BharatVote: A Blockchain-Based Secure Voting System

Secure Voting, Stronger India

Archee Arjun • Shivangi Priya • Mohd Sultan • Keshav Gupta

Under the guidance of Professor Pravin Pawar

Agenda

1	**Introduction**	
2	**Deliverables & Study Outputs**	
3	**Existing Solutions**	
4	**Core Features**	
5	**Methodology & Stack**	
6	**Architecture**	
7	**Roadmap & Team Roles**	

Introduction

🔒 What is BharatVote?

- BharatVote is a **secure, remote voting prototype** designed for web and mobile platforms. It uses **blockchain** and **commit–reveal cryptography** to ensure tamper-proof, verifiable, and privacy-preserving elections.

🗂️ Scope of the Study Project (3 Months)

This presentation summarizes the outcomes of our study-focused work, which included:

- **Phase 1 – Proposal**: Core idea, motivation, and project goals
- **Phase 2 – SRS**: Functional and non-functional requirements
- **Phase 3 – Technical Design**: Architecture, UI mock-ups, workflows, and test plan

🎯 **Goal:** *Build a well-defined design plan to guide the actual implementation next semester.*

👥 Who is BharatVote For?

Voters – Participants in campus, housing, or local municipal elections
Election Administrators – Organizers managing small, community-based elections
Auditors – Reviewers ensuring fairness in trusted, small-scale setups

Problem statement

🎯 The Problem with Current Voting Systems

Institutions still rely on **hardware-heavy**, **non-verifiable**, and **physically restrictive** voting systems like EVMs + VVPATs — which aren't built for remote or academic use.

🔍 Key Pain Points:

Opaque voting, insider tampering risks, no remote access, and hardware too costly for campuses.

✅ Our Aim

To design a **simple, auditable, remote-friendly voting prototype** that solves these issues using **blockchain** and **cryptography**, without relying on expensive equipment.

Deliverables

✅ Study Project Submissions (This Semester)

- Phase 1: Project Proposal
- Phase 2: SRS (Requirements Spec)
- Phase 3: Technical Design Document
- Supporting assets: UML diagrams, OpenAPI spec, UI mock-ups
- This presentation

🔧 Foundation for Next Semester (Prototype Phase)

- Smart contracts (Solidity, Sepolia)
- Backend services (mock KYC, OTP, Merkle proof)
- Web & mobile apps (React stack)

Usefulness

🌟 Why BharatVote Is Built for Small-Scale Elections

India hosts thousands of **small and mid-scale elections** — campus councils, union elections, professional body polls — often **run manually by a handful of officials** under immense pressure.
BharatVote replaces manpower-intensive voting with cryptographic trust.

🚀 What It Solves:

- ✅ **Fewer boots, more trust** – No need for dozens of officials; one admin controls the process digitally
- ✅ **No paper, no queues** – Voters cast their ballot from any device, in minutes
- ✅ **No confusion, no disputes** – Blockchain-backed receipts & audit trails eliminate tally errors and bias
- ✅ **No booths, no barriers** – Works for remote students, interns, exchange programs, NRIs

🏛️ Where It Fits:

- Campus elections (BITS, IITs, NITs)
- Student unions, alumni bodies, research councils
- Professional or institutional elections where scale is moderate but **credibility is critical**

BharatVote brings auditability, ease, and inclusion — without needing EVMs, polling stations, or an army of staff.

Review of existing product

EVM + VVPAT (India's National Standard)

🔧 How it works

- Press button → vote stored in standalone hardware
- Paper slip (VVPAT) flashes for 3 s and drops into sealed box

📍 Where used

- 2+ million machines in every Lok Sabha & Assembly election since 2019

⚠️ Key gaps

- Only ~1.5 % slips hand-audited → limited statistical confidence
- Firmware closed-source → insider chip re-flash risk
- No remote voting; logistics of moving machines across 543 constituencies

💡 Lesson for BharatVote

- Make the **audit path public & end-to-end** (blockchain events)
- Design for **remote, device-agnostic access** to cut logistics

Review of existing product

Remote Voting Machine (RVM) Demo 2023

🔧 How it works

- One RVM can map up to 72 constituencies via dynamic ballot screen
- Voter verifies identity at booth, then selects home constituency

📍 Where used

- Prototype demo by ECI and IIT-Madras; not yet rolled out

⚠️ Key gaps

- Still requires physical booth travel for migrants
- Complex configuration; needs new staff training
- No robust biometric / wallet authentication

💡 Lesson for BharatVote

→ Remove hardware dependency: use **browser + MetaMask**

→ Keep admin UI minimal so existing staff need no retraining

Review of existing product

IIT-M × ECI × CDAC Blockchain Kiosk (2020 Pilot)

🔧 How it works

- Special kiosk with fingerprint + webcam → writes vote to private chain
- Commit-reveal not used; vote stored directly

📍 Where used

- Limited on-campus dry-run at IIT-Madras; lab environment

⚠️ Key gaps

- Booth-bound; compromised kiosk = compromised vote
- Private blockchain → public can't audit; code not released
- Extra biometric hardware raises cost

💡 Lesson for BharatVote

→ **Public testnet (Sepolia)** for open auditability
→ Use **commit-reveal** so even a tampered client can't leak the vote
→ Leverage existing webcams/phones, not custom biometrics

Review of existing product

Telangana Smartphone Voting Pilot 2021

🔧 **How it works**

- Android-only “TSEC-eVote” app
- 3-factor auth: Aadhaar name match, selfie liveness, EPIC face match
- Votes stored on permissioned blockchain backend

📍 **Where used**

- Khammam municipal elections (dry run, Oct 2021)

⚠️ **Key gaps**

- Low turnout; many users failed KYC on 2G/3G networks
- Closed code + private ledger drew transparency criticism
- Device lock-in: lose phone = lose vote

💡 **Lesson for BharatVote**

→ **Browser first** with optional RN app = zero install barrier
→ **Public, open-source contracts** to build trust
→ Wallet-based identity allows switching devices seamlessly

Review of existing product

Helios (Open-Source Academic System)

🔧 How it works

- Web-based bulletin board; homomorphic tally; public audit of ciphertexts
- Used in universities & labour-union elections worldwide

📍 Where used

- 2008-present: Princeton grad council, Belgian student unions, etc.

⚠️ Key gaps

- Browser malware can leak your vote before encryption
- User must actively download proof and self-audit → low adoption
- No built-in voter eligibility check (trusts email lists)

💡 Lesson for BharatVote

→ **Commit-reveal** hides vote even on compromised device
→ Auto-generate simple **Tx-hash receipt** instead of manual proofs
→ Add **Merkle whitelist** to enforce eligibility on-chain

Important features of the product

✅ Feature	Why It Matters to Users
Commit–Reveal Privacy	Ballot stays secret until polls close → no early-trend manipulation
Merkle-Proof Voter Eligibility	Only authorised wallets can vote → blocks double / fake ballots
Device-Agnostic Remote Access	Browser + MetaMask or mobile app → vote anywhere, no special hardware
Real-Time Admin Dashboard	One-click phase switch & live tally charts → zero manual counting errors
Immutable On-Chain Audit Trail	Tx-hash receipt for every voter → 100% publicly verifiable results

Methodology

Our Build Stack

- 🖌️ Docs → Google Docs, Figma (key screens)
- ⚙️ Backend → Node.js + Express (mock KYC/OTP)
- ⛓️ Smart Contracts → Solidity on Hardhat, Sepolia fallback
- 🌐 Web → React + Vite (MetaMask)
- 📱 Mobile → React Native (WalletConnect)
- 🛠️ Testing → Hardhat test, Slither, manual click-through

Our 5-Step Workflow

1️⃣ Phase 1 — Brainstorm → Proposal draft → Mentor sign-off

2️⃣ Phase 2 — Interviews → SRS → Peer proof-read

3️⃣ Phase 3 — Architecture + ERD + UI → Build prototype on dev laptop

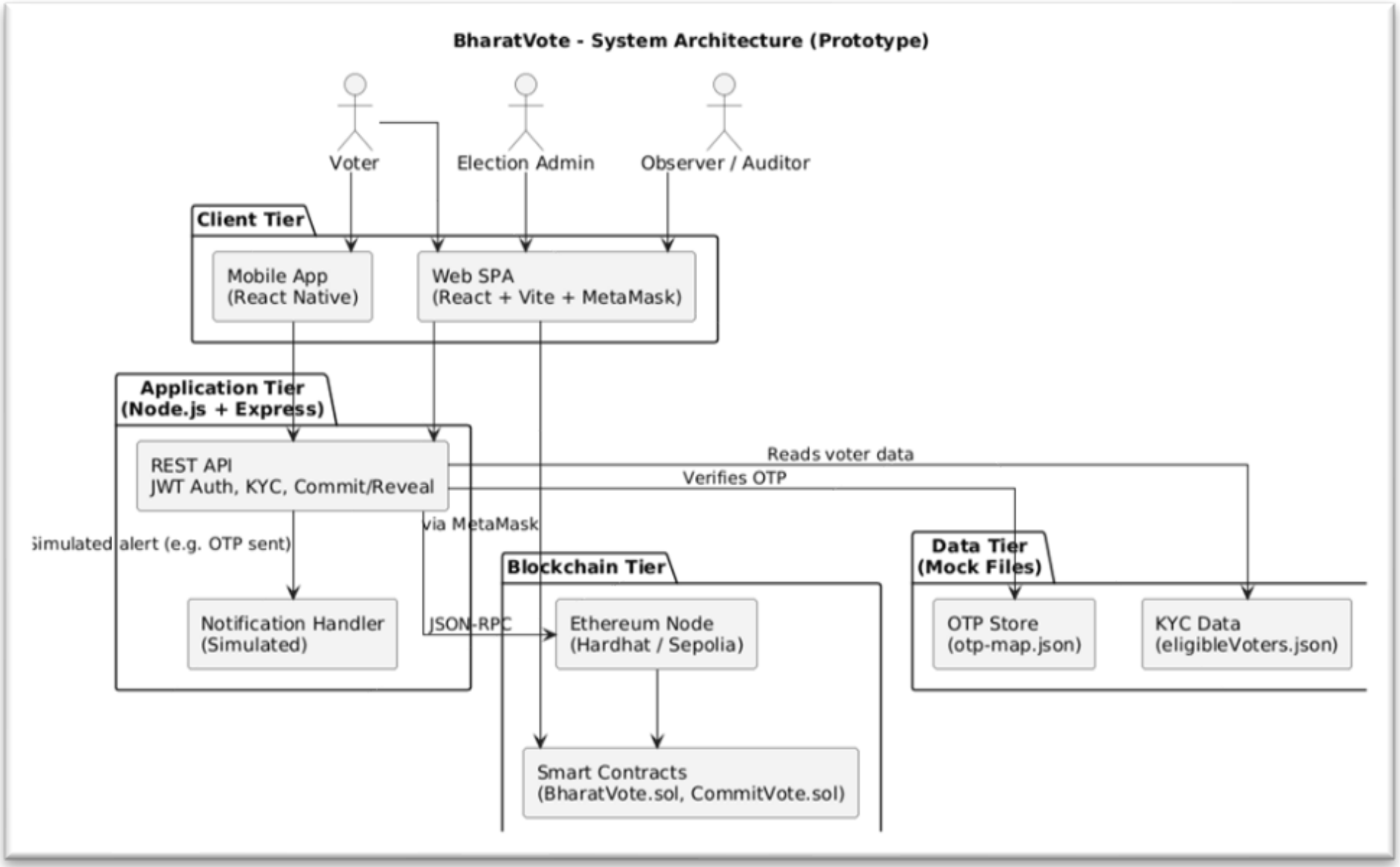
4️⃣ Weekly sync — 30 min Google Meet with mentor; feedback logged in shared sheet

5️⃣ Prototype demo — Localhost + Android emulator; Sepolia as public-chain fallback

Uses of Software

🛠️ Tool	What We Use It For
VS Code + GitHub	Coding + version control (all repos)
Hardhat + Sepolia faucet	Compile / deploy / test Solidity contracts
Node.js 18 + Express	Mock KYC + OTP REST API
React 18 (Vite)	Web SPA — admin dash & voter portal
React Native + Expo	Mobile voter flow on Android / iOS
MetaMask / WalletConnect	Sign commit / reveal txns

System Architecture



Budget

Budget Snapshot
💸 **Total Development Cost: ₹0**

Our prototype is built entirely using **free, student-accessible tools**:

Resource	Tool/Service	Cost
Smart Contract Dev	Hardhat, Sepolia Testnet	₹0 (free test ETH)
Web & Mobile UI	React, React Native, Expo Go	₹0
Backend API	Node.js + Express (local)	₹0
Hosting (demo)	Netlify, Render (free tier)	₹0
Face Matching	TensorFlow.js (local, mock)	₹0
Documentation	Google Docs, Figma (edu plan)	₹0

🛠 **Goal:** Validate design feasibility **without paid infrastructure**.
📌 For campus-scale pilot, only **cloud credits or SMS API** may need minimal funding.

How do you plan to implement the project?

Component	Technology	Key Features
Smart Contracts	Solidity + Hardhat	Commit-reveal voting, Admin controls, Merkle tree verification
Backend API	Node.js + Express	KYC verification, Voter eligibility, Merkle proof generation
Web Frontend	React TypeScript	MetaMask integration, Responsive UI, Admin panel
Mobile App	React Native	Cross-platform, MetaMask deep links, Touch-optimized

Implementation Phases

1. **Phase 1** - Smart Contract Development & Backend API
2. **Phase 2** - Web & Mobile User Interfaces
3. **Phase 3** - Integration & Security Hardening
4. **Phase 4** - Testing & Deployment

Expected Timeline: 10 weeks | **Target**: 99.9% uptime, <3s transactions

Responsibilities

Team Member	Role	Responsibilities
Archee Arjun	Blockchain	Smart contracts, Hardhat, Team coordination
Shivangi Priya	Frontend	React web app, MetaMask integration, UI/UX
Mod Sultan	Backend	Node.js API, KYC services, Database, Security
Keshav Gupta	Mobile	React Native app, Testing, Quality assurance

Conclusion

BharatVote aims to transform how small-scale elections are conducted by addressing key challenges like lack of transparency, fraud risk, limited accessibility, and paper-based inefficiencies. Using blockchain for tamper-proof records, commit–reveal cryptography for privacy, and Merkle proofs for voter validation, it ensures secure, verifiable voting. Its web and mobile interfaces with MetaMask integration make voting accessible to all, including remote and disabled users. With real-time tallying, open-source smart contracts, and a user-friendly admin dashboard, BharatVote delivers trust, transparency, and ease — a prototype built to restore faith in democratic participation and pave the way for scalable, cost-effective digital elections.