**LAB PRACTICAL REPORT**

**ON**

**ADVANCED WEB DESIGN FRAMEWORKS & DEVELOPMENT**

**22CS402**

**BACHELOR OF ENGINEERING**

**in**

**COMPUTER SCIENCE AND ENGINEERING**

****

**Submitted by: Supervised By:**

**Sneha Singla (2310990975) Name of Trainer: Mr Vikas**

**Tanisha (2310990982) Designation: Professor**

**Titiksha (2310990988)**

**Vanshika Raina (2310990995)**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**CHITKARA UNIVERSITY**

**CHANDIGARH-PATIALA NATIONAL HIGHWAY, RAJPURA, PUNJAB, INDIA**

**CONTENTS**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **Content** | **Page No** |
|  | **Declaration** | **3** |
|  | **Acknowledgement** | **4** |
| **1.** | **Introduction to Web Design , Frameworks & Developments** | **5-7** |
| **2.** | **Problem Statement** | **8** |
| **2.1** | **Abstract** | **9-10** |
| **2.2** | **Tools & Technology** | **11** |
| **2.3** | **Methodology** | **12-16** |
| **2.4** | **Implementation** | **17-21** |
| **3.** | **Results** | **22-27** |
| **3.1** | **Future Scope** | **28** |
| **4** | **Conclusion** | **29** |

**DECLARATION**

We hereby declare that the project work titled, “**CampusGig**” submitted as part of Bachelor’s degree in CSE, at Chitkara University, Punjab, is an authentic record of our own work carried out under the supervision of **Mr. Vikas.**

**Name:Mr Vikas**

**Signature(s):**

**ACKNOWLEDGEMENT**

It is our pleasure to be indebted to various people, who directly or indirectly contributed in the development of this work and who influenced my thinking, behavior, and acts during the course of my study.

We express our sincere gratitude to all for providing me an opportunity to undergo Integrated Project as part of the curriculum.

We are thankful to **Mr. Vikas** for his support, cooperation, and motivation provided to us during the training for constant inspiration, presence, and blessings.

Lastly, we would like to thank the almighty and our parents for their moral support and friends with whom we shared our day-to-day experience and received lots of suggestions that improve our quality of work.

* + - 1. **Introduction To Web Design Frameworks and Development**

Web design is the process of creating the appearance, layout, and overall presentation of websites and web applications. It involves designing web pages in a way that is visually appealing, easy to use, and functional across different devices like computers, tablets, and smartphones. Web design is not just about making a website “look nice”; it also ensures that users can easily navigate, find information, and perform tasks efficiently.

The main aspects of web design include:

1. **Layout and Structure:** The layout refers to how different elements like headers, menus, buttons, and content sections are arranged on a webpage. A well-structured layout helps users understand the website’s flow and find the information they need quickly. In CampusGig, the layout is organized so students can easily navigate through job listings, internship opportunities, and profile settings.
2. **Visual Design:** This involves the use of colors, fonts, images, icons, and other visual elements to make a website attractive and consistent. Visual design also helps in highlighting important information. For example, CampusGig uses clean and simple colors and icons to make the interface friendly and professional, helping users focus on key actions like applying for jobs or logging in.
3. **User Experience (UX):** UX design ensures that users can interact with the website or application smoothly without confusion or errors. Good UX considers how users think and behave while using the application. In CampusGig, UX principles are applied so students can quickly sign up, log in, browse jobs, and apply for internships without facing difficulties or delays.
4. **Responsiveness:** Modern web applications need to work on devices of all sizes. Responsive design ensures that a website adjusts its layout and elements according to the screen size. CampusGig is designed to be responsive, allowing students to access the portal comfortably from desktops, laptops, tablets, or mobile phones.
5. **Functionality and performance:** A well-designed website must not only look good but also function properly. Features like search, filters, login authentication, and application submission should work correctly. CampusGig integrates these functionalities using a Node.js backend and MongoDB database, ensuring that user actions like signing up, logging in, and applying for jobs are processed efficiently.

**Role of Web Design in CampusGig**

In the CampusGig project, web design plays a critical role in enhancing the user experience. Since the platform is targeted at students looking for internships and jobs, it is essential that the interface is simple, intuitive, and visually appealing. Effective web design in CampusGig allows:

* Students to easily find job and internship listings.
* Quick access to personal profiles and application history.
* Smooth interaction with the system without confusion or errors.
* Better engagement, ensuring students spend more time using the platform.

In conclusion, web design is the foundation of any successful web application. It combines aesthetics, functionality, and usability to provide a seamless experience. In CampusGig, good web design ensures that students can achieve their goals efficiently, making the platform both user-friendly and effective.

**What are Frameworks?**

A framework is a pre-built set of tools, libraries, and conventions that help developers build applications faster and more efficiently. Frameworks provide structure, reusable components, and best practices, allowing developers to focus on building the core functionality rather than writing everything from scratch.

Frameworks can be classified into two main types:

* **Frontend Frameworks**: Used for designing the user interface and handling interactions on the client-side (browser).
* **Backend Frameworks**: Used for handling server-side logic, database communication, authentication, and application functionality.

**Frameworks Used in CampusGig**

**Frontend – React.js with Vite**

* React allows building interactive UI with reusable components.
* Vite provides fast hot-reloading, optimized builds, and an improved developer experience.
* In CampusGig, React-Vite powers dashboards, job listings, signup/login forms, and search functionalities.

**Backend – Node.js and Express.js**

* Node.js runs JavaScript on the server and handles requests efficiently.
* Express.js simplifies routing, API creation, and middleware management.
* CampusGig uses Express.js to implement RESTful APIs for authentication, job management, and database communication.

**Database – MongoDB**

* MongoDB is a NoSQL database storing data in JSON-like documents.
* In CampusGig, MongoDB stores user data, job postings, and applications for quick access and easy scalability.

**Development Workflow in CampusGig**

1. Users interact with the React-Vite frontend.
2. API requests are sent to Express.js backend.
3. Backend processes requests, interacts with MongoDB, and sends responses.
4. Vite updates the frontend in real-time during development.

Using frameworks and proper development tools ensures CampusGig is **fast, scalable, and user-friendly**.

* + - 1. **PROBLEM STATEMENT**

Finding internships and job opportunities is one of the biggest challenges for college students. Many students face the following problems:

1. **Scattered Job Information:** Job listings are spread across multiple websites, social media platforms, and college notice boards. Students often spend a lot of time searching for relevant opportunities.
2. **Difficulty Finding Relevant Jobs:** Students may not know which jobs or internships match their skills, interests, or career goals. There is no single platform that provides filtered or role-specific listings.
3. **Lack of Centralized Application System**: Even if students find a job, they may have to apply separately on different websites or send emails manually. This process is time-consuming and can lead to missed opportunities.
4. **No Secure Tracking of Applications:** Students often cannot track their applications, see responses, or maintain a history of applied jobs in a centralized place.
5. **Authentication and Privacy Issues:** Many platforms require multiple logins, and students’ personal information may not be securely stored or managed.

**2.1 Abstract**

Finding suitable internships and job opportunities is one of the biggest challenges for college students. Often, students spend a lot of time searching multiple websites, social media pages, and college notice boards to find relevant opportunities. This process is time-consuming, inefficient, and sometimes leads to missing important job opportunities. There is also no centralized platform that allows students to apply for jobs, track applications, and communicate with employers in a secure and organized way.

**CampusGig** is a web-based internship and job portal developed to solve these challenges. It provides a **centralized platform** where students can browse internships and job listings, apply directly through the system, and manage their applications efficiently. The platform is built using **React with Vite** for the frontend, which provides a fast, interactive, and responsive user interface. **Node.js with Express** is used for the backend to handle server-side operations, API requests, authentication, and data management. **MongoDB** serves as the database, securely storing user data, job listings, applications, chat messages, and ratings.

The key features of CampusGig include:

1. **Secure Signup and Login:** Students and employers can create accounts, log in securely, and manage their profiles. Authentication is handled using **JWT tokens** and cookies, ensuring that data is protected and access is restricted to authorized users only.
2. **Job Search and Filtering:** Students can easily search for internships and job opportunities using filters such as role, location, or keywords. This helps students quickly find opportunities that match their skills and interests.
3. **Application Management:** Students can apply for jobs directly through CampusGig. The platform tracks all applications and maintains a history, allowing students to see the status of each application and avoid confusion.
4. **Real-Time Chat Application:** CampusGig includes a chat feature that allows students and employers to communicate instantly. This improves engagement, helps clarify queries, and makes the application process more interactive. Chat messages are securely stored in the database for future reference.
5. **Rating System:** Students can rate jobs or employers after completing applications. This feature provides feedback for employers and helps future students make informed decisions about which opportunities to pursue.
6. **Real-time email notifications:** CampusGig sends instant email notifications to keep users updated on all important activities happening on the platform.

### Real-Time Communication System: The platform provides instant messaging along with support for voice notes and video calls, enabling smooth and direct communication between employers and students during the hiring and project phases.

1. **Verified Reviews**: Ensure only users who completed a gig can leave reviews.
2. **Job Approval Workflow:** Moderators approve high-value gigs before they go live.(admin)

In summary, CampusGig is a **comprehensive, user-friendly, and secure platform** that addresses the challenges faced by students in finding and applying for internships and jobs. By combining modern frontend and backend technologies, real-time communication, and feedback mechanisms, CampusGig not only simplifies the job search process but also enhances the overall experience for both students and employers.

**2.2 Tools and Technology**

The CampusGig project uses a combination of frontend, backend, and database technologies, along with development tools, to create a full-stack web application.

**2.2.1 Technology stack**

|  |  |  |
| --- | --- | --- |
| Category | Technology/Tool | Purpose in CampusGig |
| Frontend | React.js with Vite | Build interactive user interface, reusable components, and fast development workflow |
| Backend | Node.js & Express.js | Handle server-side logic, API requests, authentication, and routing |
| Database | MongoDB | Store user data, job listings, and application records efficiently |
| Authentication | JWT & Cookies | Secure login, session management, and authorization of users |
| Real-time Chat | Socket.io | Enable real-time chat, instant messaging, live notifications, typing indicators, and real-time updates across the platform |
| Email Service | Nodemailer | Send real-time email notifications for job postings, application updates, ratings, and other important activities |
| Development Tools | VS Code, Git, Postman | Code development, version control, API testing |
| Environment Management | .env files | Store sensitive information like MONGO\_URI, JWT\_SECRET, COOKIE\_SECRET |
| Other Libraries | Mongoose | Simplify MongoDB operations, schema design, and validations |
| Browser Testing | Chrome, Edge | Test application responsiveness and UI functionality |

**2.3 Methodology**

The development of CampusGig followed a systematic methodology to ensure that the platform is efficient, secure, and user-friendly. The methodology outlines the steps from planning to implementation, covering both frontend and backend development as well as integration with the database.

**1. Requirement Analysis**

* Identify the challenges students face while searching for internships and jobs: scattered information, manual application processes, and lack of centralized tracking.
* Determine features needed for CampusGig:
  + User authentication (signup/login)
  + Job search and filtering
  + Application management
  + Real-time chat system
  + Rating system for jobs/employers
  + Real-time email notifications for updates such as job posted, application submitted, application accepted/rejected, and rating received
  + Secure Payment Gateway Integration
* Decide on technologies for development:
  + **Frontend:** React with Vite
  + **Backend:** Node.js with Express
  + **Database:** MongoDB

**2. System Design**

* **Frontend Design:**
  + Use React components for the dashboard, job listings, chat interface, rating forms, and authentication pages.
  + Design UI to be responsive and easy to navigate.
* **Backend Design:**
  + Define RESTful API endpoints for authentication (/api/auth), job management (/api/jobs), chat (/api/chat), and ratings (/api/rating).
  + Introduce notification handlers that trigger email alerts when specific events occur (e.g., job creation, application updates, ratings).
  + Implement middleware for authentication, error handling, and data validation to ensure secure and reliable processing of all user actions, including **payment verification and protected financial transactions**.
* **Database Design:**
  + Collections for users, jobs, applications, chat messages, and ratings.
  + Use Mongoose schemas to define data structure and enforce validations.

**3. Implementation Planning**

* Break the project into modules:
  + **Authentication Module:** Signup, login, password encryption, JWT token generation, and cookie management.
  + **Job Module:** Create, read, update, delete jobs, and allow students to apply for jobs.
  + **Chat Module:** Real-time messaging between students and employers using **Socket.io**.
  + **Rating Module:** Students can rate jobs or employers after applying; ratings stored in MongoDB.
  + **Email Notification Module**: Sends real-time emails using Nodemailer for job postings, application submissions, acceptance/rejection, and ratings.
  + **Payment Gateway Module:** Handles secure employer-to-freelancer payments using Razorpay.  
     This module ensures that all job-related transactions happen directly within the platform, creating a safe and efficient payment flow.
* Plan frontend components and API integration:
  + JobList, JobDetails, ChatBox, RatingForm, Dashboard, Profile.
  + Axios or fetch used for API requests to backend endpoints.

**4. Development Process**

* **Frontend Development:**
* Build React components using Vite for fast builds and hot reloading.
* Implement forms for login, signup, and job application.
* Display job listings dynamically using data fetched from backend APIs.
* **Backend Development:**
* Create Express routes for authentication, jobs, chat, and ratings.
* Connect backend to MongoDB using Mongoose for CRUD operations.
* Implement JWT authentication and cookie management for secure sessions.
* Integrate Nodemailer into the backend to send automated emails based on server events (job posted, application updates, rating submitted, bidding jobs etc).
* **I**ntegrate Razorpay to manage secure payment processing, including order creation, payment verification, and transaction storage.
* **Real-Time Features:**
* Use Socket.io to implement real-time chat between students and employers.
* Ensure messages are stored in MongoDB for history tracking.

**4. Rating System:**

* Allow students to rate jobs or employers after applying.
* Display average ratings on job listings for guidance to future applicants.

#### 5. Real-Time Email Notification System:

* Configure Nodemailer using Gmail service to send notifications.
* Create functions to trigger emails automatically when:  
   - A new job is posted
* whenever a job is accepted by a freelancer
* Rating is given based on the work done
* Bidding on the jobs or when a job given is marked as completed
* Ensure email templates are clean, responsive, and informative.

6. **Integrate Payment Gateway:**

* A secure payment system was added using Razorpay to allow employers (job posters) to pay freelancers once a job is completed.
* The backend creates payment orders, verifies payment signatures, and stores transaction details in MongoDB to ensure trust, transparency, and smooth payouts.
* Email confirmations are also triggered after successful payments to notify both parties.

**7. Admin & Analytics Features**

* Admin Dashboard: Monitor users, gigs, disputes, and revenue.
* Dispute Management: Resolve conflicts between client and provider.
* Analytics & Reports: Insights into earnings, popular categories, and active users.
* Content Moderation: Approve gigs or flag inappropriate content.

**8. Marketing & Growth Features**

* Referral System: Users earn credits for inviting friends.
* Coupons & Discounts: Promote services with limited-time offers.
* Email/SMS Marketing: Notify users about new gigs, deals, or updates.
* Social Sharing: Share gigs to social media.

**9. Testing and Debugging**

* Test backend APIs using **Postman** to verify routes and data handling.
* Test frontend UI across different devices to ensure responsiveness.
* Check authentication flow and secure cookie handling.
* Test chat and rating modules to ensure real-time updates and correct data storage.
* Test email notifications by simulating events and verifying that emails are received correctly, without delays or spam issues.
* Test payment flows by creating and verifying Razorpay orders to ensure successful transactions and secure signature validation.

**10. Deployment and Maintenance**

* Prepare .env file to manage environment variables like MONGO\_URI, PORT, JWT\_SECRET.
* Deploy the backend on servers like Heroku or Render.
* Deploy frontend on Vercel for a responsive, accessible platform.
* Monitor Nodemailer email logs and ensure reliable delivery of notifications.
* Continuous monitoring and debugging to ensure smooth operation.
* Monitor Razorpay dashboard and server logs to ensure payment transactions are processed smoothly and securely.

**2.4 Implementation**

The implementation of CampusGig focuses on turning the design and methodology into a working, functional web application. The project consists of a frontend, backend, database, and real-time features, all integrated to provide a seamless user experience.

**1. Backend Implementation**

**a. Authentication Module**

* **Signup & Login:** Students and employers can register and log in. Passwords are encrypted using bcrypt, and JWT tokens are generated for secure sessions.
* **Session Management:** JWT tokens are stored in cookies and validated for protected routes (e.g., applying for jobs, accessing dashboard).
* **Routes:**
  + POST /api/auth/signup → Create new user
  + POST /api/auth/login → Authenticate user and issue token
  + GET /api/auth/me → Fetch logged-in user details

**b. Job Module**

* **CRUD Operations:** Employers or admins can create, update, and delete job listings.
* **Job Browsing:** Students can fetch jobs, filter by role or search keywords, and view details.
* **Application:** Students can apply for jobs directly; applications are stored in the database with reference to user and job IDs.
* **Routes:**
  + GET /api/jobs → Fetch job listings
  + POST /api/jobs → Add a new job
  + POST /api/jobs/apply → Apply for a job

**c. Chat Module**

* **Real-Time Communication:** Uses **Socket.io** to allow students and employers to chat instantly.
* **Message Storage:** Messages are stored in MongoDB for tracking conversation history.
* **Features:**
  + One-to-one chat between student and employer
  + Chat history retrieval

**d. Rating Module**

* **Submit Ratings:** After applying for a job, students can rate the job or employer.
* **Store Ratings:** Ratings stored in MongoDB and associated with job and user IDs.
* **Display Ratings:** Average rating is displayed on job listings to guide future students.
* **Routes:**
  + POST /api/rating → Submit rating
  + GET /api/jobs/:id → Fetch job with ratings

#### e. Payment Gateway Module

#### To enhance the reliability and transparency of job transactions, CampusGig now includes a secure payment gateway integration. This allows employers (posters) to pay freelancers directly through the platform once the job is completed.

* **Secure Payments:** Integrated using **Razorpay** ensuring encrypted and authenticated payment processing.
* **Order Creation:** When an employer decides to pay a student, an order is created on the backend with amount, receipt ID, and currency.
* **Payment Verification:** After the payment is made, the backend verifies the payment signature to prevent fraud.
* **Transaction Tracking:** All payments and receipts are stored in MongoDB for auditing and future reference.
* **Email Notifications:** Once payment is completed, both employer and freelancer receive real-time email confirmation.

**2. Frontend Implementation**

**a. Dashboard & Navigation**

* React components provide a dashboard where students can see applied jobs, messages, and ratings.
* Navigation allows easy access to job listings, chat, and profile management.

**b. Job Listings & Application**

* **JobList Component:** Displays all jobs fetched from the backend.
* **Filter/Search:** Students can search jobs by role or keyword.
* **Apply Button:** Sends API request to backend to save application.

**c. Chat Application**

* **ChatBox Component:** Real-time chat using Socket.io-client.
* Messages are displayed in a scrolling interface with timestamps.
* New messages are sent to the backend and broadcast to recipients.

**d. Rating System**

* **RatingForm Component:** Allows students to submit ratings after applying.
* Backend calculates average ratings and displays them on job listings.
* Ensures only applicants can rate a job.

**e. Authentication & Security**

* Login and signup forms are connected to backend APIs.
* Protected routes prevent unauthenticated users from accessing dashboards or applying for jobs.

**f. Payment Interface**

* Payment UI integrated using Razorpay’s checkout popup.
* Employers can initiate payments from the job details page.
* Frontend triggers order creation, opens the payment window, and handles payment success/failure callbacks.

**3. Database Implementation**

* **MongoDB Collections:**
  + **Users:** {name, email, password, role}
  + **Jobs:** {title, description, company, role, location, ratings}
  + **Applications:** {studentId, jobId, status}
  + **Chats:** {senderId, receiverId, message, timestamp}
  + **Ratings:** {studentId, jobId, rating, comment}
* **Mongoose:** Used for schema validation and simplified database operations.

**4. Development Workflow**

* Student signs up → JWT token generated → stored in cookies.
* Student logs in → token validated → dashboard displayed.
* Student browses jobs → applies for a job → application stored in database.
* Student chats with employer → messages exchanged in real-time → chat history stored.
* Student rates the job or employer → rating stored and displayed on job listings.

**5. Testing and Debugging**

* **Backend APIs:** Tested with Postman to ensure authentication, job operations, chat, and rating modules work correctly.
* **Frontend:** Tested across different devices to verify responsiveness and user interface.
* **Integration:** Confirmed proper communication between frontend and backend APIs.
* **Real-Time Features:** Chat , rating systems and payment were tested for correctness and data persistence.

**6. Outcome**

CampusGig is now a **fully functional web application** with the following features:

* Secure authentication for students and employers.
* Centralized job listings with search and filtering.
* Job application management.
* Real-time chat for better communication.
* Real-time email notifications for important activities such as job postings, acceptance updates,bid selection and ratings—ensuring users stay informed instantly.
* Rating system for post-application feedback.

The platform provides a **complete solution for students to find, apply, communicate, and evaluate job opportunities**, making it a valuable career portal.

**3.Results**

A screenshot of a website

AI-generated content may be incorrect.

**3.1 Home Page**

**A screenshot of a login form

AI-generated content may be incorrect.**

**3.2 Login Page**

A screenshot of a website

AI-generated content may be incorrect.

**3.3 Available Jobs Page**

**A screenshot of a web page

AI-generated content may be incorrect.**

**3.4 Saved Jobs Page**

A screenshot of a computer

AI-generated content may be incorrect.

**3.5 Posting Jobs**

**A screenshot of a computer

AI-generated content may be incorrect.**

**3.6 My Recent Bids & Earnings**

A screenshot of a web page

AI-generated content may be incorrect.

**3.7 Profile**

**A screenshot of a chat

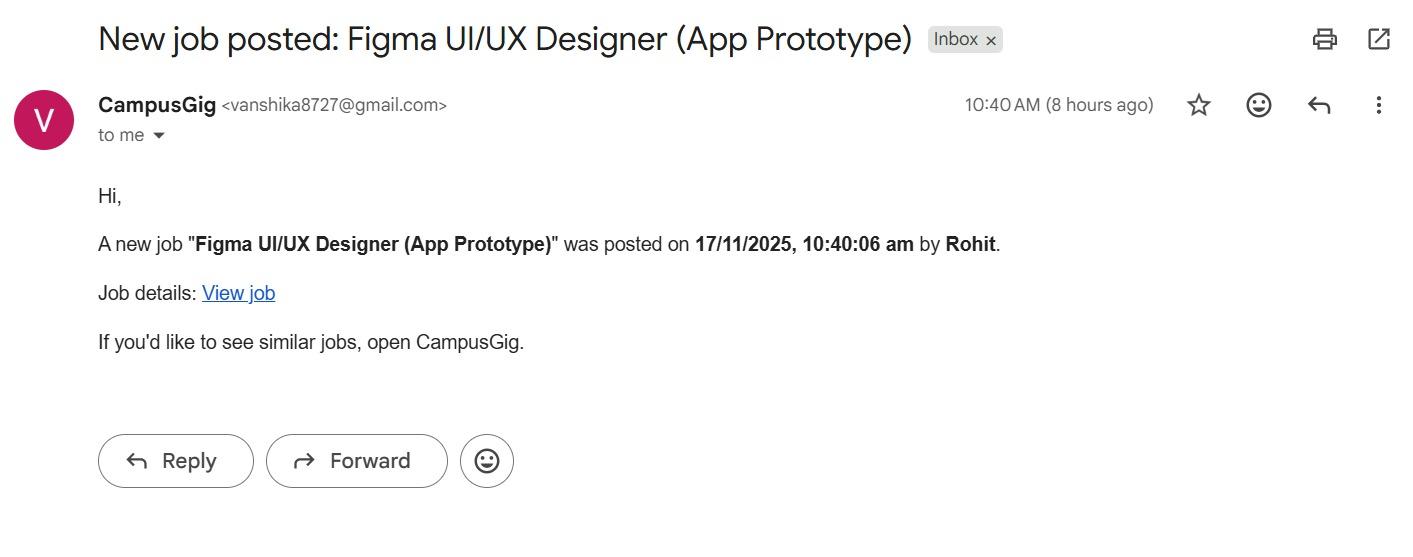
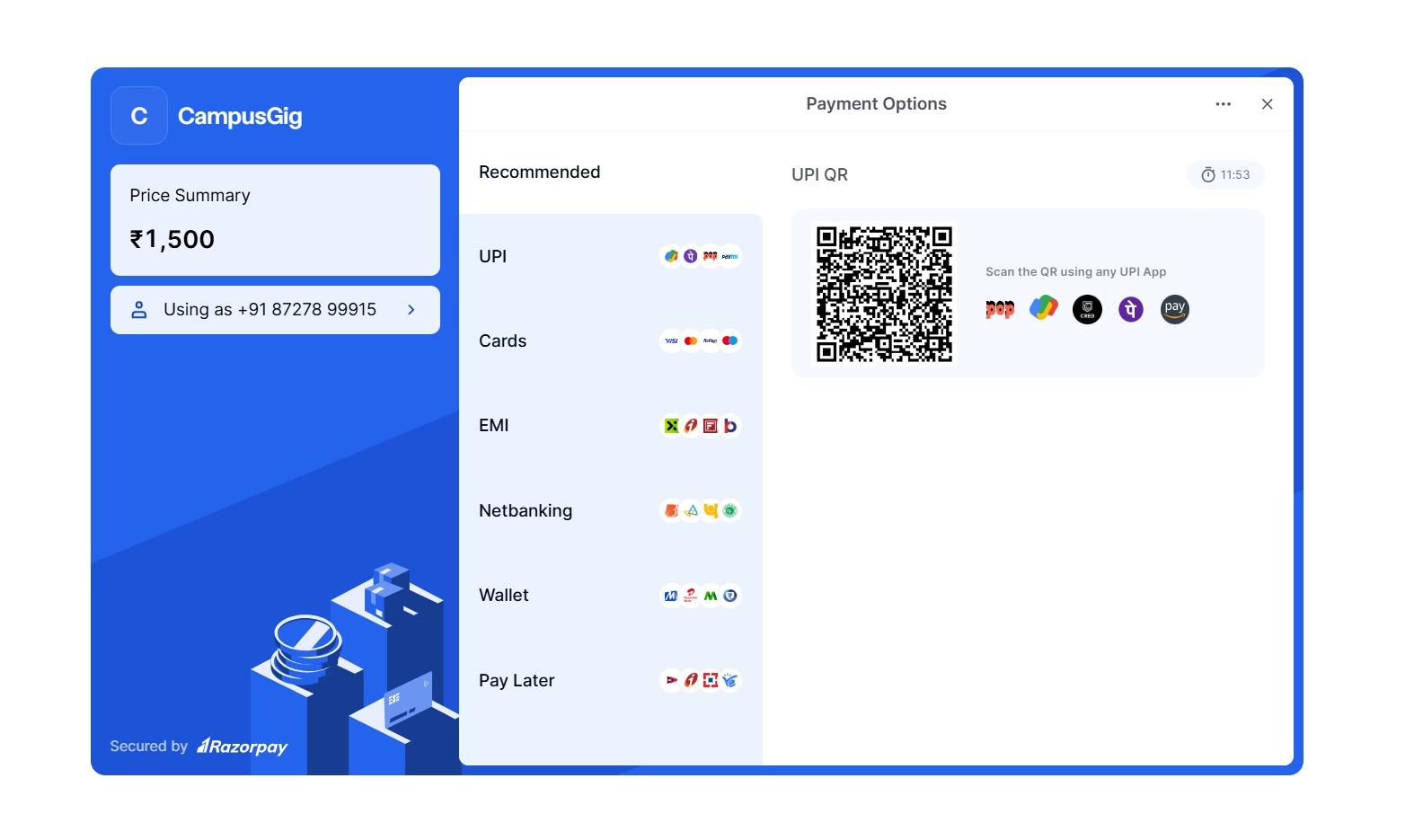
AI-generated content may be incorrect.**

**3.8 Customer Support**

Screens screenshot of a chat

AI-generated content may be incorrect.

**3.9 Chat and Video Call Feature**

**3.10 Notification Feature**

**3.11 Razorpay Dashboard for payment**

**3.1 Future Scope :**

CampusGig has been developed as a comprehensive platform to simplify internship and job discovery for students. However, the platform still has vast potential for expansion. Several improvements and advanced features can be added to enhance efficiency, scalability, security, and user engagement in the future:

### **Resume Builder & Skill Assessment:** The platform can provide:

* Automated resume generation tools.
* AI-based resume scoring.
* Online assessment tests to verify skills.  
  This will help employers find verified and well-prepared candidates.

### **Internship Tracking & Progress Monitoring:** A dedicated module for tracking internship progress:

* Weekly report submissions
* Task management
* Employer performance evaluations

This ensures transparency and accountability.

### **Blockchain for Secure Certifications:** Blockchain can be used to issue:

* Verified offer letters
* Experience certificates
* Skill badges  
  This prevents fraud and increases trust.

### **Advanced Analytics Dashboard**:**** Admins, employers, and students can benefit from:

* Trend analysis
* Job popularity tracking
* Student performance analytics
* Employer activity insights

**4.Conclusion**

CampusGig is designed as an innovative and efficient web application that centralizes internship and job opportunities for college students. By integrating modern technologies such as React-Vite for a fast and responsive frontend, Node.js and Express.js for scalable backend operations, MongoDB for flexible and secure data storage, Socket.io for real-time communication, Nodemailer for instant email notifications, and Razorpay for secure payments, the platform successfully addresses the major challenges faced by students in today’s competitive environment.

The platform provides a smooth, user-friendly experience where students can:

* Explore job listings,
* Apply instantly,
* Communicate with employers in real-time,
* Track their applications,
* Receive email alerts,
* Manage payments securely, and
* Give ratings for transparency and quality.

CampusGig not only simplifies the hiring process but also enhances communication, trust, and efficiency between students and employers. It demonstrates the power of full-stack development and modern web frameworks in solving real-world problems.

The project effectively meets its objectives and lays a strong foundation for future enhancements. With continuous improvements and advanced features, CampusGig has the potential to grow into a large-scale professional career platform serving students, freelancers, and employers across multiple domains.