Both Sides - PRD

Product Requirements Document (PRD)

Project: Both Sides ? AI-Powered Debate & Critical Thinking App (TimeBack-Optional Integration)

1. Overview

The Both Sides app is an AI-powered platform designed to foster critical thinking, empathy, and ideological flexibility in students by pairing them with peers of differing viewpoints. Students participate in moderated debates, often defending perspectives opposite to their own, and reflect on the experience. This PRD outlines an MVP architecture that can run independently or integrate with the school's TimeBack database via API.

2. Goals

- Promote civil discourse and empathy among students.
- Teach evidence-based reasoning through structured debates.
- Build adaptable architecture that can work with or without TimeBack integration.
- Ensure FERPA compliance and school data privacy standards.

3. Non-Goals

- The app is not intended for public political advocacy.
- Will not use real student identities in debates (anonymized to peers).
- Will not serve as a public-facing social media platform.

4. Key Features

- Onboarding survey to map belief openness, ideological axes, and opinion plasticity.
- AI-powered debate matching based on profile embeddings.
- Anonymous, real-time debate sessions with AI moderation.
- Post-debate reflection prompts and learning summaries.
- Teacher dashboard for session management.
- Optional TimeBack integration for class rosters and enrollments.

5. Architecture

Core Components

- **Frontend**: Next.js, Tailwind, shadcn/ui, Clerk authentication.

Both Sides - PRD

- **Backend**: NestJS, REST + WebSockets, BullMQ for jobs.
- **Database**: PostgreSQL + pgvector, S3 storage.
- **AI Services**: OpenAI for moderation, summarization, and embeddings.
- **Realtime**: Ably (MVP) or API Gateway WS (Enterprise).

Integration Layer

- **RosterProvider Interface** with two implementations:
  - `TimeBackRosterProvider`: Syncs rosters/enrollments from TimeBack API.
  - `MockRosterProvider`: Uses CSV/JSON for demo environments.
- Integration can be enabled/disabled without impacting core functionality.

6. Data Model (Key Tables)

- `users`, `profiles`, `classes`, `enrollments`, `matches`, `conversations`, `messages`, `reflections`, `moderation_events`, `audit_log`.
- External ID fields for TimeBack mappings: `timeback_*` columns.

7. Privacy & Compliance

- Row-Level Security for per-org isolation.
- Data retention policies configurable per school.
- TLS in transit, encrypted storage at rest, KMS keys in enterprise deployments.
- Audit logs for all sensitive operations.

8. Deployment Plan

MVP:

- Frontend: Vercel
- API: Railway
- DB: Neon Postgres
- Cache/Jobs: Upstash Redis
- Realtime: Ably

Enterprise:

- Frontend: AWS Amplify or SST on CloudFront

Both Sides - PRD

- API/Workers: AWS ECS Fargate
- DB: Aurora Postgres
- Cache: ElastiCache Redis
- Realtime: API Gateway WebSockets

9. Risks & Mitigations

- **Risk**: Building directly into TimeBack without approval.
  - **Mitigation**: Abstract integration and build with mock data first.
- **Risk**: API dependency causing downtime.
  - **Mitigation**: Cache last synced rosters and allow offline mode.

10. Next Steps

1. Build core app with MockRosterProvider.
2. Present demo to school using fake data.
3. Upon approval, enable TimeBack integration in staging.
4. Conduct pilot with one class.