences with fellow users, providing insights into the difficulty level, outcomes, and overall experience of the interview process. This feature promotes collaboration and knowledge sharing among students.

Placement Notifications: Upon successful placement in a company, students receive personalized notifications congratulating them on their achievement. This feature serves as a source of motivation and recognition for students' accomplishments.

2. Admin Module:

The Admin Module of PlaceX is tailored to the needs of placement officers and administrators responsible for managing the placement process within the educational institution. This module provides robust tools for data management, company administration, and placement tracking. Key functionalities of the Admin Module include:

Student Data Management:

Placement officers can access and manage comprehensive student data, including academic records, application status, and placement outcomes. This feature facilitates efficient tracking of student progress throughout the placement process.

Company Management:

Admins can add, update, and delete company listings within the platform. They can include details such as company name, job description, compensation package, and eligibility criteria, ensuring accurate and up-to-date information for students.

Interview Reports:

A reporting dashboard allows admins to generate reports on student interviews, company engagement, and placement outcomes. This feature provides valuable insights and analytics to inform decision making and program evaluation.

Placement Status Tracking:

Admins can track the placement status of individual students and generate reports based on criteria such as branch, academic performance, and placement status. This feature helps identify trends and patterns to optimize the placement process.

Actionable Insights:

An actionable insights dashboard provides placement officers with actionable insights and recommendations based on data analysis and trends. This feature enables admins to make informed decisions and implement strategic initiatives to improve placement outcomes.

SOFTWARES/ API's/ DATABASE USED WITH DESCRIPTION

1. Node.js:

Description: Node.js is a JavaScript runtime environment that allows you to execute JavaScript code outside of a web browser. It is used to build scalable network applications and is particularly well-suited for building server-side web applications.

2. Express.js:

Description: Express.js is a minimalist web application framework for Node.js. It provides a robust set of features for building web applications and APIs, including routing, middleware support, and template engines.

3. MongoDB:

Description: MongoDB is a NoSQL database that stores data in flexible, JSON-like documents. It is known for its scalability, flexibility, and high performance, making it an ideal choice for handling diverse and rapidly evolving data structures.

4. JSON Web Tokens (JWT):

Description: JSON Web Tokens (JWT) is an open standard for securely transmitting information between parties as a JSON object. JWTs are commonly used for authentication and authorization in web applications, providing a secure mechanism for user validation.

5. bcrypt:

Description: bcrypt is a password-hashing function designed to securely hash passwords. It uses salt to protect against rainbow table attacks and is widely used for storing password hashes securely in databases.

6. Nodemailer:

Description: Nodemailer is a module for Node.js applications that allows you to send emails easily. It supports sending plain text, HTML, and attachments, making it a versatile solution for email communication in web applications.

7. Cookie-parser:

Description: Cookie-parser is a middleware for Express.js that parses cookies attached to the client request object. It simplifies the process of parsing and managing cookies in web applications, enabling developers to handle session management more effectively.

8. Axios:

Description: Axios is a popular HTTP client for making asynchronous HTTP requests in JavaScript applications. It provides an easy-to-use API for sending HTTP requests and handling responses, making it a versatile tool for interacting with APIs and backend services.

9. React:

Description: React is a JavaScript library for building user interfaces. It enables developers to create reusable UI components and build rich, interactive web applications using a declarative and component-based approach.

10. Redux Toolkit:

Description: Redux Toolkit is a package that simplifies the process of managing state in React applications using Redux. It provides utilities and conventions that streamline common Redux patterns, making it easier to write and maintain Redux code.

METHODS IMPLEMENTED

User Endpoints:

1. **User Registration (POST /register):**
   - **Method**: POST
   - **Endpoint:** /register
   - **Description:** Registers a new user by saving their details to the database after validating uniqueness of the email. Password is hashed using bcrypt before saving.

2. **User Login (POST /):**
   - **Method:** POST
   - **Endpoint**: /
   - **Description:** Authenticates user login by comparing hashed passwords from the database. If authentication is successful, a JWT token is generated and sent as a cookie.

3. **Token Verification (GET /verify):**
   - **Method:** GET
   - **Endpoint:** /verify
   - **Description**: Verifies the authenticity of the JWT token sent as a cookie. The middleware function verifyUser is used to check token validity.

4. **Fetch Current User (GET /currentUser):**
   - **Method:** GET
   - **Endpoint:** /currentUser
   - **Description**: Fetches details of the current authenticated user based on the JWT token sent in the request.
5. **Forgot Password (POST /forgotpassword):**
   - **Method:** POST
   - **Endpoint:** /forgotpassword
   - **Description:** Initiates the process for resetting user password by sending a reset password link via email.

6. **Reset Password (POST /resetPassword/:token):**
   - **Method:** POST
   - **Endpoint:** /resetPassword/:token
   - **Description:** Resets user password by verifying the token provided in the request params, then updating the password with the new hashed password.

7. **Apply to Company (POST /applyCompany/:userId/:companyId):**
   - **Method**: POST
   - **Endpoint:** /applyCompany/:userId/:companyId
   - **Description:** Allows a user to apply to a company by adding the company ID to the user's appliedCompanies array.

8. **Scheduled Interviews (GET /scheduledInterviews/:userId):**
   - **Method:** GET
   - **Endpoint:** /scheduledInterviews/:userId
   - **Description:** Retrieves scheduled interviews for a user by fetching companies the user has applied to and their interview dates.

9. **Add Interview Experience (POST /add-interview):**
   - **Method:** POST
   - **Endpoint:** /add-interview
   - **Description:** Allows a user to add their interview experience to the database by providing details such as company name, position, experience, interview level, and result.

10. **Fetch Interview Experiences (GET /fetchinterviewexperience):**
    - **Method:** GET
    - **Endpoint:** /fetchinterviewexperience
    - **Description:** Fetches interview experiences posted by users from the database.

Admin Endpoints:

1. **Add Company (POST /add-companies):**
   - **Method:** POST
   - **Endpoint:** /add-companies
   - **Description:** Allows an admin to add new company details to the database.

2. **Get Users (GET /getUsers):**
   - **Method**: GET
   - **Endpoint:** /getUsers
   - **Description:** Retrieves all users from the database for generating user reports.

3. **Get Companies (GET /getCompanies):**
   - **Method:** GET
   - **Endpoint:** /getCompanies
   - **Description:** Retrieves all companies from the database for admin view.

4. **Update Company Details (PUT /updatecompany/:id):**
   - **Method**: PUT
   - **Endpoint:** /updatecompany/:id
   - **Description:** Updates company details based on the provided ID.

5. **Delete Company (DELETE /deletecompany/:id):**
   - **Method:** DELETE
   - **Endpoint**: /deletecompany/:id
   - **Description:** Deletes company details based on the provided ID.

6. **Get Specific Company (GET /getCompanies/:id):**
   - **Method:** GET
   - **Endpoint:** /getCompanies/:id
   - **Description:** Retrieves specific company details based on the provided ID.

7. **Company Applicants (GET /companyApplicants):**
   - **Method:** GET
   - **Endpoint:** /companyApplicants
   - **Description:** Fetches users who have applied to each company along with their details.

8. **Update Placement Status (POST /updatePlacementStatus):**
   - **Method:** POST
   - **Endpoint:** /updatePlacementStatus
   - **Description:** Updates the placement status of a user based on the provided user ID and company ID.

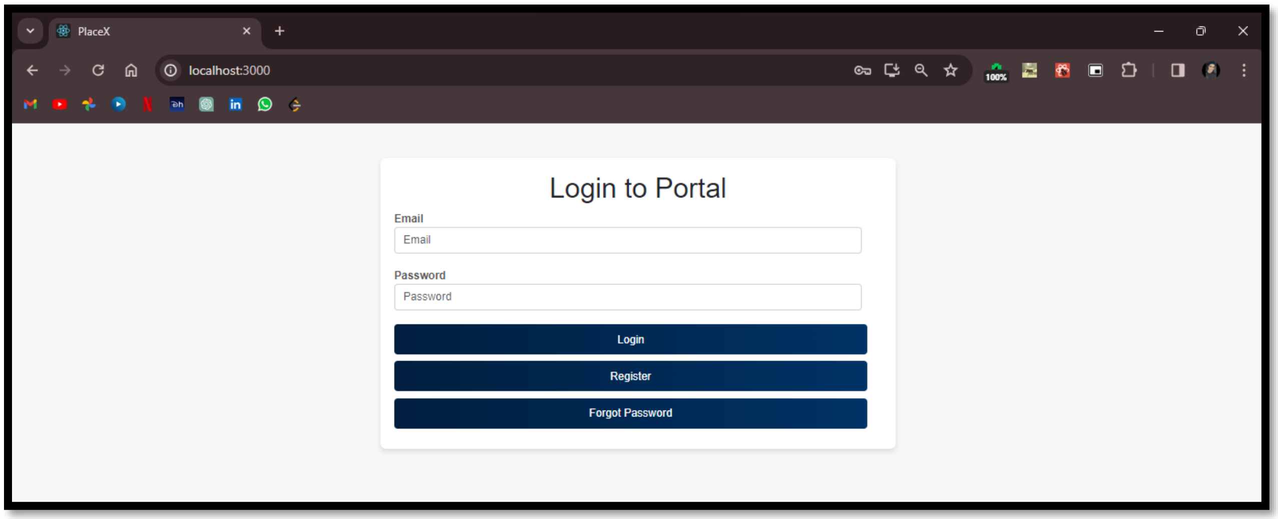
CODE SNIPPETS

routes/user.js

```
import express from "express";
import bcryt from "bcrypt";
const router = express.Router();
import { User } from "../models/user.js";
import { Company } from "../models/Company.js";
import { Interview } from "../models/Experience.js";
import jwt from "jsonwebtoken";
import nodemailer from "nodemailer";

//---------------------------------------------USER ENDPOINTS--------------------
------------------------------//

//User Registeration API
router.post("/register", async (req, res) => {
  const {
    name,
    email,
    password,
    contactNumber,
    sapId,
    rollNo,
    gender,
    dob,
    tenthPercentage,
    tenthSchool,
    twelfthPercentage,
    twelfthCollege,
    graduationCollege,
    graduationCGPA,
    stream,
    sixthSemesterCGPA,
    isAdmin,
  } = req.body;
  const user = await User.findOne({ email });
  if (user) {
    return res.json({ message: "User already existed" });
  }

  const hashpassword = await bcryt.hash(password, 10);
  const newUser = new User({
    name,
    email,
    password: hashpassword,
    contactNumber,
    sapId,
    rollNo,
    gender,
    dob,
    tenthPercentage,
    tenthSchool,
    twelfthPercentage,
    twelfthCollege,
    graduationCollege,
    graduationCGPA,
    stream,
    sixthSemesterCGPA,
    isAdmin,
  });

  await newUser.save();
  return res.json({ message: "User Registered" });
});

//User and Admin Login API
router.post("/", async (req, res) => {
  const { email, password } = req.body;
  const user = await User.findOne({ email });

  if (!user) {
    console.log("Invalid User");
    return res.json("Invalid User");
  }

  const validPassword = await bcryt.compare(password, user.password);
  if (!validPassword) {
    console.log("Password Incorrect");
    return res.json("Password Incorrect");
  }

  if (user.isAdmin === "1") {
    console.log("User is admin");
  }

  const token = jwt.sign(
    { _id: user._id, username: user.username },
    process.env.KEY,
```

```javascript
    { expiresIn: "1h" }
  );

  res.cookie("token", token, { httpOnly: true, maxAge: 300000 });

  return res.json(user.isAdmin === "1" ? "Admin" : "Success");
});

// Middleware function to verify the authenticity of a user's token before
granting access to protected routes.
const verifyUser = async (req, res, next) => {
  try {
    const token = req.cookies.token;
    if (!token) {
      return res.json({ status: false, message: "No Token" });
    }
    const decoded = jwt.verify(token, process.env.KEY);
    next();
  } catch (err) {
    return res.json(err);
  }
};

// Route to verify the authenticity of a user's token.
// It utilizes the verifyUser middleware to ensure that the user is
authenticated.
router.get("/verify", verifyUser, (req, res) => {
  return res.json({ status: true, message: "Authorized" });
});

// Route to fetch the current user's details.
// It verifies the user's token and retrieves the user's information.
router.get("/currentUser", verifyUser, async (req, res) => {
  try {
    const token = req.cookies.token;
    const decoded = jwt.verify(token, process.env.KEY);
    const userId = decoded._id;

    const user = await User.findById(userId);
    if (!user) {
      return res.status(404).json({ message: "User not found" });
    }

    return res.json({ user });
  } catch (error) {
    console.error(error);
    return res.status(500).json({ message: "Internal Server Error" });
  }
});

// This API is designed to handle the functionality of sending a reset password
link via email to the user.
router.post("/forgotpassword", async (req, res) => {
  const { email } = req.body;
  try {
    const user = await User.findOne({ email });
    if (!user) {
      return res.json({ message: "User not registered" });
    }
    const token = jwt.sign({ id: user._id }, process.env.KEY, {
      expiresIn: "5m",
    });

    var transporter = nodemailer.createTransport({
      service: "gmail",
      auth: {
        user: "jashmehtaa@gmail.com",
        pass: "ulda isbm aail fzof",
      },
    });

    var mailOptions = {
      from: "jashmehtaa@gmail.com",
      to: email,
      subject: "Reset Password Link",
      text: `http://localhost:3000/resetPassword/${token}`,
    };

    transporter.sendMail(mailOptions, function (error, info) {
      if (error) {
        return res.json({ status: true, message: "Error sending the email" });
      } else {
        return res.json({ status: true, message: "Email Sent" });
      }
    });
  } catch (err) {
    console.log(err);
  }
});

// This API endpoint is responsible for resetting the password of a user. It
verifies the provided token,
// then updates the user's password with the new hashed password. If the token is
invalid, it returns an
// error response indicating the token is invalid.
router.post("/resetPassword/:token", async (req, res) => {
  const { token } = req.params;
  const { password } = req.body;

  try {
    const decoded = await jwt.verify(token, process.env.KEY);
    const id = decoded.id;

    const hashPassword = await bcryt.hash(password, 10);

    await User.findByIdAndUpdate({ _id: id }, { password: hashPassword });

    return res.json({ status: true, message: "Updated Password Successfully" });
  } catch (err) {
    console.error(err);
    return res.status(400).json({ status: false, message: "Invalid Token" });
  }
});

//API to add a company ID to appliedCompanies array for a user
router.post("/applyCompany/:userId/:companyId", async (req, res) => {
  const { userId, companyId } = req.params;
  console.log("User ID: ", userId);
  console.log("Company ID:", companyId);

  try {
    const user = await User.findById(userId);
    console.log("User:", user);

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

    if (!user.appliedCompanies) {
      user.appliedCompanies = [];
    }

    if (user.appliedCompanies.includes(companyId)) {
      return res
        .status(400)
        .json({ message: "User already applied to this company" });
    }
    user.appliedCompanies.push(companyId);
    await user.save();

    return res.json({ message: "Company applied successfully" });
  } catch (error) {
    console.error(error);
    return res.status(500).json({ message: "Internal Server Error" });
  }
});

// Endpoint to retrieve scheduled interviews for a user
router.get("/scheduledInterviews/:userId", async (req, res) => {
  const { userId } = req.params;

  try {
    const user = await User.findById(userId);
    if (!user) {
      return res.status(404).json({ message: "User not found" });
    }

    const appliedCompanyIds = user.appliedCompanies;
    const companies = await Company.find({ _id: { $in: appliedCompanyIds } });

    const scheduledInterviews = companies.map((company) => ({
      companyName: company.companyname,
      interviewDate: company.doi,
    }));

    return res.json({ scheduledInterviews });
  } catch (error) {
    console.error(error);
    return res.status(500).json({ message: "Internal Server Error" });
  }
});

//API to post interview experience
router.post("/add-interview", async (req, res) => {
  try {
    const {
      username,
      companyName,
      position,
      experience,
      interviewLevel,
      result,
    } = req.body;
    const newInterview = new Interview({
      username,
      companyName,
      position,
      experience,
      interviewLevel,
      result,
    });

    await newInterview.save();

    return res.json({ message: "Interview experience added successfully" });
  } catch (error) {
    console.error("Error adding interview experience:", error);
    return res.status(500).json({ message: "Internal Server Error" });
  }
});

//API to fetch the interview experiences on the feed
router.get("/fetchinterviewexperience", async (req, res) => {
  try {
    const interviews = await Interview.find({});
    return res.json({ data: interviews });
  } catch (error) {
    console.error("Error fetching interview experiences:", error);
    return res.status(500).json({ message: "Internal Server Error" });
  }
});

router.get('/placementStatus/:userId', async (req, res) => {
  try {
    const { userId } = req.params;

    // Find the user by userId
    const user = await User.findById(userId);
    if (!user) {
      return res.status(404).json({ message: 'User not found.' });
    }

    // Get the placement status
    const status = user.placementStatus;

    if (status === 'Placed') {
      // If the status is placed, get the company name from the user document
      const companyName = user.companyPlaced;
      return res.json({ status, companyName });
    }

    return res.json({ status });
  } catch (error) {
    console.error('Error fetching placement status:', error);
    res.status(500).json({ message: 'Internal Server Error' });
  }
});


//---------------------------------------------ADMIN ENDPOINTS------------------
-------------------------------//

// This API endpoint is responsible for adding new company details to the
database.
router.post("/add-companies", async (req, res) => {
  const {
    companyname,
    jobprofile,
    jobdescription,
    website,
    ctc,
    doi,
    eligibilityCriteria,
    tenthPercentage,
    twelfthPercentage,
    graduationCGPA,
    sixthSemesterCGPA,
  } = req.body;

  try {
    const newCompany = new Company({
      companyname,
      jobprofile,
      jobdescription,
      website,
      ctc,
      doi,
      eligibilityCriteria,
      tenthPercentage,
      twelfthPercentage,
      graduationCGPA,
      sixthSemesterCGPA,
    });
    await newCompany.save();
    console.log(newCompany);
    return res.json({ message: "Company Registered" });
  } catch (error) {
    console.error(error);
    return res.status(500).json({ message: "Internal Server Error" });
  }
});

// Route to fetch all users and generate user reports.
// It retrieves all users from the database and sends them as a response.
router.get("/getUsers", async (req, res) => {
  try {
    const allUsers = await User.find({});

    res.send({ data: allUsers });
  } catch (error) {
    console.log(error);
  }
});

// Route to fetch all companies.
// It retrieves all companies from the database and sends them as a response.
router.get("/getCompanies", async (req, res) => {
  try {
    const allCompanies = await Company.find({});

    res.send({ data: allCompanies });
  } catch (error) {
    console.log(error);
  }
});

// Route to update company data.
// It updates the company details based on the provided ID.
router.put("/updatecompany/:id", (req, res) => {
  const id = req.params.id;
  Company.findByIdAndUpdate(id, {
    companyname: req.body.companyname,
    jobprofile: req.body.jobprofile,
    ctc: req.body.ctc,
    doi: req.body.doi,
    tenthPercentage: req.body.tenthPercentage,
    twelfthPercentage: req.body.twelfthPercentage,
    graduationCGPA: req.body.graduationCGPA,
    sixthSemesterCGPA: req.body.sixthSemesterCGPA,
  })
    .then((company) => res.json(company))
    .catch((err) => res.json(err));
});

// Route to delete company data.
// It deletes the company based on the provided ID.
router.delete("/deletecompany/:id", (req, res) => {
  const id = req.params.id;
  Company.findByIdAndDelete({ _id: id })
    .then((response) => res.json(response))
    .catch((err) => res.json(err));
});

// Route to fetch a specific company by ID.
// It retrieves the company details based on the provided ID.
router.get("/getCompanies/:id", async (req, res) => {
  const id = req.params.id;
  try {
    const company = await Company.findById(id);

    res.send({ data: company });
  } catch (error) {
    console.log(error);
  }
});

//This API fetches the users and the companies they have applied to
router.get("/companyApplicants", async (req, res) => {
  try {
    const companies = await Company.find(); // Assuming you have a Company model

    const companyData = [];

    for (const company of companies) {
      const applicants = await User.find({ appliedCompanies: company._id });

      const companyInfo = {
        companyId: company._id,
        companyName: company.companyname,
        applicants: applicants.map((applicant) => ({
          userId: applicant._id,
          name: applicant.name,
          email: applicant.email,
        })),
      };
      companyData.push(companyInfo);
    }

    res.json(companyData);
  } catch (error) {
    console.error("Error fetching company applicants:", error);
    res.status(500).json({ message: "Internal Server Error" });
  }
});

// Backend API to update placementStatus
router.post("/updatePlacementStatus", async (req, res) => {
  try {
    const { userId, companyId, status } = req.body;
    const user = await User.findById(userId);
    if (!user) {
      return res.status(404).json({ message: "User not found." });
    }
    if (user.placementStatus === "Placed" && status === "Placed") {
      return res.status(200).json({ message: "User is already placed." });
    }
    const company = await Company.findById(companyId);
    console.log(company.companyname);
    if (!company) {
      return res.status(404).json({ message: "Company not found." });
    }
    user.placementStatus = status;
    user.companyPlaced = company.companyname;
    await user.save();
    res.json({
      message: `Placement status updated to ${status} successfully.`,
    });
  } catch (error) {
    console.error("Error updating placement status:", error);
    res.status(500).json({ message: "Internal Server Error" });
  }
});

export { router as UserRouter };
```

GUI/DATABASE SCREENSHOTS:



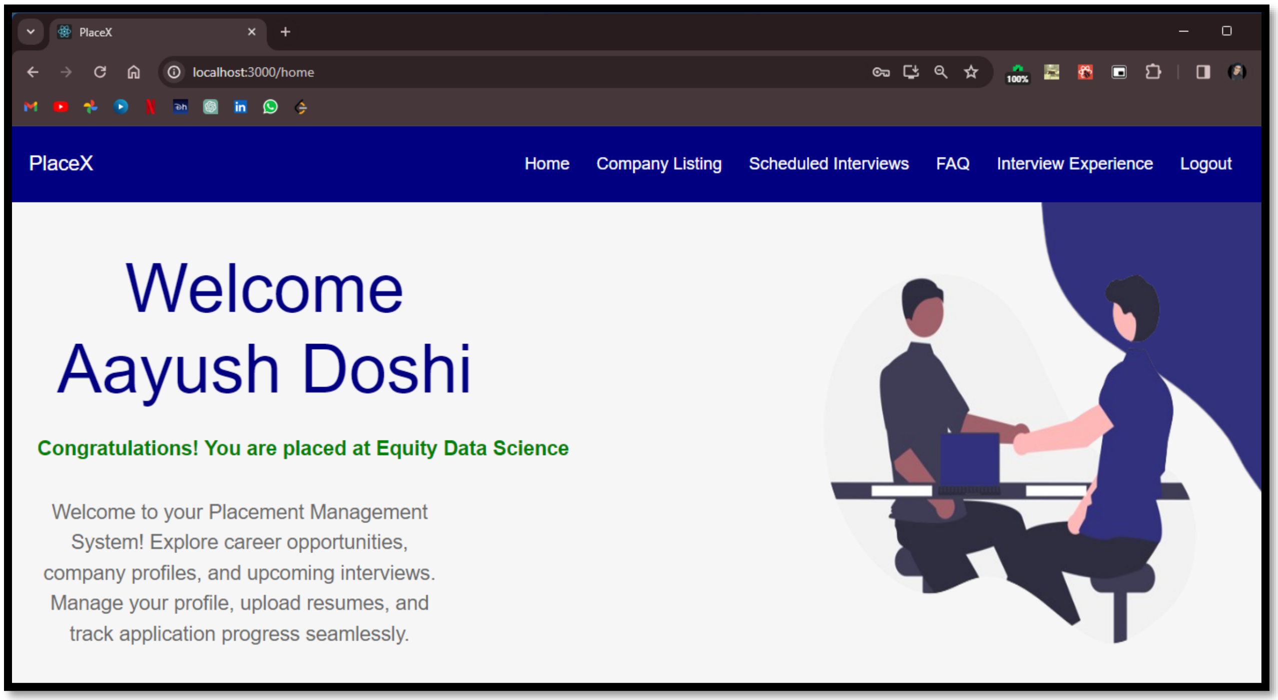
Login to Portal

Email

Email

Password

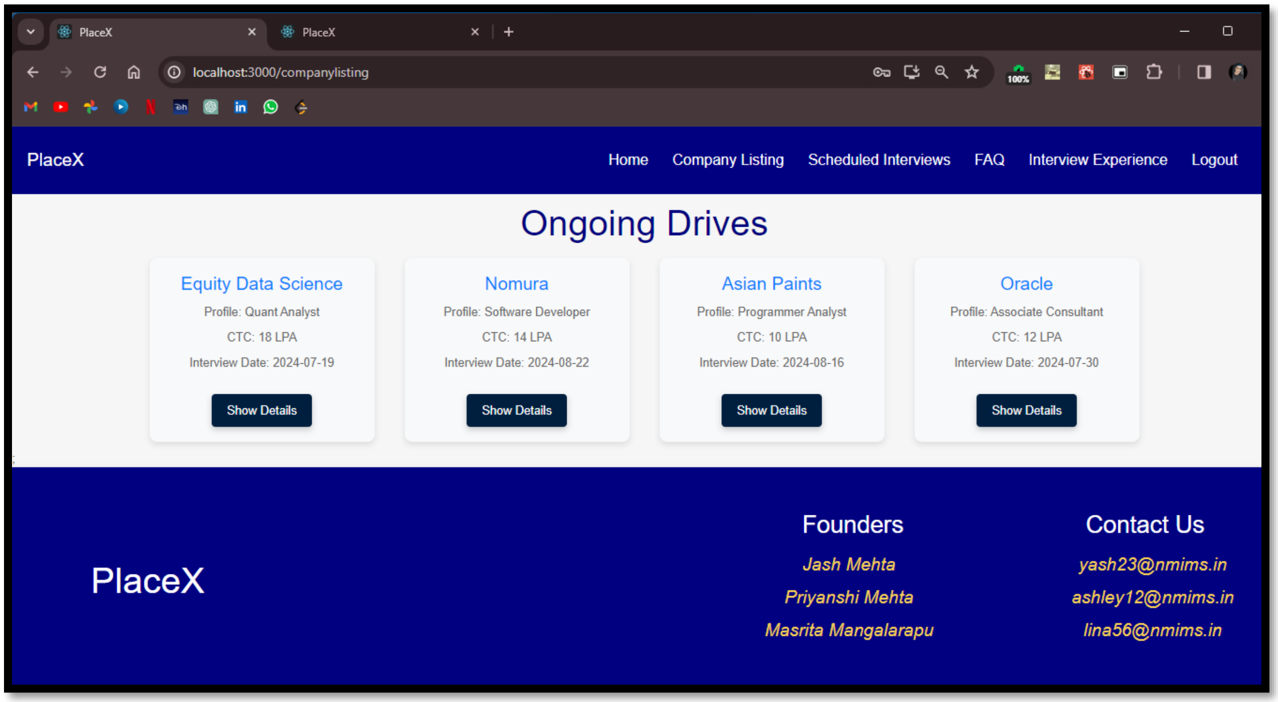
Password



Welcome
Aayush Doshi

Congratulations! You are placed at Equity Data Science

Welcome to your Placement Management System! Explore career opportunities, company profiles, and upcoming interviews. Manage your profile, upload resumes, and track application progress seamlessly.

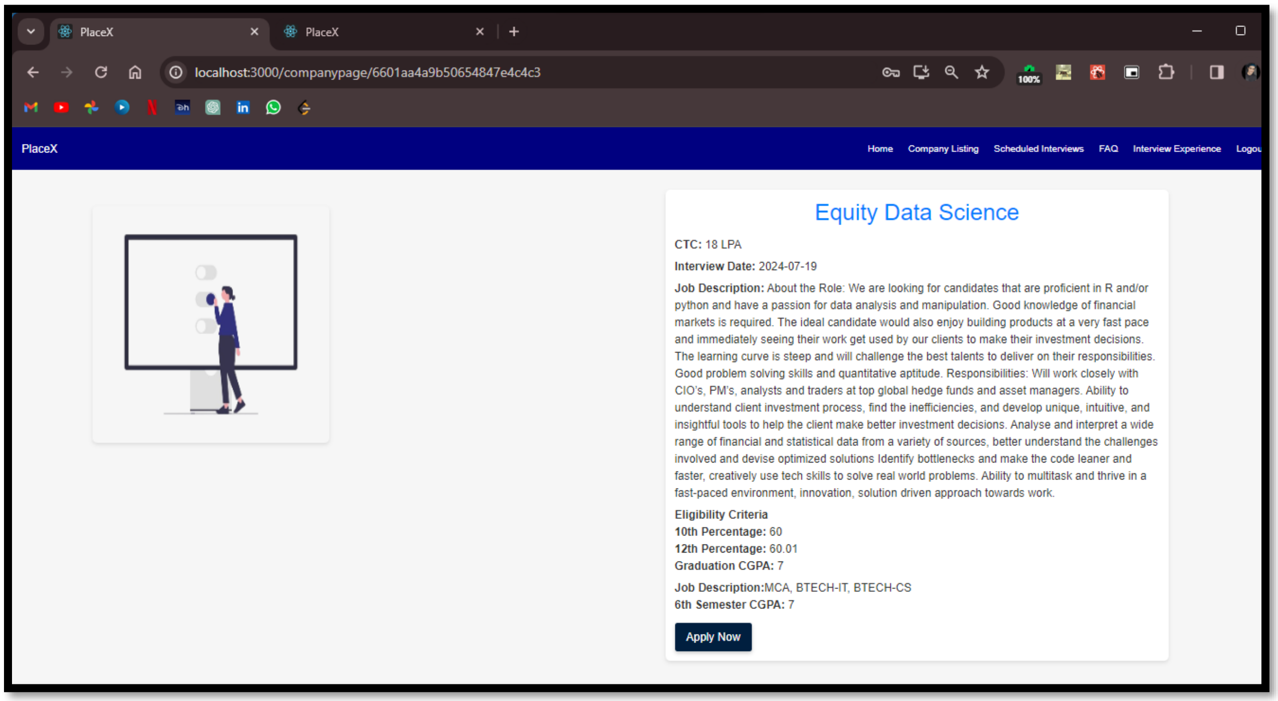


Ongoing Drives

Equity Data Science	Nomura	Asian Paints	Oracle
Profile: Quant Analyst	Profile: Software Developer	Profile: Programmer Analyst	Profile: Associate Consultant
CTC: 18 LPA	CTC: 14 LPA	CTC: 10 LPA	CTC: 12 LPA
Interview Date: 2024-07-19	Interview Date: 2024-08-22	Interview Date: 2024-08-16	Interview Date: 2024-07-30
Show Details	Show Details	Show Details	Show Details

PlaceX

Founders	Contact Us
Jash Mehta	*yash23@nmims.in*
Priyanshi Mehta	*ashley12@nmims.in*
Masrita Mangalarapu	*lina56@nmims.in*



Equity Data Science

CTC: 18 LPA

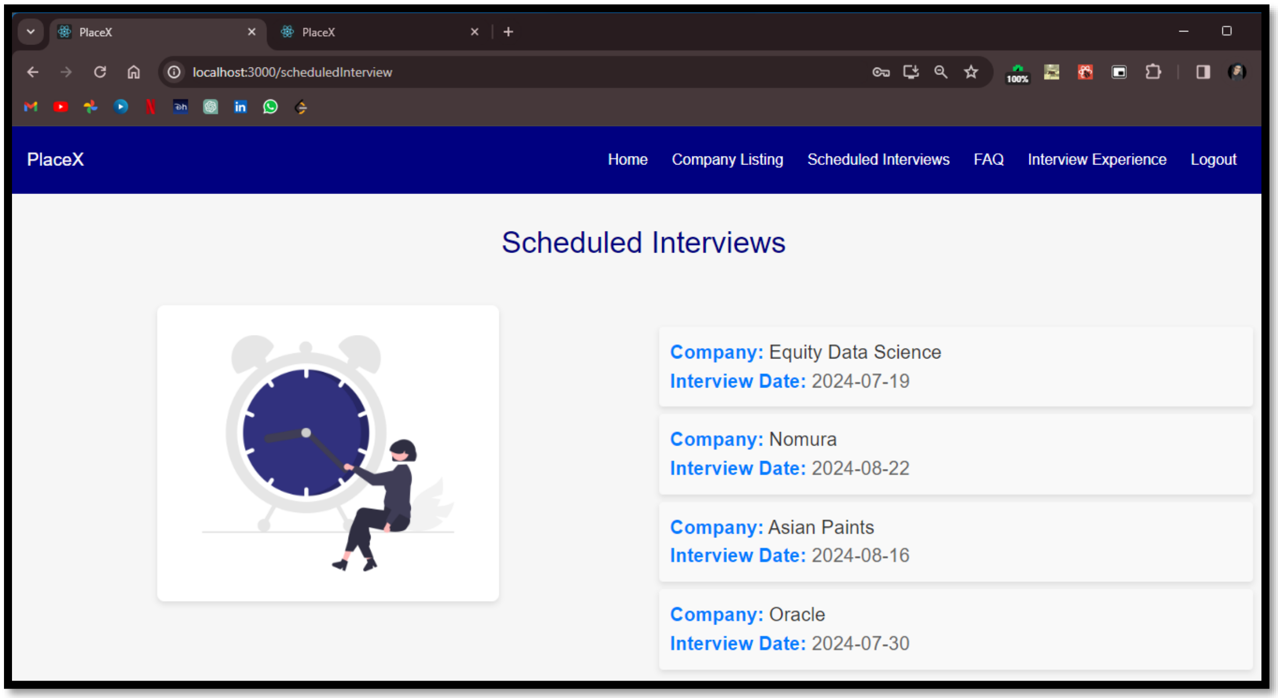
Interview Date: 2024-07-19

Job Description: About the Role: We are looking for candidates that are proficient in R and/or python and have a passion for data analysis and manipulation. Good knowledge of financial markets is required. The ideal candidate would also enjoy building products at a very fast pace and immediately seeing their work get used by our clients to make their investment decisions. The learning curve is steep and will challenge the best talents to deliver on their responsibilities. Good problem solving skills and quantitative aptitude. Responsibilities: Will work closely with CIO's, PM's, analysts and traders at top global hedge funds and asset managers. Ability to understand client investment process, find the inefficiencies, and develop unique, intuitive, and insightful tools to help the client make better investment decisions. Analyse and interpret a wide range of financial and statistical data from a variety of sources, better understand the challenges involved and devise optimized solutions Identify bottlenecks and make the code leaner and faster, creatively use tech skills to solve real world problems. Ability to multitask and thrive in a fast-paced environment, innovation, solution driven approach towards work.

Eligibility Criteria
10th Percentage: 60
12th Percentage: 60.01
Graduation CGPA: 7

Job Description:MCA, BTECH-IT, BTECH-CS
6th Semester CGPA: 7

Apply Now



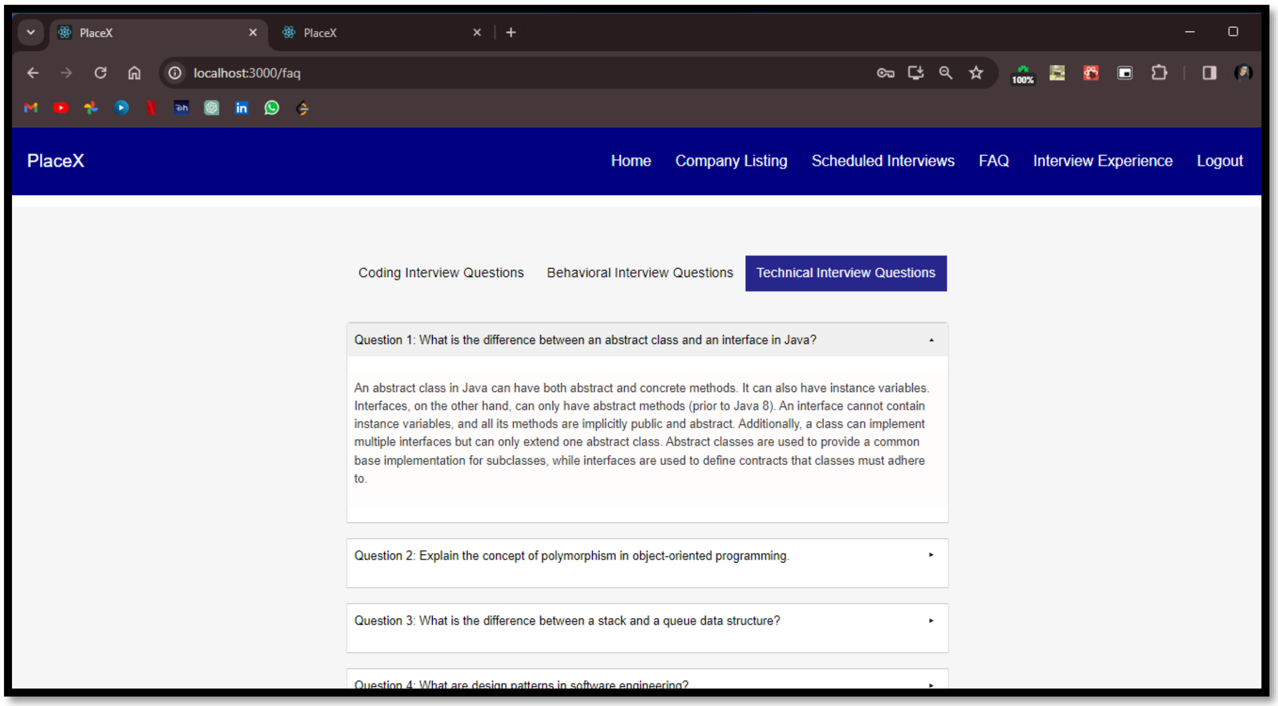
Scheduled Interviews

Company: Equity Data Science
Interview Date: 2024-07-19

Company: Nomura
Interview Date: 2024-08-22

Company: Asian Paints
Interview Date: 2024-08-16

Company: Oracle
Interview Date: 2024-07-30



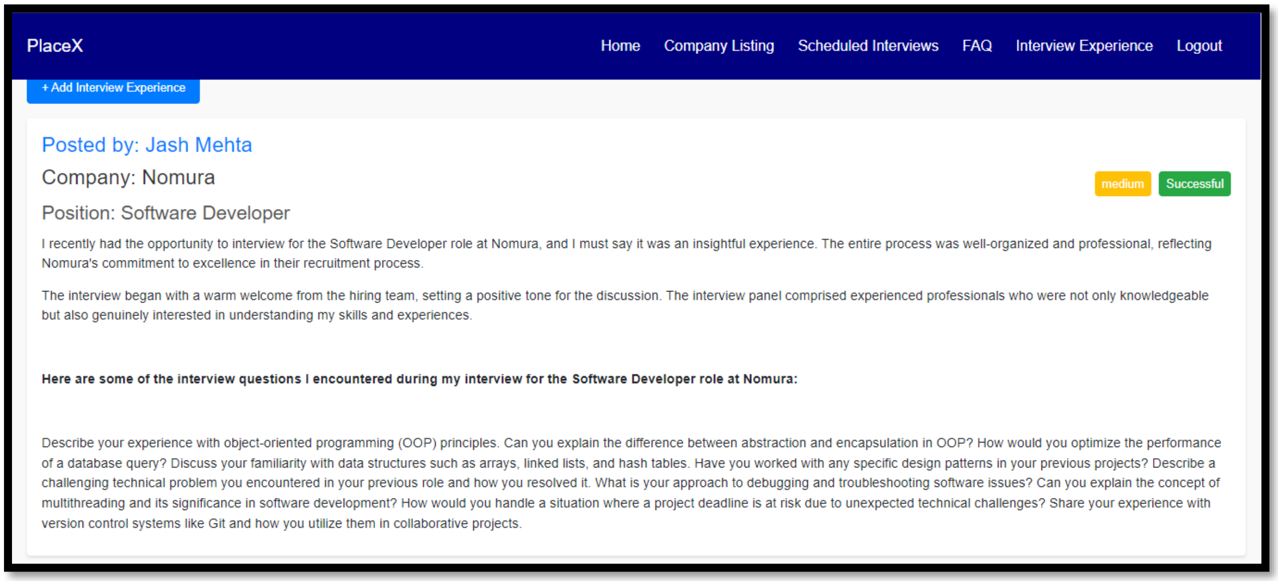
Question 1: What is the difference between an abstract class and an interface in Java?

An abstract class in Java can have both abstract and concrete methods. It can also have instance variables. Interfaces, on the other hand, can only have abstract methods (prior to Java 8). An interface cannot contain instance variables, and all its methods are implicitly public and abstract. Additionally, a class can implement multiple interfaces but can only extend one abstract class. Abstract classes are used to provide a common base implementation for subclasses, while interfaces are used to define contracts that classes must adhere to.

Question 2: Explain the concept of polymorphism in object-oriented programming.

Question 3: What is the difference between a stack and a queue data structure?

Question 4: What are design patterns in software engineering?



Posted by: Jash Mehta

Company: Nomura

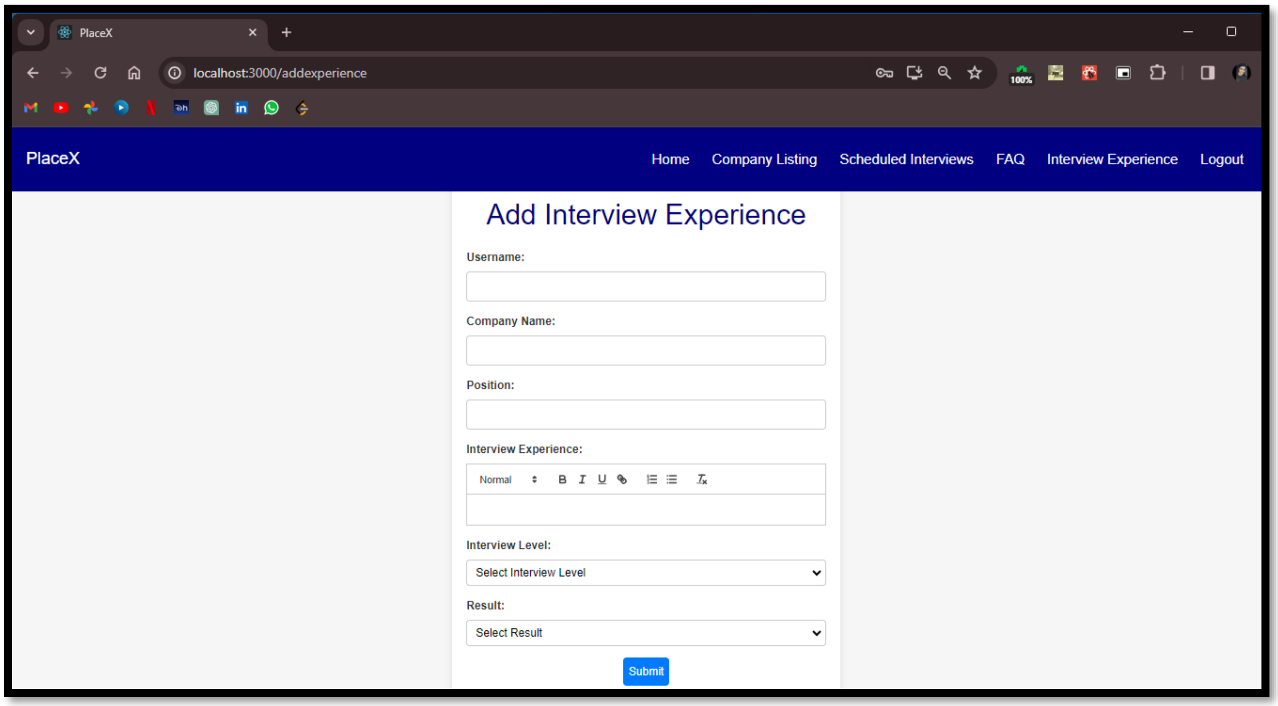
Position: Software Developer

I recently had the opportunity to interview for the Software Developer role at Nomura, and I must say it was an insightful experience. The entire process was well-organized and professional, reflecting Nomura's commitment to excellence in their recruitment process.

The interview began with a warm welcome from the hiring team, setting a positive tone for the discussion. The interview panel comprised experienced professionals who were not only knowledgeable but also genuinely interested in understanding my skills and experiences.

Here are some of the interview questions I encountered during my interview for the Software Developer role at Nomura:

Describe your experience with object-oriented programming (OOP) principles. Can you explain the difference between abstraction and encapsulation in OOP? How would you optimize the performance of a database query? Discuss your familiarity with data structures such as arrays, linked lists, and hash tables. Have you worked with any specific design patterns in your previous projects? Describe a challenging technical problem you encountered in your previous role and how you resolved it. What is your approach to debugging and troubleshooting software issues? Can you explain the concept of multithreading and its significance in software development? How would you handle a situation where a project deadline is at risk due to unexpected technical challenges? Share your experience with version control systems like Git and how you utilize them in collaborative projects.



Add Interview Experience

Username:

Company Name:

Position:

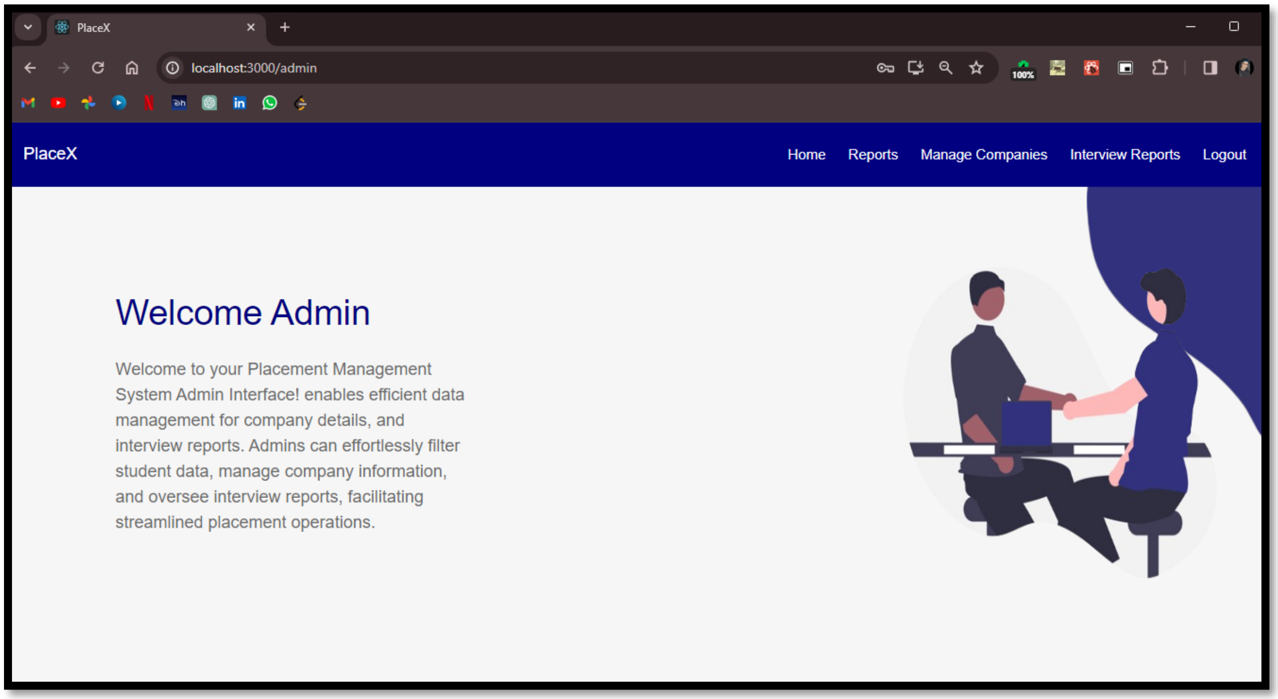
Interview Experience:

Interview Level:

Select Interview Level

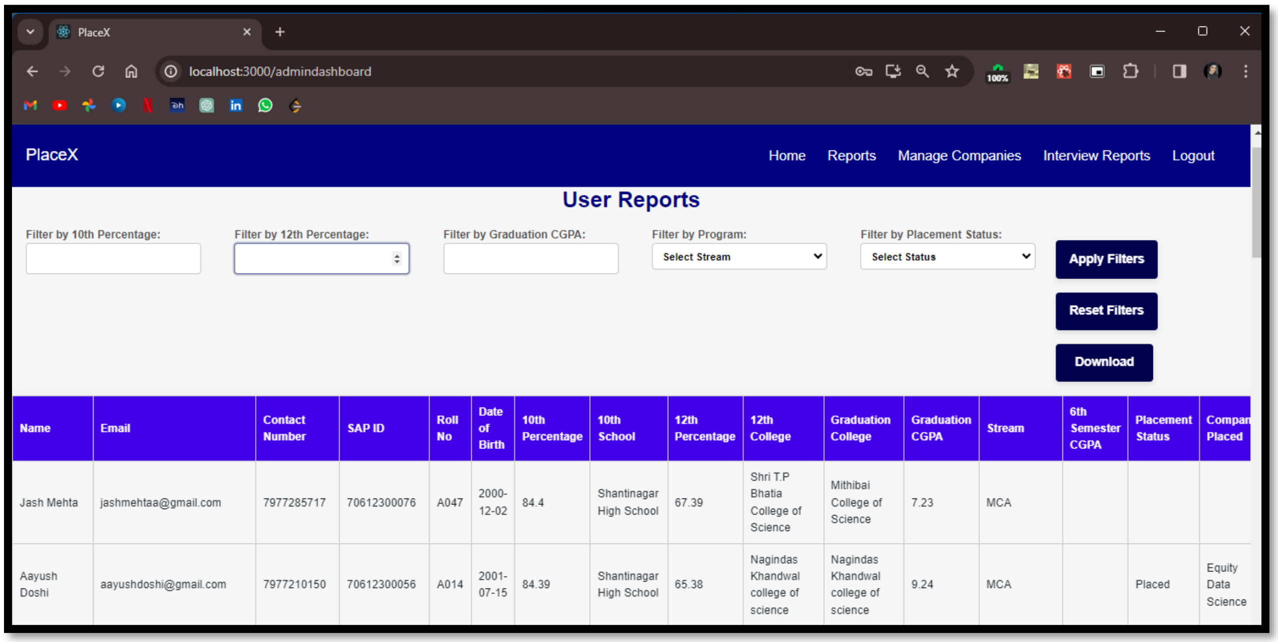
Result:

Select Result



Welcome Admin

Welcome to your Placement Management System Admin Interface! enables efficient data management for company details, and interview reports. Admins can effortlessly filter student data, manage company information, and oversee interview reports, facilitating streamlined placement operations.



User Reports

Filter by 10th Percentage:

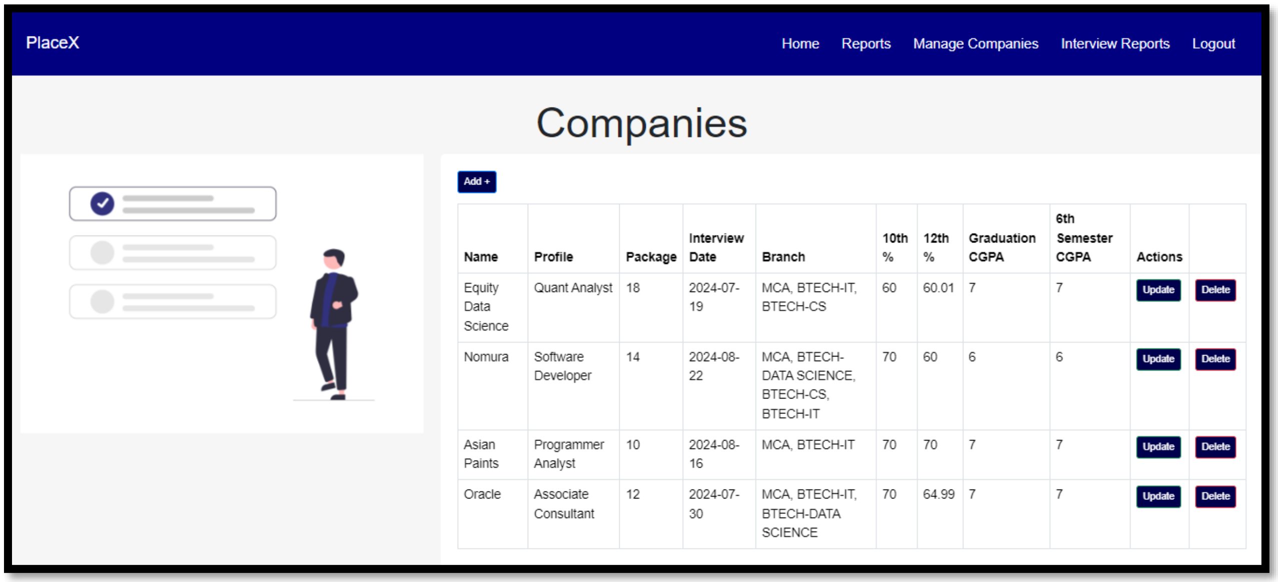
Filter by 12th Percentage:

Filter by Graduation CGPA:

Filter by Program: Select Stream

Filter by Placement Status: Select Status

Name	Email	Contact Number	SAP ID	Roll No	Date of Birth	10th Percentage	10th School	12th Percentage	12th College	Graduation College	Graduation CGPA	Stream	6th Semester CGPA	Placement Status	Compan Placed
Jash Mehta	jashmehtaa@gmail.com	7977285717	70612300076	A047	2000-12-02	84.4	Shantinagar High School	67.39	Shri T.P Bhatia College of Science	Mithibai College of Science	7.23	MCA			
Aayush Doshi	aayushdoshi@gmail.com	7977210150	70612300056	A014	2001-07-15	84.39	Shantinagar High School	65.38	Nagindas Khandwal college of science	Nagindas Khandwal college of science	9.24	MCA		Placed	Equity Data Science

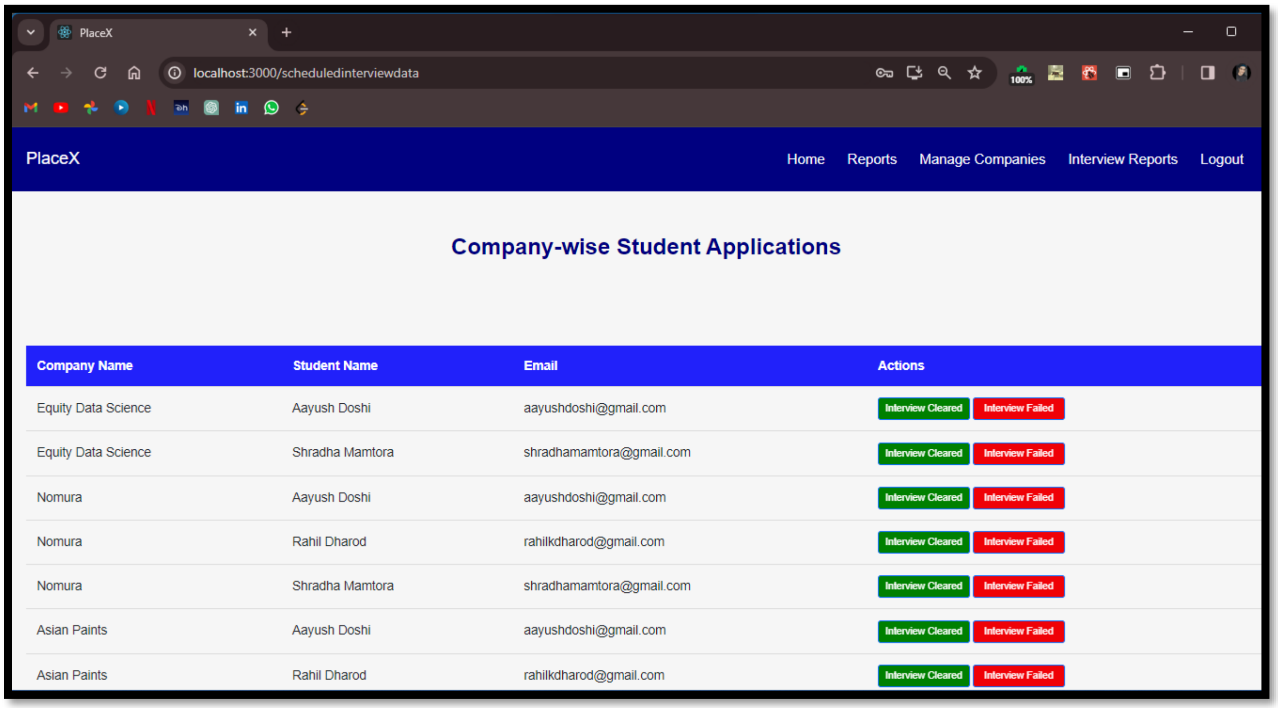


PlaceX — Home | Reports | Manage Companies | Interview Reports | Logout

Companies

Add +

Name	Profile	Package	Interview Date	Branch	10th %	12th %	Graduation CGPA	6th Semester CGPA	Actions	
Equity Data Science	Quant Analyst	18	2024-07-19	MCA, BTECH-IT, BTECH-CS	60	60.01	7	7	Update	Delete
Nomura	Software Developer	14	2024-08-22	MCA, BTECH-DATA SCIENCE, BTECH-CS, BTECH-IT	70	60	6	6	Update	Delete
Asian Paints	Programmer Analyst	10	2024-08-16	MCA, BTECH-IT	70	70	7	7	Update	Delete
Oracle	Associate Consultant	12	2024-07-30	MCA, BTECH-IT, BTECH-DATA SCIENCE	70	64.99	7	7	Update	Delete

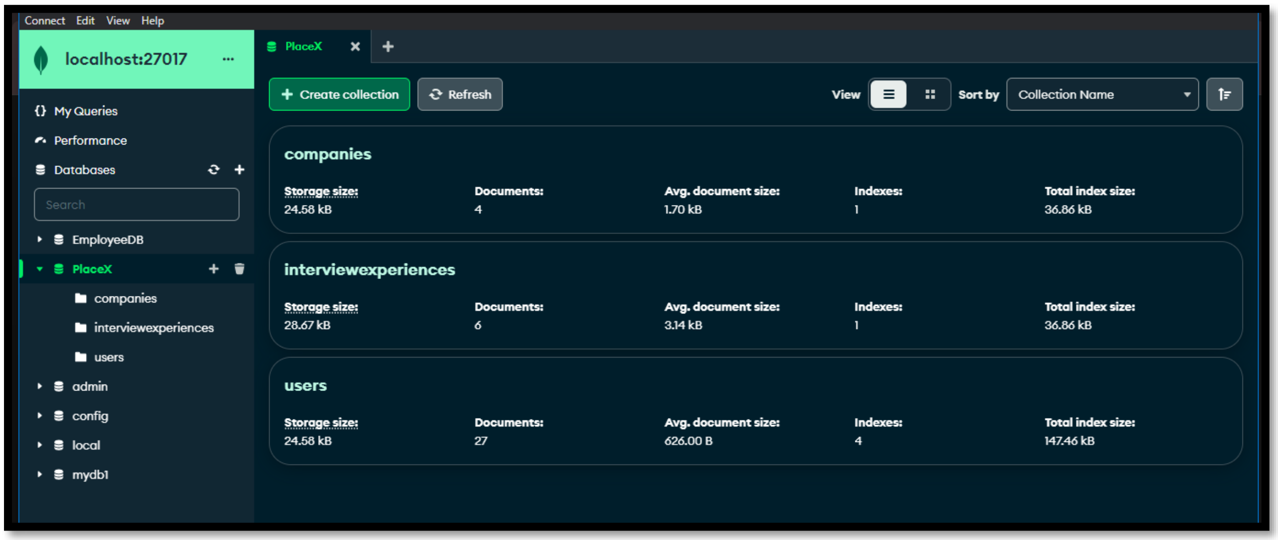


localhost:3000/scheduledinterviewdata

PlaceX — Home | Reports | Manage Companies | Interview Reports | Logout

Company-wise Student Applications

Company Name	Student Name	Email	Actions
Equity Data Science	Aayush Doshi	aayushdoshi@gmail.com	Interview Cleared / Interview Failed
Equity Data Science	Shradha Mamtora	shradhamamtora@gmail.com	Interview Cleared / Interview Failed
Nomura	Aayush Doshi	aayushdoshi@gmail.com	Interview Cleared / Interview Failed
Nomura	Rahil Dharod	rahilkdharod@gmail.com	Interview Cleared / Interview Failed
Nomura	Shradha Mamtora	shradhamamtora@gmail.com	Interview Cleared / Interview Failed
Asian Paints	Aayush Doshi	aayushdoshi@gmail.com	Interview Cleared / Interview Failed
Asian Paints	Rahil Dharod	rahilkdharod@gmail.com	Interview Cleared / Interview Failed



companies

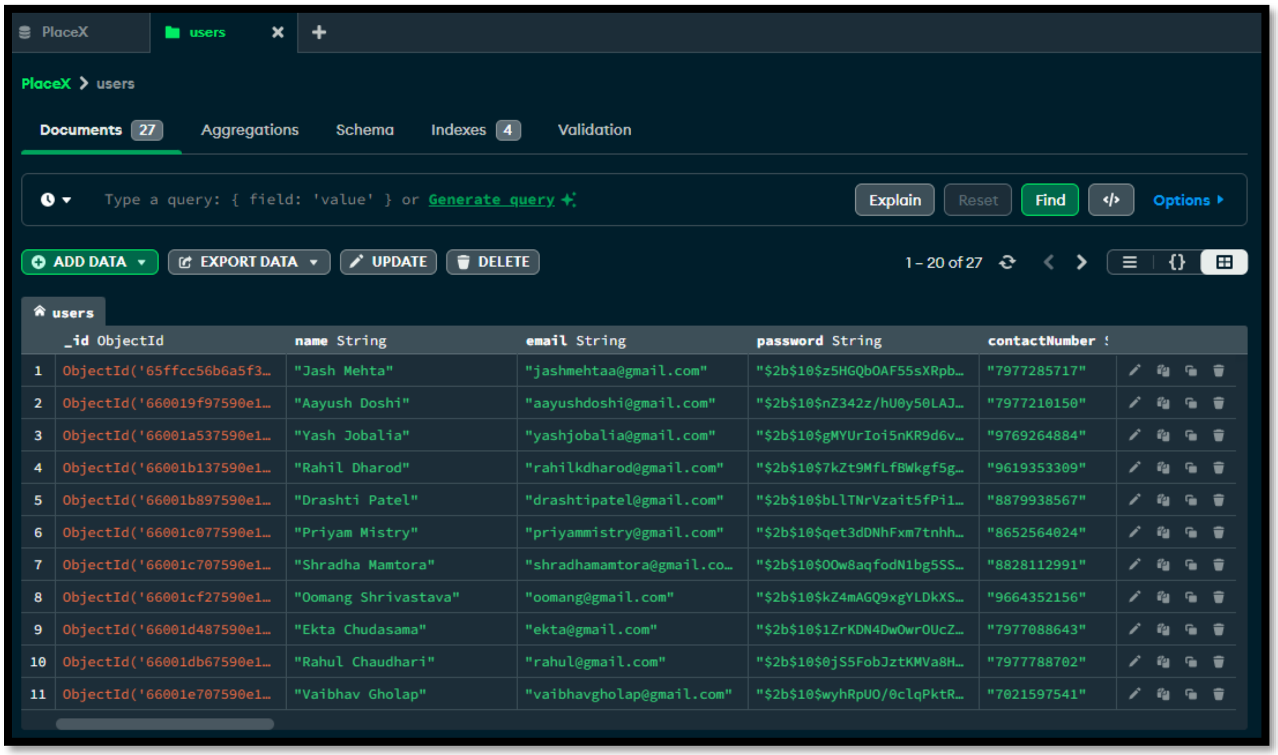
Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
24.58 kB	4	1.70 kB	1	36.86 kB

interviewexperiences

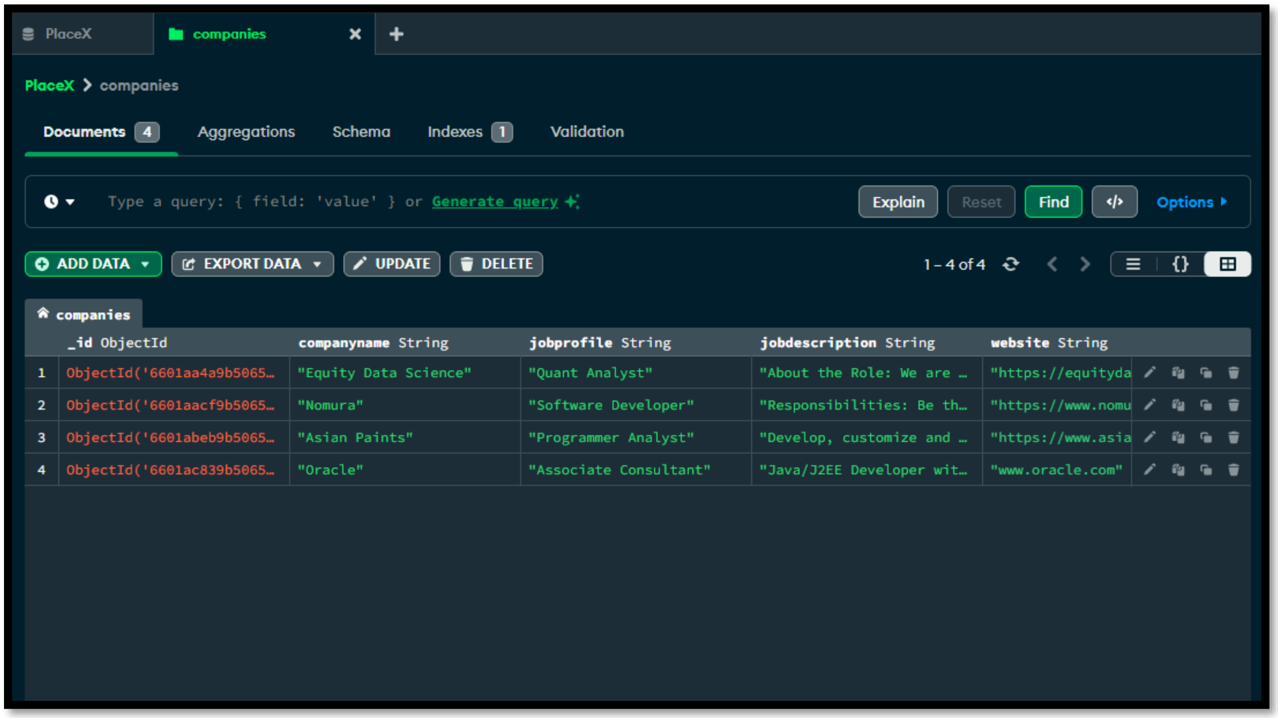
Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
28.67 kB	6	3.14 kB	1	36.86 kB

users

Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
24.58 kB	27	626.00 B	4	147.46 kB

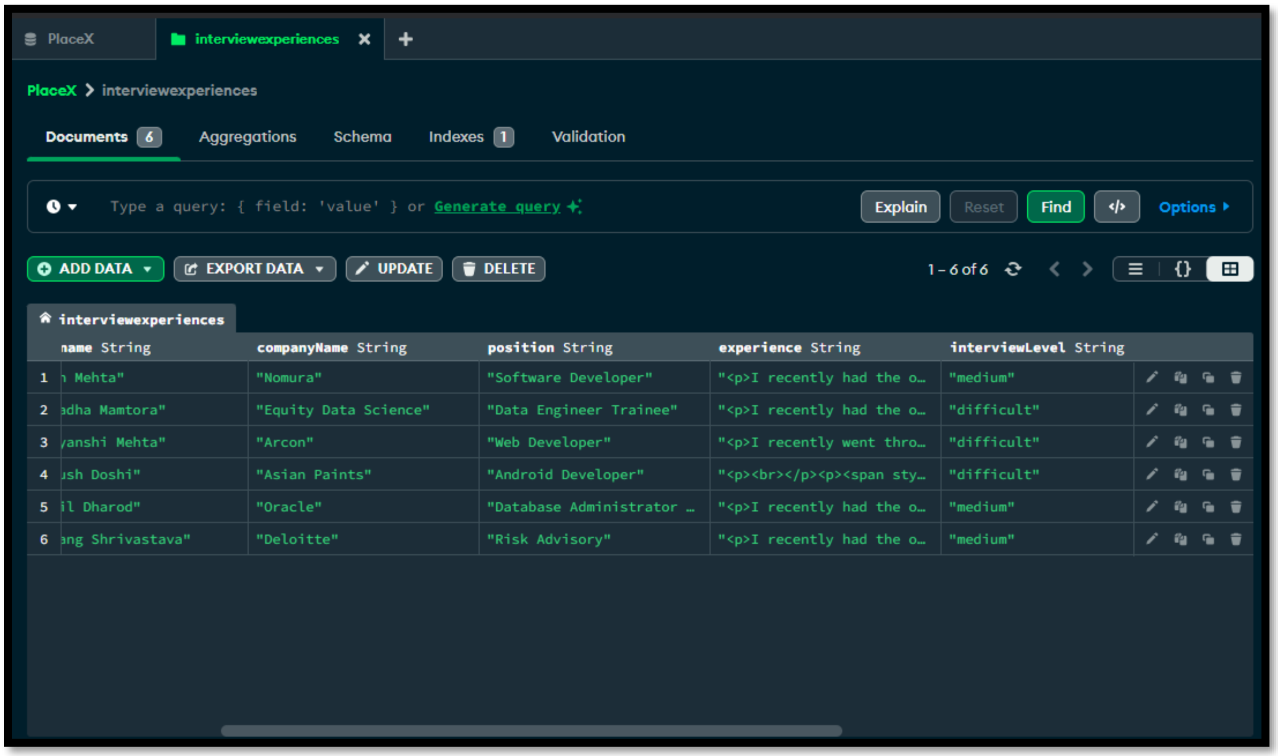


	_id ObjectId	name String	email String	password String	contactNumber !
1	ObjectId('65ffcc56b6a5f3…	"Jash Mehta"	"jashmehtaa@gmail.com"	"$2b$10$z5HGQbOAF55sXRpb…	"7977285717"
2	ObjectId('660019f97590e1…	"Aayush Doshi"	"aayushdoshi@gmail.com"	"$2b$10$nZ342z/hU0y50LAJ…	"7977210150"
3	ObjectId('66001a537590e1…	"Yash Jobalia"	"yashjobalia@gmail.com"	"$2b$10$gMYUrIoi5nKR9d6v…	"9769264884"
4	ObjectId('66001b137590e1…	"Rahil Dharod"	"rahilkdharod@gmail.com"	"$2b$10$7kZt9MfLfBWkgf5g…	"9619353309"
5	ObjectId('66001b897590e1…	"Drashti Patel"	"drashtipatel@gmail.com"	"$2b$10$bLlTNrVzait5fPi1…	"8879938567"
6	ObjectId('66001c077590e1…	"Priyam Mistry"	"priyammistry@gmail.com"	"$2b$10$qet3dDNhFxm7tnhh…	"8652564024"
7	ObjectId('66001c707590e1…	"Shradha Mamtora"	"shradhamamtora@gmail.co…	"$2b$10$OOw8aqfodN1bg5SS…	"8828112991"
8	ObjectId('66001cf27590e1…	"Oomang Shrivastava"	"oomang@gmail.com"	"$2b$10$kZ4mAGQ9xgYLDkXS…	"9664352156"
9	ObjectId('66001d487590e1…	"Ekta Chudasama"	"ekta@gmail.com"	"$2b$10$1ZrKDN4DwOwrOUcZ…	"7977088643"
10	ObjectId('66001db67590e1…	"Rahul Chaudhari"	"rahul@gmail.com"	"$2b$10$0jS5FobJztKMVa8H…	"7977788702"
11	ObjectId('66001e707590e1…	"Vaibhav Gholap"	"vaibhavgholap@gmail.com"	"$2b$10$wyhRpUO/0clqPktR…	"7021597541"



companies

	_id ObjectId	companyname String	jobprofile String	jobdescription String	website String
1	ObjectId('6601aa4a9b5065…	"Equity Data Science"	"Quant Analyst"	"About the Role: We are …	"https://equityda
2	ObjectId('6601aacf9b5065…	"Nomura"	"Software Developer"	"Responsibilities: Be th…	"https://www.nomu
3	ObjectId('6601abeb9b5065…	"Asian Paints"	"Programmer Analyst"	"Develop, customize and …	"https://www.asia
4	ObjectId('6601ac839b5065…	"Oracle"	"Associate Consultant"	"Java/J2EE Developer wit…	"www.oracle.com"



interviewexperiences

	name String	companyName String	position String	experience String	interviewLevel String
1	ı Mehta"	"Nomura"	"Software Developer"	"<p>I recently had the o…	"medium"
2	ıdha Mamtora"	"Equity Data Science"	"Data Engineer Trainee"	"<p>I recently had the o…	"difficult"
3	yanshi Mehta"	"Arcon"	"Web Developer"	"<p>I recently went thro…	"difficult"
4	ısh Doshi"	"Asian Paints"	"Android Developer"	"<p> </p><p><span sty…	"difficult"
5	il Dharod"	"Oracle"	"Database Administrator …	"<p>I recently had the o…	"medium"
6	ang Shrivastava"	"Deloitte"	"Risk Advisory"	"<p>I recently had the o…	"medium"

CONCLUSION

In conclusion, the PlaceX application offers a comprehensive solution for automating tasks typically performed by the Placement Department of colleges. Leveraging the power of MERN stack (MongoDB, Express.js, React, and Node.js), PlaceX provides a user-friendly platform with distinct modules for both users and administrators.

For users, PlaceX streamlines the process of registering for placements by capturing essential academic details and facilitating seamless access to company listings. Users can browse through available job opportunities, apply to companies of interest, and track their scheduled interviews conveniently. Additionally, features such as FAQ sections and interview experience sharing foster a collaborative environment, empowering users to prepare effectively for interviews and make informed decisions.

Administrators benefit from robust functionalities for managing user data, company listings, and placement processes. The ability to generate comprehensive reports based on various criteria enables administrators to gain insights into user demographics and recruitment trends. Furthermore, features such as updating company details, managing company applicants, and updating placement statuses provide administrators with the tools necessary for efficient decision-making and oversight.

The implementation of industry-standard libraries and technologies, including JWT for authentication, bcrypt for password hashing, and nodemailer for email communication, ensures the security and reliability of the application. Moreover, the modular architecture and clear separation of concerns facilitate scalability, maintainability, and extensibility, making PlaceX a robust solution capable of meeting the evolving needs of college placement departments.

Overall, PlaceX represents a significant advancement in streamlining placement processes, enhancing user experience, and empowering administrators with actionable insights. By leveraging cutting-edge technologies and best practices, PlaceX is poised to revolutionize the way colleges manage placements, ultimately contributing to the success of both students and recruiters.

FUTURE SCOPE

The future scope of PlaceX encompasses several avenues for expansion, enhancement, and refinement, aiming to further optimize the placement management process and deliver added value to users and administrators alike. Some potential areas for future development include:

1. Advanced Analytics and Insights:

Integrate advanced analytics tools to analyse user data, company interactions, and placement trends. This could include predictive analytics to forecast placement outcomes, identify emerging job trends, and provide personalized recommendations to users.

2. Machine Learning for Matching:

Implement machine learning algorithms to enhance the matching process between users and companies. By analysing user profiles, preferences, and historical data, PlaceX can intelligently recommend relevant job opportunities to users and optimize the recruitment process for companies.

3. Enhanced User Experience:

Continuously improve the user interface and experience based on user feedback and usability studies. This involves refining navigation flows, optimizing page load times, and incorporating interactive elements to enhance engagement and satisfaction.

4. Expanded Collaboration Features:

Introduce features for fostering collaboration and networking among users, such as discussion forums, mentorship programs, and virtual career fairs. These platforms can facilitate knowledge sharing, skill development, and professional networking within the PlaceX community.

5. Integration with Learning Management Systems:

Integrate PlaceX with learning management systems (LMS) used by educational institutions to provide seamless access to placement-related resources, training materials, and skill development courses. This integration can facilitate holistic career preparation for students and streamline administrative workflows for institutions.

6. Mobile Application Development:

Develop dedicated mobile applications for PlaceX to provide users with on-the-go access to placement-related information and functionalities. Mobile apps can offer features such as push notifications, offline access, and location-based services to enhance convenience and accessibility.