AN INDUSTRIAL ORIENTED MINI PROJECT REPORT

ON

"PROVISION SYSTEM FOR REGION BASED JOBS AND FACILITIES USING FULL STACK DEVELOPMENT"

Submitted in partial fulfillment of the requirement for the award of degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE & ENGINEERING

Submitted by

JOSHI SAYALI SHRIPAD (20JJ1A0521)

MANIKANTA PRAMOD KUMAR REDDY (20JJ1A0530)

KOLUGOORI BHAVANA REDDY (20JJ1A0524)

CHUKKA HASINI (20JJ1A0512)

Under the Esteemed Guidance of

DR. S. VISWANADHA RAJU

Senior Professor, Department of CSE.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

(Accredited by NBA: UG(CSE))

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY

HYDERABAD UNIVERSITY COLLEGE OF ENGINEERING JAGTIAL

(Accredited By NAAC with A+ Grade)

Nachupally (Kondagattu), Jagtial Dist. -505501. T.S

2023-2024

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY

HYDERABAD UNIVERSITY COLLEGE OF ENGINEERING JAGTIAL

(Accredited By NAAC with A+ Grade)

NACHUPALLY (KONDAGATTU), JAGTIAL DIST, TELANGANA STATE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

(Accredited by NBA: UG(CSE))

CERTIFICATE

This is to certify that the mini project entitled"**PROVISION SYSTEM FOR REGION BASED JOBS AND FACILITIES USING FULL STACK DEVELOPMENT**" is a bonafide work carried out by **JOSHI SAYALI SHRIPAD (20JJ1A0521), MANIKANTA PRAMOD KUMAR REDDY (20JJ1A0530), KOLUGOORI BHAVANA REDDY (20JJ1A0524), CHUKKA HASINI (20JJ1A0512)** in the partial fulfillment of the requirements for the award of the degree of **BACHELOR OF TECHNOLOGY** in **COMPUTER SCIENCE AND ENGINEERING** by the Jawaharlal Nehru Technological University Hyderabad during the academic year 2023-2024.

The results embodied in this report have not been submitted to any other University or Institution for the award of any degree.

- - - - - - - - - - - - - - - -	- -
Project Guide	**Head of the Department**
Dr. S. VISWANADHA RAJU	**Dr. B. SATEESH KUMAR**
Senior Professor, Dept of CSE	**Professor & Head, Dept of CSE**

- - - - - - - - - - - - - - -

External Examiner

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY
HYDERABAD UNIVERSITY COLLEGE OF ENGINEERING JAGTIAL

(Accredited By NAAC with A+ Grade)

NACHUPALLY (KONDAGATTU), JAGTIAL DIST, TELANGANA STATE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

(Accredited by NBA: UG(CSE))

DECLARATION

We hereby declare that the project work entitled "**PROVISION SYSTEM FOR REGION BASED JOBS AND FACILITIES**" submitted in partial fulfillment of the requirements for the award of the degree **BACHELOR OF TECHNOLOGY** in the **COMPUTER SCIENCE AND ENGINEERING**, which was carried out under the supervision of **Dr. S. VISWANADHA RAJU** , Senior Professor, Department of CSE, JNTUH University College of Engineering, Jagtial.

Also, we declare that the matter embedded in this thesis has not been submitted by us in full or partial thereof for the award of any degree/diploma of any other institution or University previously.

JOSHI SAYALI SHRIPAD (20JJ1A0521)

MANIKANTA PRAMOD KUMAR REDDY (20JJ1A0530)

KOLUGOORI BHAVANA REDDY (20JJ1A0524)

CHUKKA HASINI (20JJ1A0512)

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of the task would be put incomplete without the mention of the people who made it possible, whose guidance and encouragement crown all the efforts with success.

We wish to express our deep sense of gratitude to **Dr. S. VISWANADHA RAJU** Senior Professor, department of CSE, and our Project Guide for his able, vigorous guidance and useful suggestions, which helped us in completing the project work in time.

We are extremely grateful to our HOD- **Dr. B. SATEESH KUMAR**, Professor of CSE& Head for his immense support and cooperation that contributed a lot in the completion of the task.

We show gratitude to our beloved Principal **Dr. V. KAMAKSHI PRASAD** , Professor of CSE and vice principal **Dr. T. VENUGOPAL** Sir, for providing necessary infrastructure and resources for the accomplishment of our project report at JNTUHCEJ.

We show our immense gratitude towards our college library for providing us with necessary references required for completing this project.

We also thank all the staff members of Computer Science and Engineering department and H&S department,JNTUH UCEJ for their valuable support and generous advice.

Finally, thanks to all my friends and family members for their continuous support and enthusiastic help.

JOSHI SAYALI SHRIPAD (20JJ1A0521)

MANIKANTA PRAMOD KUMAR REDDY (20JJ1A0530)

KOLUGOORI BHAVANA REDDY (20JJ1A0524)

CHUKKA HASINI (20JJ1A0512)

ABSTRACT

The current unemployment crisis highlights the need for solutions that not only connect job seekers with vacancies but also promote local job opportunities. The PROVISION SYSTEM FOR REGION–BASED JOBS AND FACILITIES aims to address this issue by enabling individuals to find work in their local area and support nearby businesses.

For instance, someone looking for part-time employment could register as a driving teacher on the platform. Another person in need of driving lessons could easily find and hire them through the system, allowing both parties to discuss payment and proceed with their arrangement.

Additionally, this system would assist individuals in locating local businesses that offer services such as carpentry, plumbing, or bicycle repair. It eliminates the hassle of searching for these services by providing direct contact information, enabling users to easily connect with and hire the appropriate professionals.

To implement this system, a web application would be developed using Web Development tools and integrated with a Database Management System. This technology can be implemented in various local areas and scaled up to serve larger communities. By facilitating local job opportunities and supporting small businesses, this system has the potential to reduce unemployment rates and contribute to local economic growth.

To construct this website, we'll utilize HTML, CSS, and JavaScript for the front-end development. For the back-end, we'll employ Node.js along with MongoDB as the database. This tech stack will enable us to build a responsive and dynamic platform, ensuring a user-friendly interface and seamless data management.

TABLE OF CONTENTS

LIST OF FIGURES

1.INTRODUCTION

1.1 PROJECT OVERVIEW

The current unemployment crisis highlights the need for solutions that helps in solving the issues related to the unemployment. It requires not only connecting job seekers with vacancies but also promote local job opportunities. “Provision system for region- based jobs and facilities” is a website which assists individuals in locating local businesses that offer services such as carpentry, plumbing, or bicycle repair.

It eliminates the hassle of searching for these services by providing direct contact information, enabling users to easily connect with and hire the appropriate professionals.

For this purpose, we use web development tools for creating a website that takes user information and stores that information and helps them to find the jobs / promote the jobs.

This project uses the respective user’s data provided by the individuals. This data contains information about the individuals. Users post their jobs in respective categories and find jobs in an easier way.

To protect users, employ strong authentication, encryption for data transmission, input validation, access control, and regular updates. Securely store data, define clear privacy policies, monitor activities, ensure safe payment gateways, and conduct regular audits to bolster website security and user confidence.

1.1.1 PROBLEM STATEMENT

- In today's dynamic and fast-paced world, the current unemployment crises is the major issue that is being faced by the people all over the world. Job opportunities and facilities is crucial for the well-being and development of communities.
- Many regions face challenges in ensuring that job seekers find suitable employment and the contractors/manufacturers face issues in searching for the workers. To address these issues,the proposed development of a Region-Based Jobs and Facilities Provision System.

1.1.2 MOTIVATION

The current unemployment crisis highlights the need for solutions that not only connect job seekers with vacancies but also promote local job opportunities. The PROVISION SYSTEM FOR REGION–BASED JOBS AND FACILITIES aims to address this issue by enabling individuals to find work in their local area and support nearby businesses.

Many young people today lack the readiness to work under someone else and are resistant to relocating from their hometowns to find employment. Moreover, there are those who aspire to establish their own businesses but struggle to find suitable marketing platforms for promoting their products or services. Therefore, the implementation of PROVISION SYSTEM FOR REGION–BASED JOBS AND FACILITIES could assist these youngsters in advertising their businesses.

In today's scenario, locating a suitable painter for a family looking to paint their house can be quite daunting. Hence, a web application consolidating comprehensive details about local painters would prove immensely beneficial for their search.

1.2 OBJECTIVES

This regional based jobs and facilities provision system is a website that helps people in a region for finding jobs and the small businesses to promote their businesses. For this purpose, web development tools are used for the development of the website. This bridge the unemployment gaps between the people in the local regions / local areas by

- Showing various jobs and businesses present in respective regions.
- Improving the publicity of small regional businesses.
- Providing smooth communication between the producers and consumers.
- Helping local people to find the local businesses in an easier way.
- Promote skill growth in remote regions through our platform.

1.3 EXISTING SYSTEM

Indeed, many websites and applications are dedicated to connecting individuals with job opportunities globally and aiding users in finding and evaluating local businesses. Here's a breakdown of these two aspects:

Job Opportunities:

1. Job Search Engines: Websites like Indeed, LinkedIn, and Glassdoor aggregate job listings from various sources, allowing users to search for positions based on industry, location, and qualifications.
2. Company Career Pages: Many companies have dedicated career pages on their websites where they post job openings, provide information about their culture, and offer insights into the application process.
3. Freelance Platforms: Platforms like Upwork, Freelancer, and Fiverr connect freelancers with clients worldwide, offering a range of job opportunities in fields such as writing, design, programming, and more.
4. Industry-Specific Platforms: Some industries have specialized job boards or platforms, such as Dribbble for designers, GitHub for developers, and Behance for creative professionals.

Local Business Discovery:

1. Google Maps: Google Maps not only provides navigation but also allows users to discover local businesses, read reviews, view photos, and get information about hours of operation, contact details, and directions.
2. Yelp: Yelp is a platform where users can find and review local businesses, including restaurants, shops, and service providers. It helps people make informed decisions based on the experiences of others.
3. Yellow Pages Online: An online version of the traditional Yellow Pages, it provides a directory of businesses categorized by industry and location.
4. Local Business Directories: Many regions have local business directories specific to their area, helping residents and visitors discover nearby services.

5. Social Media Platforms: Platforms like Facebook and Instagram allow businesses to create pages, share updates, and interact with customers. Users can find local businesses through these platforms and read reviews and recommendations.
6. Review and Aggregator Sites: Websites like TripAdvisor (for travel-related businesses) and Angie's List (for home services) aggregate reviews and ratings to help users choose local services.

These websites and applications play a crucial role in bridging the gap between job seekers and employers globally, as well as aiding consumers in discovering and evaluating local businesses for various services.

1.3.1 Drawbacks of Existing System:

- Fragmented User Experience:

Users often have to navigate multiple platforms for job search and local business discovery, leading to a fragmented user experience.

- Lack of Location-Based Precision:

While existing platforms offer geolocation features, they might not prioritize region-specific job listings or services effectively.

- Limited Direct Engagement:

Current platforms often act as intermediaries, displaying listings and reviews without enabling direct engagement between users.

- Incomplete Information for Decision-Making:

While users can access reviews and basic information, they might lack comprehensive details required for informed decision-making.

- Limited Support and Assistance:

Users might encounter challenges with limited support or assistance in navigating the platforms.

The proposed system aims to address these limitations by providing an integrated platform that emphasizes direct engagement, location-based precision, and comprehensive information, fostering a more interactive and personalized user experience for both job seekers and local service providers.

1.4 PROPOSED SYSTEM

The "Provision System for Region-Based Jobs and Facilities" is a platform designed to empower individuals by facilitating their engagement in local employment opportunities and connecting them with nearby businesses. The system goes beyond job listings and includes features that enable direct communication between producers (job seekers or service providers) and consumers (employers or service users), allowing them to make deals and transactions within the platform.

Here's an elaboration on the key features and steps of the system:

1. **User Registration and Profile Creation**:
   - Users (both job seekers and employers or service providers and users) create accounts on the platform.
   - They provide essential information, such as personal details, skills, qualifications, and preferences.
2. **Job Listings and Services Offered:**
   - Job seekers or service providers create listings detailing the type of jobs they are seeking or services they offer.
   - Employers or service users can browse through these listings based on location, skills, and other relevant criteria.
3. **Deal Making and Agreements:**
   - Once the communication phase is complete, users can make deals or agreements within the platform.
   - For job seekers and employers, this could involve discussing terms of employment.
   - For service providers and users, this could include finalizing service details, pricing, and other relevant terms.
4. **Privacy and Security Measures:**
   - The system implements robust privacy and security measures to protect user data, transactions, and communication within the platform.

By integrating these features, the "Provision System for Region-Based Jobs and Facilities" aims to create a dynamic and interactive ecosystem that empowers individuals to connect with local opportunities and businesses, fostering a sense of community and economic growth.

Proposed system addresses the drawbacks of existing system by following methods :

- The proposed system integrates job listings and service offerings in a unified platform, streamlining the user experience by offering a one-stop solution for both job opportunities and local services.
- It focuses on region-based precision, connecting users with opportunities and businesses specifically within their vicinity, enhancing relevance and accessibility.
- It promotes direct communication and deal-making, allowing users to negotiate terms and finalize agreements within the platform, fostering active participation and autonomy.
- It also incorporates user assistance and support features to aid users throughout their journey, offering help centers and customer support services.

2. SOFTWARE REQUIREMENT ANALYSIS

2.1 REQUIREMENT ANALYSIS

Project perspective

The ultimate perspective of any technology or innovation is to make the human life much easier to sustain either by providing a basic need or adding on a luxury to basic life. “This project, Provision system for region-based jobs and facilities” can be used for both types.

Project Features

The features can be considered as the in-dependency, Ease-to-use, Interactive & Responsive.

a. In-dependency.

b.Ease-to-use.

c.Interactive & responsive.

2.2 REQUIREMENT SPECIFICATIONS

External Interface Requirements

External Interface requirements are the necessities that should be satisfied for successful development and working the application. Such interface requirements are basically UI requirements, Hardware and software requirements

User Interfaces:

1 Laptop/Desktop with Graphic card.

2 Ability to login and access their account.

Hardware Interfaces:

- Laptop, Desktop : Minimum i5 configuration
- RAM Capacity : 8 GB
- Hard Disk : 60 GB

Software Interfaces:

- Languages use : HTML5, CSS3, JavaScript, MongoDB, Node.JS
- OS : Windows/Linux
- Platform : GitHub

Functional Requirements

1. **User Authentication:** Users should be able to create accounts, log in, and reset passwords securely.
2. **User Roles:** Different user roles (e.g., admin, regular user) should have specific permissions and access levels.
3. **User Profile Management:** Users should be able to update their profiles, including personal information and preferences.
4. **Content Creation and Management:** Authorized users should create, edit, and delete content (e.g., articles, posts).
5. **Search and Filtering:** Users should search for content and filter results based on various criteria.
6. **Data CRUD Operations:** Full CRUD (Create, Read, Update, Delete) operations for managing data entities (e.g., products, orders).
7. **Notification System:** Send notifications to users (e.g., email, in-app) for various events (e.g., new messages, order updates).
8. **File Uploads:** Allow users to upload files (e.g., images, documents) and manage them.
9. **Messaging and Chat:** Implement real-time messaging or chat functionality for user communication.
10. **Integration with External APIs:** Integrate with third-party APIs (e.g., social media, mapping services) for enhanced functionality.
11. **Security Features:** Implement security measures, including authorization checks,
12. data validation, and protection against common vulnerabilities.

Non-Functional Requirements

Non-Functional Requirements are those which describe the threshold capability of the project for functioning without any breaks or errors. They are usually performance and safety requirements.

1. **Performance:** Ensure the system responds quickly and efficiently under various loads, with acceptable page load times.
2. **Scalability:** The application should handle increased user and data loads by scaling horizontally or vertically.
3. **Reliability:** The system should be available with minimal downtime and recover gracefully from failures.
4. **Security:** Protect user data, implement encryption, and follow best practices for authentication and authorization.
5. **Usability:** The user interface should be intuitive and user-friendly, with accessibility features for all users.
6. **Compatibility:** The application should work on different browsers and devices, with responsive design for mobile devices.
7. **Data Integrity:** Ensure data consistency and accuracy throughout the application.
8. **Backup and Recovery:** Implement regular data backups and disaster recovery plans.
9. **Compliance:** Comply with relevant laws and regulations (e.g., GDPR for data privacy).
10. **Monitoring and Logging:** Set up monitoring tools to track system health and log events for debugging and auditing.
11. **Availability:** Aim for high availability (e.g., 99.9% uptime) based on project requirements.

12. **Resource Efficiency:** Optimize resource usage, including memory and CPU, to reduce costs.
13. **Documentation:** Maintain comprehensive documentation for code, APIs, and system architecture.
14. **Response Times:** Define acceptable response times for various actions within the application.
15. **Data Storage and Retrieval:** Specify data storage requirements, such as database choices and data retrieval times.
16. **Error Handling:** Implement clear and informative error messages for users and detailed error logs for developers.

2.3 FEASIBILITY STUDY

A feasibility study for the provision system for region based jobs and facilities provision system project involves assessing the website, audience response, cost-effectiveness, and practicality of implementing the proposed system. Here are the key components of the feasibility study:

- Economic Feasibility
- Technical Feasibility
- Social Feasibility
- Market Feasibility

2.3.1 Economic Feasibility

- Initial development costs for the project, that include design, programming, hosting the website.
- Ongoing operational costs of the website are maintenance of website, marketing, customer support.
- Revenue model and potential income streams.

2.3.2 Technical Feasibility

- Evaluating the technical requirements for developing and maintaining the website.
  - Technical Requirements:
    - Frontend (HTML, CSS, JavaScript):
      - Ensure compatibility with modern browsers.
      - Responsive design for various devices.
    - Backend (Node.js, MongoDB):
      - Evaluate server-side requirements for Node.js (version, modules).
      - Design MongoDB schema and ensure efficient data storage and retrieval.
- Evaluating the availability of technology for the website development and expertise required.
  - Technology Availability and Expertise:
    - Assess the availability of Node.js and MongoDB for the chosen hosting platform.
    - Evaluate the expertise of the development team in HTML, CSS, JavaScript, Node.js, and MongoDB.
- Software, hardware, and development tools needed for the development and maintenance of the website.
    - Assess server requirements based on expected traffic and database size.
    - Ensure that developers have the necessary code editors (e.g., Visual Studio Code).
- Consideration of scalability and future updates of the website.

2.3.3 Social Feasibility

- Identifying how the website will positively contribute to the community.
- Assess if the platform can address any existing social issues related to job access and facility availability.
- Consider how the website promotes businesses by providing opportunities for job seekers and service users.

- Evaluate if the platform can reduce existing disparities in jobs and facility access.
- Evaluate the accessibility of the website for users with diverse needs.
- Showcasing local businesses, and facilitating local service interactions to positively impact the community and foster a sense of local economy growth.

2.3.4 Market Feasibility

- Evaluate the current job and facilities market and the scope of the website in the market.
- Identify competitors in the jobs and facilities market, and analyze their strengths and weaknesses.
- Determine the demand for similar platforms in the market.

3. SOFTWARE DESIGN

3.1 ARCHITECTURE

Architecture describes the overall functionality of the project with all the modules specified in it. It gives a clear picture of the internal process of the projects. The architecture along with modules can be displayed as the following.

Fig 3.1: Architecture

3.2 MODULE DESCRIPTION

According to the architecture above,the modules used in this project are : front end development,backend development and database integration with the front end and backends

Modules of the project can be defined as

3.2.1 Front-end Development

Frontend development refers to the process of creating the user interface (UI) and user experience (UX) of a website or web applications It involves the design, development, and implementation of the visual elements that users interact with on the screen.

Developers use combination of languages, frameworks, and tools to build responsive, engaging and interactive interfaces that provide a seamless experience for users.

3.2.2 Back-end Development

Backend development is a critical component of web development, Backend development refers to the server-side of web development, where the server, database, and application logic reside. Backend developers focus on building the infrastructure that supports the frontend and enables the functionality of a web application

3.2.3 Database Integration

Database integration is a crucial aspect of software development, enabling applications to store, retrieve, and manage data efficiently. It involves connecting the backend of an application with a database system, allowing application to interact with and manipulating data.

Database integration includes integration of database with both the frontend and backend.Integrating the database with both frontend and backend systems facilitates a smooth exchange of data, ensuring that user inputs are processed, stored securely, and made accessible as needed throughout the website.

In frontend database integration, user information retrieved from the database, like profile details during login, is presented to users. User-entered data on the frontend, via forms or interfaces, is transferred to the backend for processing and storage in the database. The backend manages various data operations such as registration and service requests, applying specific rules before database interaction to ensure accuracy and security. For instance, when a user submits a service request, the backend processes the data, interacts with the database to securely store the information, and sends confirmation to the frontend upon successful storage.

3.2.4 API - Application programming interface

APIs (Application Programming Interfaces) in a website act as intermediaries that enable different software systems or components to communicate and interact with each other. In the context of the website various APIs implemented to facilitate its functionalities:

1.User Registration/Login API: An API handling user authentication and registration, allowing users to create accounts, log in, and access personalized features within the platform.

2. Job Listings/Services API: APIs managing the creation, retrieval, and modification of job listings or service offerings. These APIs can allow users to post jobs/services and retrieve relevant listings based on specific criteria.

These APIs collectively form the backbone of the system, enabling seamless interactions between users, facilitating key functionalities like user registration, job/service management. Each API serves as an interface allowing different components of the system to exchange data and perform specific functions necessary for the platform's operation.

3.3 UML DIAGRAMS

UML diagrams, crucial in software engineering, visually represent system components, behaviors, and relationships. They serve as blueprints, aiding system analysis, design, and communication among teams and stakeholders. These diagrams simplify complexities, acting as collaborative tools for discussing and resolving issues. They document software systems, aiding understanding, maintenance, and updates.

Class, use case, sequence, activity, component, and deployment diagrams are common types. UML's value lies in providing standardized, clear representations, facilitating efficient communication, and ensuring a shared understanding of system architecture and behavior, ultimately enhancing planning, decision-making, and system development processes.

3.3.1 Data Flow Diagram

A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination. Data flowcharts can range from simple, even hand-drawn process overviews, to in-depth, multi-level DFDs that dig progressively deeper into how the data is handled.

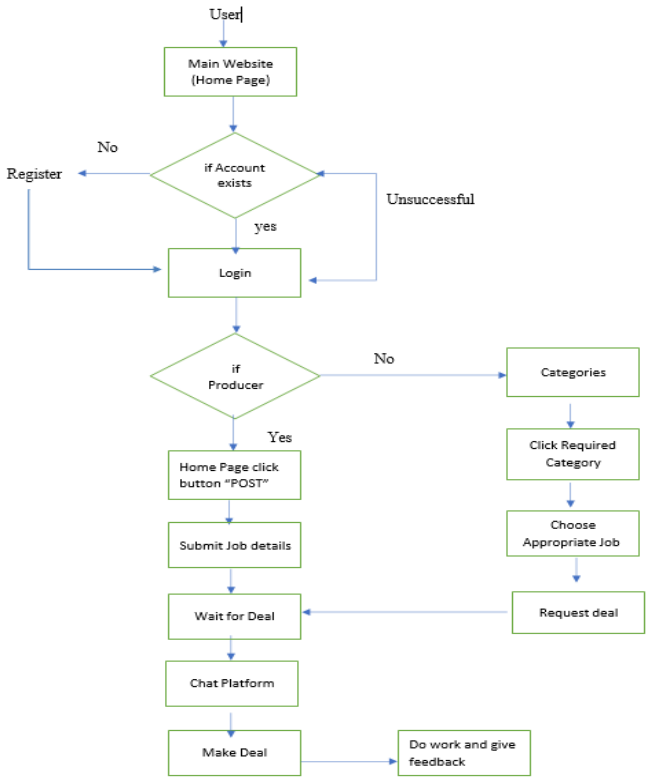


Fig 3.3.1: Data flow diagram of the project.

Fig 3.3.2 Home page

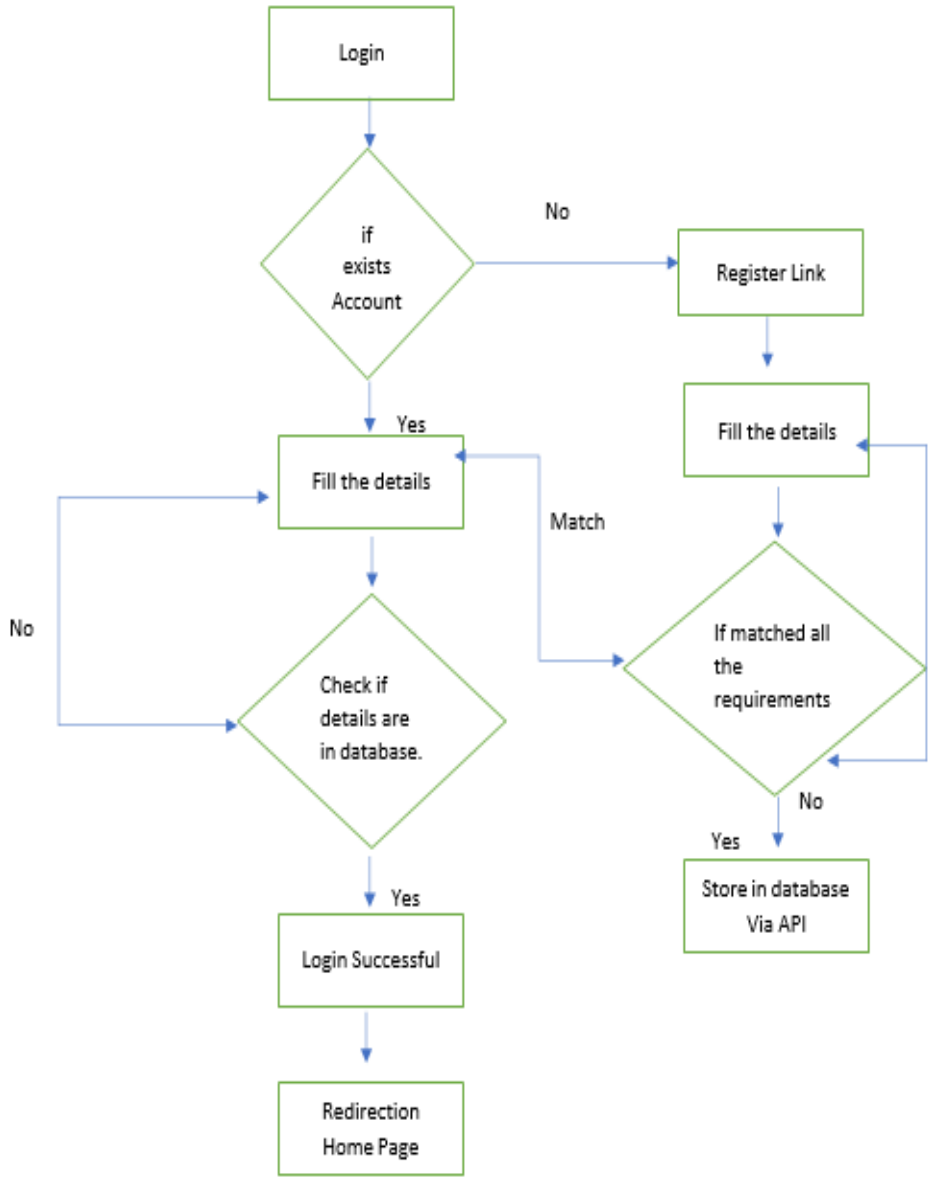


Fig 3.3.3 Login / Register

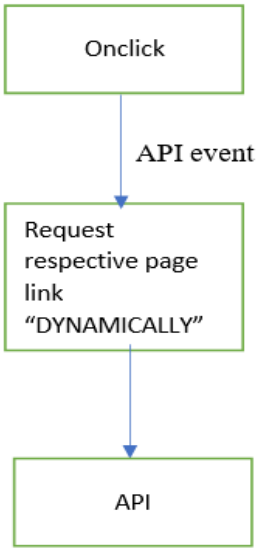
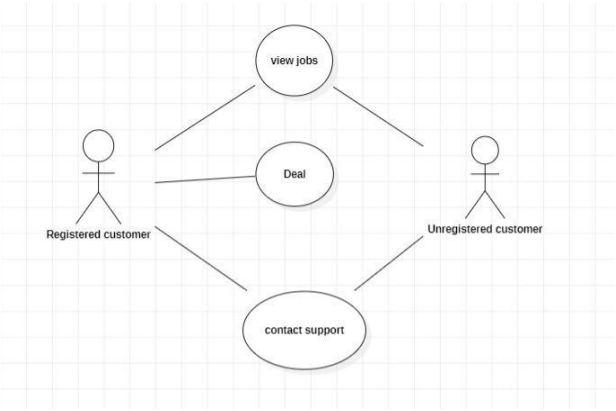


Fig 3.3.4 Categories Link

3.3.2 Use Case Diagram

Use case diagrams specify the events of a system and their flows. But use case diagram never describes how they are implemented. Use case diagram can be imagined as a black box where only the input, output, and the function of the black box is known.

These diagrams are used at a very high level of design. This high-level design is refined again and again to get a complete and practical picture of the system. A well-structured use case also describes the pre-condition, post condition, and exceptions. These extra elements are used to make test cases when performing the testing.



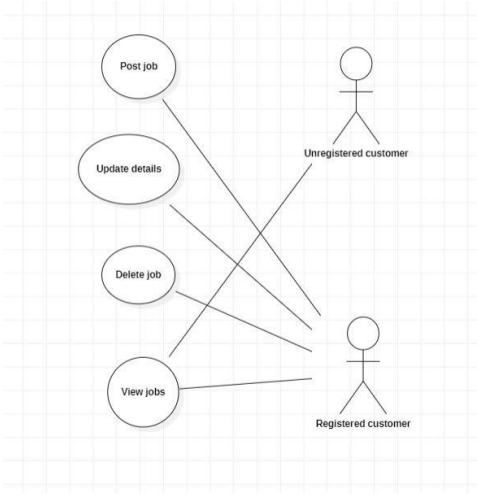


Fig 3.3.5: Use Case Diagram

3.3.3 Class Diagram

Class diagram is a static diagram and it is used to model the static view of a system. The static view describes the vocabulary of the system.Class diagram is also considered as the foundation for component and deployment diagrams.

Class diagrams are not only used to visualize the static view of the system but they are also used to construct the executable code for forward and reverse engineering of any system.Generally,UML diagrams are not directly mapped with any object - oriented programming languages but the class diagram is an exception.

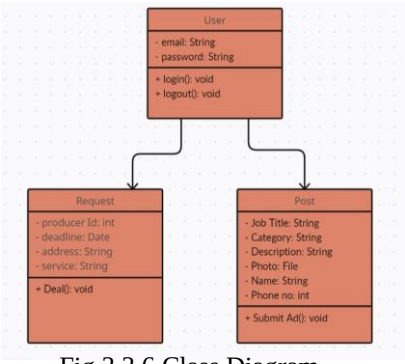


Fig 3.3.6 Class Diagram

4. TECHNOLOGY STACK

A technology stack, often referred to as a tech stack, is a collection of programming languages, software tools, frameworks, libraries, servers, databases, and other technologies used by developers to build and operate a web or software application. It's essentially the combination of technologies and tools employed to develop a particular project.

4.1 HTML5

HTML5, or Hypertext Markup Language version 5, is the latest standard for structuring and presenting content on the World Wide Web. It is a core technology of the Internet and is widely used for creating web pages and web applications. HTML5 introduces several new features and enhancements over its predecessor, HTML 4.01, and provides better support for multimedia, offline capabilities, and improved semantics.

Key features of HTML5:

- HTML5 is a remarkably powerful structuring language that is used in a wide variety of application domains.
- Some of its key distinguish features include
- Semantic elements
- Multimedia support
- Offline web applications
- Local storage
- Web storage
- Geolocation API
- Web workers and web socket

4.2 CSS3

CSS3, or Cascading Style Sheets level 3, is the latest evolution of the CSS language used to style and format the visual presentation of web pages. CSS is a fundamental technology in web development and works in conjunction with HTML to define the

layout, colours, fonts, and other stylistic aspects of a web page. CSS3 introduces several new features and enhancements over CSS2, providing web developers with more flexibility and creative control.

Key Features of CSS3:

- Selectors.
- Box model enhancements.
- Flexible box layout.
- Grid layout.
- Transitions and transformations.
- Animations.

4.3 JAVASCRIPT:

JavaScript is a versatile programming language hat is primarily used for creating dynamic content on the web. As one of the core technologies of web development, JavaScript enables developers to build interactive and responsive user interfaces.

Key features of JavaScript:

- Client-side scripting
- ECMA script
- Variables and datatypes
- Browser compatibility
- DOM manipulation
- Events
- Asynchronous programming
- JSON and AJAX
- Frameworks and libraries

4.4 NODE.JS:

Node.js is a JavaScript runtime built on the V8 JavaScript engine, and it allows developers to execute javaScript code on the server side. It enables the development of scalable and high-performance web applications by utilizing an event-driven, non-blocking I/O model.

Key features of Node.js:

- Event driven architecture.
- NPM (Node Package Manager).
- Non-blocking IO.
- Common JS modules.
- Single threaded, Event loop architecture.
- Microservices architecture.

4.4.1 EXPRESS

Express.js is a popular web application framework for Node.js, designed to simplify the process of building robust and scalable web applications and APIs. It provides a set of features and tools to streamline common web development tasks, making it an excellent choice for both beginners and experienced developers.

Key features of Express:

- Middleware, Routing.
- Error handling.
- Template engines.
- Static file serving.

4.4.2 BODY PARSER

Body-parser is a middleware module for Express.js, a web application framework for Node.js. The primary purpose of body-parser is to parse the incoming request bodies

in middleware before your handlers, making it easier to work with data submitted by clients through forms or other means.

Key features of BodyParser:

- Parsing form data.
- Parsing JSON data.
- Handling various content types.
- Raw body access.
- Limiting request size.

4.4.3 MONGOOSE

Mongoose is an Object Data Modelling (ODM) library for MongoDB and Node.js. It provides a higher-level, schema-based abstraction over the MongoDB Node.js driver, making it easier to work with MongoDB databases in a Node.js application. Mongoose allows developers to define schemas, models, and queries in a way that is more reminiscent of traditional relational databases.

Key features of Mongoose:

- Schema definition.
- Model creation.
- Connecting to MongoDB.
- CRUD operations (Create, Read, Update, Delete).

4.4.4 JSON WEB TOKEN

JSON Web Token (JWT) is a compact, URL-safe means of representing claims to be transferred between two parties. This information can be verified and trusted because it is digitally signed. JWTs are commonly used for authentication and authorization purposes in web development.

4.4.5 PATH

The term "path" in the context of Node.js typically refers to the built-in **path** module, which is used for working with file paths and directories. It provides utilities for handling and manipulating file paths in a platform-independent manner.

4.5 MONGODB:

MongoDB is a widely used NoSQL (Not Only SQL) database management system that provides a flexible, scalable, and document-oriented approach to data storage. Developed by MongoDB, Inc., it is designed to handle large volumes of unstructured or semi-structured data.

Key features of MongoDB:

- Document Oriented database.
- Horizontal scalability.
- Aggregation framework.
- Query language.
- Schema less design.

4.5.1 MONGO COMPASS

MongoDB Compass is the official graphical user interface (GUI) for MongoDB, provided by MongoDB,Inc. It is designed to simplify the process of interacting with MongoDB databases, allowing developers and administrators to visually explore and manipulate their data.

Key features of MongoCompass:

- GUI for MongoDB.
- Explore and query data.
- Schema analysis.
- Realtime index management.
- Data validation rules.
- Document validation and connection management.

5. CODING/CODE TEMPLATES

5.1 SOURCE CODE:

1. **Importing libraries**

```
const express = require('express');
const app= express();
const bodyParser = require('body-parser');
const mongoose  = require('mongoose');
const multer = require('multer');
const path=require('path');
const jwt = require("jsonwebtoken");
```

2. **Developing Home page**

```
<section class="home">
            <div class="container-fluid background sec1" id="page1">
              <div class="mainBox">
                <div class="first-half">
                  <p class="bigText">JOBS AND FACILITIES</p>
                  <p class="smallText">Find the local businesses near you and apply
for jobs</p>
                  <div class="buttons">
                    <button                                    class="postB"
onclick='window.location="submitJob.html"'>Post a
                      business</button>
                  </div>
                </div>
                <div class="second-half">
                  <img src="group-clipart-handshake-8.png" alt="logo">
                </div>
              </div>
            </div>
```

```
</section>
```

3. **Developing Home page- Navigation bar**

```
<nav class="navbar navbar-expand-lg navbar-light bg-light fixed-top">
        <a class="navbar-brand" href="#"></a>
    <button class="navbar-toggler ml-auto" type="button" data-toggle="collapse" data-target="#navbarNavAltMarkup"aria-controls="navbarNavAltMarkup"aria-expanded="false" aria-label="Toggle navigation">
        <span class="navbar-toggler-icon"></span>
        </button>
        <div class="collapse navbar-collapse" id="navbarNavAltMarkup">
          <div class="navbar-nav">
            <a class="nav-link active" href="#sectionHome">Home<span class="sr-only">(current)</span></a>
            <a class="nav-link" href="login.html">Login/Register</a>
          </div>
        </div>
    </nav>
```

4. **Categories**

```
<div>
        <ul class="catogriesRow">
          <div class="col-12 col-md-4 col-lg-3 mb-5">
            <li>
        <a href="/trial"><img src="group-clipart-handshake-8.png" alt="" id="all"></a>All locals
            <script>document.getElementById("all").addEventListener("click", function(event){
              event.preventDefault()
              fetchJobs()
            });</script>
            </li>

            <li>
          <a href="/teachers"><img src="images/teacher.jpg" alt="" id ="teacher"></a>Teaching
            <script>
```

```
                    document.getElementById("teacher").addEventListener("click",
function(event){
                    event.preventDefault()
                        searchTeacher()});
                    </script>
                </li>
            </div>
            <div class="col-6 col-md-4 col-lg-3 mb-5">
                <li>
                    <ahref="/electrician"><imgsrc="images/electrician.jpeg"alt=""
                        id="electrician"></a>Electrician
                    <script>
                    document.getElementById("electrician").addEventListener("click",
function(event){
                    event.preventDefault()
                    searchElectrician()});
                    </script>
                </li>
                <li>
                    <a href="/driving"><img src="images/driver.png" alt=""
id="driver"></a>Drivers
                    <script>
                        document.getElementById("driver").addEventListener("click",
function(event){
                    event.preventDefault()
                    searchDriver()});
                    </script>
                </li>
            </div>
            <div class="col-6 col-md-4 col-lg-3 mb-5">
                <li>
                    <a href="/repair"><img src="Images/rpair.jpeg" alt=""
id="repair"></a>Repairs
                    <script>
                        document.getElementById("repair").addEventListener("click",
function(event){
                    event.preventDefault()
                    searchRepair()});
                    </script>
                </li>
                <li>
                    <ahref="/carpentry"><imgsrc="Images/carpentry.png"alt=""
```

```
                id="carpentry"></a>Carpentry
                    <script>
                    document.getElementById("carpentry").addEventListener("click",
function(event){
                    event.preventDefault()
                    searchCarpentry()});
                    </script>
                  </li>
                </div>
                <div class="col-6 col-md-4 col-lg-3 mb-5">
                  <li>
          <a       href="/plumber"><img       src="Images/plumbing.png"       alt=""
id="plumber"></a>Plumbing
                    <script>
                    document.getElementById("plumber").addEventListener("click",
function(event){
                    event.preventDefault()
                    searchPlumber()});
                    </script>
                  </li>
                </div>
              </ul>
            </div>
```

5. **Creating login page**

```
<div class="login">
      <h1 class="heading">login</h1>
      <form action='/login' method='post'>
     <input type="email" placeholder="email" autocomplete="off" class="email"
id="email"              name="email"
       required>
      <input      type="password"      placeholder="password"      autocomplete="off"
class="password" id="password"
        name="password" required>
      <button type="submit" class="submit-btn">log in</button>
    </form>
      <a href="register.html" class="link">don't have an account? Register one</a>
    </div>
```

Javascript:

```
app.post('/login', async (req, res) => {
  try {
    const email = req.body.email;
    const password = req.body.password;
   // console.log(db.collection('users'));
    // Perform further validation or authentication logic
    const user=await User.findOne({email: email});
    console.log(user);
         if(user===null){
            console.log('email does not exist');
         }
        // else if (err) throw(err);
         if(user.password===password){
            req.session.user = user;
            let token = jwt.sign(
              {
               userId: user._id.toString(),
               organization: "FunctionUp",
              },
              "My$Secret",
              { expiresIn: "24h" }
            );

           return res.redirect('/');
         }
         else{
            console.log('password is incorrect,email:',email,'password:',password);
            res.send("password is incorrect");
         }
    //console.log({'email': email, 'password': password});

  } catch (error) {
    console.log({ error });
  }
});
```

6.Creating register page

```
<div class="registerForm">
      <h1 class="heading">Register</h1>
      <form action='/register' method="POST">
      <input type="text" placeholder="username*" autocomplete="off" class="name"
   name="username" required>
      <input type="email" placeholder="email*" autocomplete="off" class="email"
   name="email" required>
      <input type="password" placeholder="password*" autocomplete="off"
   class="password"
   name="password" required>
      <input type="password" placeholder="confirm password*" autocomplete="off"
   class="password" name="confirmpassword" required>
      <button class="submit-btn" type="submit">register</button>
   </form>
      <a href="login.html" class="link">already have an account ? log in here</a>
   </div>

app.post('/register', async (req, res) => {
   try {
     const password = req.body.password;
     const cpassword = req.body.confirmpassword;
    // console.log(req.body.username,password,cpassword)
     //res.send(req.body.username);
     // Perform further validation or authentication logic
       if(password===cpassword){
          const registerUser = new User({
             name: req.body.username,
             email: req.body.email,
             password: req.body.password,
             confirmpassword: req.body.confirmpassword

          })

         const registered=await registerUser.save();
         console.log(registered);
         res.redirect('login.html')
       }
       else{
          console.log("passwords do not match");
          res.send("passwords do not match");
```

```
      }
     //console.log({'email': email, 'password': password , 'name': name});
   } catch (error) {
    console.log({ error });
   }
 });
```

7.Job submission

```
app.get('/submitJob',auth.isLogin, async(req, res) => {
   let conn=await connect();
   //return res.redirect('submitJob.html');
   res.send('submitJob.html');
 })
 app.post('/submitJob',upload.single('UploadImage'), async (req, res) => {
     try{
      const jobsubmit = new Job({
           ownername: req.body.ownerName,
           jobOrbusiness: req.body.JobTitle,
           category: req.body.Category,
           jobDescription: req.body.JobDescription,
           images:'uploads/'+req.file.originalname,
           BNumber: req.body.BNumber,
           location: req.body.location
       })
       console.log(req.body.ownerName,
    req.body.JobTitle,
           req.body.Category,
            req.body.JobDescription,
          req.body.UploadImage,
            req.body.BNumber,
           req.body.location)


     const jobSubmitted=await jobsubmit.save();
     console.log(jobSubmitted);
     res.redirect('index.html')
    // const duplicateDocuments = await Job.find({ name: null }).exec();
    // res.redirect('/')
```

```
    }
    catch (error) {
      console.log({ error });
    }
  });
```

8.MongoDB connection

```
mongoose.connect("mongodb+srv://sayali21joshi:BSOInyNN6usdpMQB@cluster0.vo
    ozemb.mongodb.n
   et/miniProject",{
   useNewUrlParser: true,
   useUnifiedTopology: true,
}).then(
   () => {
      console.log("database connected");
   }
).catch(
   (err) => {
      console.log(err);
   }
);
```

9.Database – User schema

```
const mongoose= require('mongoose');
const userSchema=new mongoose.Schema({
   name:{
      type:String,
      required:true
   },
   email:{
      type:String,
      required:true,
      unique:true
   },
   password:{
      type:String,
      required:true
   },
   confirmpassword:{
      type:String,
```

```
        required:true
    },

})

const User=mongoose.model('User',userSchema);

module.exports=User;
```

10.Database – Job schema

```
const mongoose= require('mongoose');
const requestSchema=new mongoose.Schema({
    userId: { type: ObjectId, ref: "User", required: true },
    request: [
      {
        producerId: { type: ObjectId, ref: "Producer", required: true },
        deadline: { type: Date, required: true },
        adrs: { type: String, required: true },
        service: { type: String, required: true },
        _id: false,
      },
    ],

    status: {
      type: String,
      default: "pending",
      enum: ["pending", "completed", "cancled"],
    },
   requestedOn: { type: Date },
  },
  { timestamps: true }
);

const Request=mongoose.model('Request',requestSchema);
module.exports=Request;
```

11.Database – Request schema

```
const mongoose= require('mongoose');
const requestSchema=new mongoose.Schema({
    userId: { type: ObjectId, ref: "User", required: true },
    request: [
```

```
      {
        producerId: { type: ObjectId, ref: "Producer", required: true },
        deadline: { type: Date, required: true },
        adrs: { type: String, required: true },
        service: { type: String, required: true },
        _id: false,
      },
    ],

    status: {
      type: String,
      default: "pending",
      enum: ["pending", "completed", "cancled"],
    },
   requestedOn: { type: Date },
  },
  { timestamps: true }
);

const Request=mongoose.model('Request',requestSchema);
module.exports=Request.
```

6. TESTING

6.1 TESTING STRATEGIES

Testing is the process of detecting errors. Testing performs a very critical role for quality assurance and for ensuring the reliability of software. The results of testing are used later even during maintenance also.

6.2 TESTING OBJECTIVES

The main objective of testing is to uncover a host of errors, systematically and with minimum effort and time. Stating formally, we can say,

➢ Testing is a process of executing a program with the intent of finding an error.

➢ A successful test is one that uncovers a yet undiscovered error.

➢ A good test case is one that has a high probability of finding error, if it exists.

➢ The tests are inadequate to detect possibly present errors.

➢ The software more or less confirms the quality and reliable standards.

6.3 LEVELS OF TESTING (test cases)

6.3.1 Unit Testing

Unit testing focuses verification effort on the smallest unit of software i.e., the module.

Scope: Individual units or components (functions, methods, modules) of the system.

Purpose: Verify the correctness of each unit's functionality in isolation.

Examples: Testing login authentication functions, database interactions for user registration, validation of service request creation.

Unit testing done in our project :

1. Login and Register Pages:

Scenarios:

- Testing if user credentials (username, password) are validated correctly during login.

Validate successful login with correct credentials and failure with incorrect ones.

- Testing registration by creating new user accounts and checking if they are stored correctly in the database.

1. Positive Test - Successful Login:

Scenario: Test if a user can successfully login with valid credentials.

Test Steps:

- Enter valid username and password.
- Click the login button.
- Verify that the user is redirected to the homepage.

Ex:entering details of registered user lee@gmail.com and correct password

Result: User successfully logged in and access the homepage.

2. Negative Test - Invalid Credentials:

Scenario: Test login with invalid username or password.

Test Steps:

- Enter an invalid username or password.
- Click the login button.
- Verify the error message or prompt.

Ex:entering details of registered user lee@gmail.com and incorrect password

Result: System displays an error message indicating invalid credentials.

3. Negative Test - Empty Fields:

Scenario: Verify handling of exceptional cases like empty input fields.

Test Steps:

- Keep the username or password field empty.
- Click the login button.
- Verify the error message or prompt.

Result: System shows users to fill in both username and password fields.

2. Homepage:

Scenarios:

Load and Display:

1.Positive Test - Successful Page Load:

Scenario: Ensure the homepage loads without errors.

Test Steps:

- Access the homepage URL.

- Verify that the homepage loads within an acceptable time frame.
- Check for the presence of essential elements (header, navigation, footer).

Result: Homepage loads without errors, displaying essential elements.

Navigation:

1.Positive Test - Navigation Links:

Scenario: Test functionality of navigation links on the homepage.

Test Steps:

- Click on navigation links (categories).
- Verify redirection to the respective pages without errors.

Result: Navigation links direct users to the correct pages without any issues.

2.Negative Test - Broken Links:

Scenario: Test for broken or incorrect navigation links.

Test Steps:

- Attempt to access non-existent or broken URLs through navigation links.
- Verify the behavior of the system.

Result: System should handle non-existent or broken links gracefully (e.g., display a 404 error page).

3. Listing Local Businesses:

Scenarios:

Business Listing Functionality:

1.Positive Test - Successful Business Addition:

Scenario: Verify the functionality of listing businesses.

Test Steps:

- Add a new business listing with valid details (name, contact, address,job description,photos etc).
- Retrieve the added business from the database.

Result: Business successfully added and retrievable with accurate details.

2.Negative Test - Error Handling for Business Addition:

Scenario: Test error handling during business addition.

Test Steps:

- Attempt to add a business with invalid or missing details.
- Check for error messages or system response.

Result: System prompt for valid details and handle errors appropriately.

Data Consistency and Integrity:

1.Positive Test - Database Integrity:

Scenario: Ensure consistency between displayed and stored business data.

Test Steps:

- Verify that the displayed information matches the stored data in the database for added businesses.

Result: Information displayed on the website matches the details stored in the database.

2.Negative Test - Data Mismatch Verification:

Scenario: Test for data mismatches or inconsistencies between displayed and stored data.

Test Steps:

- Introduce discrepancies in stored data (e.g., incorrect contact details,not a unique address,not unique phone no.).
- Verify the displayed information against the expected data.

Result: System handles data inconsistencies and display accurate information.

Sorting and Filtering Options:

1.Positive Test - Sorting Businesses:

Scenario: Test sorting options for businesses.

Test Steps:

- Sort businesses by various criteria (e.g., alphabetical, rating, proximity).
- Check the arrangement of displayed businesses.

  Ex : click on all jobs logo in categories and check if all are filtered properly.

Result: Businesses are displayed in the correct order based on the selected sorting criterion.

2.Positive Test - Filtering Businesses:

Scenario: Verify the functionality of filtering businesses.

Test Steps:

- Apply different filters (e.g., category, location) to display specific businesses.
- Verify that only relevant businesses are displayed according to the applied filters.

Result: Displayed businesses should match the specified filter criteria.

4. Service Requests:

Scenarios:

Request Submission:

1.Positive Test - Successful Service Request:

Scenario: Ensure a user can successfully submit a service request to a business.

Test Steps:

- Log in as a registered user.
- Navigate to the deal on searching required local business service..
- Fill in valid details for the deal..
- Submit the request.

Result: The request is successfully submitted without errors and is visible in the user's profile.

2.Negative Test - Invalid Request Submission:

Scenario: Test the system's response to invalid or incomplete service request submissions.

Test Steps:

- Attempt to submit a service request with missing or invalid information.
- Verify system response (error message or handling).

Result: System prompts the user to provide necessary details and does not allow submission with incomplete or invalid data.

Request Processing:

1.Positive Test - Business Receives Request:

Scenario: Verify that businesses receive service requests from users.

Test Steps:

- Submit a deal as a user.
- Check the backend or database to confirm the request's presence from the business perspective i.e if it is in requests database.

Result: The deal request is successful and it is correctly stored in the system.

2.Negative Test - Failed Request Processing:

Scenario: Test system behavior if the request fails to reach the business or is not stored correctly.

Test Steps:

- Submit a service request as a user.
- Verify the backend or database for the absence or incorrect storage of the request.

Result: System fails to store or process the service request, indicated by missing or incorrect data in the backend.

Request Status and Tracking:

1.Positive Test - Tracking Service Request Status:

Scenario: Test user ability to track the status of their service requests.

Test Steps:

- Submit a service request as a user.
- Navigate to the user's profile or service requests section.
- Check for status updates or tracking options for the submitted request.

Result: Users can view the status (e.g., pending, accepted, completed) of their service requests.

2.Negative Test - Incorrect Request Status or Tracking:

Scenario: Test for incorrect or missing service request status updates.

Test Steps:

- Submit a service request as a user.
- Verify the user profile or service requests section for status updates.

Result: System inaccurately displays or fails to update the service request status.

5. Profile and Service Listing:

Scenarios:

Profile Management:

User Profile Display:

1.Positive Test - Profile Information Display:

Scenario: Ensure that user profile information displays correctly.

Test Steps:

- Access a user's profile page.
- Verify that user details (name, email, etc.) are displayed accurately.

Result: User profile information is correctly displayed without any discrepancies.

2.Negative Test - Unauthorized Access:

Scenario: Test unauthorized access to user profiles.

Test Steps:

- Attempt to access another user's profile without appropriate permissions.
- Verify system behavior or error message.

Result: System should restrict unauthorized access and display an appropriate error message.

Service Request Listing:

1.Positive Test - Display of Service Requests:

Scenario: Validate if service requests made by users are correctly listed in their profiles.

Test Steps:

- Create service requests from a user account.
- Access the user's profile and verify the displayed service requests.

Result: Service requests made by the user are accurately displayed in their profile.

2.Negative Test - Empty Service Requests:

Scenario: Test handling of empty service request listings.

Test Steps:

- Access a user profile with no service requests made.
- Verify the behavior of the system.

Result: System should handle empty listings gracefully, displaying appropriate messages.

Service Listing Functionality:

Service Listing:

1.Positive Test - Successful Service Listing:

Scenario: Ensure that the service request details are correctly stored in the database.

Test Steps:

- Request services from businesses.

- Access the database or backend to verify the storage of service request details.

Result: Service request details are accurately stored in the database or backend.

2.Negative Test - Unsuccessful Service Requests:

Scenario: Test unsuccessful service requests handling.

Test Steps:

- Submit service requests with invalid or missing information.
- Check for error messages or system behavior.

Result: System should handle invalid service requests and provide appropriate feedback to users.

6.3.2 Integration Testing:

After the unit testing, we have to perform integration testing. The goal here is to see if modules can be integrated properly, the emphasis being on testing interfaces between modules.

Scope: Interaction between integrated units or components.

Purpose: Verify the interaction and data flow between units to ensure they work together seamlessly.

Examples: Testing the integration between user authentication and user profile creation, verifying interactions between business listings and service requests.

Integration testing done in our project :

1. Login and Registration Integration Testing:

Scenario:

- Test the interaction between the login and registration components with the database and session management.

2. Homepage Integration Testing:

Scenario:

- Verify interactions between the homepage components and the backend services fetching dynamic content or featured sections.

3. Business Listing and Service Request Integration Testing:

Scenario:

- Test the interaction between the business listing component and the service request feature.
- Ensure service requests are correctly linked and sent to the respective businesses listed on the website.

4. User Profile and Service Listing Integration Testing:

Scenario:

- Verify the interaction between the user profile and service listing components.
- Test if the service requests listed in a user's profile correspond to the requests sent by the user.

Sample Integration Test Scenarios:

Login and Profile Update:

- Test if user profile details are updated after successful login or registration.

Service Request and Business Interaction:

- Verify if a service request is properly linked and sent to the corresponding business listed on the website.

6.3.3 System Testing

System testing involves examining the entire software application to ensure it meets the project requirements.Both unit testing and integration testing mentioned above comprise in system testing.

6.3.4 White Box Testing

This is a unit testing method where a unit will be taken at a time and tested thoroughly at a statement level to find the maximum possible errors. Test has been done step wise for every piece of code, taking care that every statement in the code is executed at least once. The white box testing is also called Glass Box Testing.

- **Backend Functionality Testing -** User Authentication:

Objective: Validate the backend code handling user login/authentication.

Test different scenarios (valid credentials, invalid credentials, null inputs) to ensure all code paths are executed, covering conditions, loops, and error-handling statements.

- **Database Interaction** - Data Validation:

Objective: Verify how data validation is performed before storing information in the database.

Directly assess the backend code responsible for data validation, ensuring that all validation rules are rigorously tested.

- **Error Handling** - Frontend and Backend Integration:

Objective: Assess error-handling mechanisms in the frontend and backend integration.

Force errors by providing incorrect inputs or manipulating data to observe how the system responds and handles exceptions at code levels.

By applying white box testing methodologies like statement-level testing, code coverage assessments, and thorough validation of internal code pathways, the aim is to expose errors and vulnerabilities within the system, ensuring robustness and reliability in the codebase.

6.3.5 Black Box Testing

This testing method considers a module as a single unit and checks the unit at interface and communication with other modules rather getting into details at statement level. Here the module will be treated as a block box that will take one input and generate output.

Black Box Testing Examples:

- **User Interface Testing** - Login Page:

Objective: Evaluate the login page functionality without examining the backend code.

Input valid and invalid credentials, observing if correct login grants access and incorrect credentials result in denied access.

- **Business Logic** - Service Request Submission:

Objective: Test the service request submission process without examining the internal codebase.

Submit various types of service requests (different categories, details) to ensure proper acceptance and processing without knowing how the backend handles the requests.

- **Integration Testing** - Frontend and Backend Interaction:

Objective: Validate the interaction between frontend and backend systems.

Submit requests through the frontend interface and verify if corresponding actions are reflected in the user profile or database without inspecting the internal communication mechanisms.

- **Database Interaction** - Data Retrieval:

Objective: Check the correctness of displayed data without inspecting database handling code.

Validate if information retrieved from the database (like user profiles or business listings) corresponds accurately to the expected data displayed on the frontend.

- **Input Validation** - Error Handling

Objective: Test how the system handles incorrect inputs without examining the backend code.

Submit erroneous or incomplete data and observe how the system responds, ensuring it handles errors gracefully without revealing internal workings.

These tests aim to ensure that the system functions as expected from a user's perspective, irrespective of the underlying code complexities.

7. OUTPUT SCREENS

7.1 HOME PAGE

Fig 7.1 Home page

On the website's homepage, users have the option to log in to the site.

7.2 LOGIN PAGE

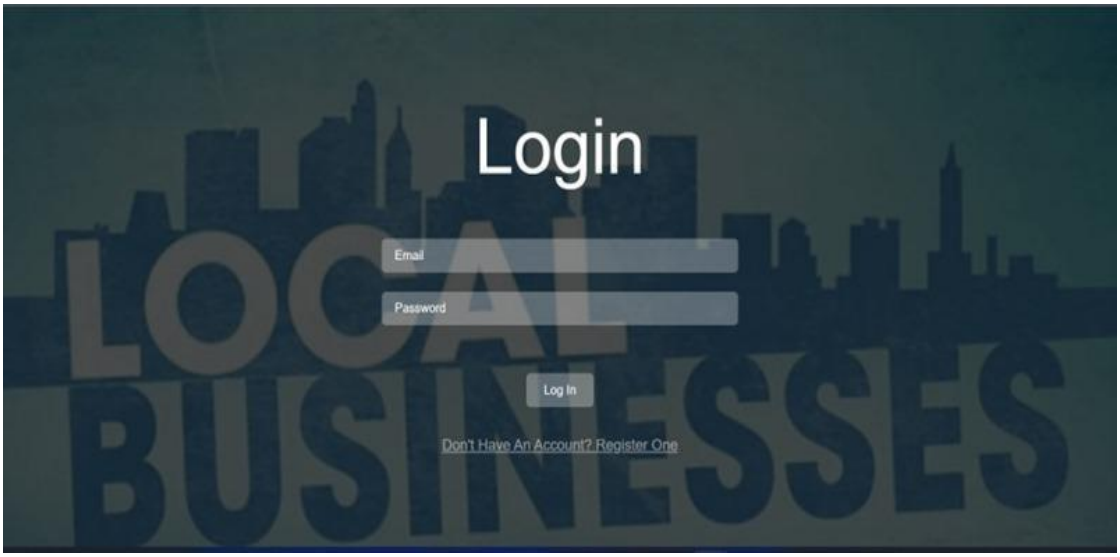


Fig 7.2 Login Page

On the login page, users can access the website by logging in. For new users, registration is required before logging in; without registration, access to the website is restricted. During login, users need to input their email and password (e.g., lee@gmail.com, pwd: 1234). If the provided credentials are incorrect, the website displays an "incorrect credentials" message.

7.3 REGISTER PAGE

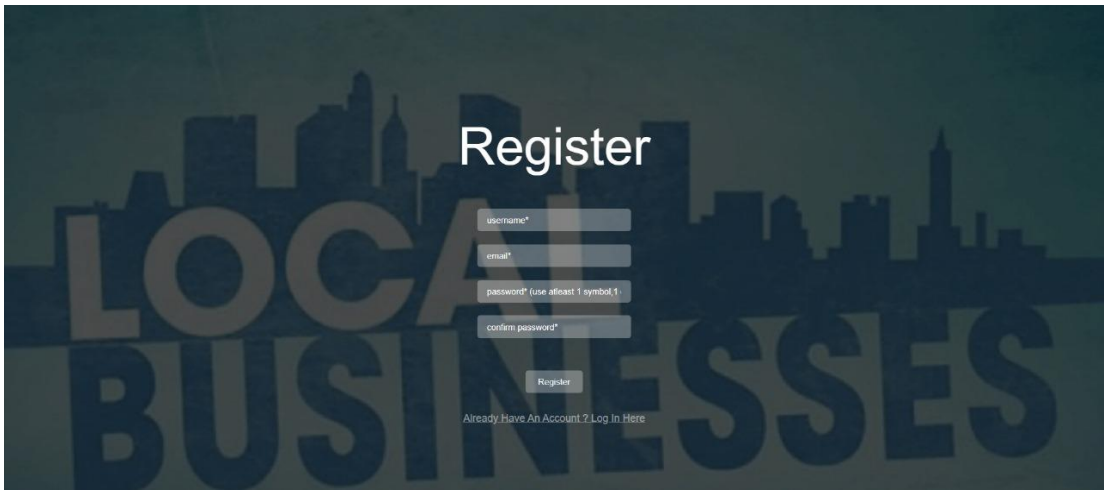


Fig 7.3 Register Page

On the registration page, new users are required to furnish their name, email details, and create a robust password for their account. A valid email ID must be provided during this process.

7.4 CATEGORIES

Fig 7.4 Categories

The website displays all available job categories. Users can search for specific categories based on the services they require.

7.5 SUBMIT A JOB

Fig 7.5 job selection

To post their business/job on the website, users must complete a form named "Submit a Job." This form requires details such as job role, job title, job description, category, name, phone number, and a photo. Users are expected to provide accurate and valid information.

7.6 JOBS

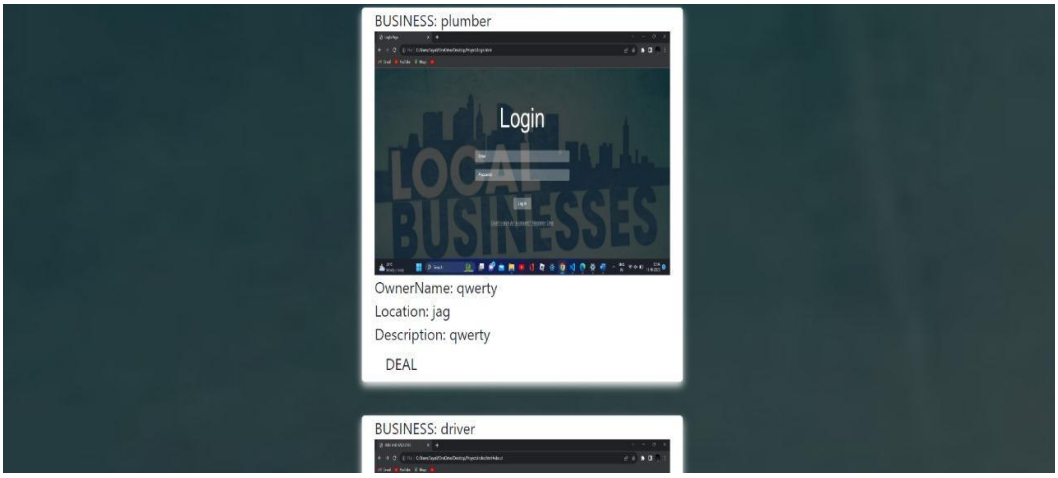


Fig 7.6 jobs

Upon selecting any category, the website displays all the businesses listed within that specific category. If users find a suitable category, they can click on the "deal" button to initiate a transaction or negotiate an agreement.

7.7 MONGODB DATABASE

The database resembles the image provided below. It comprises three collections, namely users, requests, and jobs.Each collection stores information in documents.

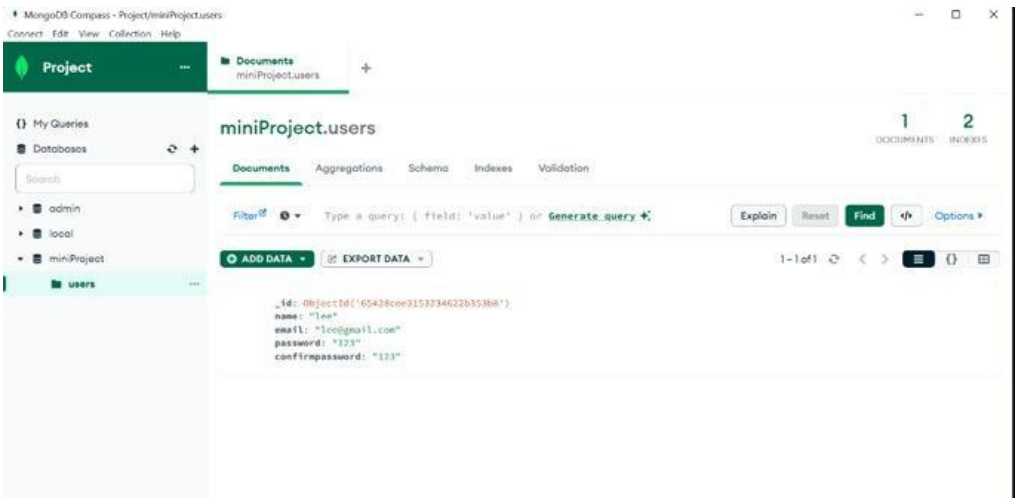


Fig 7.7 Database Mongodb

	_id ObjectId	name String	email String	password String	confirmpassword String
1	ObjectId('65428cee3153234622b…	"lee"	"lee@gmail.com"	"123"	"123"
2	ObjectId('654b6175f41f9f264fc…	"hello"	"hello@gmail.com"	"54321"	"54321"
3	ObjectId('655c8febfc927a1d9d5…	"sayali"	"lee21@gmail.com"	"09876"	"09876"
4	ObjectId('655d95a66766cebaf73…	"qwerty"	"qwerty@gmail.com"	"6543"	"6543"
5	ObjectId('655d99c3f1f9348fdf8…	"asdf"	"asdf@gmail.com"	"12345"	"12345"
6	ObjectId('655d9ffd271fa2caa89…	"hasini"	"hasinichukka9@gmail.com"	"098765"	"098765"
7	ObjectId('655da531756e61c05ab…	"sayalii"	"sayali21joshi@gmail.in"	"qwerty"	"qwerty"
8	ObjectId('655daece4c59398a65a…	"hasini"	"hasini@gmail.com"	"09876"	"09876"
9	ObjectId('655db3cbe0a2bedd303…	"bhavana"	"bhavana@gmail.com"	"1234"	"1234"

Fig 7.8 User's table

	d	ownername String	jobOrbusiness String	category String	jobDescription String	images String	BNumber String	location String
1	54690fda95f77d8061…	"qwerty"	"plumber"	"Plumbing"	"qwerty"	"uploads/Screenshot (8).png"	"123456876543"	"jag"
2	54691c7a95f77d8061…	"lee"	"driver"	"Driving"	"safe driver"	"uploads/Screenshot (7).png"	"1234567890"	"jntuhcej"
3	54691e7a95f77d8061…	"leeJ"	"electrician"	"Electrician"	"great experience"	"uploads/Screenshot (9).png"	"0987654321"	"jnuthucej"
4	5469238a95f77d8061…	"ytrewq"	"teacher"	"Teaching"	"Tuition teacher : math scien…	"uploads/Screenshot (5).png"	"5432167890"	"jnuthcej"
5	5473b4391097 4f4d57…	"asd"	"driver"	"Driving"	"customer-friendly"	"uploads/1699167043237-Screen…	"14235678906"	"jgtl"
6	550c636ffa8b658dae…	"qaz"	"carpenter"	"Carpentry"	"asdfghjk"	"uploads/Screenshot 2023-09-1…	"123456"	"kmnr"
7	560757a16634188eb9…	"qwertyu"	"teacher"	"Teaching"	"5 yrs exp"	"uploads/Screenshot 2023-11-1…	"98765432100"	"jntuhh"
8	5607d6c3b1b7056d40…	"1234567890"	"electrician"	"Electrician"	"qwertyuiopxfcgvj"	"uploads/team.png"	"7654321890"	"jggtl"
9	5607de23b1b7056d40…	"sayalii"	"repair"	"AutomobileRepair"	"asdfghjkl"	"uploads/Screenshot 2023-11-2…	"6305079024"	"jgtll"

Fig 7.9 Job's table

8.CONCLUSION

- The development of a well-organized website with secure login, seamless access, and a protected environment for both producers and consumers requires a comprehensive approach. By ensuring efficient performance, scalability, robust security, user-friendliness, and easy maintenance, the created system aims to provide a reliable and enjoyable experience for all users.
- In the realm of frontend development of this project, the creation of the user interface and user experience involves the utilization of HTML5, CSS3, and Javascript.
- Simultaneously, the backend development focuses on crafting the business logic(request, response), employing robust frameworks like Node.js.
- Establishing connections with databases and seamlessly integrating them into the backend logic for a cohesive and functional web application.
- Evaluating the website using the specified test cases outlined above. Employing the mentioned test cases to assess the functionality and performance of the website.

- In conclusion, our website aimed to bridge local businesses and users, incorporating features like user authentication, business listings, service requests, and secure database integration. The main focus was on seamless frontend and backend interaction, emphasizing data security, user engagement, and community impact. Through robust development, rigorous testing, and adherence to security measures, we successfully created a user-centric platform facilitating efficient service connections and fostering skill development in remote areas.

9. FUTURE ENHANCEMENTS

- The website could incorporate a chat platform to facilitate seamless communication between producers and consumers.
- The website has the potential to be transformed into a mobile application.
- The website could be accessible to a broader audience in extensive regions due to its scalability.
- User assistance should be improved.

10. REFERENCES

[1] 'Software Engineering A practitioner's Approach', Roger S Pressman, 6th Edition. McGrawHill InternationalEdition.

[2] 'Web Technologies and Applications' by Dr.P.Sammulal, Peddi Kishor,2nd Edition, BS Publications

[3] 'Beginning Node.js', Basarat Ali Syed,1st edition, Apress Berkeley,CA .

[4] Node Js website as a platform to download node js framework. : https://nodejs.org

[5] Freelancer as a reference to existing systems : https://www.freelancer.com

[6] Linkedin as a reference to existing systems. : https://in.linkedin.com

[7] Glassdoor app : https://www.glassdoor.co.in

[8] Geeks for Geeks as a reference to basics of front end and back end technologies : https://www.geeksforgeeks.org

[9]Github as a reference for sample projects https://github.com/yousufshaikh/react-vend-app

[10] w3schools as a reference to front end technologies. : https://www.w3schools.com

[11] Mongodb Website as a platform to download mongodb and as a reference to queries used in mongodb : https://www.mongodb.com/atlas

APPENDICES

1. **Abstract :** An abstract is a concise summary that provides an overview of the key points and findings of a research paper, thesis, or project.

2. **Acceptance testing:** The level of testing that is performed with realistic data to demonstrate that the software is working satisfactorily and meets the requirements of the client.

3. **Access:** Access in software refers to the permissions and rights granted to users or programs, determining their ability to interact with and manipulate data, resources, or functionalities within a system.

4. **Admin panel:** A user interface that allows administrators to manage and control a system or application.

5. **Analysis:** Analysis in software refers to the process of gathering and evaluating requirements, problems, and data in order to understand, identify, and define the needs and objectives of a software project, facilitating its successful development and implementation. It involves breaking down complex systems into smaller components to identify the problem areas and generate effective solutions.

6. **Animations:** Animations involve the dynamic and visually appealing movement or transition of elements, often used in web design and digital media for enhanced user experience.

7. **API:** API, or Application Programming Interface, is a set of rules and tools that allows different software applications to communicate with each other, enabling them to access and exchange data or functionality.

8. **Architecture:** Architecture in software refers to the high-level structure and organization of a system, encompassing design principles, components, and their interactions to achieve specific functionalities and goals.

9. **Asynchronous programming:** It is a technique that enables your program to start a potentially long-running task and still be able to be responsive to other events while that task runs, rather than having to wait until that task has finished. Once that task has finished, your program is presented with the result.

10. **Authentication:** Authentication is the process of verifying the identity of a user, device, or system, often through the use of credentials such as usernames and passwords.

11. **Authorization:** Authorization is the process of granting or denying access rights and permissions to a user, device, or system based on their authenticated identity and predefined rules.

12. **Back-end Development:** The server-side of web development, where the server, database, and application logic reside. Back-end developers focus on building the infrastructure that supports the front-end and enables the functionality of a web application.

13. **Best practices:** Industry-recognized methods or techniques that are considered the most effective or efficient approach to a particular task or process.

14. **Black box testing:** It is a functional testing that relies on input/output behaviour of the system.

15. **BodyParser:** A middleware module for Express.js that parses the incoming request bodies, making it easier to work with client-submitted data.

16. **Client:** A client is an individual, organization, or software application that seeks services, resources, or information from a server or service provider in a networked environment.

17. **Client-side scripting:.** The execution of scripts, such as JavaScript, on the client-side (browser) to enhance web page interactivity.

18. **Compatibility:** Compatibility refers to the ability of different systems, software, or components to work together smoothly and interchangeably without issues.

19. **Compliance:** Adherence to laws, regulations, and standards related to data privacy and security.

20. **Content creation and management:** Creating and editing articles, posts, or other forms of content.

21. **CSS3:** The latest version of Cascading Style Sheets used to style and format the visual presentation of web pages, introducing new features and enhancements for more flexibility and creative control.

22. **Data Flow Diagram (DFD):** A visual representation of information flow within a system, showing the inputs, outputs, storage points, and routes between each destination.

23. **Database:** A database is a structured collection of organized data that can be easily accessed, managed, and updated. It serves as a centralized repository for storing and retrieving information in a systematic and efficient manner.

24. **Database Integration:** The process of connecting the backend of an application with a database system, allowing for efficient data storage, retrieval, and management.

25. **Dynamic:** Dynamic refers to the quality of constant change, activity, or progress, often associated with systems, processes, or environments that evolve over time.

26. **ECMA script:** It is also known as JavaScript, is a programming language adopted by the “European Computer Manufacturer’s Association” as a standard for performing computations in Web applications.

27. **Errors:** Mistakes or flaws in software that can cause it to behave incorrectly or not as intended.

28. **Express.js:** A popular web application framework for Node.js, providing features and tools to simplify the process of building robust and scalable web applications and APIs.

29. **Flexibility:** Flexibility refers to the ability of a system or object to adapt, bend, or adjust easily to changing conditions or requirements. It implies versatility and responsiveness to accommodate various needs or scenarios.

30. **Front-end Development:** The process of creating the user interface and user experience of a website or web application, focusing on the design and development of the visual elements that users interact with.

31. **Geolocation API:** It is used to locate user's position.

32. **Graphic card:** It is an expansion card for your PC that is responsible for rendering images to the display.

33. **GPU** : A Graphics Processing Unit (GPU) is a specialized electronic circuit that accelerates graphics rendering and performs parallel computations for various tasks beyond visuals.

34. **Implementation:** Implementation in software refers to the process of translating a design or specification into a working and executable program or system, involving coding, testing, and integrating various components to achieve the desired functionality. It involves the actual writing and assembly of the program code, as well as configuring and deploying the software on the intended platform or environment.

35. **Integration testing:** The level of testing that focuses on testing the integration and interaction between different modules of a software system.

36. **JavaScript:** A versatile programming language used for creating dynamic content on the web, enabling developers to build interactive and responsive user interfaces.

37. **JSON:** “JavaScript Object Notation” is an open data interchange format that is both human and machine-readable.

38. **JSON Web Token (JWT):** A compact and URL-safe means of representing claims between two parties, commonly used for authentication and authorization purposes in web development.

39. **Libraries:** Libraries in software are collections of pre-written code or routines that can be reused by various programs to perform common tasks, enhancing efficiency and reducing development time.

40. **Local Storage:** Local storage refers to a web browser's capability to store data on a user's device, allowing websites to save and retrieve information persistently. It provides a way for web applications to store data locally, improving performance and enabling offline functionality.

41. **Loop:** A loop in software is a control structure that repeats a set of instructions or statements until a specific condition is met, facilitating efficient iteration and automation in code.

42. **Middleware :** Middleware is software that enables one or more kinds of communication or connectivity between applications or application components in a distributed network.

43. **Module:** A specialized unit or component within a project that performs a specific task or functionality.

44. **MongoDB:** A widely used NoSQL database management system, designed to handle large volumes of unstructured or semi-structured data in a document-oriented approach.

45. **Mongo Compass:** The official graphical user interface for MongoDB, providing a visual tool for exploring, querying, and manipulating data in MongoDB databases.

46. **Multimedia:** Multimedia involves the integration of various forms of content, such as text, graphics, audio, and video, to create a rich and interactive user experience. It encompasses the combination of different media elements to convey information, entertain, or communicate effectively.

47. **Node.js:** A JavaScript runtime that allows developers to execute JavaScript code on the server-side, enabling the development of scalable and high-performance web applications through event-driven, non-blocking I/O.

48. **Objective:** Objective in software refers to an explicit and measurable goal that a software system aims to achieve. It helps define the purpose, scope, and desired outcomes of the software project.

49. **Parsing:** Parsing in software is the process of analyzing and interpreting structured data, often represented in a specific format or language, to extract meaningful information or perform further processing.

50. **Path:** The built-in module in Node.js used for working with file paths and directories in a platform-independent manner.

51. **Problem Statement:** A problem statement in software development is a concise description of an issue or challenge that needs to be addressed. It helps define the scope and goal of the project, guiding the development process towards finding a solution.

52. **Quality assurance:** A set of activities and processes performed during software development to ensure the quality and reliability of the software.

53. **Reliability:** The ability of software to perform consistently and accurately over a period of time.

54. **Requirements:** Requirements in software refer to the specific functionalities, behaviors, and constraints that a system or application should possess to meet the

needs and expectations of its stakeholders. They serve as a foundation for software development by defining what needs to be achieved and providing guidelines for design, implementation, and testing.

55. **Robust:** Robust in software refers to the ability of a system to withstand errors, handle unexpected inputs, and maintain stability under various conditions.

56. **Safety:** Safety relates to the prevention of accidents, errors, and failures that may impact the application's usability and user experience. It involves ensuring that the application is robust and resilient against any potential issues, such as handling input validation, error handling, and proper exception management.

57. **Scalability:** The ability of software to handle increasing amounts of data or users without impacting its performance.

58. **Search and filtering:** The ability to search for specific content and narrow down results based on criteria.

59. **Semantic elements:** HTML elements that provide additional meaning to the structure and content of a web page.

60. **Security:** Security is a critical aspect of web-based applications, and a Java Full Stack Developer must have a good understanding of security protocols and measures. They need to ensure that the application is secure against common attacks, such as SQL injection or cross-site scripting (XSS).

61. **Selectors:** Selectors in CSS are patterns that define which HTML elements a style rule should apply to. They enable the styling of specific elements or groups of elements on a web page.

62. **Semantics:** Semantics refers to the meaning and interpretation of words, symbols, or language constructs within a specific context. It explores the relationships and understanding of how language elements convey intended messages or information.

63. **Server:** A server is a computer or system that provides resources, services, or functionality to other devices or programs, typically over a network.

64. **System testing:** The level of testing that focuses on testing a software system as a whole to uncover errors and ensure its functionality.

65. **Syntax**: Syntax refers to the set of rules governing the arrangement and combination of words or symbols in a programming language or formal language. It ensures the correct structure and format for creating valid statements or expressions within a given system.

66. **Technical:** Technical refers to aspects or characteristics related to the practical application of scientific and mathematical knowledge, often involving specific skills and methods.

67. **Technology stack :**A technology stack refers to a set of software tools, programming languages, frameworks, libraries, databases, and other technologies used to develop and deploy a software application.

68. **Testing:** The process of detecting errors in software through the execution of a program with the intent of finding errors.

69. **Transitions and transformations:** CSS3 properties for animating and transforming elements, providing visual effects.

70. **Transformations:** Transformations in the context of web design or graphics involve modifying the size, position, or orientation of elements, often using CSS or similar technologies. They enable dynamic and visually engaging effects, enhancing the layout and presentation of digital content.

71. **Unemployment crisis:** The current situation of high unemployment rates and a lack of job opportunities.

72. **Usability testing:** Evaluating the ease of use, functionality, and user experience of software through testing with actual users.

73. **User authentication:** The process of verifying the identity of a user before granting access to a system or application.

74. **User data:** Information about individuals stored in the system**.**

75. **User profile management:** The ability to update personal information and preferences.

76. **User roles:** Different levels of access and permissions given to individuals using the system.

77. **Variables:** Variables in programming are symbolic names that represent stored values, allowing data manipulation and storage in a computer program.

78. **Verification:** Verification is the process of evaluating whether a system or component meets specified requirements or conditions, often through testing or inspection.

79. **Versatile:** Versatile refers to the ability of something or someone to adapt easily and perform effectively in various situations or roles.

80. **Web application:** A computer program accessed through a web browser.

81. **Web Development Tools:** Software applications used for creating websites.

82. **Web socket:** A communication protocol that provides full-duplex communication channels over a single TCP connection.