Resilient Live Polling System

Live Link: https://livepollingsystem-kappa.vercel.app

Project Overview

The **Resilient Live Polling System** is a real-time web application that enables teachers to create and manage live polls while students participate in real-time. The system is designed with resilience as a core principle, ensuring that users can recover from page refreshes, network interruptions, and late joins without losing state or data.

Key Differentiators

- **State Recovery**: Survive page refreshes during active polls

- **Timer Synchronization**: Server-authoritative time with client sync

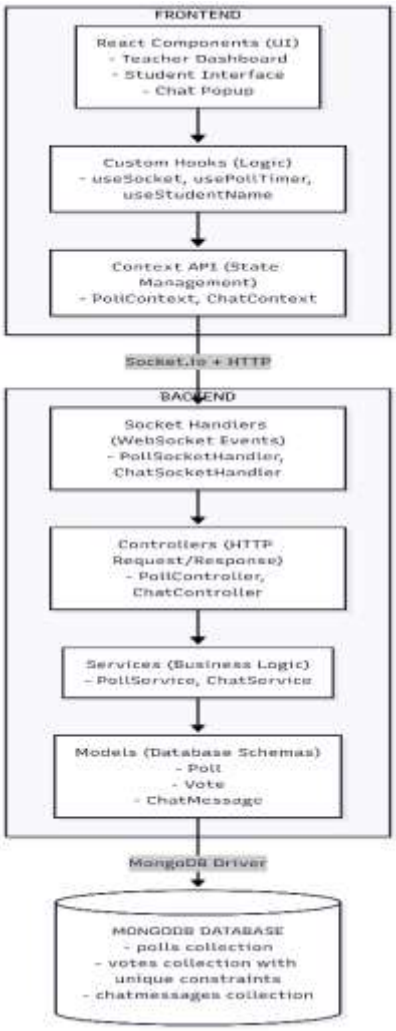
- **Race Condition Prevention**: Database-level duplicate vote prevention

- **Real-time Communication**: Socket.io for instant updates

- **Persistent Storage**: MongoDB for poll history and chat

System Architecture

Architecture Pattern: Clean Architecture + MVC



Backend Architecture Principles

1. Separation of Concerns

- **Models**: Only define data schemas, no logic

- **Services**: All business logic and database operations

- **Controllers**: Only handle HTTP request/response

- **Socket Handlers**: Only handle WebSocket events, delegate to services

2. Single Responsibility

- PollService.js: Create polls, record votes, fetch results

- `PollController.js: Parse HTTP requests, validate, respond

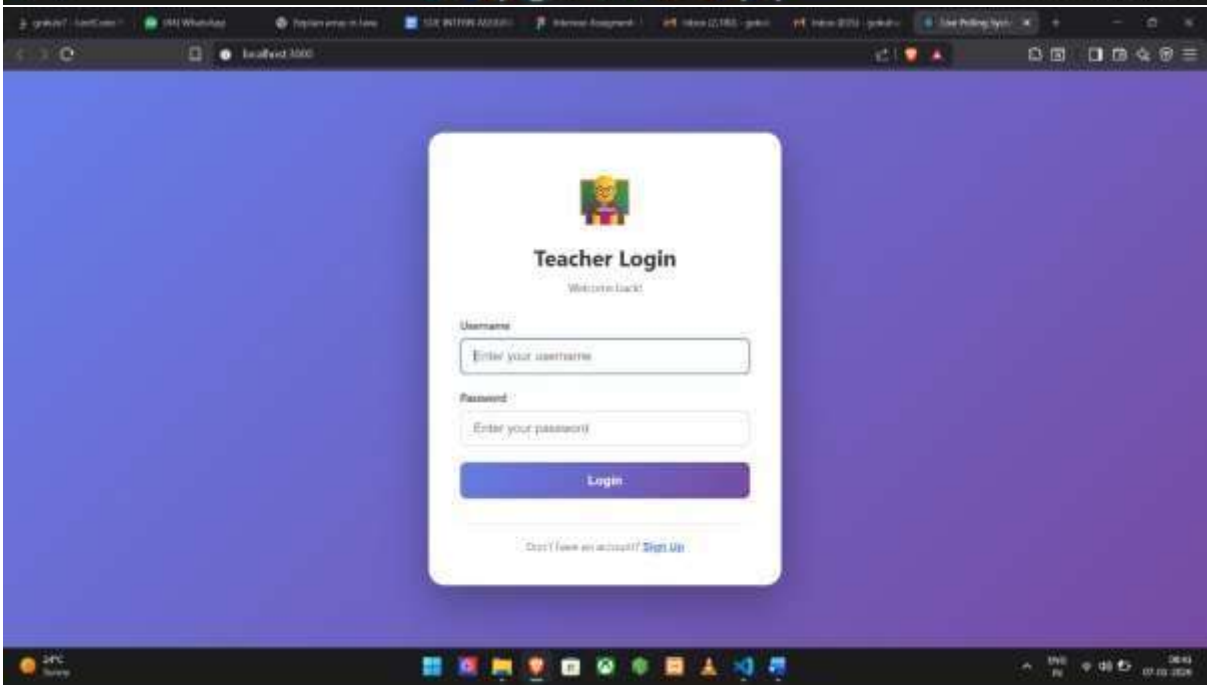
- PollSocketHandler.js: Manage WebSocket connections, emit events

3. DRY (Don't Repeat Yourself)

- Services are shared between HTTP controllers and Socket handlers

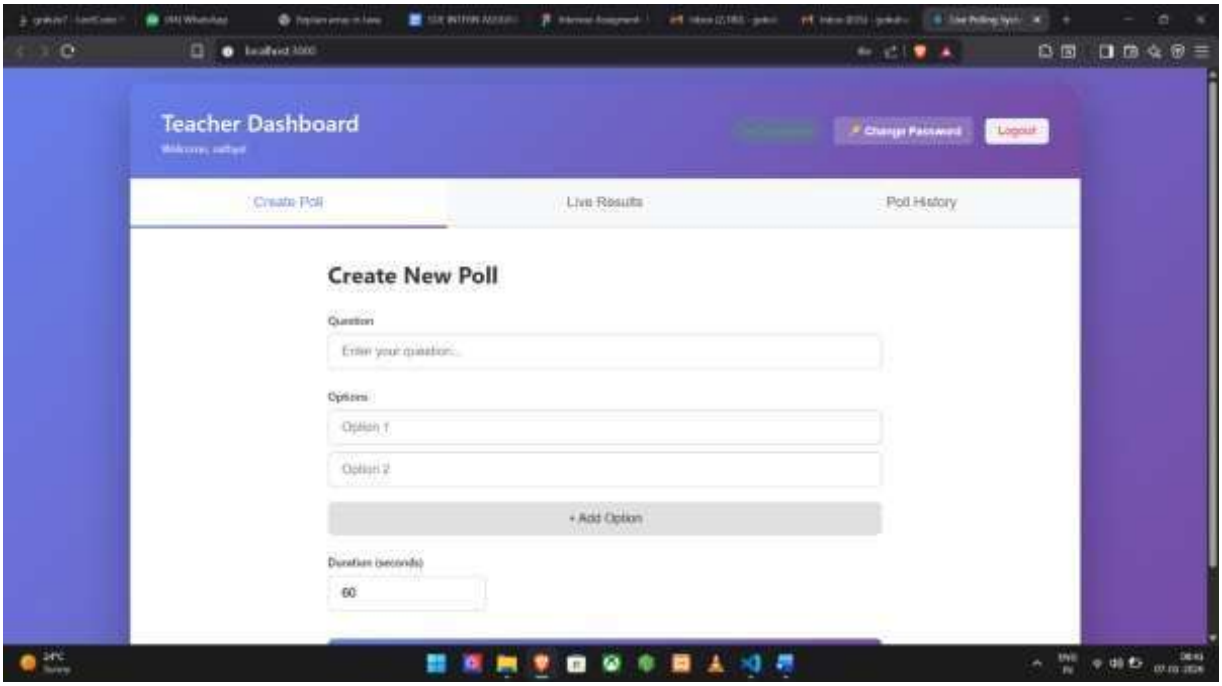
- No duplicate business logic

Teacher Persona



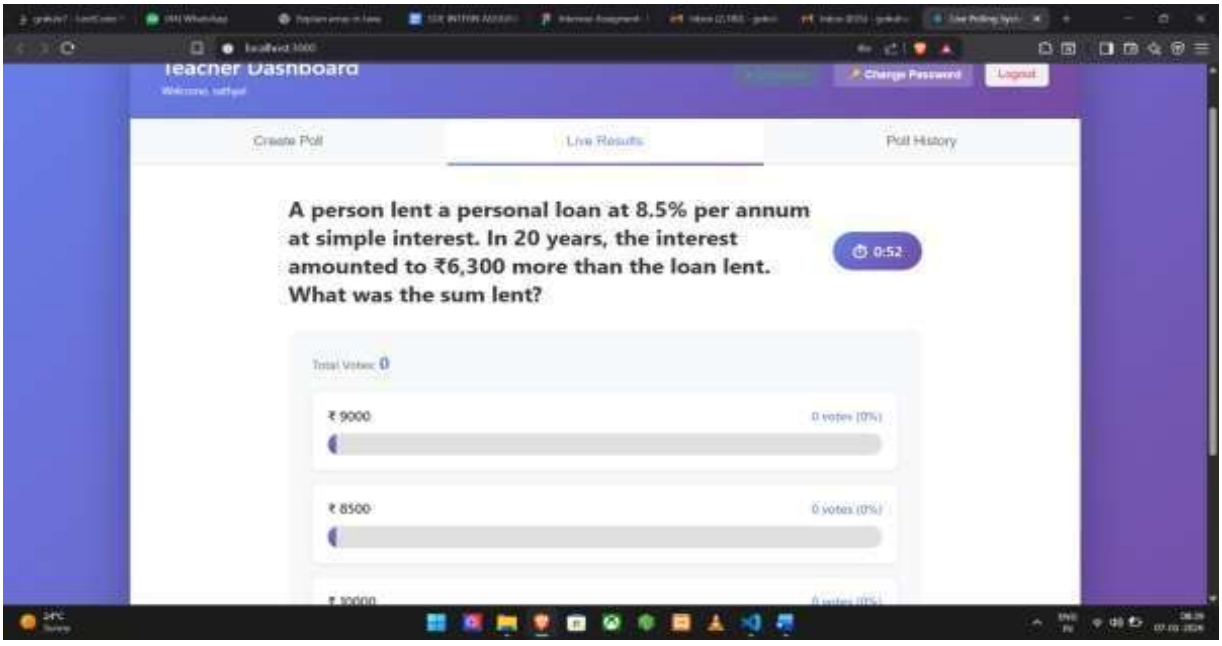
1. **Poll Creation**

- Create question with text input
- Add 2-6 options dynamically
- Set timer duration (10-300 seconds)
- Validation: Question and at least 2 options required



2. **Live Dashboard**

- Real-time vote counts per option
- Percentage calculation
- Visual progress bars
- Auto-updates via Socket.io



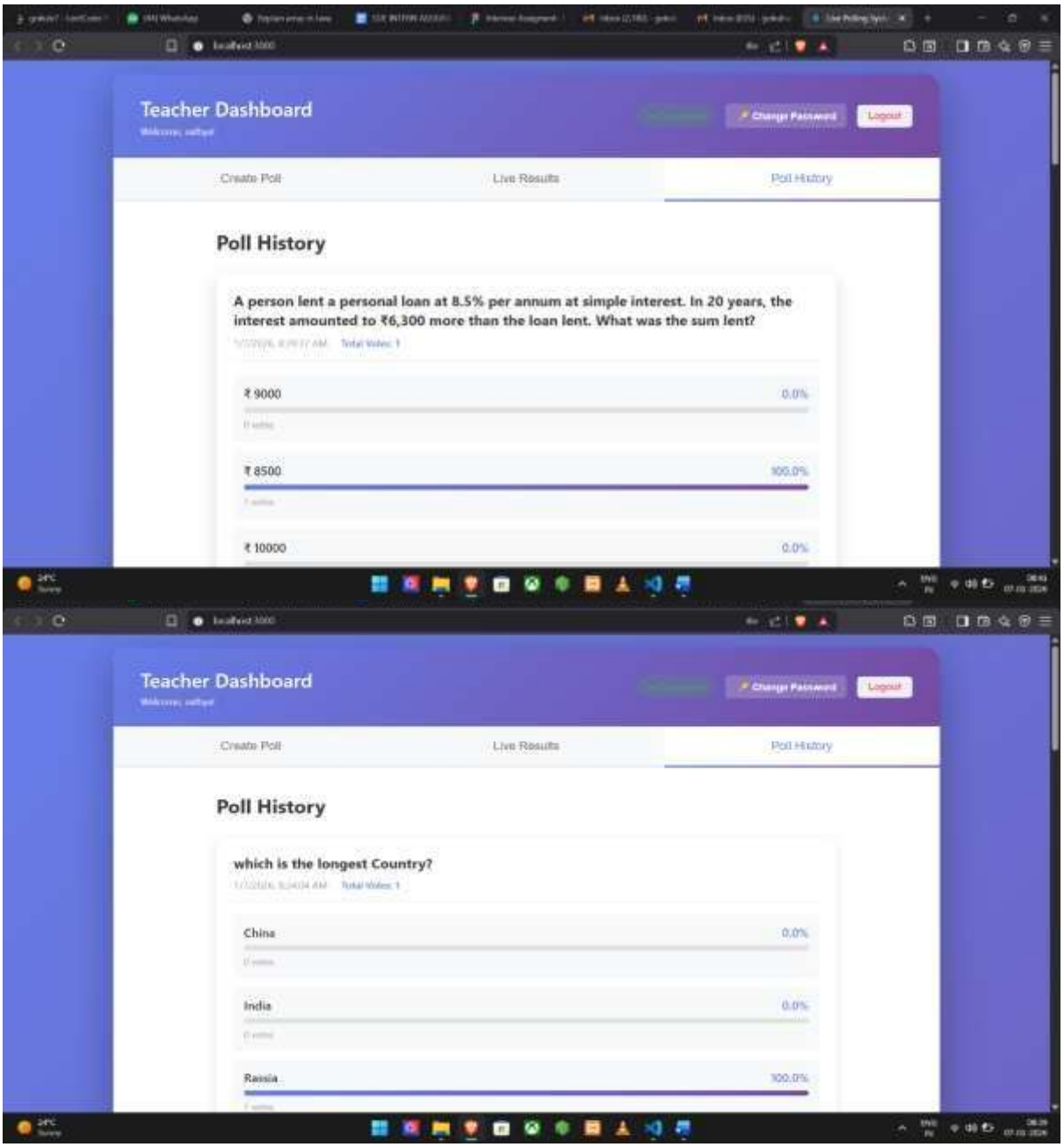
3. **Poll History**

- Fetch past polls from MongoDB

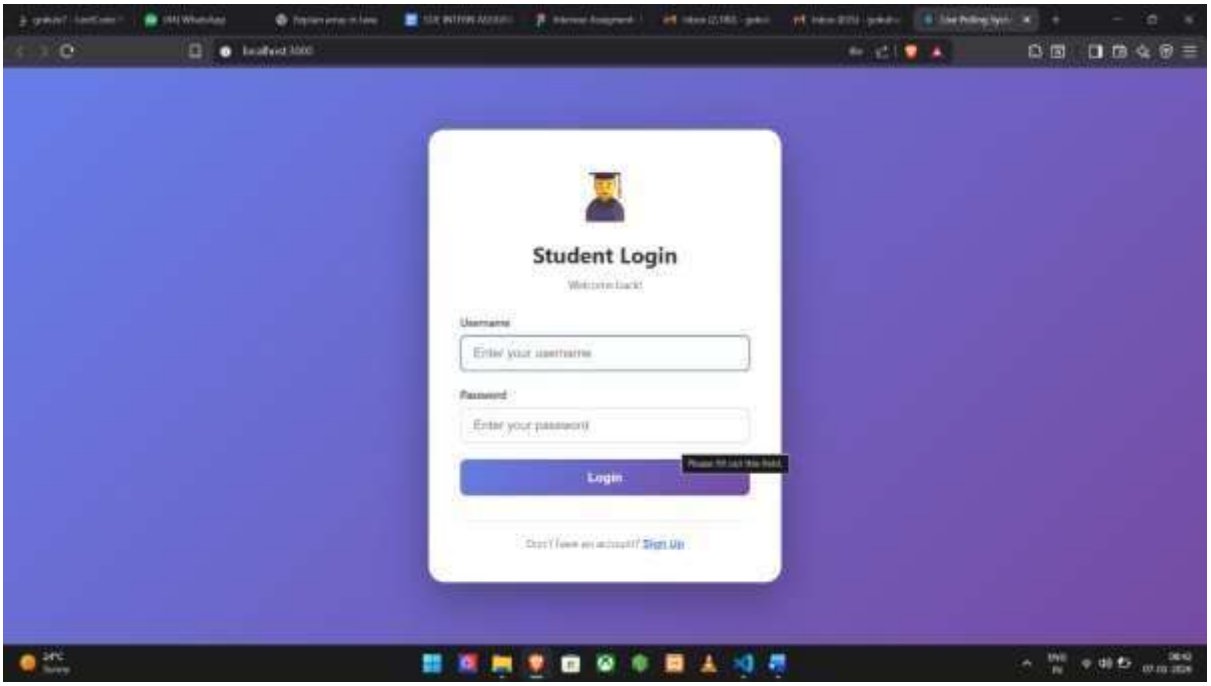
- Display question, options, final results

- Timestamps (created, ended)

- No local storage - all from database



Student Persona



1. **Onboarding**

   - Enter name on first visit

   - Unique per browser tab (sessionStorage)

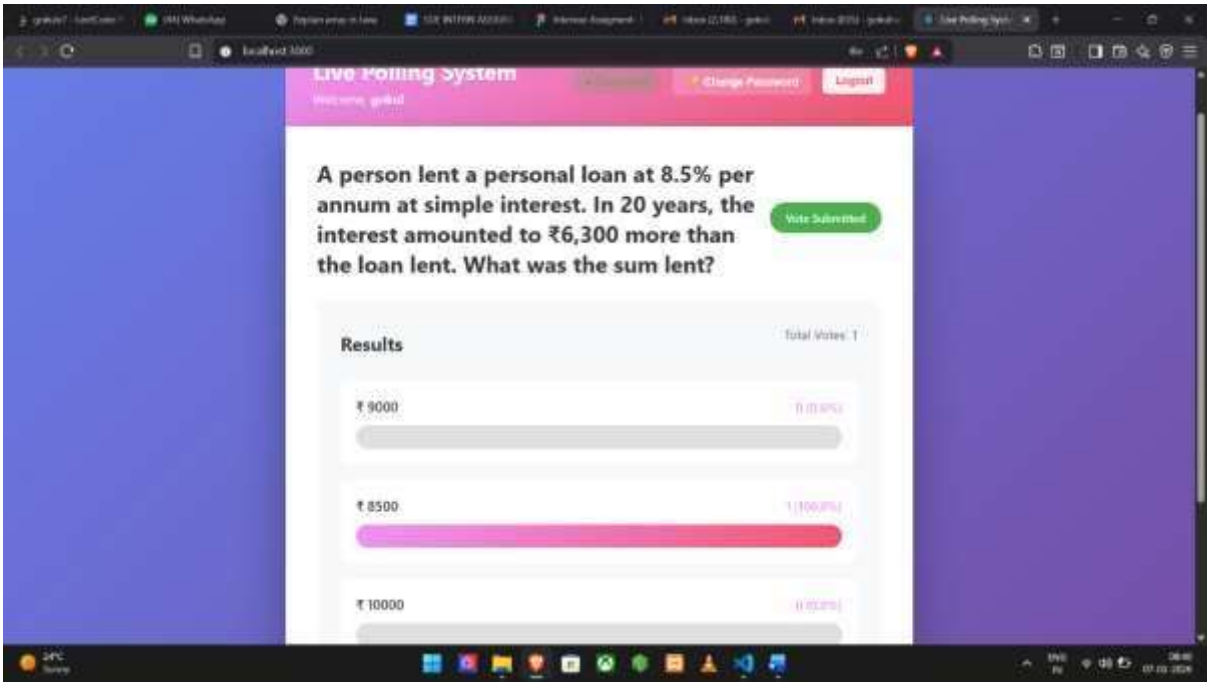
   - Name persists until tab closes

2. **Real-time Interaction**

   - Receive poll instantly when teacher creates it

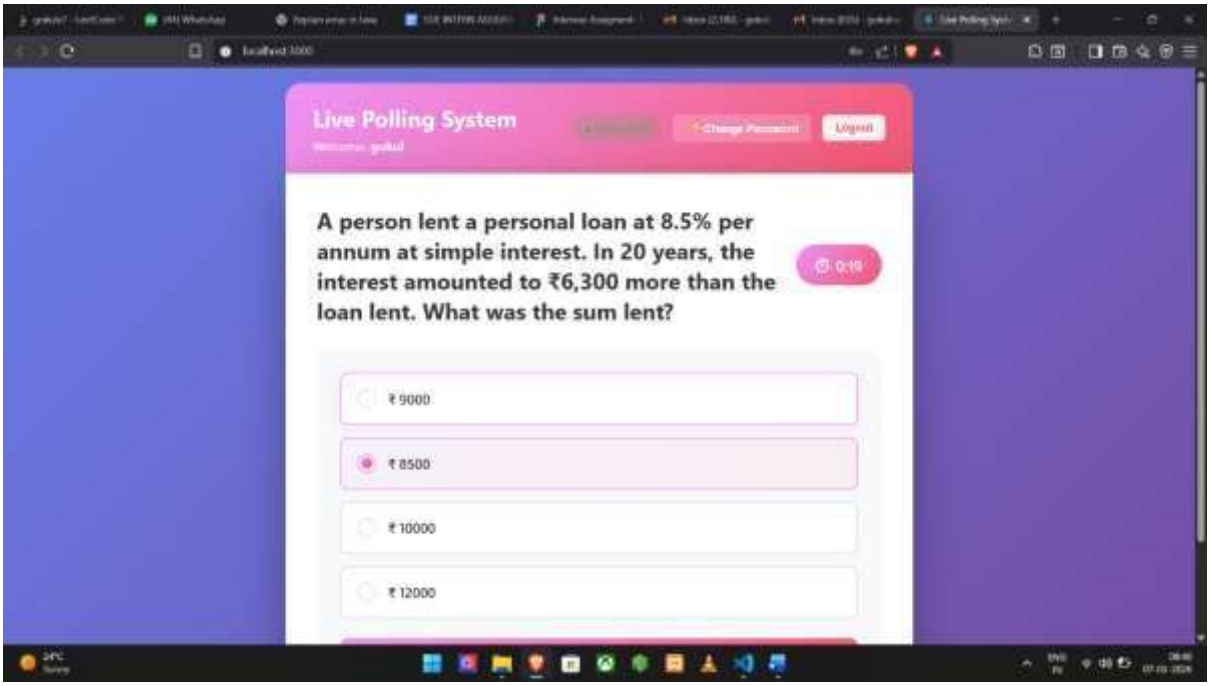
   - See all options

   - Submit vote within time limit



- Late joiners see accurate countdown

- Example: 60s poll, join at 10s → shows 50s

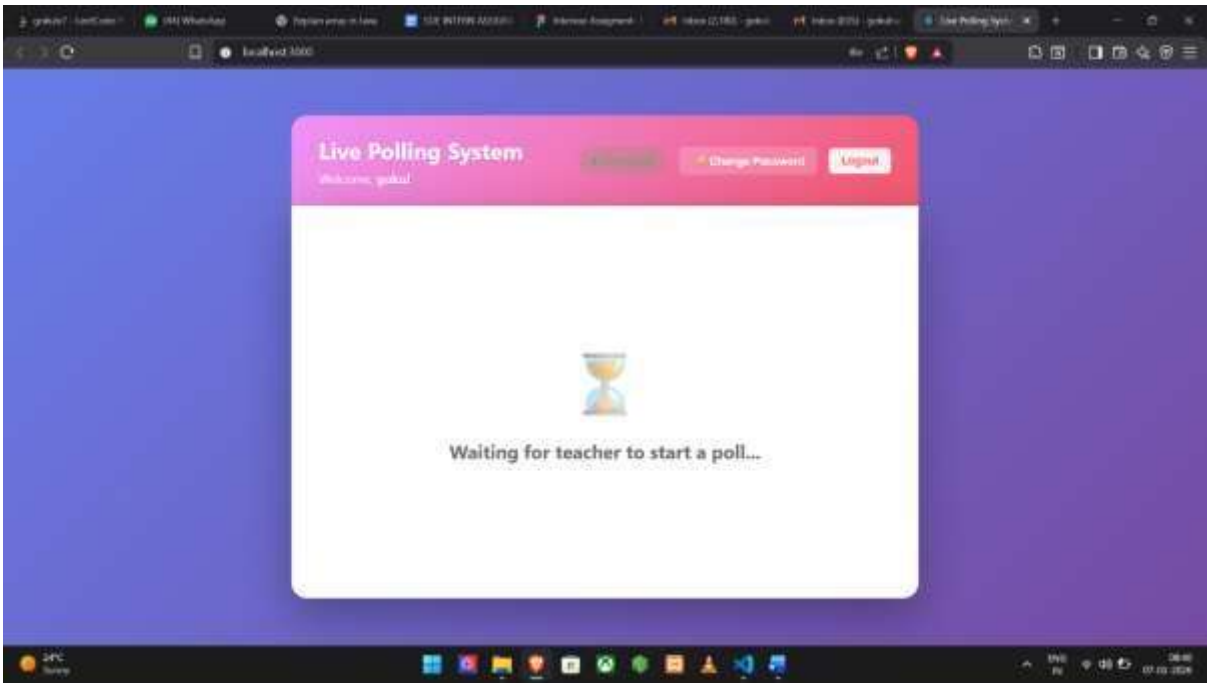


4. **Voting**

- One vote per poll per student

- Immediate feedback on submission

- Cannot vote after time expires



Resilience Features

1. State Recovery

Scenario: Teacher refreshes browser during active poll

Expected Behavior:

- Poll remains active on server

- On refresh, frontend fetches current poll state via `/api/polls/current`

- UI resumes showing live results

- Timer continues from actual remaining time

2. Timer Synchronization

Scenario: Student joins 30s into a 60s poll

Expected Behavior:

- Server stores poll creation time in MongoDB

- Server calculates: remainingTime = duration - (now - createdAt)

- Student receives remainingTime: 30 seconds

- Client countdown starts from 30s, not 60s

3. Race Condition Prevention

Scenario: Student spams vote button or opens dev tools to call API multiple times

Expected Behavior:

- First vote succeeds and is recorded

- Subsequent votes fail with error

- Final results show only 1 vote

Last Updated: January 11, 2026

Version: 1.0.0