Product Requirements Document (PRD): Scrum Poker MVP

## **1. Purpose & Vision**

**Purpose:** Deliver a minimal viable Scrum Poker application enabling remote-first teams to collaboratively estimate user stories. The MVP will showcase:

* **Real‑time voting** for live sprint planning
* **Asynchronous voting** for distributed participants
* **AI‑driven estimation suggestions** using a local Llama.cpp model
* **Basic authentication** and role‑based access

**Vision:** Lay the foundation for a full-featured Scrum Poker suite by building a lean, stable prototype with essential workflows, ready for demos and initial user feedback.

## **2. Objectives & Success Metrics**

| **Objective** | **Metric** | **MVP Target** |
| --- | --- | --- |
| Enable live estimation sessions | Time to reflect votes (95th pct) | <200 ms |
| Support async voting | % sessions completed async | ≥80% |
| Provide AI estimation suggestions | AI suggestion latency (95th pct) | <300 ms |
| Achieve basic auth and role control | Auth flow completion time | <3 s |
| Deliver polished UI for demos | UX satisfaction (qualitative) | Positive feedback in demo |

## **3. MVP Scope**

**In‑Scope:**

* User login via Keycloak OAuth2 (public client)
* Create/join session (real‑time & async modes)
* Submit and reveal votes (Fibonacci scale)
* Display AI suggestion per story
* Basic session summary export (CSV)
* Responsive web UI (React + Tailwind)
* Backend API & WebSocket (Node.js + Socket.IO)
* Persistent storage (PostgreSQL + Redis)
* Local AI service (FastAPI + llama.cpp)

**Out‑of‑Scope:**

* Mobile native app
* Multi‑tenant data segregation
* Advanced customization (themes, custom scales)
* Detailed analytics & velocity tracking
* Enterprise features (SSO, audit logs)

## **4. Business Requirements (MVP)**

| **ID** | **Description** |
| --- | --- |
| BR-MVP-1 | Real-time vote broadcasting: votes appear to all participants within 200 ms. |
| BR-MVP-2 | Asynchronous voting: users can vote up to 24 h after session creation. |
| BR-MVP-3 | AI estimation: generate and display a suggested story point for each story. |
| BR-MVP-4 | Authentication: users must log in; only ScrumMasters can create/terminate sessions. |
| BR-MVP-5 | Session summary: export simple CSV with stories, votes, and AI suggestions. |

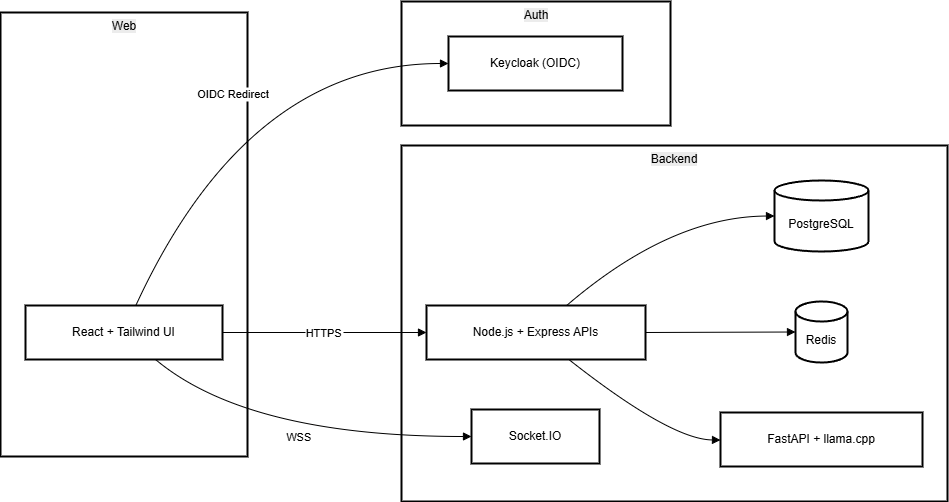
## **5. Functional Requirements & User Stories**

| **Story ID** | **As a…** | **I want to…** | **So that…** |
| --- | --- | --- | --- |
| US-1 | Developer | join a live estimation session | I can cast my vote in real time |
| US-2 | Scrum Master | reveal all votes when ready | we avoid anchoring bias |
| US-3 | Remote Developer | vote asynchronously within 24 h | I can estimate when convenient |
| US-4 | Developer | see AI’s suggested estimate alongside my vote | I have a baseline to discuss |
| US-5 | Scrum Master | download session results as CSV | I can share the outcomes |

## **6. Non‑Functional Requirements (MVP)**

| **Category** | **Requirement** |
| --- | --- |
| Performance | API responses <200 ms (95th percentile) |
| Reliability | 99.5% uptime for main endpoints |
| Usability | First‑time users onboard in <3 min without training |
| Security | TLS for all traffic; JWT validation for API/WebSocket |
| Maintainability | 70%+ automated test coverage; CI pipeline for builds and tests |

## **7. Architecture Overview**



## **8. Technical Specification (MVP Tech Stack)**

* **Frontend**: React 18 + Vite + Tailwind CSS; Socket.IO client; oidc-client-ts for auth
* **Backend**: Node.js 18 + Express; Socket.IO server; JWT validation via jwks-rsa; Axios HTTP client
* **AI Service**: Python 3.10 + FastAPI; llama.cpp; HTTPX for client
* **Database**: PostgreSQL 15; Knex.js for migrations/queries
* **Cache**: Redis 7 for pub/sub and async session state
* **Auth**: Keycloak 21 (public realm, public client)
* **Containerization**: Docker multi‑stage
* **Orchestration**: Docker Compose for MVP (local/demo)
* **CI/CD**: GitHub Actions for build, test, Docker push, and Compose deployment

## **9. Low‑Level Technical Specifications (MVP)**

### **9.1 Frontend**

* **Pages**: Login, Home (session list), Session (story list & voting)
* **Components**: Card, VoteList, AIHint, SessionHeader, SessionSummary
* **State**: Zustand store: session, votes, suggestions
* **Services**: api.ts for REST; socket.ts for real-time
* **Auth**: oidc-client-ts with PKCE
* **Styling**: Tailwind JIT, custom Eduvibz color palette

### **9.2 Backend (API & WebSocket)**

* **Entry**: index.js launching Express and Socket.IO
* **Routes**: /api/sessions, /api/sessions/:id/vote, /api/sessions/:id/results
* **WebSocket Events**: join-session, vote-submitted, reveal-votes
* **Middleware**: JWT validation, requireRole('ScrumMaster')
* **Services**: SessionService, VoteService, AISuggestionService

### **9.3 AI Service**

* **Endpoint**: GET /estimate?storyId=&description=
* **Model**: llama.cpp quantized LLaMA 7B loaded on startup
* **Prompt**: Fibonacci estimation template
* **Cache**: Redis key ai:{storyId} TTL 24h

### **9.4 Database & Cache**

* **Migrations**: Flyway scripts for tables users, sessions, stories, votes, ai\_suggestions
* **Indexes**: Primary keys on UUID fields; FK indexes on session/story relations
* **Cache Keys**: session:{id}:state, ai:{storyId}

### **9.5 DevOps (Local MVP)**

* **Docker Compose**: services: frontend, api, ai-service, postgres, redis, keycloak
* **Compose ports**: 3000 (frontend), 4000 (API), 8000 (AI), 5432 (DB), 6379 (Redis), 8080 (Keycloak)
* **Startup Order**: Keycloak → DB + Redis → AI → API → Frontend

## **10. Demo & Prototype Goals**

* **User Flow**: login → create/join session → add stories → cast votes (real-time & async) → reveal votes → view AI suggestion → export CSV
* **UI Polish**: responsive, clear card animations, simple dialogs, session summary modal
* **Deployment**: one-click demo via Docker Compose on a single VM or dev laptop

*End of Scrum Poker MVP PRD*