Project Document: Internship Warm-up Assignment - Company Registration & Verification Module

Project Overview

The **Company Registration & Verification Module** is a web application designed to enable companies to register, log in, and manage their profiles through a secure, scalable, and responsive interface. The module includes a multi-step registration form displayed in the dashboard post-registration, with profile details editable in the settings section. The backend handles all logic via APIs, using Firebase for authentication (email/password and SMS OTP) and JWT for session management. The application adheres to the provided Figma design, uses Cloudinary for image storage, and maintains a normalized PostgreSQL database. All code will follow standard practices, include test cases, and be documented clearly.

Objective: Develop a high-quality, scalable module with comprehensive testing and documentation, ensuring confidentiality per the NDA with Bluestock Fintech.

Deadline: August 13, 2025

Demo: Demonstrate the working model (frontend, backend, API testing) via Microsoft Teams/Google Meet in English, Hindi, or Marathi, showcasing all features and any pending tasks.

Important : *Based on the quality of your work on this assignment, you will be assigned to the respective project teams. Please ensure that you fully understand all requirements before proceeding, and avoid unnecessary messages or emails.*

Imp Link -

UI UX Design - https://www.figma.com/design/KL8dxFw2iWjnRpiPEJ8ljO/3-Warm-UP-Assignment?node-id=0-1&t=Iv99Ams1PYQFSMSN-1

PostgreSQL Database File - https://bluestock.in/backoffice-tech/company_db

Sample Frontend Screenshot (For Reference) - https://bluestock.in/backoffice-tech/company-module-sample/index.html

Tech Stack

Frontend

- **ReactJS 19**: For building the UI.
- **Vite**: For fast build and development server.
- **Redux Toolkit**: For state management (no Context API or plain Redux).
- **Dependencies** (from provided package.json):
    - @emotion/react, @emotion/styled, @mui/material: For styling and UI components.
    - @tanstack/react-query: For API data fetching and caching.
    - axios: For API requests.
    - firebase: For email/password and SMS OTP authentication.
    - react-hook-form: For form handling and validation.
    - react-phone-input-2: For mobile number input with country codes.
    - react-toastify: For user notifications.
    - react-responsive: For mobile responsiveness.
    - react-router-dom: For routing.
    - cloudinary: For storing company logo and banner images.
    - Other packages as listed (e.g., react-datepicker, sweetalert2, recharts).

Backend

- **Node.js 20.x (LTS)**: Server runtime.
- **Express**: Web framework for API development.
- **PostgreSQL 15**: Database for user and company data.
- **Dependencies** (from provided package.json):
    - jsonwebtoken: For JWT authentication (90-day validity).
    - bcrypt: For password hashing.
    - pg: For PostgreSQL integration.
    - express-validator: For input validation.
    - sanitize-html: For input sanitization.
    - libphonenumber-js: For phone number validation.
    - Other packages as listed (e.g., helmet, cors, compression).

External Services

- **Firebase**: For email/password authentication and SMS OTP verification.
- **Cloudinary**: For storing and retrieving company logo and banner images (URLs stored in database).
- **Figma**: UI design reference .
- **Database**: Import provided SQL file (bluestock.in/backoffice-tech/company_db.sql) into PostgreSQL on localhost.
- **Screenshots**: Reference frontend UI from - https://bluestock.in/backoffice-tech/company-module-sample/index.html

---

Database Schema

The database is normalized with two tables: users and company_profile. The owner_id in company_profile is a foreign key referencing users(id) to link users to their company profiles.

Table: users

Column	Type	Constraints	Description
id	integer	Primary Key, Auto-increment	Unique user identifier
email	character varying(255)	Not Null, Unique	User's email address
password	text	Not Null	Hashed password (using bcrypt)
full_name	character varying(255)	Not Null	User's full name
signup_type	character varying(1)	Not Null, Default 'e'	Signup type ('e' for email)
gender	character(1)	Not Null, Values: 'm', 'f', 'o'	Gender (male, female, other)
mobile_no	character varying(20)	Not Null, Unique	Mobile number with country code
is_mobile_verified	boolean	Default false	Mobile verification status
is_email_verified	boolean	Default false	Email verification status
created_at	timestamp	Not Null, Default CURRENT_TIMESTAMP	Record creation timestamp
updated_at	timestamp	Not Null, Default CURRENT_TIMESTAMP	Record update timestamp

Table: company_profile

Column	Type	Constraints	Description

id	integer	Primary Key, Auto-increment	Unique company identifier
owner_id	integer	Foreign Key (users.id), Not Null	References user who owns the company
company_name	text	Not Null	Company name
address	text	Not Null	Company address
city	character varying(50)	Not Null	City
state	character varying(50)	Not Null	State
country	character varying(50)	Not Null	Country
postal_code	character varying(20)	Not Null	Postal code
website	text	Optional	Company website URL
logo_url	text	Optional	Cloudinary URL for company logo
banner_url	text	Optional	Cloudinary URL for company banner
industry	text	Not Null	Industry type
founded_date	date	Optional	Company founding date
description	text	Optional	Company description
social_links	jsonb	Optional	JSON object for social media links
created_at	timestamp	Not Null, Default CURRENT_TIMESTAMP	Record creation timestamp
updated_at	timestamp	Not Null, Default CURRENT_TIMESTAMP	Record update timestamp

Normalization:

- The users table stores user authentication and personal details.
- The company_profile table stores company-specific details, linked to users via owner_id.
- Triggers will update updated_at timestamps on record modifications.
- The database schema will be imported from https://bluestock.in/backoffice-tech/company_db and connected to the project on localhost.

Functional Requirements

1. **User Registration**:
    - Users register with email, password, full name, gender, mobile number, and signup_type ('e' for email).
    - Firebase handles email/password authentication and sends SMS OTP for mobile verification.
    - On successful OTP verification, set is_mobile_verified to true in the users table.
    - Send an email verification link via Firebase; set is_email_verified to true upon link click.
2. **User Login**:
    - Users log in with email and password via Firebase.
    - On successful login, generate a JWT token (90-day validity) and return it to the frontend.
    - Store the token in Redux Toolkit and include it in API request headers for authentication.
3. **Company Registration**:
    - After login, display a multi-step form in the dashboard (based on Figma design).
    - Collect company details (name, address, city, state, country, postal code, website, industry, etc.).
    - Allow upload of logo and banner images via Cloudinary; store URLs in logo_url and banner_url.
    - Save details in the company_profile table with owner_id linked to the user's id.
4. **Profile Management**:
    - Display user and company details in the dashboard's profile/settings section.
    - Allow editing of company details (update company_profile table) and user details (update users table).
    - Support re-upload of logo/banner images via Cloudinary.
5. **UI/UX**:
    - Follow Figma design and screenshots (bluestock.in/backoffice-hr/company-module-sample/).
    - Ensure mobile responsiveness using react-responsive and @mui/material.
6. **Security**:
    - Hash passwords with bcrypt.
    - Validate and sanitize inputs using express-validator and sanitize-html.

- Use helmet for secure HTTP headers and cors for cross-origin requests.
- Implement JWT authentication for protected APIs.

API Endpoints

All APIs will return JSON responses with proper status codes and error messages. Protected endpoints require JWT in the Authorization header.

Endpoint	Method	Description	Authentication
/api/auth/register	POST	Register user (email, password, etc.)	None
/api/auth/login	POST	User login (returns JWT token)	None
/api/auth/verify-email	GET	Verify email via Firebase link	None
/api/auth/verify-mobile	POST	Verify mobile via Firebase OTP	None
/api/company/register	POST	Submit company profile details	JWT
/api/company/profile	GET	Fetch company profile details	JWT
/api/company/profile	PUT	Update company profile details	JWT
/api/company/upload-logo	POST	Upload company logo to Cloudinary	JWT
/api/company/upload-banner	POST	Upload company banner to Cloudinary	JWT

Sample Request (Register):

json

```
POST /api/auth/register

{

 "email": "company@example.com",

 "password": "Password123!",

 "full_name": "John Doe",
```

```
 "gender": "m",

 "mobile_no": "+919876543210",

 "signup_type": "e"

}
```

Sample Response:

json

```
{

 "success": true,

 "message": "User registered successfully. Please verify mobile OTP.",

 "data": { "user_id": 1 }

}
```

Error Handling:

- Use http-errors for standardized errors (e.g., 400 for invalid input, 401 for unauthorized).
- Display errors on the frontend using react-toastify.

Frontend Implementation

Key Features

- **Multi-step Registration Form**: Built with react-hook-form for validation and navigation between steps.
- **Dashboard**: Displays user and company details with edit functionality (uses @mui/material for UI components).
- **State Management**: Uses Redux Toolkit for managing user, authentication, and company data.
- **API Integration**: Uses @tanstack/react-query for API calls and caching.
- **Responsive Design**: Implements Figma designs with mobile responsiveness via react-responsive.
- **Notifications**: Uses react-toastify for success/error alerts.

Directory Structure

text

```
frontend/

├── public/

|   ├── assets/                # Static assets (e.g., favicon)

├── src/

|   ├── assets/                # Images, fonts

|   ├── components/            # Reusable components (FormStep, ProfileCard,
ImageUploader)

|   ├── pages/                 # Page components (Login, Register, Dashboard,
Settings)

|   ├── store/                 # Redux Toolkit slices (authSlice,
companySlice)

|   ├── api/                   # Axios service functions for API calls

|   ├── styles/                # Global styles and MUI theme

|   ├── App.jsx                # Main app with routing

|   ├── main.jsx               # Entry point

├── vite.config.js             # Vite configuration

├── package.json
```

Backend Implementation

Key Features

- **Authentication**: Integrates Firebase for email/password and SMS OTP, with JWT (90-day validity) for sessions.
- **Database**: Connects to PostgreSQL 15 using pg and the provided SQL file.
- **Image Storage**: Uses Cloudinary for logo and banner uploads, storing URLs in company_profile.
- **Security**: Implements input validation (express-validator), sanitization (sanitize-html), and secure headers (helmet).

Directory Structure

text

```
backend/

├── src/

|   ├── controllers/           # Request handlers (authController, companyController)

|   ├── middleware/            # JWT authentication, input validation

|   ├── routes/                # API route definitions

|   ├── services/              # Firebase, Cloudinary, and database services

|   ├── models/                # PostgreSQL queries and schema

|   ├── utils/                 # JWT generation, error handling

|   ├── config/                # Environment variables (Firebase, Cloudinary, DB)

|   ├── tests/                 # Unit and integration tests

|   ├── server.js              # Main server file

├── package.json
```

Testing

- **Frontend**:
    - Use jest and react-testing-library to test components, Redux slices, and API integrations.
    - Test cases: Form validation, navigation, profile display, image upload.
- **Backend**:
    - Use jest and supertest to test API endpoints.
    - Test cases: User registration/login, company profile CRUD, JWT validation.
- **API Testing**:
    - Demonstrate APIs using Postman or VS Code Thunder Client during the demo.
    - Test all endpoints with valid/invalid inputs and edge cases.

Documentation

- **API Documentation**: Created in Google Docs, detailing endpoints, request/response formats, headers, and error codes.
- **Technical Documentation**: Includes project setup, database schema, code structure, and external service configurations.
- **Submission**: Share Google Docs link during the demo on August 13, 2025.

Implementation Plan

1. **Setup (August 7-8, 2025)**:
    - Initialize Vite + ReactJS 19 and Node.js + Express projects.
    - Import company_db.sql into PostgreSQL on localhost.
    - Set up Firebase and Cloudinary accounts; configure environment variables.
    - Install dependencies from provided package.json files.
2. **Backend Development (August 9-10, 2025)**:
    - Implement authentication APIs (register, login, verify email/mobile) with Firebase and JWT.
    - Develop company profile APIs (create, read, update) with Cloudinary integration.
    - Add input validation and error handling.
    - Write unit and integration tests for APIs.
3. **Frontend Development (August 11-12, 2025)**:
    - Build multi-step registration form with react-hook-form and @mui/material.
    - Create dashboard and settings pages to display/edit user and company details.
    - Integrate Redux Toolkit for state management and @tanstack/react-query for API calls.
    - Ensure mobile responsiveness per Figma design.
    - Write unit tests for components and Redux slices.
4. **Testing & Documentation (August 12, 2025)**:
    - Test all features (frontend, backend, APIs) with valid/invalid inputs.

    - Finalize API and technical documentation in Google Docs.
    - Prepare Postman/Thunder Client collections for API testing.
5. **Demo Preparation (August 13, 2025)**:
    - Test application on localhost (frontend, backend, database).
    - Prepare demo script to showcase all features (login, registration, profile management, image upload).
    - Highlight any pending tasks and explain workarounds.

Confidentiality

- Per the NDA with Bluestock Fintech, the module will not be shared publicly (e.g., LinkedIn, GitHub, or other platforms).
- All code and documentation will be kept secure and shared only during the demo.

Demo Requirements

- **Date**: August 13, 2025
- **Platform**: Microsoft Teams/Google Meet
- **Language**: English, Hindi, or Marathi
- **Scope**:
    - Demonstrate working frontend on localhost (login, registration form, dashboard, profile editing).
    - Show backend APIs via Postman/Thunder Client (all endpoints with sample requests/responses).
    - Explain database schema and Cloudinary/Firebase integration.
    - Highlight test cases and documentation.
    - Discuss any pending tasks and proposed solutions.

Contact

- For critical doubts or clarifications, contact via WhatsApp at 9209550273.
- Avoid unnecessary messages/emails to maintain professionalism.

Notes

- The project will adhere strictly to the provided tech stack, Figma design, and database schema.

- Code quality will follow standard practices (e.g., modular structure, error handling, scalability).
- The assignment's performance will determine allocation to further project teams, so ownership and quality are prioritized.

Next Steps:

- Download and import the database SQL file (bluestock.in/backoffice-tech/company_db.sql) into PostgreSQL.
- Set up Firebase and Cloudinary accounts for authentication and image storage.
- Begin development with the provided directory structures and follow the implementation plan.
- For any critical doubts, contact 9209550273 via WhatsApp.