Project Title	**Hourly Recruitment Application**
Technologies	MERN
Domain	Industry
Project Level	Medium

Table

Table of Contents

1. Problem Statement: 2
1.1. What is an Hourly Recruitment Application? 2
1.2. Project Objective 3
1.3. Functional and Non-Functional Requirements: - 3
1.3.1. Functional Requirements 3
1.3.2. Non-Functional Requirements 3
2. Project Evaluation metrics: 4
2.2. Database: 4
2.3. API Details or User Interface: 4
2.4. Deployment: 4
Deploy the application on your preferred service........ 4

2.5. Solutions Design: 4

2.6. System Architecture: 4

2.7. Optimization of solutions: 4

3. Submission requirements: 5

3.1. High-level Document: 5

3.2. Low-level document: 5

3.3. Architecture: 5

3.4. Wireframe: 5

3.5. Project code: 5

3.6. Detail project report: 5

3.7. Project demo video: 5

3.8. The project LinkedIn a post: 5

1. Problem Statement:

Design a web application “Hourly Recruitment Application” to allow anyone to sign up as a recruiter or as a worker.

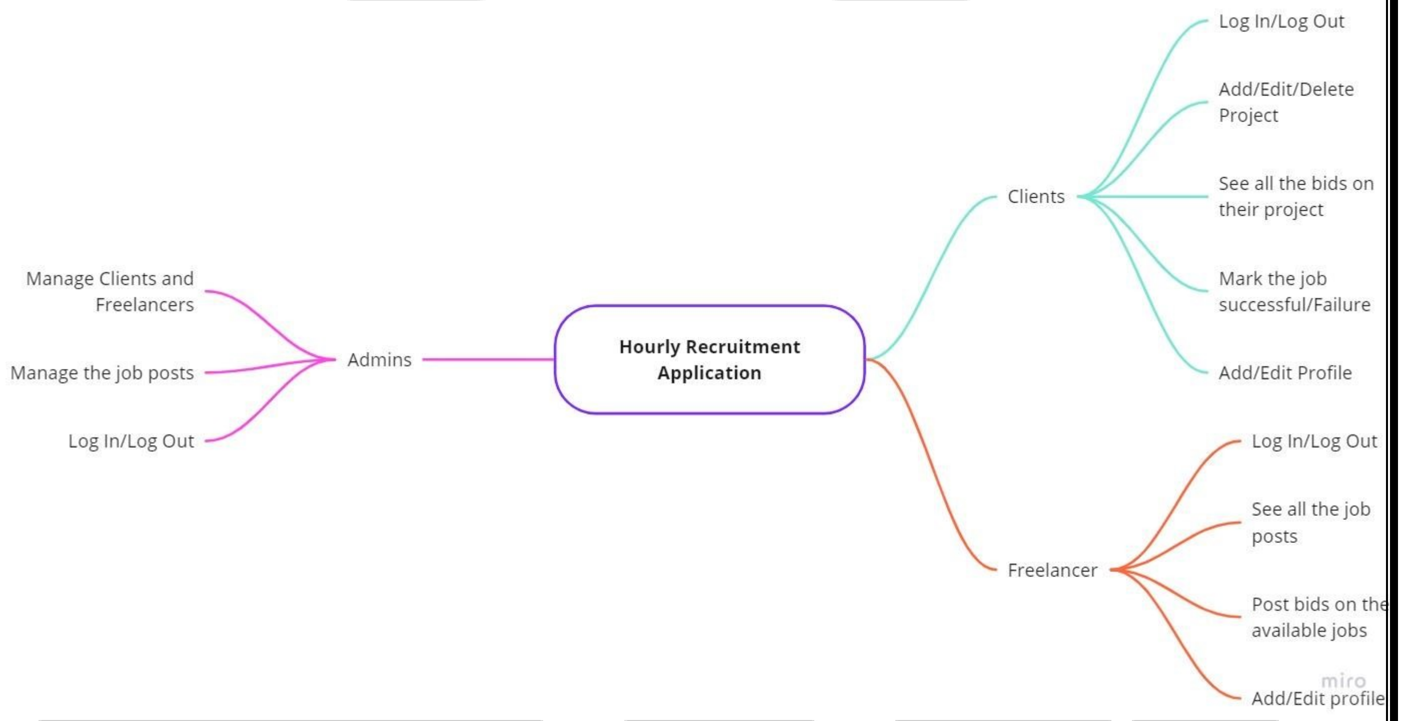
1.1. What is an Hourly Recruitment Application?

A platform for clients and freelancers from all around the world to work together for mutual benefit is Hourly Recruitment Application. If someone or a company needs professional labor for a short-term or long-term project, they can post it and invite freelancers to submit bids to complete it. The freelancers then can bid the amount they wish and at the end the recruiter can choose anyone. Upon mutual understanding from both the parties the contract work begins and upon successful completion of the work the contract ends.

Now that you understand what Hourly Recruitment Application is, let’s discuss some of the functionality of the “**Hourly Recruitment Application**” that you will design.

1. To create a web-based system that will let the users sign up as either a client or as a freelancer.
2. The clients can then post a job along with the necessary job requirements.
3. The freelancers can see the job posted and can start bidding with their quotes and time period they require to finish the task.

4. The client can then see all the bids and select any one bidder and upon mutual agreement the contract will begin.
5. As soon as both the parties agree the funds will be locked in the app wallet. The client will pay the agreed amount.
6. Once the work has been delivered and the client is satisfied the funds which are frozen in the app wallet can then be released to the freelancer, thus ending the contract.



Low-Level system flow

1.2. Project Objective

- To build a platform which makes it easier to hire people for contract work and for people to get hired for contract work.
- Give people the chance to work from anywhere for anyone without getting tied to a company.

1.3. Functional and Non-Functional Requirements: -

1.3.1. Functional Requirements

- **Login/Register:** Users must be able to register themselves as either a client or a freelancer and once registered successfully they should be able to login with their credentials.
- **Post a Gig:** Users who have signed up as clients should have an option to post a gig and the gig should then be visible to all the freelancers.

- **Bid for Gig:** Any freelancer should be able to bid on any of the gigs posted by clients, later the bid should be visible to the client with the freelancer's details.
- **Accept Bid:** The client should be able to contact the bidder and upon mutual agreement the client needs to accept the bid.
- **Complete Gig:** Once the project is completed and the client is happy with the work, the client needs to mark the project as completed thus ending the contract/Gig. At this point the money should then be sent to the freelancer.

1.3.2. Non-Functional Requirements

1. **Security:** The system should have a certain level of security such that not anybody can have access to sensitive information and the passwords should be properly encrypted.
2. **Robustness:** If the user's system crashes, a backup of the user data must be stored on remote database servers to enable recovery.
3. **Performance:** The application must be lightweight and the UI should be fast and responsive.
4. **Authentication Requirements:** There should be an authentication system in place which checks the credentials each time a user logs in.

2. Project Evaluation metrics:

2.1. Code:

- You are supposed to write code in a modular fashion ● Safe: It can be used without causing harm.
- Testable: It can be tested at the code level.
- Maintainable: It can be maintained, even as your codebase grows.
- Portable: It works the same in every environment (operating system).
- You have to maintain your code on GitHub.
- You have to keep your GitHub repo public so that anyone can check your code.
- Proper readme file you have to maintain for any project development.
- You should include the basic workflow and execution of the entire project in the readme file on GitHub.
- Follow the coding standards.

2.2. Database:

MongoDB is a source-available cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with optional schemas.

2.3. API Details or User Interface:

You have to expose your complete solution as an API or try to create a user interface for your model testing. Anything will be fine for us.

2.4. Deployment:

Deploy the application on your preferred service.

2.5. Solutions Design:
You have to submit complete solution design strategies in High-level Document (HLD), Lowlevel Document (LLD), and Wireframe documents**.**

2.6. System Architecture:
You must submit a system architecture design in your wireframe and architecture documents.

2.7. Optimization of solutions:
Try to optimize your solution on the code level, and architecture level, and mention all of these things in your final submission.

Mention your test cases for your project.

3. Submission requirements:

3.1. High-level Document:
You have to create a high-level document design for your project. You can reference the HLD form below the link.

Sample link: HLD Document Link

3.2. Low-level document:
You have to create a Low-level document design for your project; you can refer to the LLD from the link below.

Sample link: LLD Document Link

3.3. Architecture:

You have to create an Architecture document design for your project; you can refer to the Architecture from the link below.

Sample link: Architecture sample link

3.4. Wireframe:

You have to create a Wireframe document design for your project; refer to the Wireframe from the link below.

Demo link: Wireframe Document Link

3.1. Project code:

You have to submit your code to the GitHub repo in your dashboard when the final submission of your project.

Demo link: Project code sample link

3.2. Detail project report:

You have to create a detailed project report and submit that document as per the given sample.

Demo link: DPR sample link

3.3. Project demo video:

You have to record a project demo video for at least 5 Minutes and submit that link.

The project LinkedIn a post:

You have to post your project details on LinkedIn and submit that post link in your dashboard in your respective field.