**JOB PORTAL WEBSITE**

Index

1. Job Portal Website

2. The form of Personal Details

3. The form of Job Details

4. Validations

5. Front-end Tools Setup Process:

5.1. Figma UI Designing Tool

5.2. Visual Studio Code

5.3. Node.js and React

6. Back-end Tools Setup Process:

6.1. Python- Django

6.2. Postman

6.3. PostgreSQL

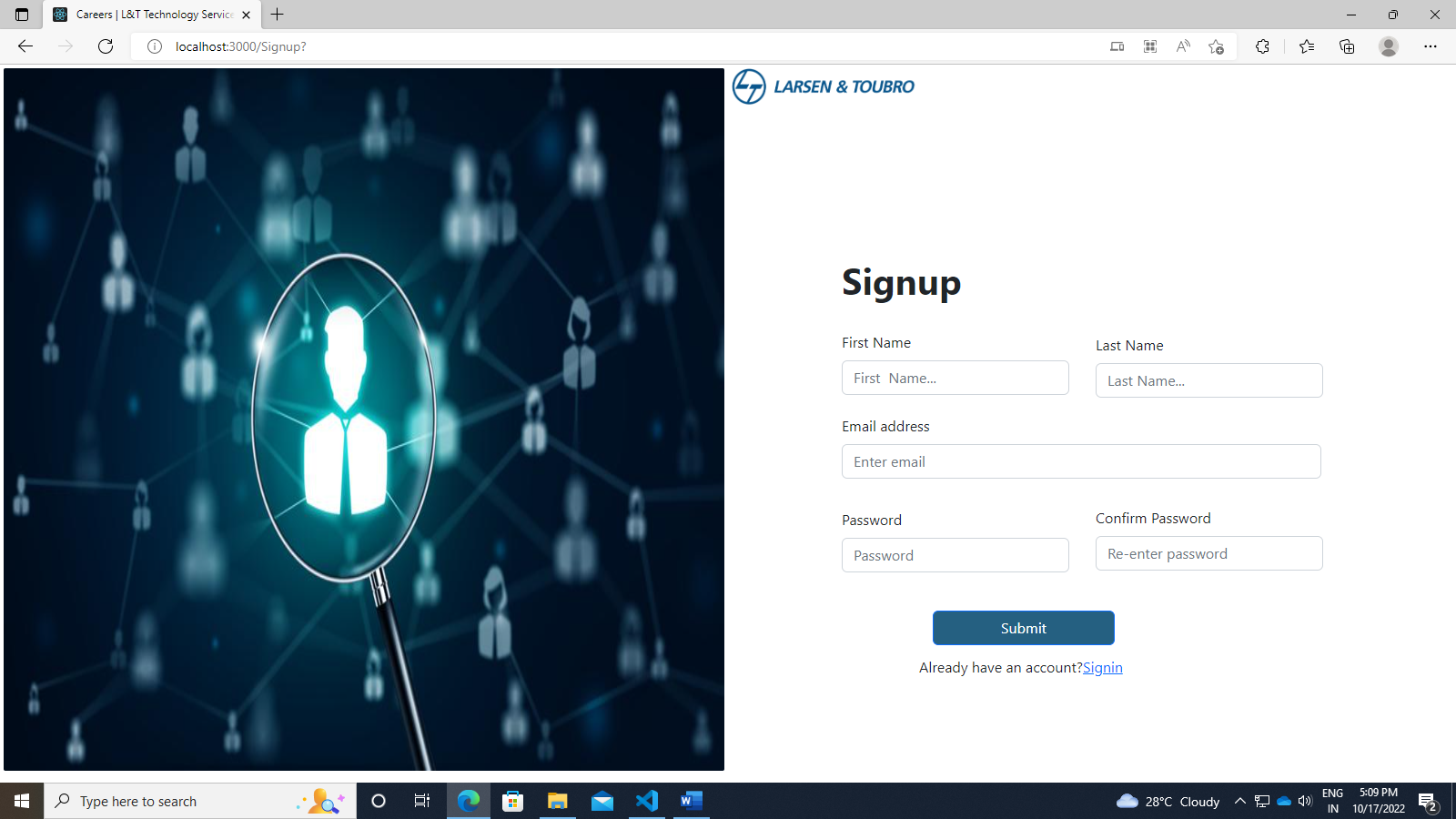
**Tools User Manuals**

In this let’s know about the Tools user manuals of each tool and how to use Job Portal Website as well.

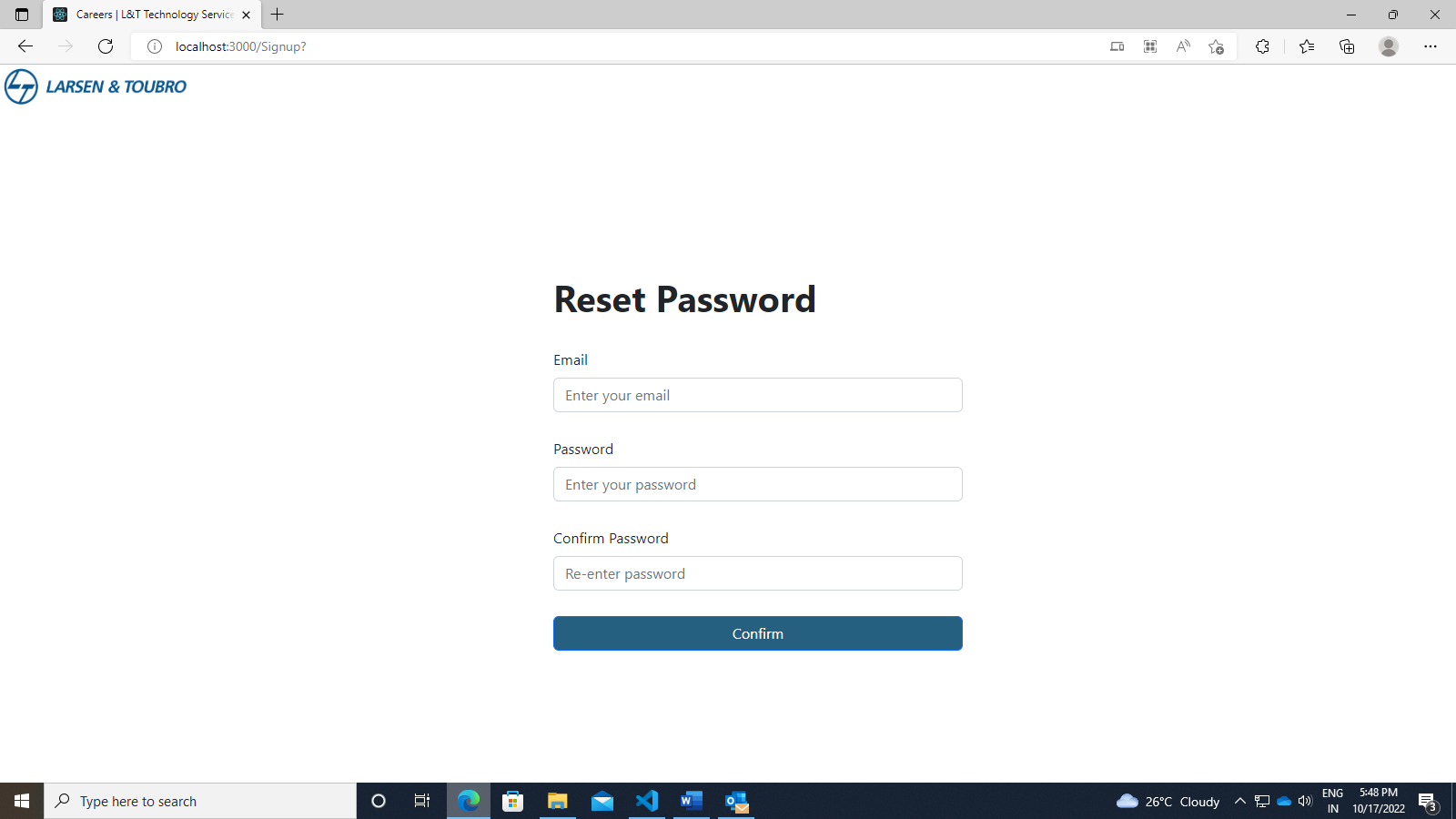
**1.Job Portal Website**:

The Job Portal Website needs an account to be created for user and using the same to login. To create an account, Signup is used and the account is secured through the password set by the user. If the user forgets the password, the forget password procedure can be used to retrieve the account. Once after the account is created, the user can login using the respective credentials as shown below.

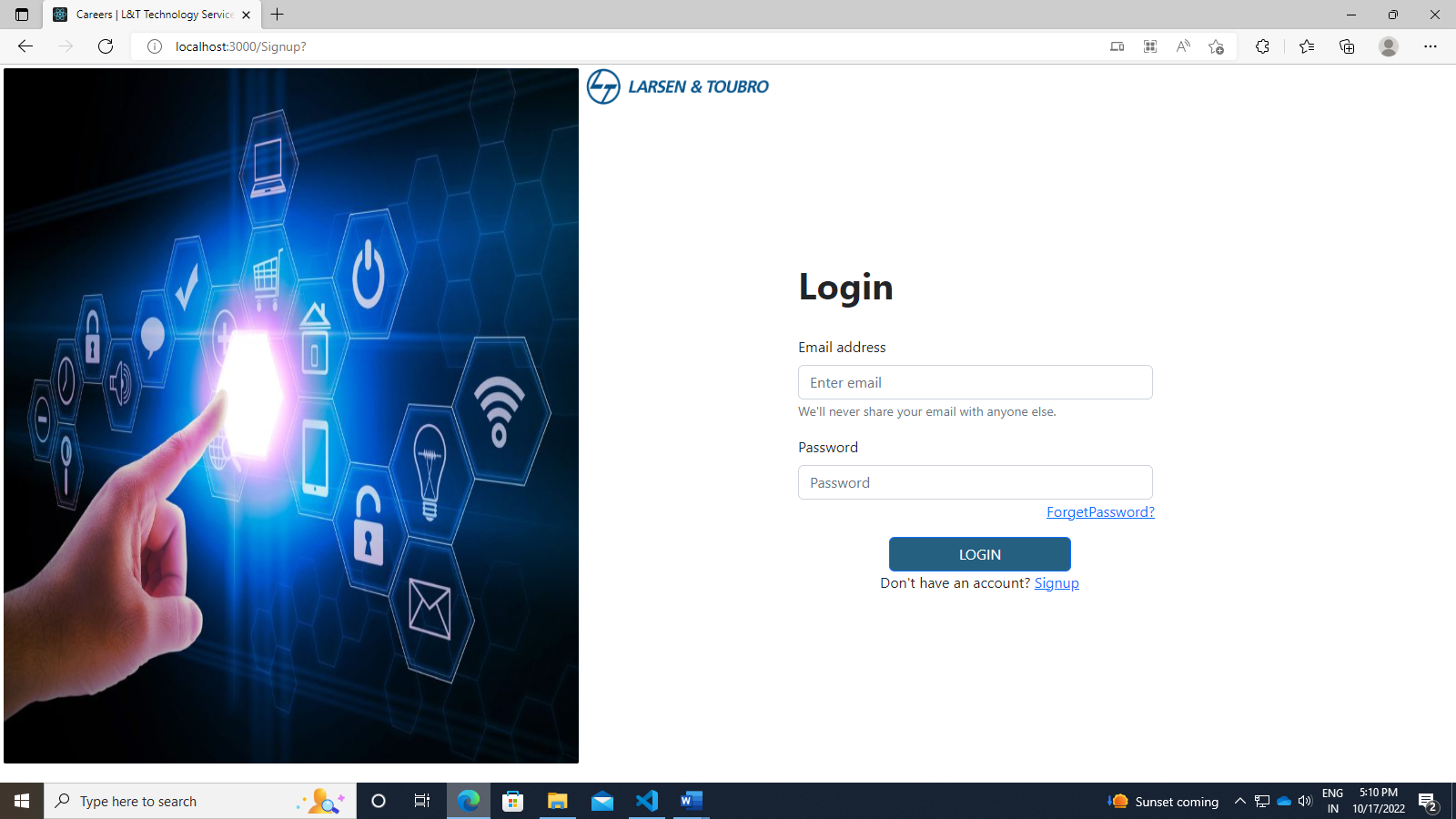
Step 1: To create an account, fill the details and click on Signup.



Step 2: The Forgot Password option is used to retrieve the account.



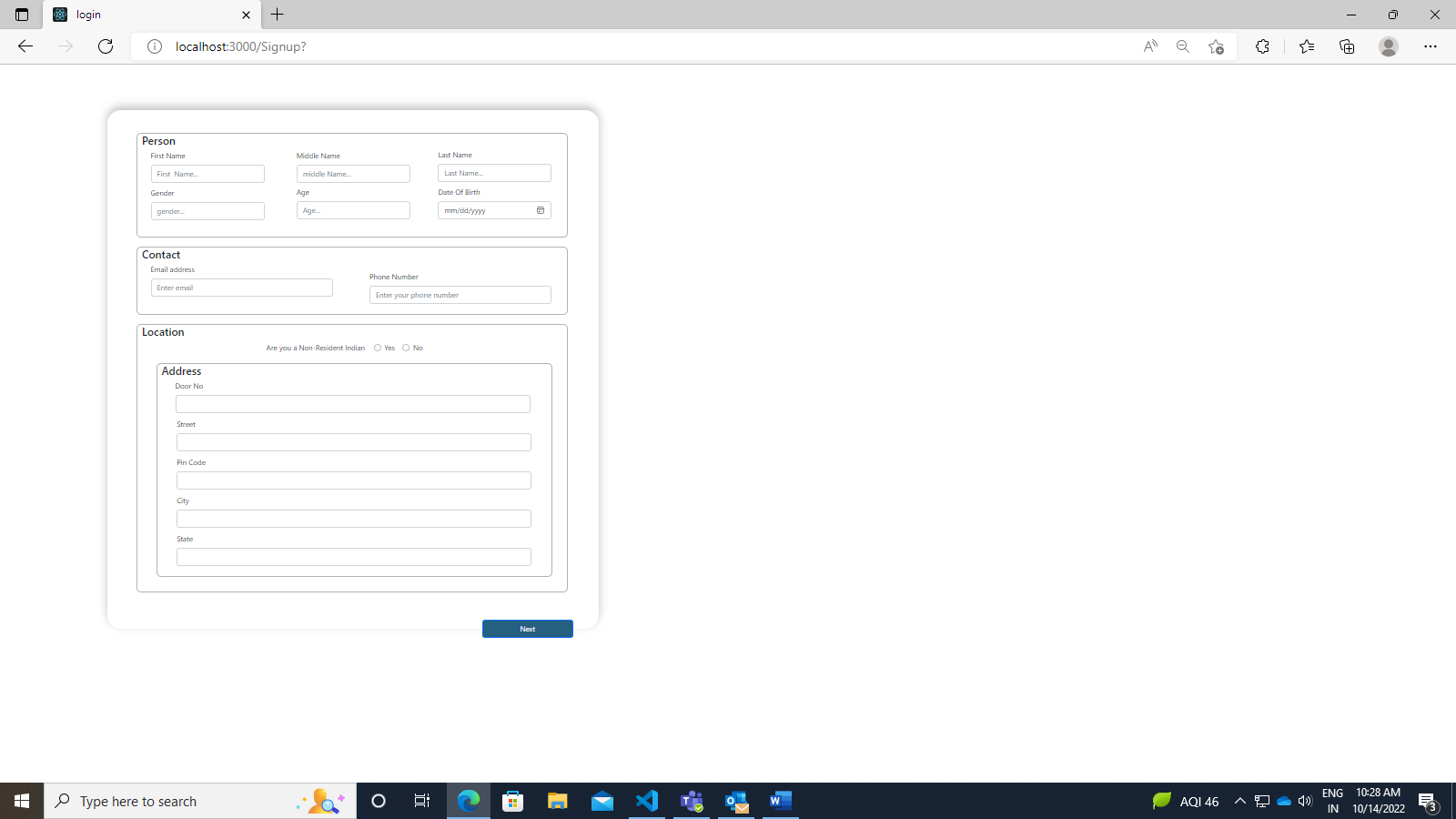
Step 3: Provide the respective credentials and Login.



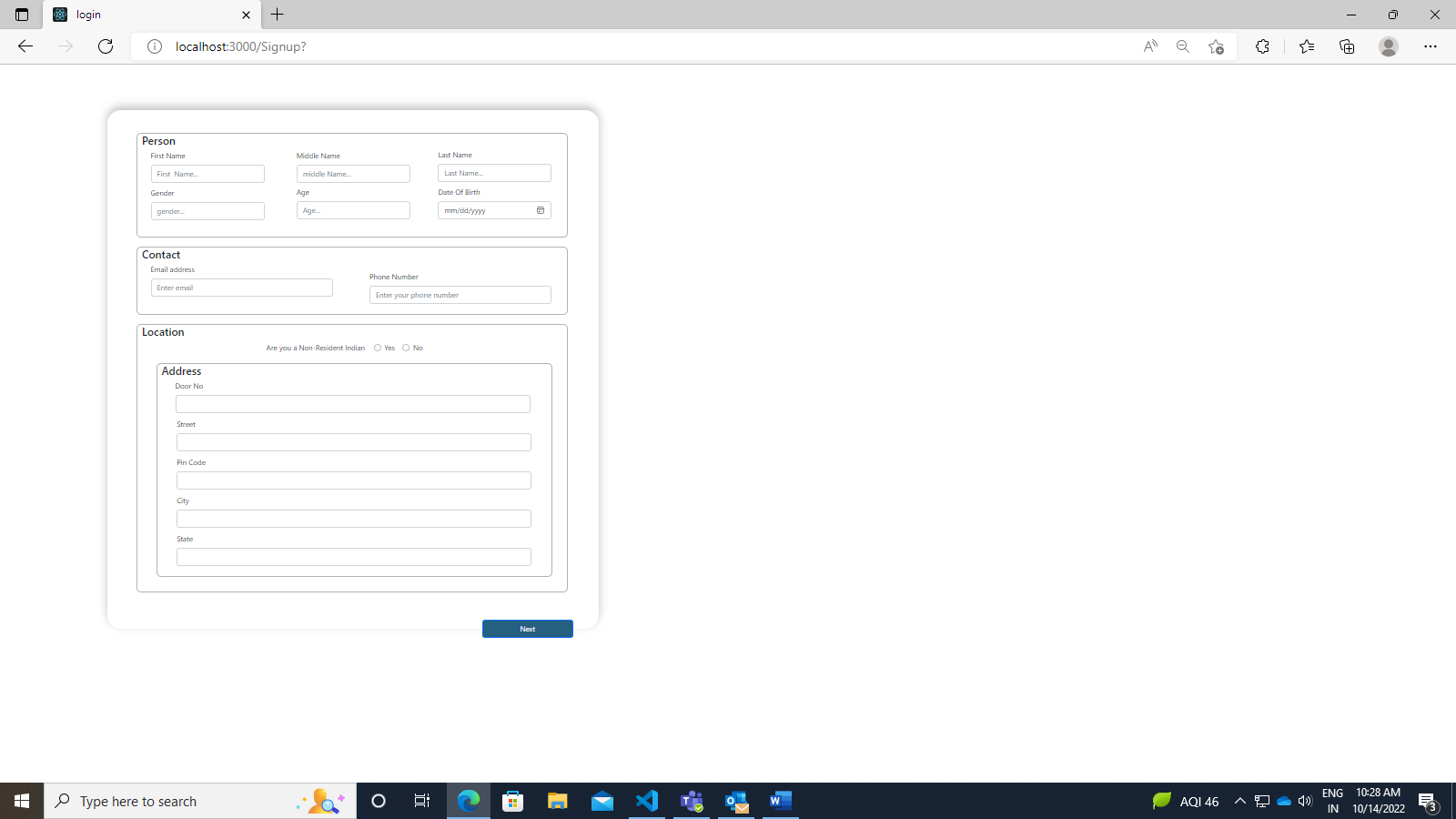
**2. The form of Personal Details:**

To apply for a Job, the personal details are required irrespective of the role preferred. The personal details include the user’s details, contact details and the user’s location details. On filling the personal details, the user can move to next page using the Save & Next button.

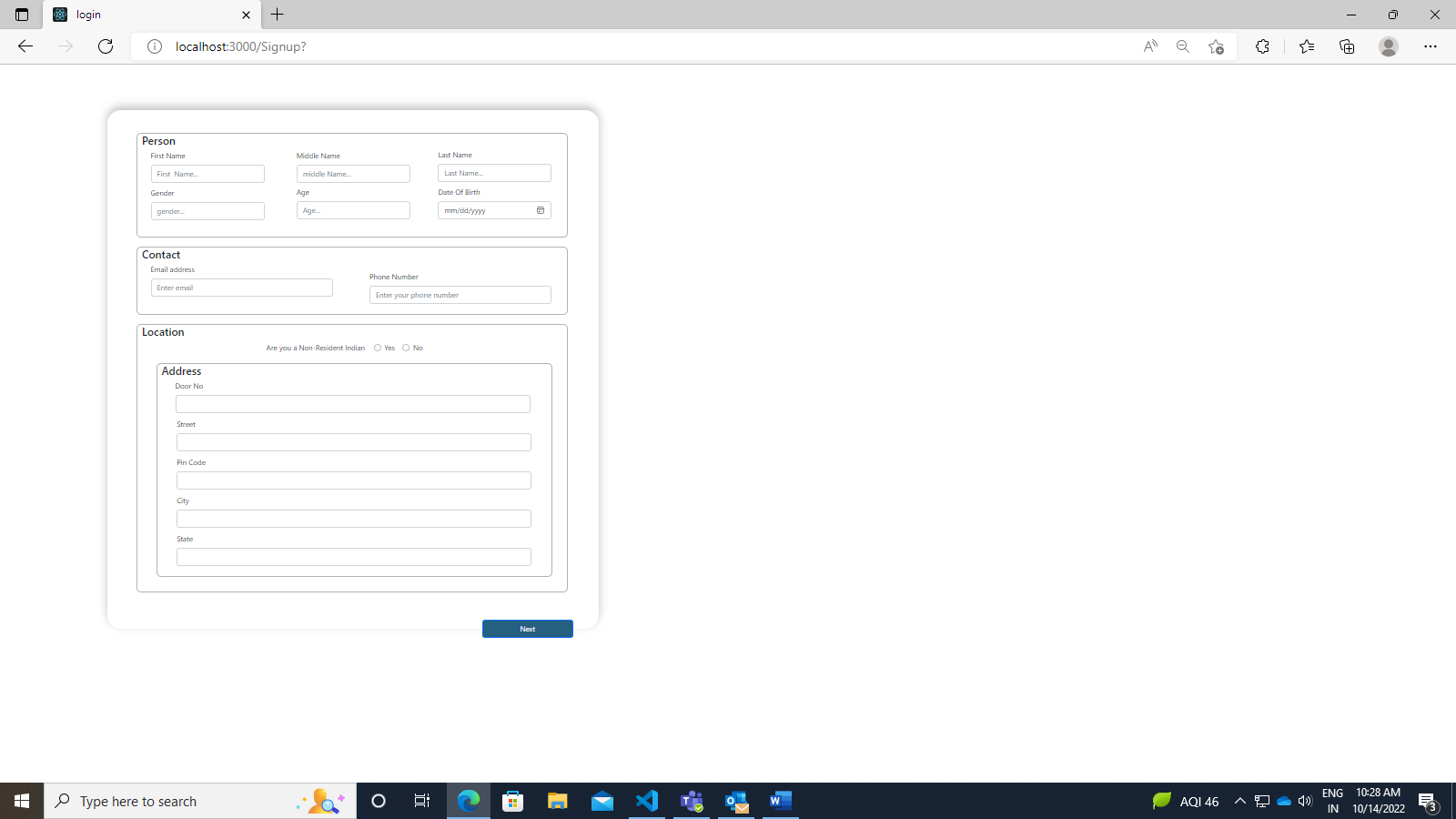
Step 1: User’s Details are set and entered as follows:



Step 2: The contact information of the user is set as follows:



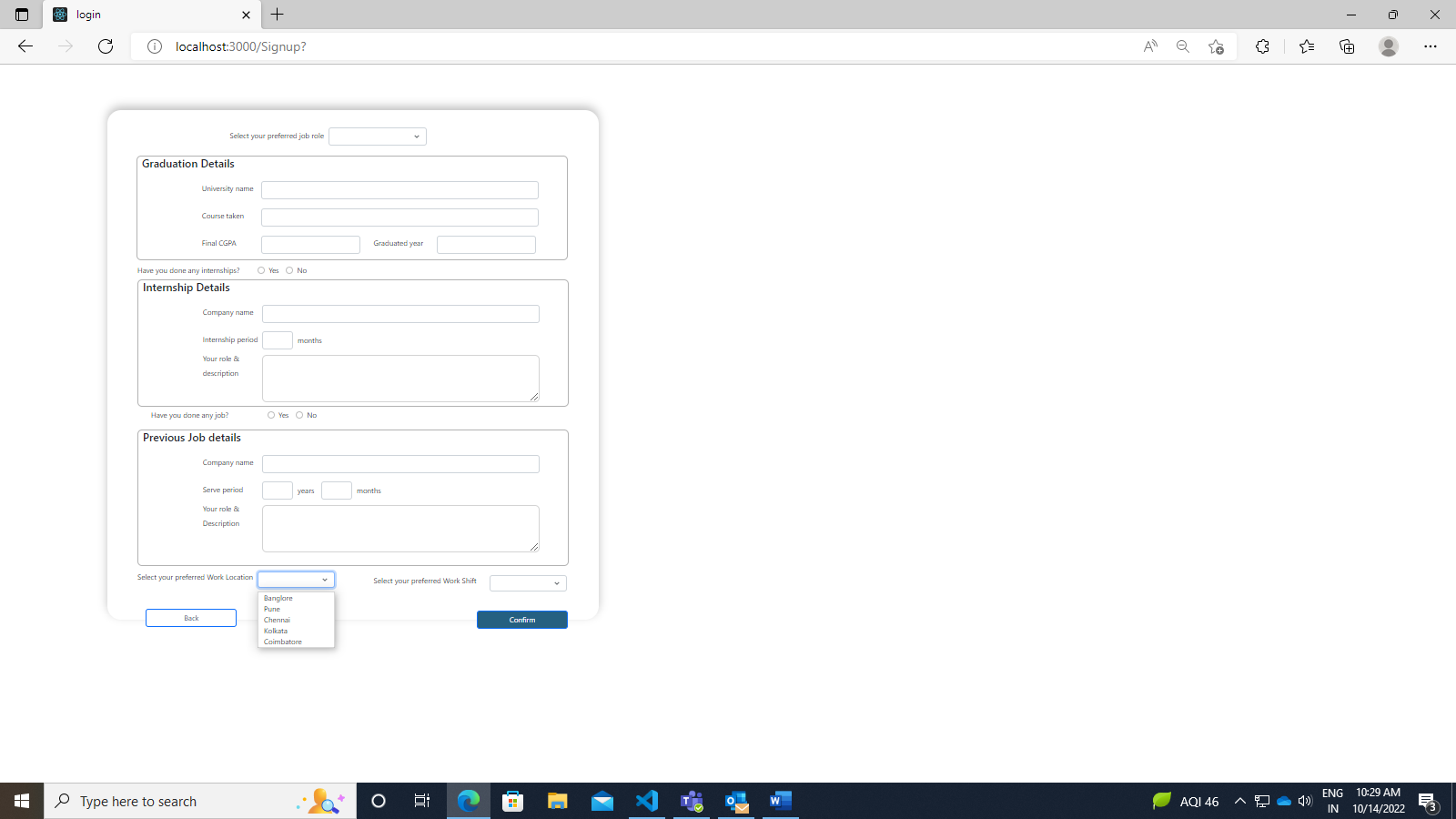
Step 3: The Location details of the user is set and the entire details is saved and moved to next page.



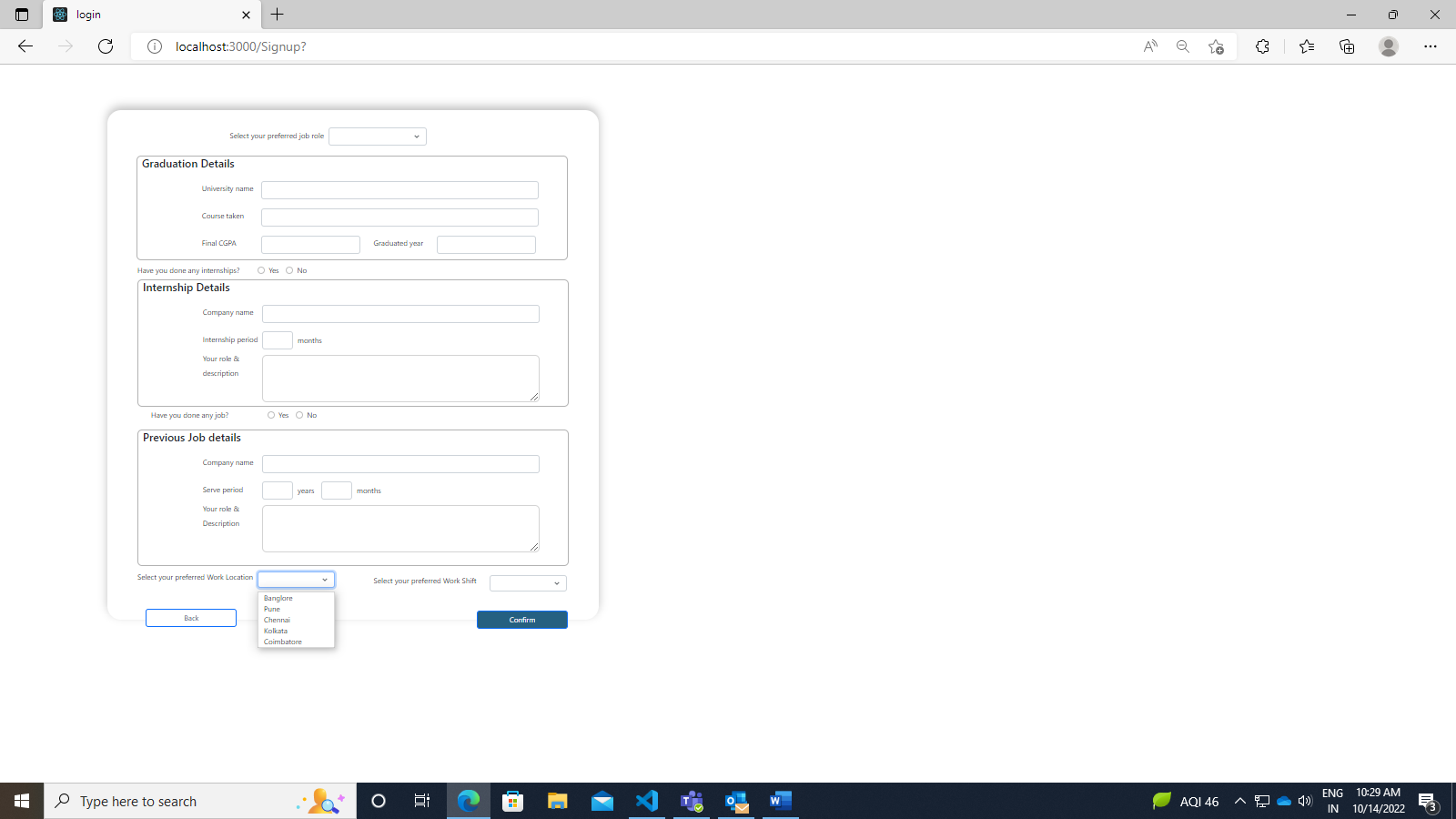
**3. The form of Job Details:**

To apply for a Job, the role preferred by the user becomes quite essential. The job details include the graduation details. The internship details and previous job details can be entered if the user possesses any of them. On filling the Job details, the user can upload the resume move to click on Confirm button. The user can also modify the previous personal details form if there are any corrections to be made. The user can navigate to previous page using back button.

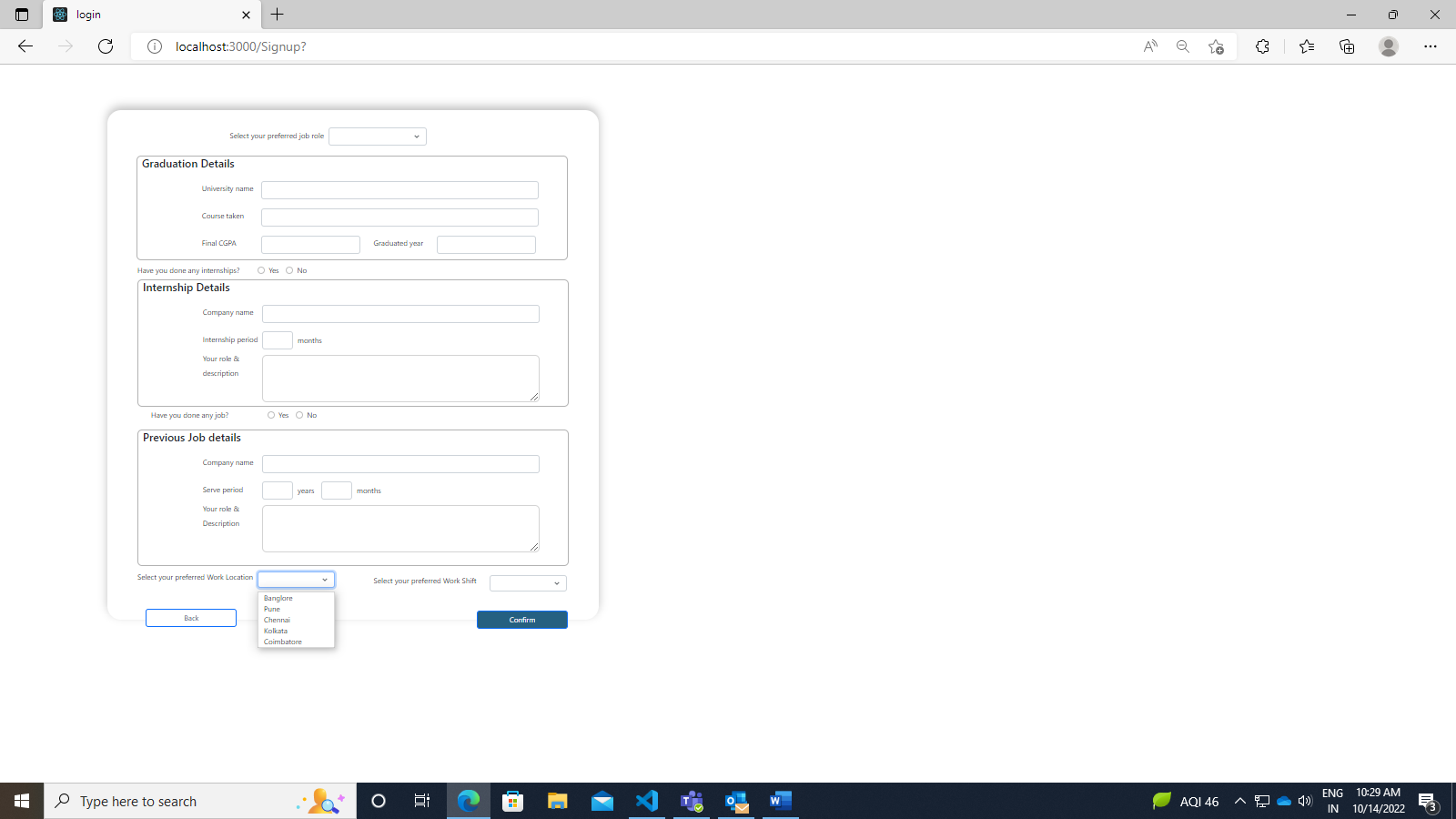
Step 1: Graduation details of the user is set and entered as follows:



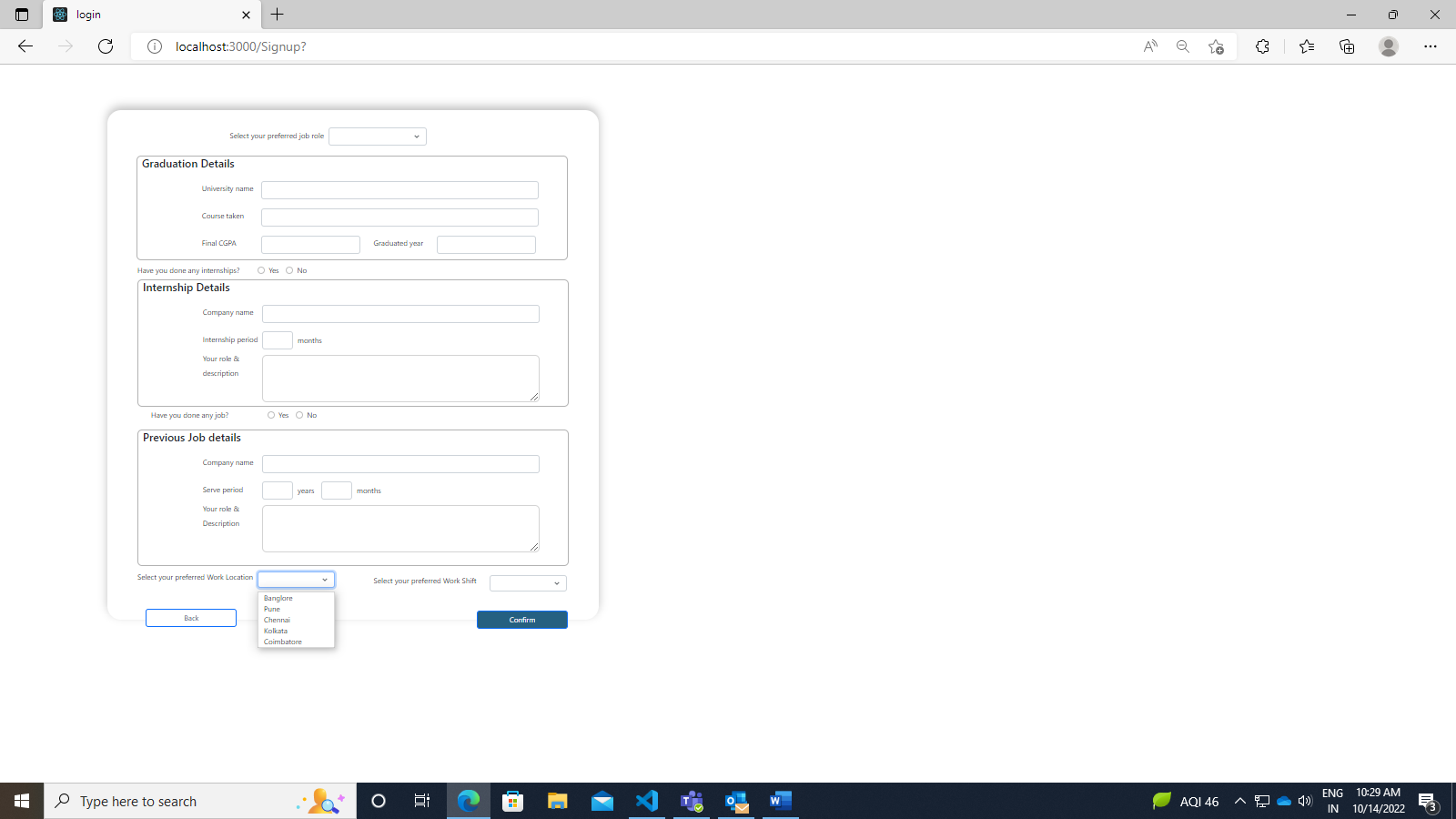
Step 2: Internship details can be entered if the user has any.



Step 3: Previous Job details can be entered if the user has any.

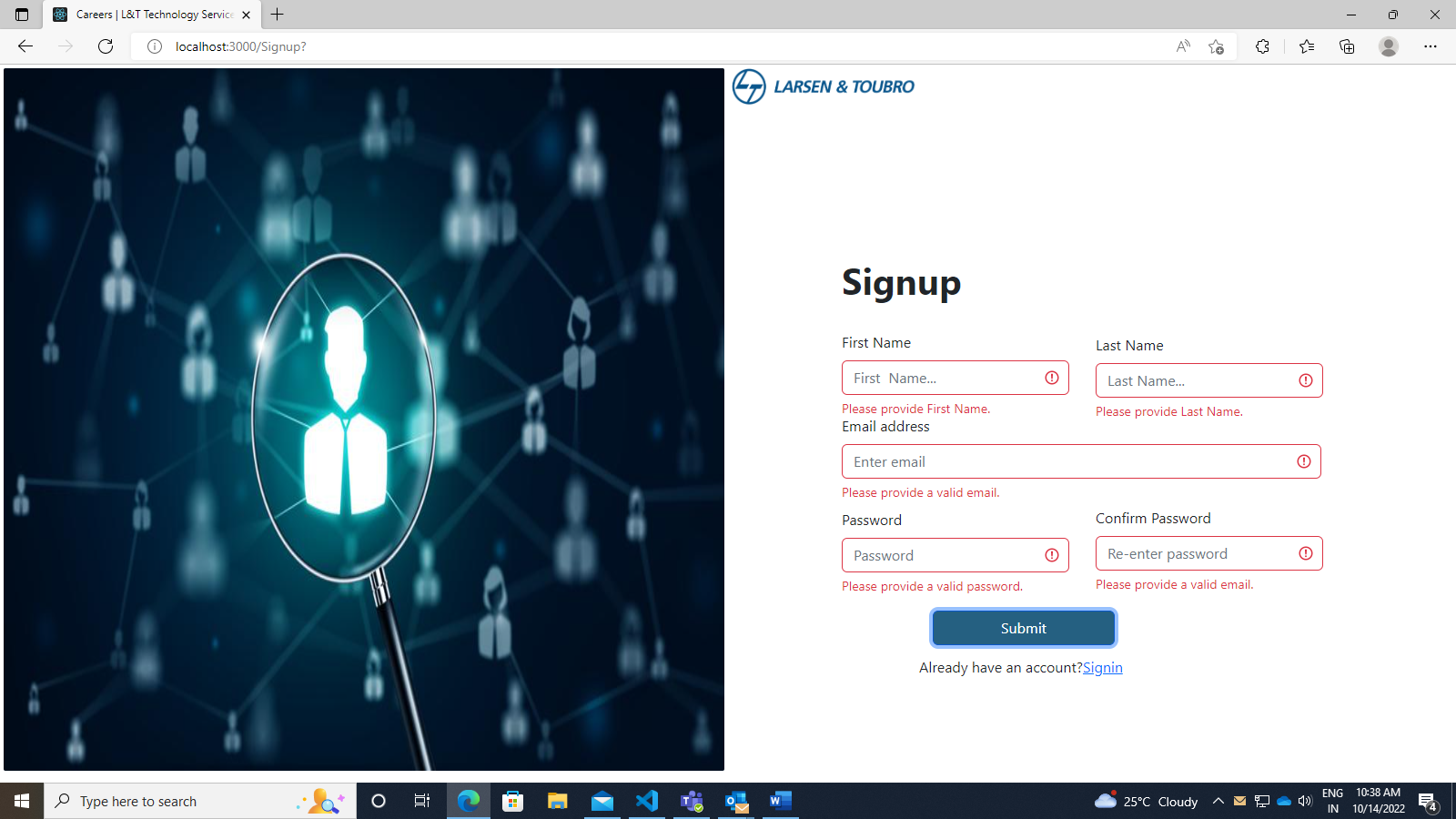


Step 4: The user’s resume can be uploaded and confirmed using confirm button or can navigate to personal details form using back button.

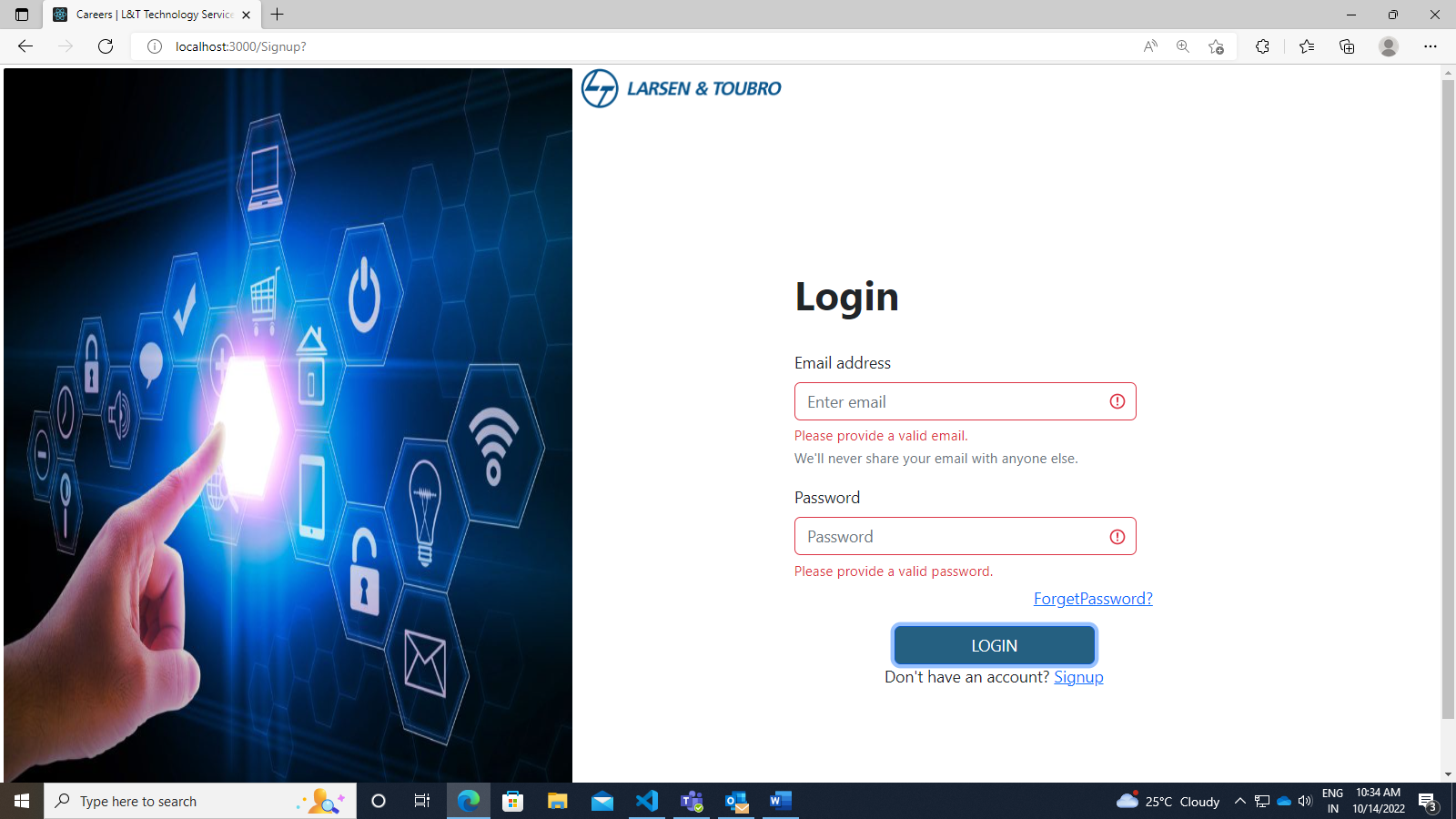


**4. Validation:**

Step 1: The validations of Signup page is designed as the ser to give the first and last names mandatorily. The e-mail address is specified as the @ symbol is set mandatorily. The password and it’s confirmation is validated as to give one uppercase, lowercase, special char and a number as constant.



Step 2: The validations of login page checks if the email and password are correct and matches with the data filled during the signup process.



**5. Front-end Tools Setup Process:**

**5.1. Figma UI Designing Tool:**

***STAGE 1: SKETCHES -*** This is the quickest, lowest cost and lowest commitment way to kick off your designs. It gets the high-level, top of mind ideas out of your head and onto your paper or screen. Generally, these are roughly captured ideas and if they look pretty bad then you're probably on the right track. It's something you should be able to do in a very short amount of time on paper, a whiteboard or using a wireframing app.

Contents:

* + Outlined boxes
  + Place holder boxes to represent images, text and graphics
  + Can be done in low, mid-or hi fidelity depending on the project needs at this

Purpose:

* + To start off the conversion about design needs and layout
  + Helps to clarify and define features needed and change ideas quickly
  + Helps visualize how screens might look on different device sizes and in different contexts

***STAGE 2: Wireframes or Gray boxing -*** Wireframes are created to show what things look like before visual design principles have been applied. It's where you can start to see the layout and elements taking shape.

Contents:

* + Filler content
  + Actual text
  + Placeholder images or stock images
  + Usually, can be done in your design software using an existing wireframing UI kit or done from scratch

Purpose:

* + Visualize the general layout
  + Establish correct proportions
  + Build trust with stakeholders and help them see the idea more fleshed out
  + Fast and cheap way to create initial ideas or to low-fidelity prototype

***STAGE 3 - Component design -*** Dynamic user interfaces (unlike static website landing pages or marketing sites) require you to think through states and conditionals. It also requires you to be consistent, apply good usability principles and decide the best way to layout and organize and disclose information and actions. This is where thinking in components and atomic elements and database views can really help.

Contents:

* UI kits
* Symbols (buttons, badges)
* Element states (hover state, clicked state)
* Components (like tables, lists, cards, forms)
* Actual text

Purpose:

* + Discover and address usability issues,
  + Uncover accessibility and responsive problems
  + Translate ideas into consistent elements that can be shared with engineers and more easily and accurately discussed and shared across teams.

***STAGE 4 - User Flows and Task flows -*** This is about understanding the user's mental models and your system's model and the orchestration of the pathways and responses that your interface will provide.

Contents:

* + Wireframes
  + Flow charts
  + Wire flows
  + Arrows
  + Conditionals

Purpose:

* + To show navigation routes
  + To check for missing states and information
  + To visualize entry, exist and decision points for your user/customer journey

***STAGE 5– Prototype -*** This tie everything together and shows how the app is expected to look and behave.

Contents:

* + All screens hi-fidelity
  + Target points to mimic what it will be like when the user interacts with the real thing

Purpose:

* Links together all of the screen and flows in a simulated environment
* Ensures that everything looks good, work well and flows as intended before it goes to production.
* This can be considered the pre-code or no-code version of your product.

**5.2. Visual Studio Code:**

* Download the Visual Studio Code Installer for Windows.
* Once it is downloaded, run the installer.

**5.3. Node.js and React:**

***Installation:***

* Download the Node.js ‘.msi’ installer.
* Run the Node.js installer.
* Double click on the .msi installer. The Node.js Setup wizard will open.
* Select “Next”. After clicking “Next”, accept the terms in the License Agreement” and Select “Next”
* Destination Folder: Set the Destination Folder where you want to install Node.js & Select “Next”
* Custom Setup: Select “Next” and Ready to Install Node.js. Click on “Install”.
* Verify whether Node.js is installed properly in the location say C:\Users\Admin> node -v.

***STAGE 1: Setting up a React Environment***

* To create a React application, run the command: *Npx create-react-app* Login Page
* The create-react-app will set and ready to run.
* Run the command *npm start* to run the React application for the pages that need to run.
* A new browser window pops up with the created React App. If not, open the browser and type localhost:3000 in the address bar.

***STAGE 2: Modify the React Application***

* Open the App.js file under the src folder found in the login page directory.
* In the src folder, a new file named components is created.
* Create jsx file for login, register, forget password, form one and two pages respectively in the components file.

***STAGE 3: Pages and their Validations:***

* The jsx file for the pages ( Login, Register and Forgot Password) is created, where the combination of html and react code for the respective page is coded.
* The required components are imported.
* **Validation**
  + The pages are validated using the :valid and :invalid pseudo selectors to apply validation styles as well as display feedback messages.
  + Bootstrap scopes the :valid and :invalid styles to parent .

const [validated, setValidated] = useState(false);

    const handleSubmit = (event) => {

        const form = event.currentTarget;

        if (form.checkValidity() === false) {

            event.preventDefault();

            event.stopPropagation();

        }

        setValidated(true);

    };

***STAGE 4: Form Pages***

* The jsx file for the forms is created, where the combination of html and react code for the forms are coded respectively .
* The required components for the form, button and image are imported from ‘react-bootstrap’.
* The pages are coded in such a way that it contains the divisions and they contain many text fields for texts, dropdown menu and radio buttons

**6. Back-end Tools Set-up Process**

**6.1. Python-Django**

**6.1.1. Installation**

* Django installation: *Pip install Django*
* Django rest framework: *Pip install djangorestframework*
* By default, Django framework comes with security block some domain so, *Pip install Django- coreheaders*

**6.1.2. Creating Project Folder**

Step 1: Create a folder for project -> start project with given name *Django- admin startproject {Django api}*

Step 2: To check the Django server is running *Python manage.py runserver*

Step 3: Create application using *Python manage.py startapp {filename}*

Step 4: Include the created app in setting.py file and include coreheaders

**6.1.3. views.py**

Django views are Python functions that takes http requests and returns http response. This is where the information is gathered and a proper response must be responded back.

Step 1: Create a file named *urls.py*in the same folder as the *views.py*file, and code.

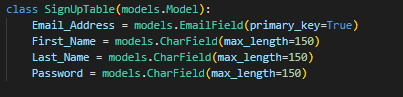
Step 2: The *urls.py* file just created is specific for the created application. We have to do some routing in the root directory.

Step 3: Open the *urls.py* located on the application folder and add the include module in the import statement including a path function in the *urlpatterns* list.

Step 4: On creating the Django project, an empty SQLite database was set by default.

Step 5: The settings.py file was altered to change database to postgreSQL.

Step 6: To create a new table, a new model must be created. Open the *models.py* file. It is empty by default, with only an import statement, *models.py*



**6.1.4. Tables in Database**

Fields are the inputs given by the user which must be stored in database.

Step 1: To add a table to the database, start creating a class and describe the table fields in it.

Step 2: Django creates a file with any new changes and stores the file in the */migrations/ folder.*

Step 3: Now, run *py manage.py migrate* Django will create and execute an SQL statement, based on the content of the new file in the migrations folder.

Step 4: To see the result in a web page, create a view for this task.

Step 5: The code for login, register and form pages are coded in *views.py*.

**6.1.5. Authentication**

The authentication is done using the following code:



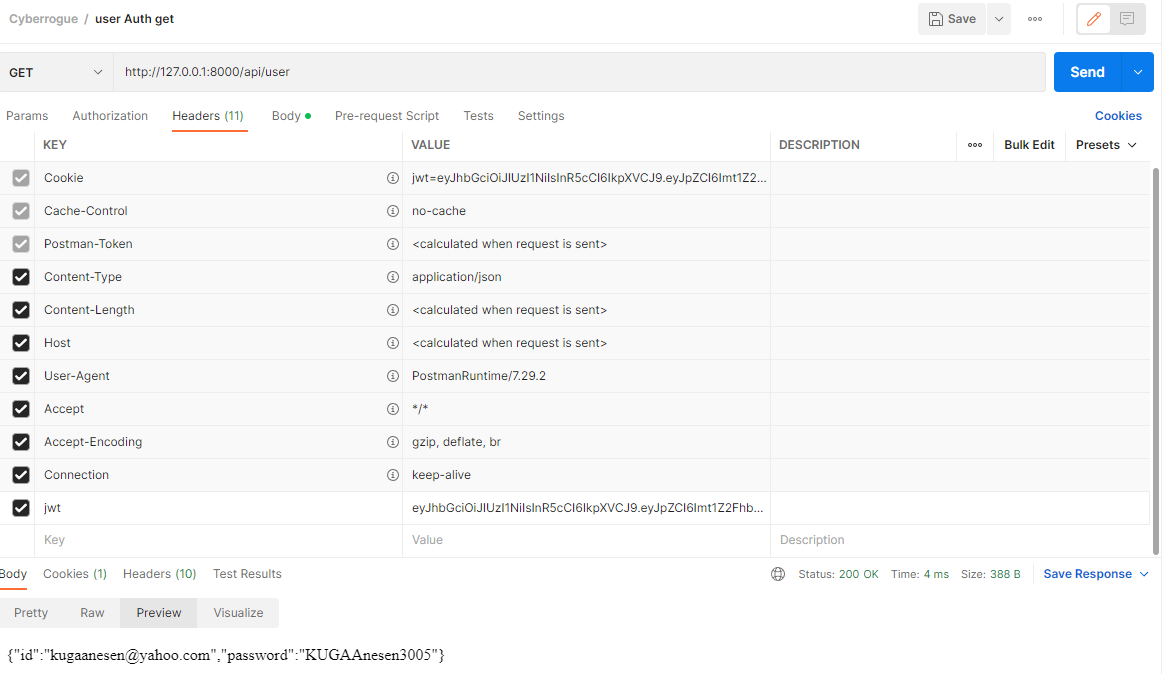
**6.2. Postman**

Step 1: The signup, login and form pages are coded using the POST method.

Step 2: Postman POST request allows appending data to the endpoint. This is a method used to add information within the request body in the server. It is commonly used for passing delicate information. Once we send some the request body via POST method, the API in turn yields certain information to us in Response.

Step 3: The User authentications using JWT token ‘cookies’ (Get Method)

Step 4: The GET method is used to retrieve information from the given server using a given URI. Requests using GET should only retrieve data and should have no other effect on the data.



**6.3. PostgreSQL**

PostgreSQL is a relational database. It stores data points in rows, with columns as different data attributes. A table stores multiple related rows.

The relational database is the most common type of database in use. It differentiates itself with a focus on integrations and extensibility.

PostgreSQL is an enterprise-class open-source database management system. It supports both SQL and JSON for relational and non-relational queries for extensibility and SQL compliance.

PostgreSQL supports advanced data types and performance optimization features, which are only available in expensive commercial databases.

