A black and white logo

Description automatically generated

**FLEXI WORK**

"Empowering Connections, Simplifying Work"

**TEAM MEMBERS**

SHIVANESH G

NEHA

TAARUN ADITHYA SK R23EM098

VIGNESH BG R23EQ132

SCRUM MENTOR

ATUL R

VIGNESH S

**Problem Statement**

Millions of people worldwide face the daily challenge of finding reliable and efficient workers for short-term tasks. Whether it’s a last-minute home renovation, a seasonal event, or emergency assistance, the current job market often falls short of meeting these urgent and specific needs. The lack of an organized, accessible platform often leaves individuals scrambling for help at critical times. For instance, finding a plumber on a weekend or a babysitter during a busy season can turn into a stressful and time-consuming ordeal. This problem is further compounded by issues like lack of trust, high costs, unverified service providers, and unreliable communication channels. As a result, individuals and businesses frequently struggle to connect with the right professionals, leading to wasted time, compromised outcomes, and heightened frustration.

This persistent gap highlights the need for a streamlined solution that connects people and businesses with qualified workers in real time, providing a flexible, secure, and efficient way to manage short-term and long-term job requirements. FlexiWork aims to address this pressing issue by bridging the gap between demand and supply for services and job opportunities, ensuring convenience, reliability, and peace of mind for all users.

**TECH STACK**

FlexiWork utilizes the **MERN stack** (MongoDB, Express.js, React, and Node.js) to build a scalable, efficient, and user-friendly platform for connecting users with skilled short-term workers.

**MongoDB (Database)**

**Role:** Stores user profiles, job requests, availability, and reviews in a flexible, scalable, schema-less format.

**Key Benefit:** Handles large data efficiently and allows easy horizontal scaling

**React (Frontend Library)**

**Role:** Builds the dynamic, responsive frontend for browsing, booking, and rating services.

**Key Benefit:** Modular, component-based architecture ensures a fast, interactive user experience on both mobile and desktop.

**Node.js (Backend Runtime Environment)**

**Role:** Provides the backend runtime, enabling fast, asynchronous processing of requests.

**Key Benefit:** Event-driven architecture ensures smooth, real-time interactions.

**Supporting Tools**

**Mongoose:** Simplifies MongoDB integration with schema-based models.

**JWT:** Secures user authentication.

**MongoDB atlas:** For database operation.

**Prerequisites**

Before setting up the project, ensure the following tools are installed:

* **Node.js**
* **MongoDB Atlas** (database management)
* **Git** (version control)
* **npm** (Node Package Manager)
* **Text Editor** (VS Code)

**CODE**

**🡪API to Add a Product**

app.post('/addproduct', async (req, res) => {

try {

const product = new JobDetails(req.body);

await product.save();

console.log("Saved:", product);

res.json({ success: true, message: "Product added successfully!" });

} catch (error) {

console.error("Error saving product:", error.message);

res.status(500).json({ success: false, message: error.message });

}

});

🡪**Backend on signup and login page**

const router = express.Router();

router.post('/signup', async (req, res) => {

const { email, password, confirmPassword } = req.body;

if (password !== confirmPassword) return res.status(400).json({ message: ' Passwords do not match' });

const existingUser = await User.findOne({ email });

if (existingUser) return res.status(400).json({ message: 'User already exists' });

const hashedPassword = await bcrypt.hash(password, 10);

const newUser = new User({ email, password: hashedPassword });

await newUser.save();

res.status(201).json({ message: 'User registered successfully' });

});

router.post('/login', async (req, res) => {

const { email, password } = req.body;

const user = await User.findOne({ email });

if (!user) return res.status(404).json({ message: 'User does not exist' });

const isPasswordValid = await bcrypt.compare(password, user.password);

if (!isPasswordValid) return res.status(401).json({ message: 'Invalid password' });

const token = jwt.sign({ id: user.\_id }, process.env.JWT\_SECRET, { expiresIn: '1h' });

res.status(200).json({ message: 'Login successful', token });

});

🡪**jsx for jobseeker and job provider**

import React from 'react';

import { BrowserRouter as Router, Routes, Route } from 'react-router-dom';

import Header from './components/Header';

import Footer from './components/Footer';

import JobSeekerAndProvider from './components/JobSeekerAndProvider';

import './App.css';

const App = () => {

return (

<Router>

<div className="page-container">

<Header />

<main className="main-content">

<Routes>

<Route path="/" element={<JobSeekerAndProvider />} />

</Routes>

</main>

<Footer />

</div>

</Router>

);

};

export default App;

🡪**frontend for job seeker and job provider**

import React from 'react';

import './JobSeekerAndProvider.css';

const JobSeekerAndProvider = () => {

return (

<div>

<div className="job-seeker-page">

<header className="job-seeker-header">

<h1 className="job-seeker-logo">FlexiWork</h1>

<p className="job-seeker-tagline">Your Flexible Job Solution</p>

</header>

<main className="job-seeker-content">

<section className="job-seeker-hero">

<h2 className="section-heading">Find Your Dream Job</h2>

<p className="section-description">

Browse thousands of jobs tailored to your skills and preferences.

</p>

<button className="explore-jobs-btn">Explore Jobs</button>

</section>

<section className="job-seeker-categories">

<h2 className="section-heading">Job Categories</h2>

<ul className="categories-list">

<li>Technology</li>

<li>Design</li>

<li>Marketing</li>

<li>Finance</li>

<li>Healthcare</li>

</ul>

<h2 className="section-heading">Complete Your Profile</h2>

🡪**Css for Jobseeker and Jobprovider**

.explore-jobs-btn {

padding: 12px 24px;

background-color: #9857d9;

color: #ffffff;

border: none;

border-radius: 5px;

cursor: pointer;

font-size: 16px;

transition: background-color 0.3s ease;

.job-seeker-categories {

padding: 40px 20px;

text-align: center;

background-color: #000000;

}

.categories-list {

List style: none;

padding: 0;

margin: 20px auto;

display: flex;

flex-wrap: wrap;

justify-content: center;

gap: 20px;

}

job-seeker-additional-info button {

display: inline-block;

background-color: #2a9d8f;

color: #ffffff;

border: none;

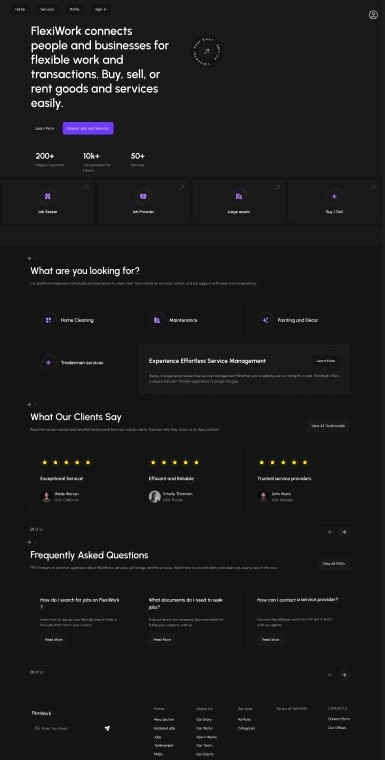
padding: 12px 24px;

border-radius: 5px;

cursor: pointer;

transition: background-color 0.3s ease;

**UI Design**



**Individual contribution**