**Project Design Phase**

**Solution Architecture**

|  |  |
| --- | --- |
| Date | 26 -06-2025 |
| Team ID | Single |
| Project Name | Freelance application mern |
| Maximum Marks | 4 Marks |

**Solution:**

Freelance application mern leverages a modular MERN architecture to deliver a secure, scalable, and intuitive platform for freelance collaboration. The system is designed to handle high concurrency while ensuring data integrity and seamless user experiences.

Core Architectural Components

1. Responsive Frontend (React.js)
   * Dynamic UI with role-based views (client/freelancer/admin)
   * Real-time updates via WebSocket integrations
   * Mobile-first design using Material-UI or TailwindCSS
2. Backend Services (Node.js + Express.js)
   * RESTful APIs with JWT authentication
   * Role-based access control (RBAC) middleware
   * Rate limiting and request validation
3. Database Layer (MongoDB)
   * Optimized schemas for:
     + User profiles (embedded portfolios/work history)
     + Job listings (geospatial/text indexing for search)
     + Transaction records (immutable logs for audits)
   * Atlas cloud deployment for automatic scaling
4. Real-Time Communication
   * Bidirectional messaging via Socket.io
   * Event-driven notifications (email/in-app) using queues
5. Payment Infrastructure
   * Escrow service with Stripe/Braintree integration
   * Webhook handlers for payment status reconciliation
   * Dispute resolution workflows with evidence attachments
6. Admin & Moderation
   * Dashboard with analytics (MongoDB Aggregation)
   * Reporting system with automated evidence capture

Key Workflows

| Feature | Technical Implementation | Scalability Consideration |
| --- | --- | --- |
| Job Posting & Bidding | React forms + Express.js validation | Redis caching for high-frequency job feeds |
| Smart Matching | MongoDB text search + custom scoring algorithm | Indexed fields for performance |
| Milestone Payments | Stripe Connect with escrow release triggers | Idempotent API design |
| Real-Time Chat | Socket.io rooms with message persistence | Load-balanced WebSocket servers |
| Fraud Detection | Node.js middleware for pattern analysis | Automated flagging + admin alerts |

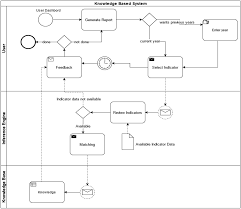
DevOps & Scalability

* CI/CD: Automated testing/deployment (GitHub Actions)
* Monitoring: Prometheus/Grafana for API metrics
* Scaling: Horizontal scaling (Node.js clusters), CDN for static assets

Why This Architecture?

1. Performance: Async I/O (Node.js) handles concurrent bids/chat efficiently.
2. Security: JWT + RBAC, encrypted payments, and immutable audit logs.
3. Flexibility: Microservices-ready for future features (e.g., AI matching).

**Example - Solution Architecture Diagram:**



2.Diagram that help you understand how it’s work:

