Architecture Document

Hourly Recruitment Application

Written By	Hardik Arora
Document Version	0.3
Last Revised Date	Date

Document Control

Change Record:

Version	Date	Author	Comments
0.1	10/09/2023	Hardik Arora	Introduction & Architecture defined
0.2	20/09/2023	Hardik Arora	Architecture & Architecture Description appended and updated
0.3	25/09/2023	Hardik Arora	Unit Test Cases defined and appended

Contents

1. Introduction

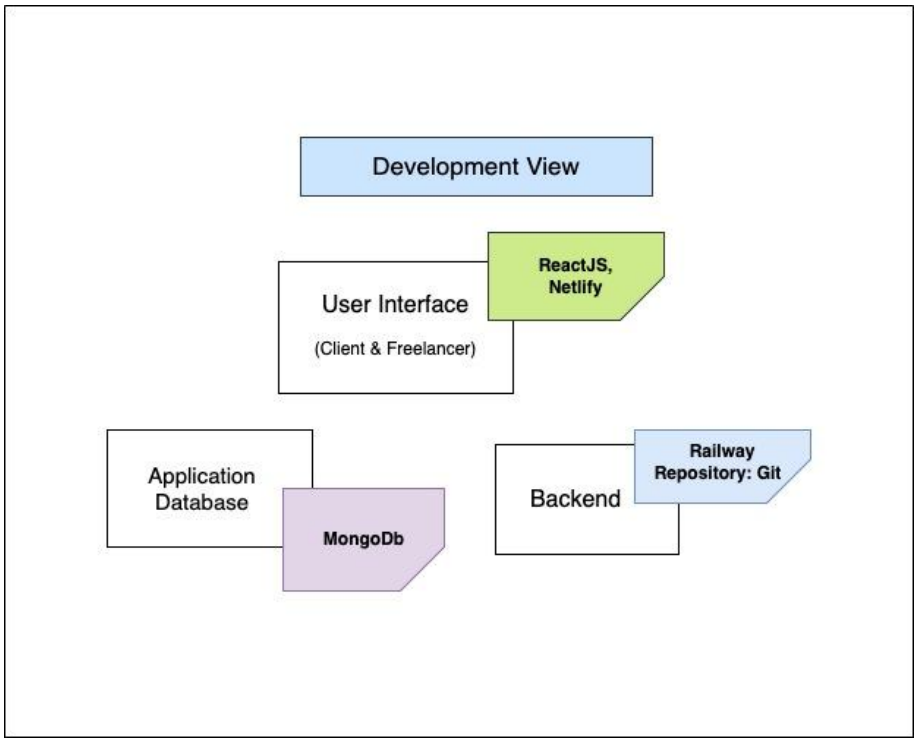
1.1. What is Low-Level design document?

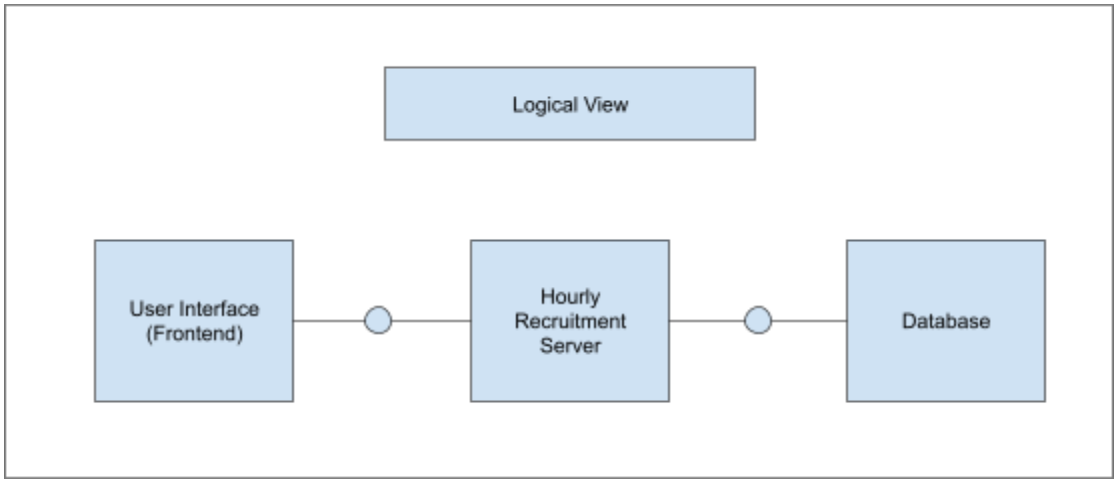
The goal of LLD or a low-level design document (LLDD) is to give the internal logical design of the actual program code for the Hourly Recruitment Application. LLD describes the class diagrams with the methods and relations between classes and program specs. It describes the modules so that the programmer can directly code the program from the document.

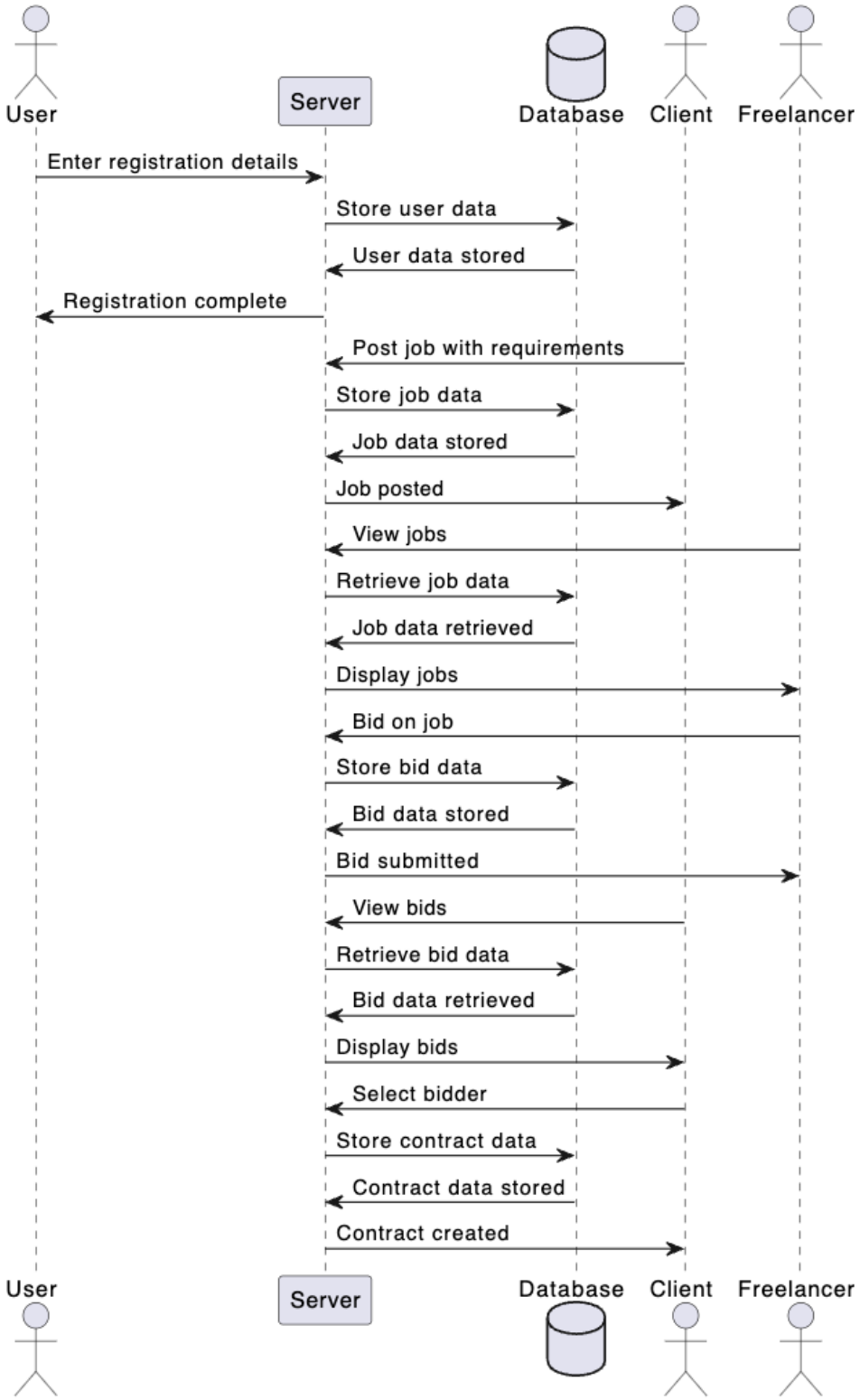
1.2. Scope

Low-level design (LLD) is a component-level design process that follows a step-by-step refinement process. This process can be used for designing data structures, required software architecture, source code and ultimately, performance algorithms. Overall, the data organization may be defined during requirement analysis and then refined during data design work.

2. Architecture






User
Server
Database
Client
Freelancer
Enter registration details
Store user data
User data stored
Registration complete
Post job with requirements
Store job data
Job data stored
Job posted
View jobs
Retrieve job data
Job data retrieved
Display jobs
Bid on job
Store bid data
Bid data stored
Bid submitted
View bids
Retrieve bid data
Bid data retrieved
Display bids
Select bidder
Store contract data
Contract data stored
Contract created
User
Server
Database
Client
Freelancer

3. Architecture Description

3.1 Frontend (React):

Authentication:

Implementing user signup and login functionality. Using JWT (JSON Web Tokens) for secure authentication.

User Dashboard:

Separating dashboards for clients and freelancers after login.

Job Posting:

Clients can create a job post with necessary details (job description, requirements, budget, etc.).
Use forms for input and validation.

Job Listing:

Displaying all available jobs to freelancers.
Using React components to render job details.

Bidding System:

Freelancers can bid on jobs, specifying their quotes and timeframes.

Job Selection:

Clients can view bids and select a freelancer.

Contract Agreement:

Creating a contract with terms agreed upon by both parties.

Billing:

Initiation of billing as soon as the contract is agreed upon.
Displaying total bill amount and details.

3.2 Backend (Node.js and Express.js):

User Management:

APIs for user registration, login, and profile management.

Job Management:

APIs for posting, updating, and deleting jobs.

Bid Management:

APIs for freelancers to submit bids and clients to view and accept bids.

Contract Management:

APIs for creating, updating, and viewing contracts.

Billing System:

Implement a billing system to calculate charges based on time, rates, etc.

3.3 Database (MongoDB):

Store user profiles, job details, bids, contracts, and billing information.

3.4 Additional Considerations:

Security:

Implementing secure coding practices.
Using HTTPS for communication.
Validating and sanitizing inputs to prevent security vulnerabilities.

Scalability:

The application should be able to handle a growing number of users and transactions.
Consider load balancing and database sharding if needed.

Testing:

Implement unit testing.

Deployment:

Deploy frontend and backend separately.

Monitoring and Logging:

Implement logging and monitoring solutions to track application performance and errors.

Documentation:

Create comprehensive documentation for developers and users.

4. Unit Test Cases

S. No.	Test Case Description	Test Input	Expected Output
1	Successful User Login	Valid username and password	Successful login
2	Unauthorized Access	Invalid or expired JWT token	Access denied
3	Successful Job Posting	Valid job details and requirements	Job posted successfully, visible in the job listing.
4	Empty Job Post	Posting a job without specifying details	Validation error with a request to provide necessary details.
5	Successful Bid Submission	Valid bid details, including quote and time period	Bid submitted successfully and visible in the list of bids.
6	Bid with Invalid Quote	Bid with a non-numeric quote	Validation error and request for valid numeric quote.
7	Project Completion	Post repository link	Request successful, Final cost is displayed
8	Project completion with Empty Link	Clicking on complete without link	Request Failed, and request for link.
9	Admin User Deletion	Admin deletes an existing user	User is removed from the system
10	Admin Project Deletion	Admin deletes an existing project	Project is removed from the system