

JOB APPLICATION TRACKER

A React, Redux & ASP.NET Core Web API Project



Submitted by:

Sameer Saurav

Batch C3

Under the Supervision of **Ms. Jyoti S Patil**

20th March 2025





Index

Serial No	Title	Slide No
1	Problem Statement	3
2	Project Goals & Objectives	4
3	Software & Hardware Requirements	5
4	Frontend & Backend Architecture	6
5	Components Breakdown & API Design with ER Diagrams	8
6	Data Flow Design	12
7	Conclusion	13
8	Future Scope	14

Problem Statement:

Many job seekers apply for multiple jobs but struggle to track their applications, status updates, and interview schedules efficiently. This leads to missed opportunities, poor organization, and delayed responses.

Key challenges Faced by Job Seekers:

-  **Tracking multiple job applications** (No proper organization)
-  **No real-time notifications** for application status updates
-  **Lack of filtering & sorting** for jobs based on category, location, or salary
-  **No way to bookmark jobs or take notes** for reference

Project Goals & Objectives

Goal: Created an efficient **Job Application Tracker** to help users manage applications and track job statuses seamlessly.

Objectives:

- ✓ **Track job applications** with ease
- ✓ **Search filter, and sort** job listings
- ✓ **Bookmark** job applications
- ✓ **Secure Authentication** with JWT

✓ **Add, Edit & Delete applications** in just one click

Software Requirements:

- **CODE EDITOR:** VS Code, Visual Studio 2022
- **FRONTEND:** React.js, NPM, Redux, CSS, Bootstrap, Toaster, Styled Components
- **BACKEND:** .NET (8), C#, Web Browser (Chrome)
- **DATABASE:** SQL Server
- **TESTING API:** Post Man, Swagger

Hardware Requirements:

- **PROCESSOR:** 3.0 GHz quad-core processor

- **RAM:** At least 4GB of memory
- **OPERATING SYSTEM:** Windows (10 or Higher)

Frontend & Backend Architecture

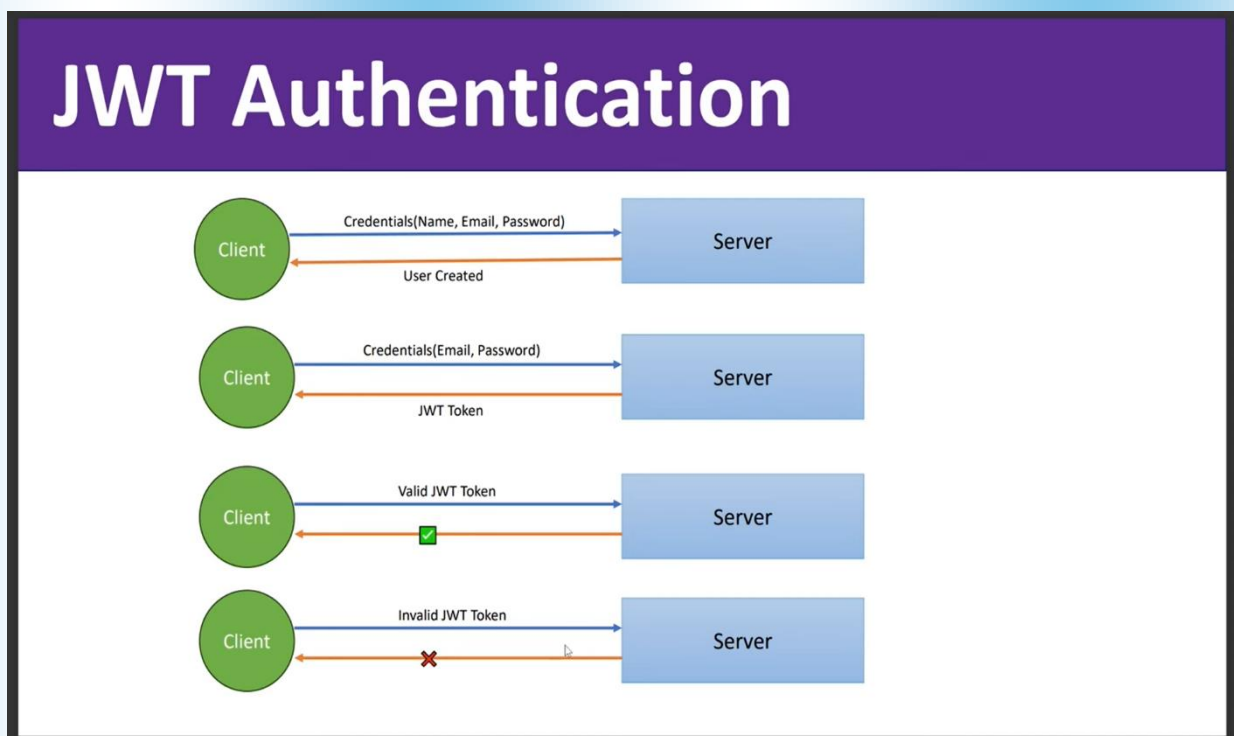
Frontend

- Developed using **React.js** with **Redux** for state management.
- **React Router** is used for navigation.
- **Bootstrap** is used for responsive UI design.
- Implemented **Lazy Loading** for performance optimization.

Backend

- Built with **ASP.NET Core Web API**.
- Used **Entity Framework Core** for database interactions.

- **Endpoints** for user authentication, job listings, applications, and notifications.
- Implemented **JWT-based authentication & authorization.**



Database

- **Microsoft SQL Server** as the database.

- Tables: Users, Jobs, Applications, etc
- Used **Primary Key** for efficient querying.

Components Breakdown & API Design

Frontend Components

- **State Management:** Redux & React is used for managing job applications and user authentication.
- **Routing:** React Router handles navigation across pages.
- **UI Components:** Separate components for job listing, application form, filters, and user dashboard.
- **Navbar.jsx** – Displays navigation links
- **JobDetails.jsx** – Shows job details

- **JobFilter.jsx** – List all jobs with filters
- **FavoriteJobs.jsx** – Bookmark a job

API Design

Endpoints:

1. **/api/register** - Register a new user.
2. **/api/login** - Authenticate users and generate JWT tokens.
3. **/api/job/getall** - Fetch all available jobs.
4. **/api/jobsearch** – Filter jobs according to location, city, country.
5. **/api/applications/me** - Retrieve user-specific applications.
6. **/api/applications/:id** - Retrieve user-specific applications by id.

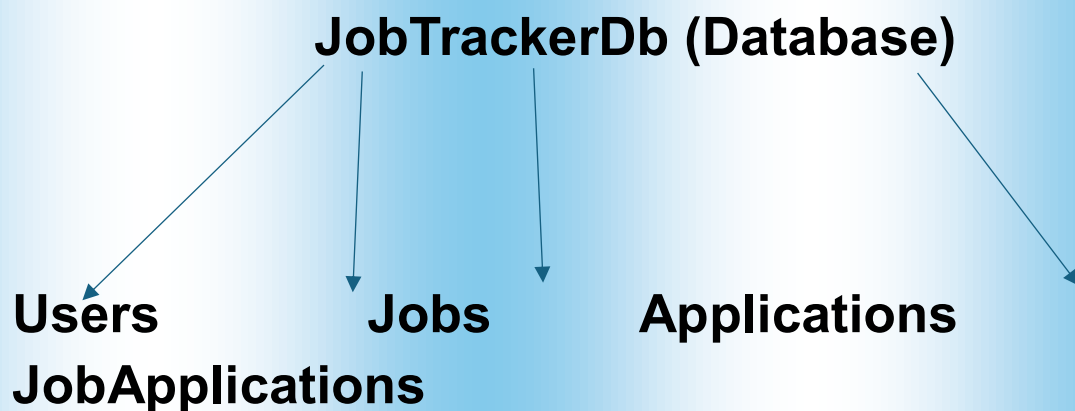
7. **/api/favorites** – Bookmark a Job.

8. **/api/applicants** – Admin can see user's profile.

9. **/api/add/jobs** – Admin can create jobs.

10. **/api/update/jobs** – Admin can update or delete existing jobs.

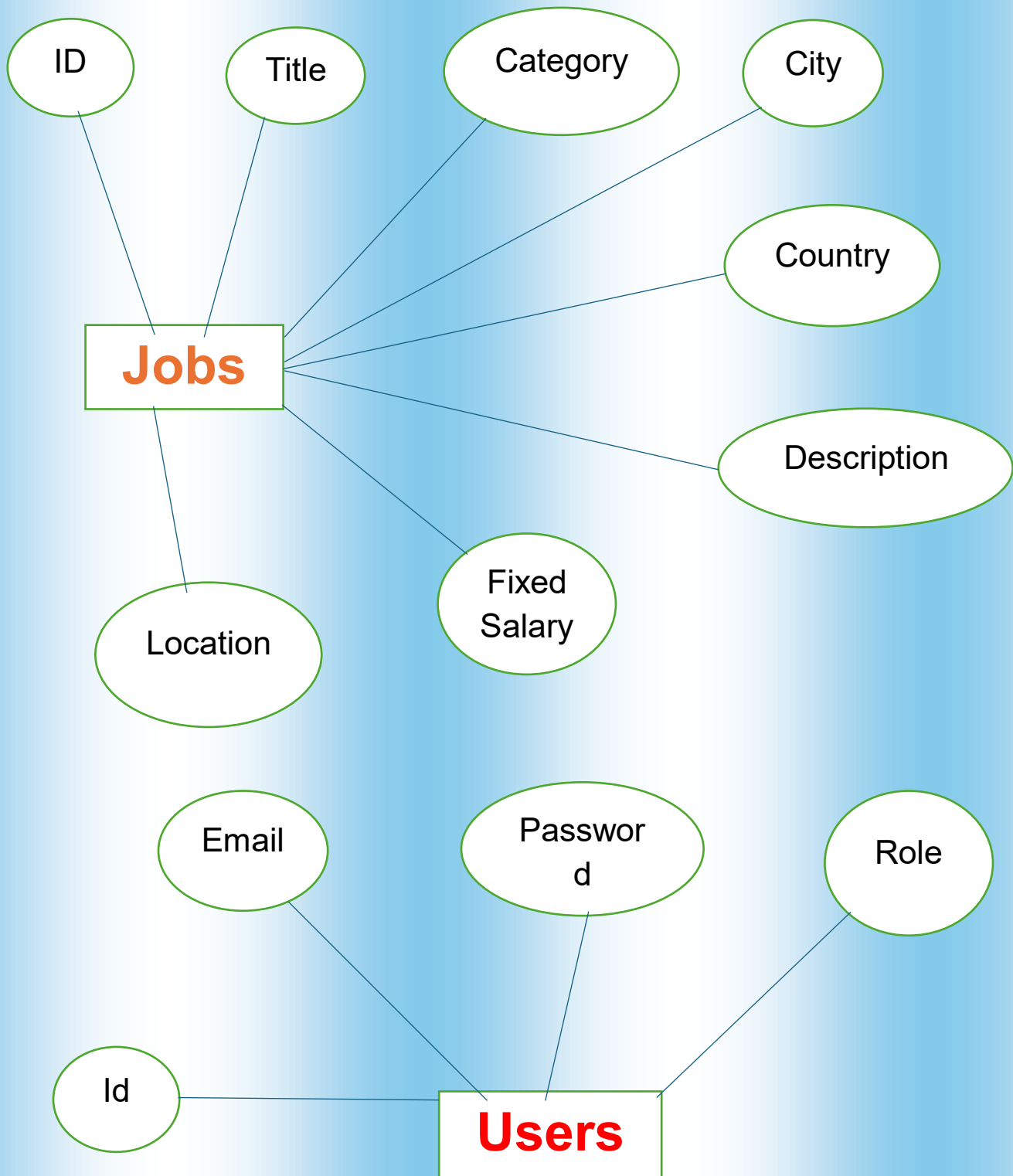
Database Design

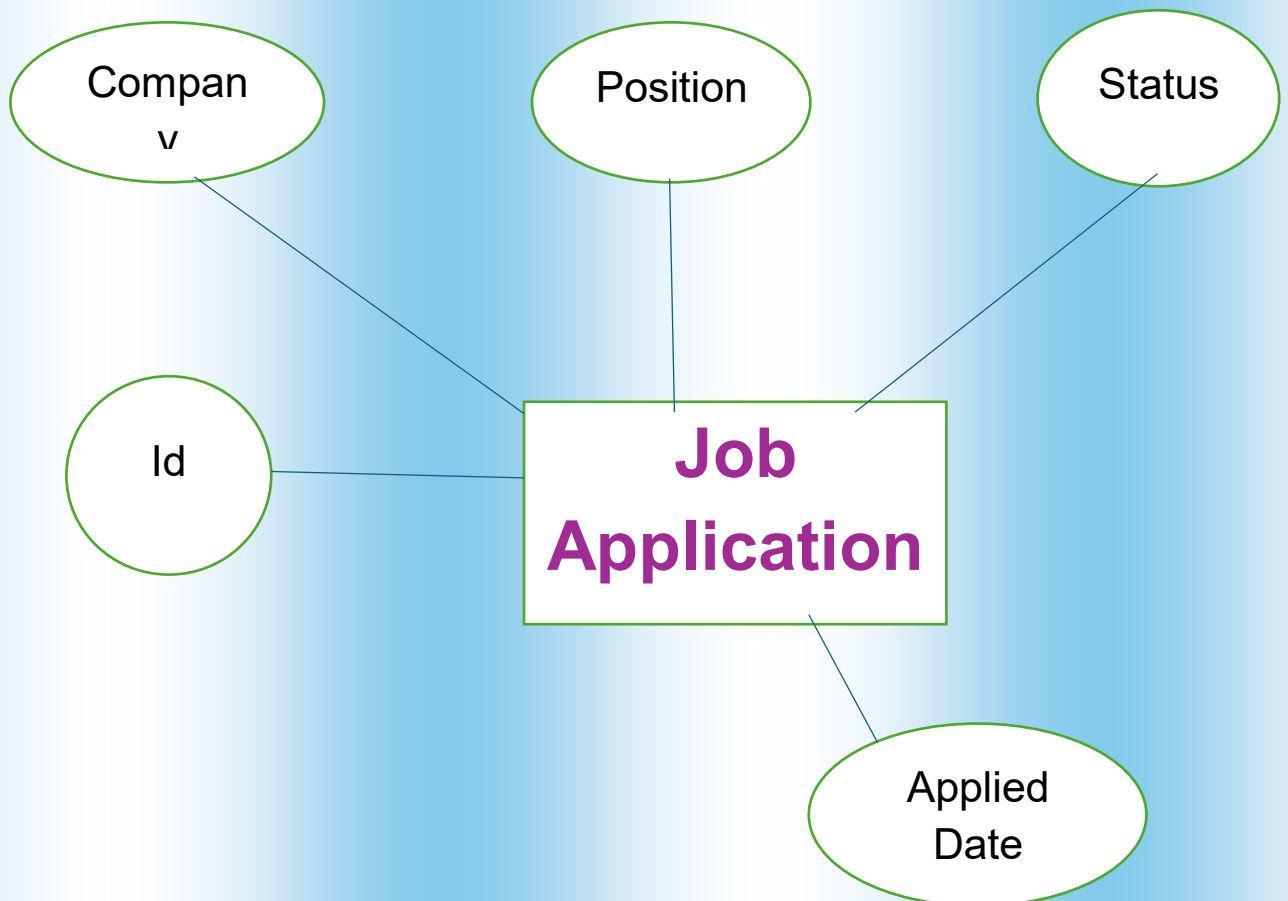
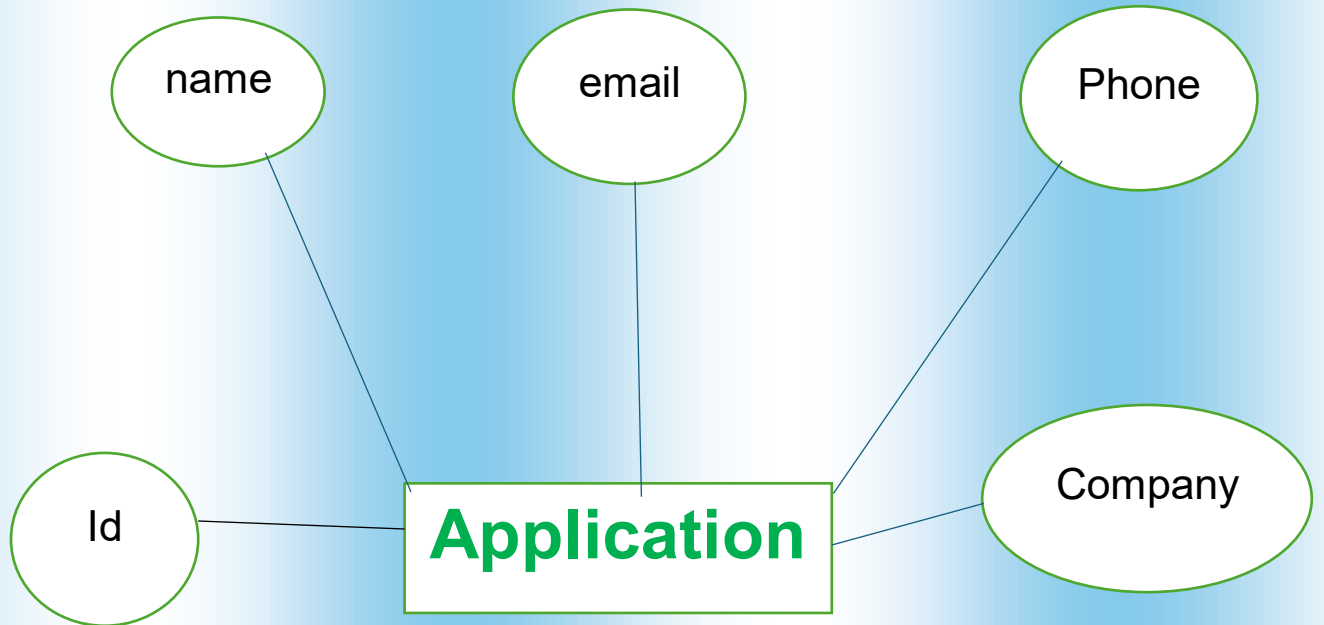


Tables: -

➤ **Users** (Id, Email, Password, Role)

- **Jobs** (Id, Title, Category, Country, City, Description, Fixed Salary)
- **Applications** (Id, Name, Email, Phone)





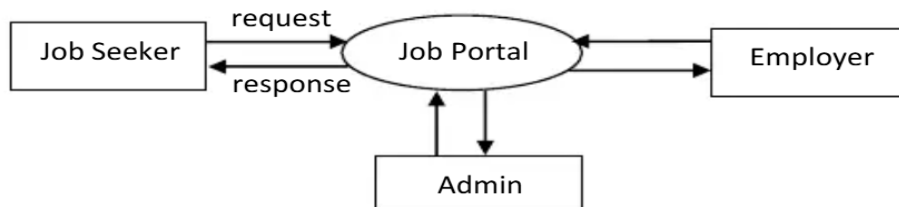
Security Considerations

- **Password Hashing:** User passwords are stored securely using BCrypt hashing.
- **JWT Authentication:** Ensures secure user login and API access.
- **Role-based Authorization:** Users and Admins have different privileges
- **CORS Policy Implementation:** Ensures secure cross-origin requests

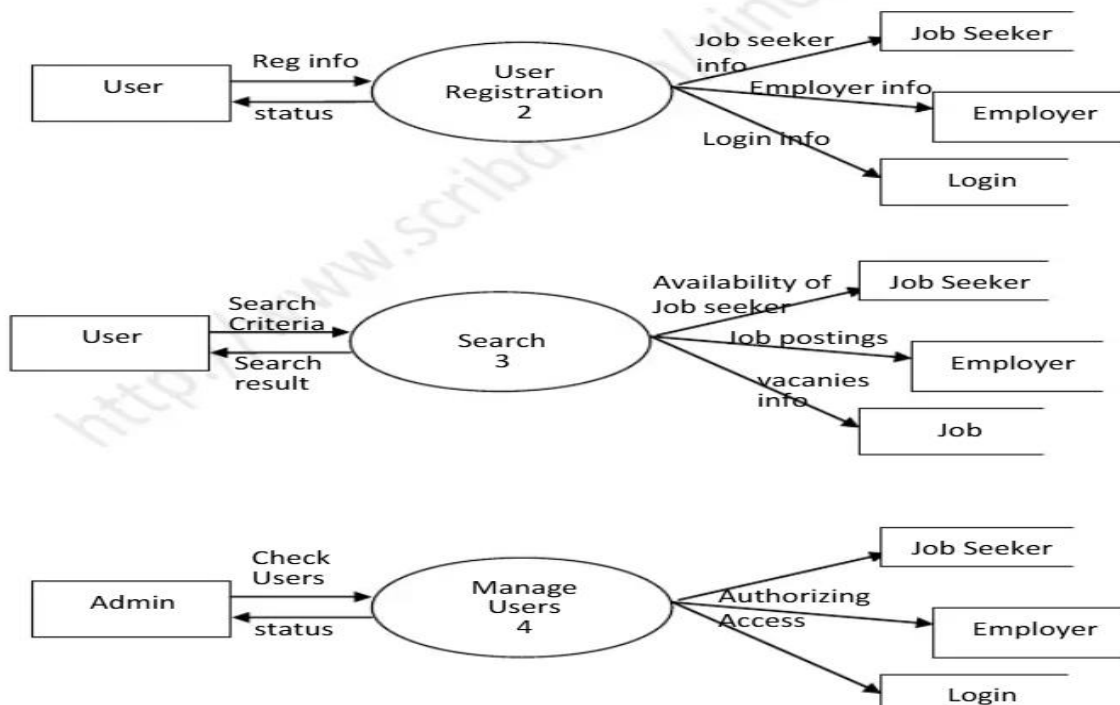
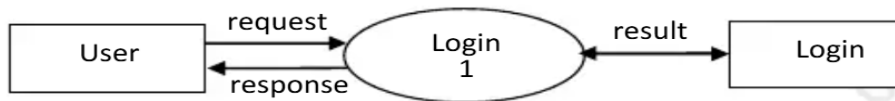
Data Flow Diagram of Project

Data Flow Diagram

Level 0



Level 1



Conclusion

The **Job Application Tracker** project successfully streamlines the job search and application process by providing a user-friendly platform for job seekers and employers. With a **React.js frontend, ASP.NET Core Web API backend, and SQL database**, the system ensures efficient job listing, application management, and real-time status updates.

Key features such as **job filtering, sorting, bookmarking, real-time notifications, and status tracking** enhance the user experience, making job applications more organized and accessible.

For More details access the complete source code on **GitHub** and also refer README to start project:

<https://github.com/sam32190/JobTracker.git>

Future Scopes

- **AI-Based Job Recommendations:** Suggest jobs based on user preferences.
- **Mobile App Integration:** Expand to Android/iOS platforms.
- **Resume Parsing:** Automatically extract skills & experience from resumes.
- **Social Media Integration:** Apply for jobs using LinkedIn profiles