InMyTime

SoftWare Engineering Project

Humbat Jamalov

Rauf Jabarov

-

MirSaid Hesenzade

Huseynova Fakhriyya

InMyTime

The Problem: Scheduling Challenges

Scheduling a meeting often turns into a long back-and-forth. It’s time-consuming, inefficient, and frustrating for groups. Most teams struggle to quickly find a time that works for everyone.

InMyTime

Solution Overview

Our solution is a web-based platform that helps groups quickly find a common available time. It collects everyone’s availability, visualizes the results, and automatically highlights the best meeting options. This makes scheduling fast, transparent, and effortless.

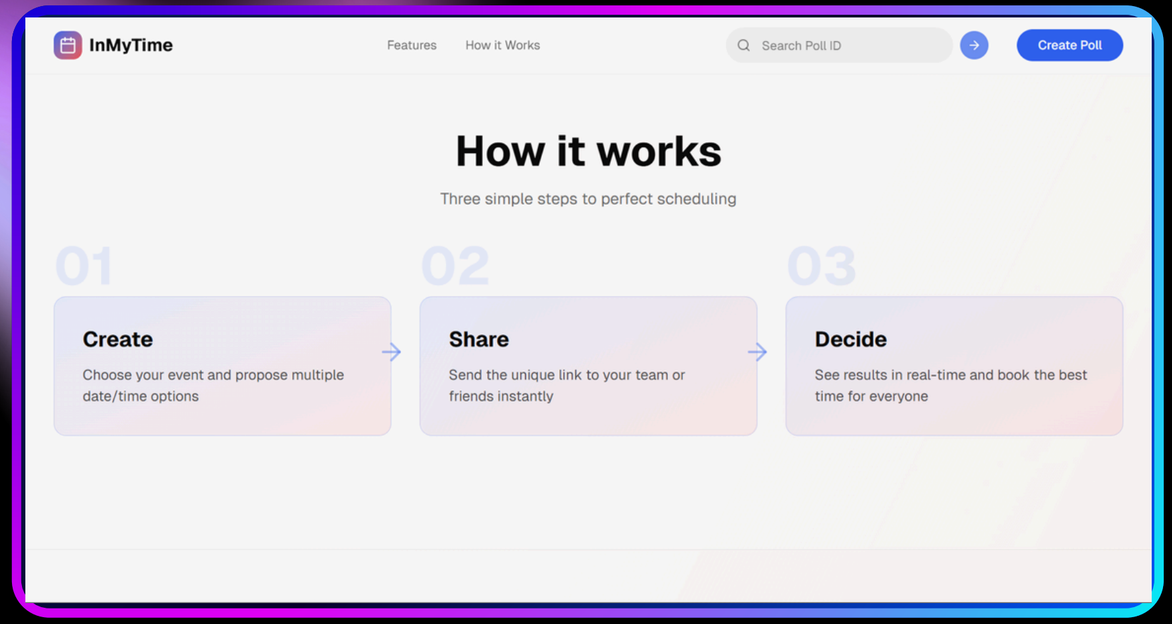
InMyTime

OUR COMPETITORS

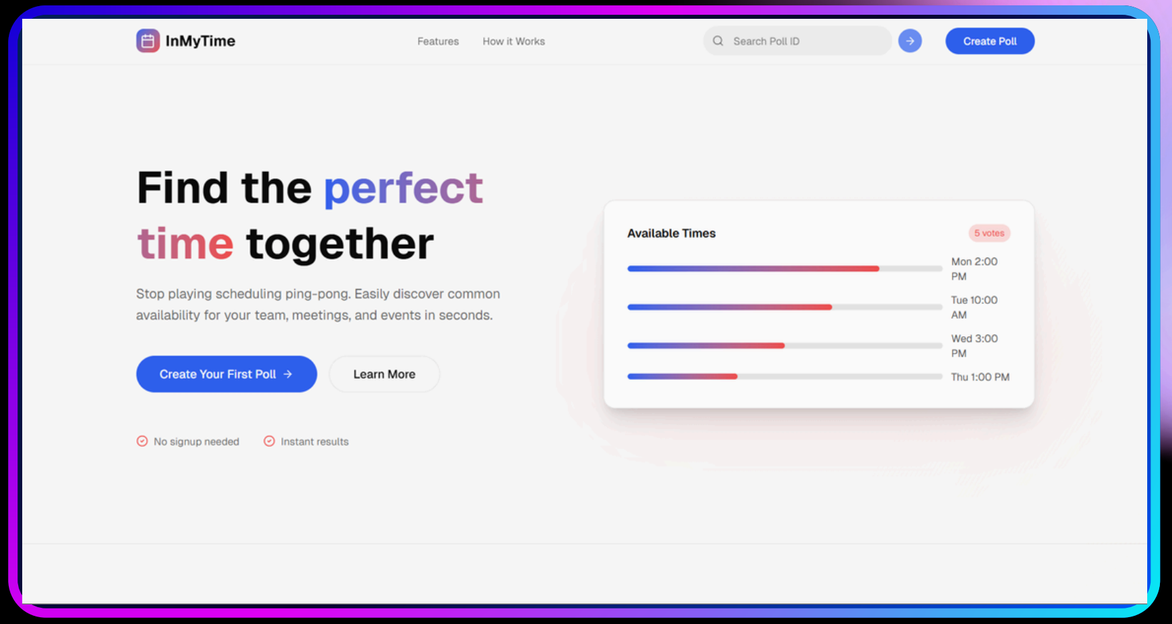
	InMyTime	When2Meet	Doodle
Real-time availability updates	✓	✗	✗
No Registration Required	✓	✓	✗
Host-controlled final selection	✓	✗	✓

InMyTime

Overview of Our Platform



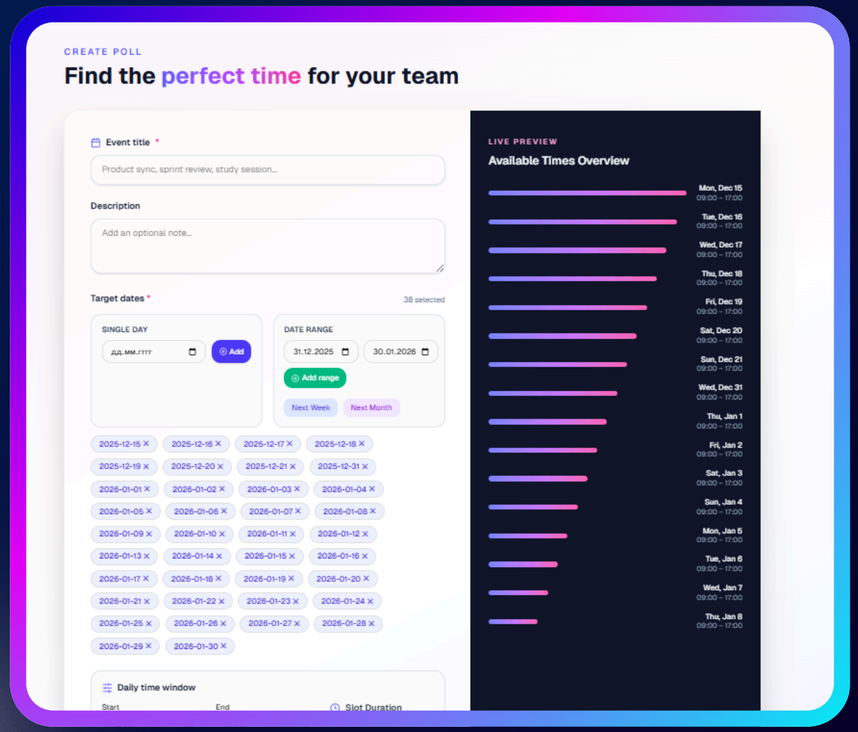
Users create a poll, share it with participants, and instantly see the combined availability. The process is simple and intuitive, making scheduling smooth and fast.



The landing page highlights the core value of the platform: finding the perfect meeting time effortlessly. It clearly showcases the main features

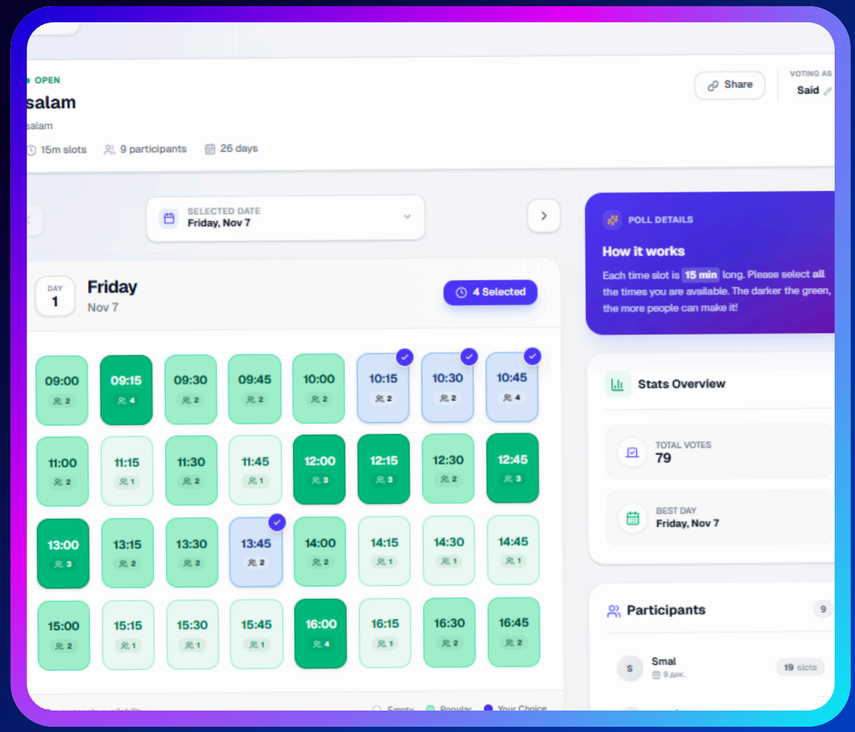
InMyTime

Create Poll



Users select dates, set time ranges, and choose slot duration. A live preview helps them understand how the final schedule will look before sharing.

Participants choose their available slots on an interactive grid, with real-time updates, color-coded availability, and clear statistics to guide scheduling decisions.

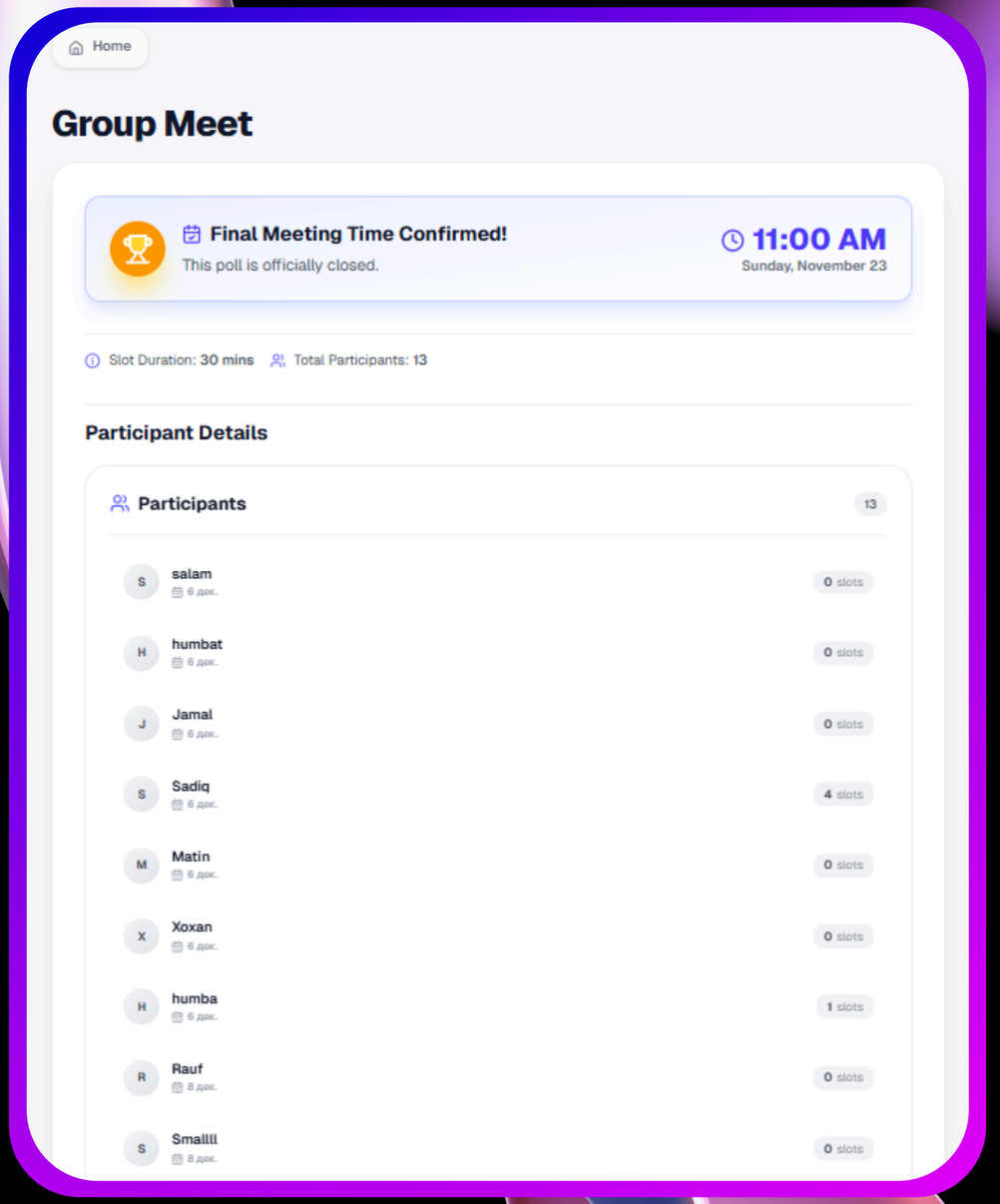


Voting Page

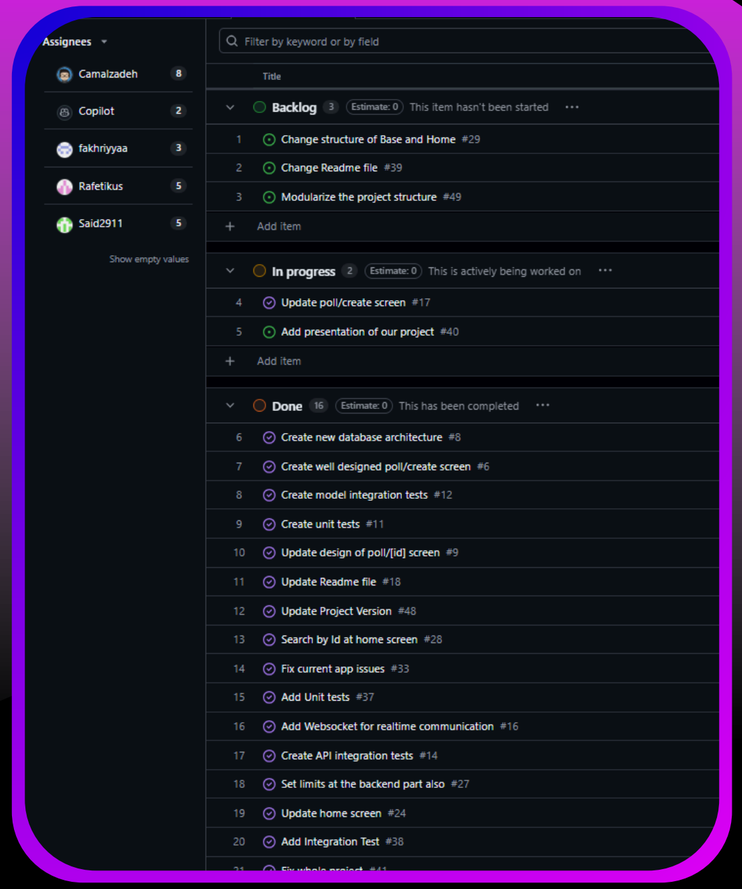
InMyTime

Finalize Feature

After everyone submits their availability, the host selects the best meeting time and closes the poll. A clear confirmation is shown to all participants, making the final step simple and transparent.



InMyTime



DEVELOPMENT WORKFLOW

✓ Task Management
- GitHub Issues used to track features and bugs

🔧 Branching Strategy
- Feature branches kept main branch stable

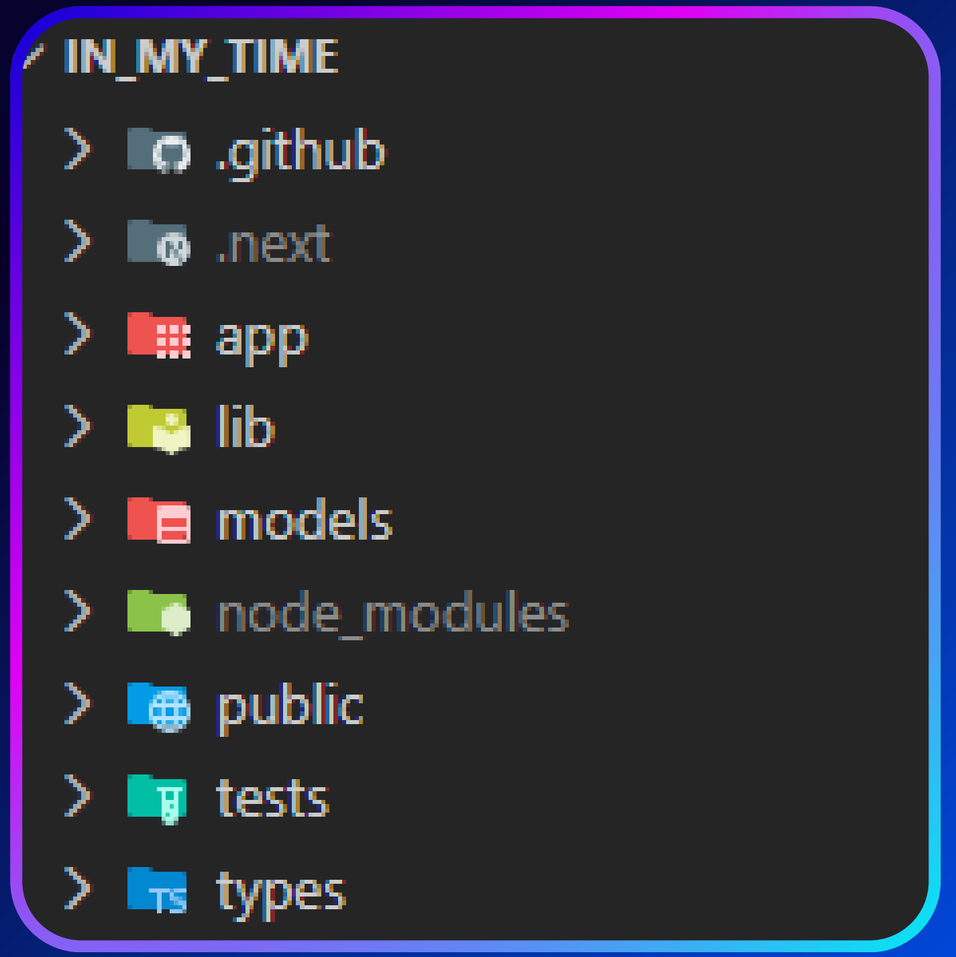
📝 Pull Requests
- Code reviewed before merging

🧪 Testing
- Integration tests run before each merge

InMyTime

Project Structure

Our project is organized into clear modules: **app** for pages and API routes, **components** for reusable UI blocks, **lib** and **utils** for shared logic, **models** for data schemas, **types** for TypeScript definitions, and **tests** for automated testing.



InMyTime

Application Architecture

The project uses TypeScript across both frontend and backend. The frontend is built with Next.js and React, while the backend uses Next.js API routes with MongoDB and Mongoose. Jest is used for integration testing.

Example of FrontEnd (Header.tsx)

```tsx
"use client";
import { useState } from "react";
import Link from "next/link";
import { Calendar, Search, X } from "lucide-react";
import { UI_PATHS} from "@/lib/routes";
import HeaderSearch from "@/app/components/search/HeaderSearch"


export default function Header() {
    const [isSearchOpen, setIsSearchOpen] = useState(false);

    return (
        <nav className="sticky top-0 z-50 border-b border-borde
            <div className="max-w-7xl mx-auto px-4 sm:px-6 lg:p
                <div className="flex items-center justify-betwe
                    <Link href="/" className="flex items-center
                        <div
                            className="w-8 h-8 bg-gradient-to-b
                            <Calendar className="w-5 h-5 text-p
                        </div>
                        <span className="text-xl font-bold text
                    </Link>

                    <div className="hidden md:flex items-center
                        <a href="#features"
                           className="text-sm text-foreground/6
                            Features
                        </a>
                        <a href="#how-it-works"
```

Example of BackEnd (route.ts)

```ts
import {NextRequest, NextResponse} from 'next/server';
import {connectDB} from '@/lib/mongodb';
import {Poll} from '@/models/Poll';
import mongoose from 'mongoose';

interface RouteContext {
    params: Promise<{
        id: string;
    }>;
}

export async function POST(
    request: NextRequest,
    context: RouteContext
) {
    const { id } = await context.params;
    const pollId = id;
    try {
        const { status, finalDate } = await request.json();

        if (status !== 'closed' || !finalDate) {
            return NextResponse.json({ message: "Invalid payload:
        }

        await connectDB();

        const poll = await Poll.findById(pollId);

        if (!poll) {
            return NextResponse.json({ message: "Poll not foun
```

InMyTime

Security Update: React RSC Vulnerability Fix

Why we updated

- A critical vulnerability was discovered in React Server Components (CVE-2025-55182)
- Affected version: 19.2.0
- The issue allowed unauthenticated remote code execution

What we changed

- Updated React RSC package from 19.2.0 → 19.2.1

Impact on our project

- Our app is now protected against the vulnerability
- No changes to UX/functionality
- Update required only back-end build dependencies

InMyTime

Thank you for your attention.

With InMyTime, scheduling becomes faster, clearer, and more collaborative — helping teams stay organized and aligned.
We look forward to improving and expanding the platform together.

Follow Us

InMyTime - Our GitHub