Technical Specification

1. Overview of Architecture

This application will adopt a Three-Tier Architecture:

1. Frontend (Client-Side): Responsible for user interaction, leveraging React.js for dynamic UI and seamless interaction.
2. Backend (Server-Side): Manages business logic, APIs, authentication, and integrations. Built using Node.js with Express.js.
3. Database: MongoDB for storing structured and semi-structured data, including user profiles, job posts, messages, and payment details.

2. Key Technologies, Libraries, and Frameworks

Layer	Technology	Reason of selection
Frontend	React Js	Highly responsive and reusable component UI
	Tailwind CSS	Customizable styling
	Axios	For the api request and error handling
Backend	Node Js	Efficient server-side scripting and scalability
	Express Js	Lightweight, fast web application framework.
	Socket io	Real-time communication for messaging.
Database	MongoDB	Flexible schema design for user profiles and job postings.
Authentication	Firebase	Streamlined social and email authentication setup
Payment Gateway	Stripe/PayPal	Secure and robust payment integration.
DevOps	AWS	Reliable hosting with CI/CD pipelines.

Outline of Core Components and Modules

Frontend Components

Authentication Module

Login/Signup Page: Handles user registration and login via email, Google, or LinkedIn.

Account Type Selection: Interface for choosing between Worker and Recruiter accounts.

Profile Management

General Info Form: Captures name, location, bio, etc.

Portfolio Form: Allows adding projects with descriptions and links.

Employment History Form: For past roles, employers, and durations.

Education Form: Academic qualifications and institutions.

Bank Details Form: Account number, IFSC, and PayPal and Stripe details.

Dashboard

Worker Dashboard:

Job Search: List and filter job postings.

Messaging: Chat interface for recruiter-worker communication.

Reports: Track hours worked, tasks, and payment status.

Recruiter Dashboard:

Job Management: Add, edit, or delete job posts.

Hired Workers: View and manage hired freelancers.

Balance Overview: Manage funds for payments.

Settings

Update contact info, payment method, and password.

Backend Modules

Authentication Service

Email verification and social login.

JWT for session management.

Profile Management

APIs to create and retrieve worker/recruiter profiles.

Data validation and file upload handling.

Job Management

CRUD APIs for job posts.

Search and filter functionality for workers.

Messaging

Real-time communication via Socket.IO.

Persistent message storage in MongoDB.

Reports & Financial Management

APIs for tracking hours, pending tasks, and payments.

Integration with Stripe/PayPal for fund transfers.

Database Design

Collections:

users: Stores user details, account type, and authentication info.

profiles: Holds worker and recruiter profiles.

jobs: Contains job posts with recruiter details.

applications: Tracks job applications and their statuses.

messages: Stores chat history.

payments: Logs financial transactions.

Collection	Fields
Users	_id, email, password, accountType, socialAuth, createdAt, updatedAt
Profiles	userId, generalInfo, portfolio, employmentHistory, education, bankDetails
Jobs	_id, recruiterId, title, description, requirements, salary, createdAt
applications	_id, jobId, workerId, status, createdAt
messages	_id, conversationId, senderId, receiverId, message, timestamp
payments	_id, userId, amount, method, transactionId, status, createdAt

4. Integration Plan

Third-Party Services:

Firebase: Social login and email-based authentication.

Stripe and PayPal: Secure payment processing.

API Documentation: Postman for clear API documentation.

5. Security Measures

Encrypt sensitive data (passwords, bank details) using bcrypt.

Implement role-based access control for APIs.

Use input validation libraries (ZOD) like Joi to prevent injection attacks.

Team Composition

Role	Responsibilities	Number Of Member
Project Manager	Define scope, milestones, and timelines.	1
	Ensure communication among team members.	
	Track progress and manage risks.	
	Act as the primary point of contact.	
UI/UX Designer	Refine the Figma designs for usability and consistency.	1
	Create interactive prototypes.	
	Ensure design alignment with the branding and user goals.	
Frontend Developer	Implement user-facing components using React.js	2
	Integrate APIs.	
	Ensure responsiveness and cross-browser compatibility.	
Backend Developer	Design and implement APIs using Node.js and Express.js.	2
	Manage database schemas and queries.	
	Handle authentication, real-time messaging, and payment systems.	
Database Engineer	Set up and manage the MongoDB database.	1

	Ensure optimal query performance and backup strategies.	
	Handle data migration and indexing.	
QA Engineer	Conduct functional, regression, and user acceptance testing.	1
	Identify and log bugs.	
	Validate performance and security requirements.	
DevOps Engineer	Set up CI/CD pipelines for automated testing and deployment.	1
	Manage cloud infrastructure and monitoring.	
	Ensure application uptime and scalability.	

2. Rationale for Each Role

1. **Project Manager**:
    - Essential for aligning team efforts, managing resources, and ensuring the project stays on track.
    - Acts as a bridge between stakeholders and the development team.
2. **UI/UX Designer**:
    - Ensures the platform is visually appealing and user-friendly, increasing user satisfaction and engagement.
    - Handles the Figma design handoff, addressing usability issues before development begins.
3. **Frontend Developers**:
    - Two developers ensure parallel work on worker and recruiter dashboards, profile sections, and job search features.
    - They will focus on responsiveness, dynamic updates, and integrating backend APIs efficiently.

4. **Backend Developers**:
    - One backend developer focuses on user authentication, profile management, and messaging APIs.
    - The other handles job posting, financial management, and report generation APIs.
5. **Database Engineer**:
    - Manages data integrity and ensures database scalability and performance.
    - Optimises database queries and implements backup strategies for disaster recovery.
6. **QA Engineer**:
    - Plays a critical role in maintaining product quality by identifying and resolving issues before deployment.
    - Ensures all features meet user requirements and are free from critical bugs.
7. **DevOps Engineer**:
    - Manages CI/CD pipelines to streamline development and deployment.
    - Monitors server health and ensures high availability of the application.

3. Anticipated Challenges and Solutions

Challenge	Solution
Communication Gaps	I will schedule daily standups and use Slack for real-time communication. I will manage tasks using Jira.
Cross-Dependency Delays	I will create detailed timelines with buffer time for interdependent tasks and regularly review progress in sprint meetings.
Resource Overload	I will distribute tasks evenly and track workload using task management tools, assigning backups for critical roles.
Integration Issues	I will set up regular integration checkpoints and encourage collaboration between frontend and backend teams early in development.
Team Onboarding and Alignment	I will conduct an initial orientation meeting to explain the project goals, workflow, and deliverables, providing access to documentation and tools.
Security and Compliance	I will ensure the DevOps engineer and QA engineer review security standards during development and testing.

4. Team Collaboration Tools

- **Communication**: Slack
- **Task Management**: Jira
- **Version Control**: GitHub with defined branching strategies.
- **Design Collaboration**: Figma.
- **Documentation**: Notion.

Project Timeline and Milestone Document

1. Breakdown of Project Phases

Phase	Description	Duration	Key Deliverables
Phase 1: Planning	Define project scope, finalise requirements, and assign roles. Set up repositories and tools.	1 Week	* Project requirements * Team onboarding * Setup tools
Phase 2: Design	Review and refine Figma designs. Prepare design specifications and prototypes for handoff.	1 Week	* Finalised designs * Interactive prototypes
Phase 3: Development	Implement core functionality for authentication, profile creation, dashboards, and messaging.	4 Weeks	**Authentication module** * Dashboards * Messaging system
Phase 4: Testing	Perform unit, integration, and user acceptance testing. Resolve bugs and finalise for deployment.	2 Weeks	**Test cases** * Bug reports * Final QA sign-off
Phase 5: Deployment	Deploy to production and conduct post-launch monitoring to ensure stability and scalability.	1 Week	* Live application * Monitoring and feedback plan

2. Timeline and Milestones

Phase	Milestone	Deadline	Dependencies
Planning	Project kickoff and tools setup	End of Week 1	None
Design	Finalised design handoff	End of Week 2	Completion of planning phase
Development	Authentication module completed.	End of Week 3	Design handoff
	Profile creation functionality implemented	End of Week 4	Authentication module
	Worker and recruiter dashboards completed	End of Week 5	Profile creation
	Messaging and job management features completed	End of Week 6	Dashboards
Testing	Functional and integration testing	End of Week 7	Completion of all development tasks
	User acceptance testing	End of Week 8	Functional testing
Deployment	Application deployed to production	Successful testing and QA sign-off	Successful testing and QA sign-off

3. Task Dependencies

1. **Design Handoff**: Development begins only after final designs are approved.
2. **Authentication Module**: Profile management and dashboard functionalities depend on authentication being completed.
3. **Dashboards**: Require both authentication and profile creation to be functional.
4. **Testing Phase**: Begins only after all development tasks are complete.
5. **Deployment**: Dependent on the successful completion of testing and QA sign-off.

4. Risk Management

- **Risk**: Delays in design handoff may impact the development schedule.
  Mitigation: Collaborate closely with the designer and involve the frontend team early to anticipate requirements.
- **Risk**: Unexpected bugs during testing.
  Mitigation: Allocate buffer time and ensure unit testing is performed during development.

Communication and Collaboration Strategy

1. Regular Check-ins and Reporting Structures

Meeting Type	Frequency	Participants	Purpose
Daily Standup	Daily (15 minutes)	Entire development team	Quick updates on progress, blockers, and priorities for the day.
Sprint Planning	Weekly (1 hour)	Project Manager, Team Leads	Define tasks for the sprint, assign roles, and clarify deliverables.
Design Review	As needed (1 hour)	UI/UX Designer, Frontend Team	Collaborate on finalising designs and resolving any ambiguities.
Development Review	Bi-weekly (1 hour)	Project Manager, Developers	Review progress on key features, discuss challenges, and plan next steps.
QA/Testing Sync-Up	Weekly (30 minutes)	QA Engineer, Developers	Review test results, track bug fixes, and align testing with new development.
Retrospective	End of each sprint	Entire team	Reflect on what went well, identify areas for improvement, and discuss team feedback.
Stakeholder Update	Bi-weekly (1 hour)	Project Manager, Stakeholders	Share progress, demo key features, and gather feedback or address concerns.

2. Preferred Tools

Function	Tool	Purpose
Communication	Slack	For real-time communication and team channels.
	Zoom	For virtual meetings and video conferencing.
Collaboration	Figma	Design handoffs and collaborative UI/UX reviews.
	GitHub/GitLab	Version control and code reviews.
Project Management	JIRA	Sprint planning, task tracking, and reporting.
	Notion	Documentation for processes, FAQs, and onboarding.
Testing & Feedback	TestRail/JIRA Test Management	Tracking test cases and integrating QA feedback into sprints.
File Sharing	Google Drive	Storing and sharing documentation, assets, and other resources.

3. Handling Remote and Asynchronous Collaboration Challenges

Challenge	Solution
Different Time Zones	Schedule daily standups at overlapping working hours. Leverage asynchronous tools like Slack for updates.
Lack of Real-Time Interaction	Use recorded video updates for key meetings. Encourage regular communication on Slack.
Misalignment on Priorities	Use JIRA to clearly define tasks and dependencies. Regularly update task statuses to maintain clarity.
Delayed Feedback	Implement clear deadlines for reviews and feedback. Utilise shared documentation tools like Notion.
Onboarding Remote Team Members	Provide access to onboarding materials in Confluence/Notion. Host a virtual orientation session.

4. Escalation Protocol

- **Minor Issues**: Address during daily standups or in Slack threads.
- **Critical Issues**: Escalate immediately to the Project Manager, who will set up a quick resolution call if needed.
- **Stakeholder Concerns**: Document feedback in JIRA and align on priorities during the next stakeholder update.

5. Key Principles

1. **Transparency**: Share progress, blockers, and achievements regularly with the entire team.
2. **Flexibility**: Accommodate varying schedules by using asynchronous updates effectively.
3. **Accountability**: Each team member is responsible for updating the status of their assigned tasks and responding to queries promptly.