Project Report: Decentralized Job Board

TEAM CRYPTOSMITHS

K. Brahmisree	230001035
Gayathri Manaswini K	230001038
Raunak Anand	230001067
J. Vidhhya Varshini	230041013
K. Vaishnavi	230041016
M. Namaswi	230041023

1. Objective

The Decentralized Job Board (DJB) is a blockchain-powered platform designed to establish a trustless job marketplace. It connects freelancers and employers without intermediaries by leveraging smart contracts for transparent and secure escrow-based payment handling.

Key Goals:

- Eliminate the need for trust between freelancers and employers.
- Enable automatic, verifiable, and immutable workflows.
- Ensure fund safety using escrow mechanisms.
- Provide an open, decentralized environment for global talent hiring.
- Integrated chat system for resolving disputes.

2. Technology Stack

Smart Contract Development:

- Solidity (v0.8.21): For writing and deploying smart contracts.
- OpenZeppelin: Security tools like ReentrancyGuard.

Development & Testing:

- Truffle: Smart contract compilation, migration, and testing.
- Ganache: Local Ethereum blockchain for simulations.

Frontend Development:

- React.js: Building dynamic user interfaces.
- Web3.js: Interaction with the Ethereum blockchain.
- MetaMask: Wallet integration for blockchain authentication.
- JSON Server: Handles off-chain data (job descriptions, messages).

3. Smart Contract Architecture

Job Struct:

- Stores all essential information:
- Job ID, Title, Description
- Budget (payable)
- Status (Enum: OPEN, ASSIGNED, AWAITING_APPROVAL, COMPLETED, DISPUTED, RESOLVED)
- Escrow Time (timestamp)

Employer & Freelancer addresses (payable)

State Variables:

- Mapping for storing jobs using Job IDs.
- Separate mapping to track escrowed amounts.

Events:

- Emit events like JobPosted, ApplicationReceived, PaymentReleased for frontend synchronization and off-chain tracking.

Security:

- ReentrancyGuard: Prevent reentrancy attacks. Used OpenZeppelin's ReentrancyGuard for critical functions.Updated state variables before transferring funds.

- Access Modifiers: Restrict sensitive functions to job creators or contract owners.

4. Core Smart Contract Functions

- postJob(): Employers post a job with a valid budget and title.
- applyForJob(): Freelancers apply to jobs with OPEN status and escrowed funds.
- escrowFunds(): Employers lock job budget into the smart contract.
- markWorkDone(): Assigned freelancer signals job completion.
- releasePayment(): Employer releases payment to freelancer; updates status.
- refundEmployer(): Employer refunds escrow if job is not taken in 7 days.
- initiateDispute() / resolveDispute(): Handles job disputes between freelancer and employer via chat.

5.Testing Strategy

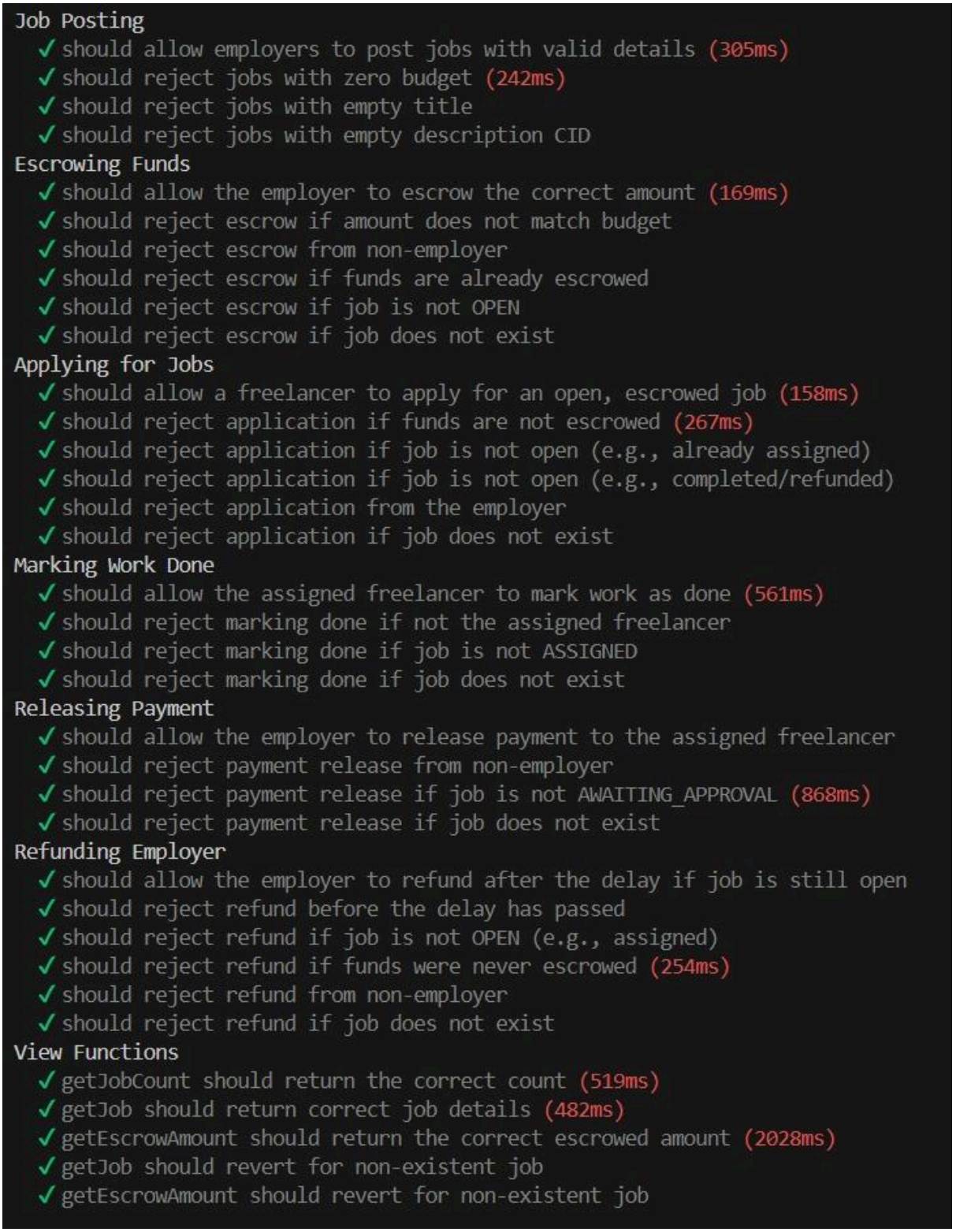
Types of Tests:

- Unit Tests: Validate each function.
- Integration Tests: Test full workflows from posting to payment.
- Edge Cases:
    - Double applications prevention.
    - Invalid release/payment attempts.
    - Refund eligibility timing.
    - Insufficient fund checks.

6. Testing Results

- All major test cases executed successfully.
- Validations passed for:
    - Access control.
    - State transitions.
    - Escrow mechanisms.

- Event emissions.



7. Frontend Features

The frontend is designed for intuitive use by both freelancers and employers, integrating on-chain and off-chain operations seamlessly.

UI Components:

- Job Board: Displays active jobs fetched from the blockchain (on-chain).
- Job Posting Form: Lets employers post jobs (on-chain interaction).
- Application Panel: Freelancers can apply directly (on-chain call).
- Employer Dashboard: Manage jobs, release payments, request refunds (on-chain).
- Freelancer Dashboard: View applied jobs and payment statuses (on-chain).
- Transaction Status Feed: Real-time blockchain update logs via emitted events.
- Messaging Interface: Off-chain chat between employer and freelancer.

On-Chain Frontend Functions:

- Smart contract calls for posting, applying, marking work done, payments.
- MetaMask-triggered transactions for all financial actions.
- Job state fetched from blockchain storage.

Off-Chain Frontend Functions:

- Portfolios, resume links stored off-chain.
- UI updates triggered via event listeners from Web3.js (on-chain events).
- Pinata + IPFS: Pinata’s API is used to upload and pin job descriptions and other metadata to IPFS, keeping the blockchain lightweight and reducing gas fees.
- Messaging System: Real-time communication between freelancers and employers is handled via Socket.io. Messages are stored only when the server is active; otherwise, they're treated as temporary and not persisted.

8. User Experience

Employer Workflow:

1. Connect wallet via MetaMask.
2. Post job (with title, description, and budget).
3. Escrow funds securely.
4. Assign applicant.
5. Release payment or raise dispute.
6. Can resolve dispute via chat with freelancer.
7. Employer can also **delete** the posted job if he feels there is no need at present.

Freelancer Workflow:

1. Connect wallet via MetaMask.
2. Browse and apply for jobs.
3. Complete work off-chain.
4. Get paid on approval or raise dispute.
5. Can resolve dispute via chat.
6. Can **withdraw** application if he wants.

9. Optimization & Gas Efficiency

- Data Types: Chosen for minimal gas (e.g., uint256).
- Mapping Structures: Enable fast lookups.
- Minimal On-Chain Storage: Large content (descriptions) stored off-chain.
- Avoids Loops: No expensive for loops.
- Efficient View Functions: Data retrieval without gas consumption.

10. Privacy Measures

- On-Chain: Job title, budget, addresses.
- Off-Chain: Descriptions, portfolios, and communication.
- User Identity: Only wallet addresses used.
- Escrow Timestamps: Avoid personal data leakage.

11. Bonus Features

- Refund System: Time-based refund to employers (7 days).
- Status Tracking: Monitors job states across lifecycle.
- Dispute System: Admin can resolve disputes fairly using the messaging facility.

12. Conclusion

The Decentralized Job Board successfully implements a trustless, secure, and scalable freelance platform on blockchain. With strong security, functional testing, optimized gas usage, and a user-friendly frontend, it serves as a promising prototype for future decentralized marketplaces

NOTE: We've added these features that we missed during the presentation.

- Multi applicants are allowed to apply for a job and the employer will review their resume/portfolio and will select suitable candidates.
- Disputes raised can be resolved via chat messaging system.
- A **Primary Key** is used in **Pinata + IPFS** to uniquely identify job descriptions, preventing conflicts when the same jobs are reposted after a certain period of time.
- A freelancer can withdraw their job application if they lose interest or decide not to pursue the opportunity.
- An employer can delete a posted job if the requirement is no longer valid or necessary