**Project Title:  
Universal Timetable Management System (TTMS)**

**Team Members:**

* SpikedSkull (Frontend & Microservices Developer, Designer)
* Teammate (Backend & Microservices Developer)

**Technology Stack:**

* Backend: Golang (Gin Framework) for high-performance REST APIs and ease of deployment.
* Frontend: HTML, CSS, and JavaScript initially, with planned migration to React for component-based architecture and maintainability.
* Database: PostgreSQL as the primary relational store, leveraging its robust transaction support and SQL capabilities; MongoDB as a fallback NoSQL option for unstructured or rapidly evolving data models.
* Caching: Redis to improve read performance, session storage, and distributed locking where necessary.
* Architecture: Microservices to isolate concerns, enable independent development and deployment, and allow horizontal scaling of individual components.
* Containerization: Docker to ensure consistent environments across development, testing, and production.

**Project Description**

The Universal Timetable Management System (TTMS) is designed to simplify and streamline the creation, modification, and distribution of schedules in educational and personal contexts. Users will interact with a modern web interface to perform tasks such as:

* Creating and Editing Schedules: Define classes, events, or tasks with customizable time slots, durations, and recurrence rules.
* Assigning Resources: Link each event to specific rooms, instructors, or equipment to avoid conflicts.
* Role-Based Access Control: Grant different permissions to students, teachers, administrators, and personal users to view, edit, or manage schedules.
* Real-Time Updates and Notifications: (Planned) Send alerts via email, in-app messages, or chatbots when schedules change.
* Performance Optimization: Use Redis caching for frequently accessed data (e.g., user sessions, public schedule views) to reduce database load and improve response times.

By decomposing the application into distinct microservices—such as Authentication Service, Schedule Service, User Management Service, Notification Service, and optional Integration Service—TTMS will achieve maintainability and scalability. Each service will be responsible for its own data store and API endpoints, communicating over HTTP/JSON or gRPC.

**Motivation and Relevance**

Academic and Personal Scheduling Challenges: Educational institutions and individuals alike face growing complexity in planning. In universities, scheduling must account for multiple departments, varying course loads, shared classrooms, and faculty availability. Outside academia, professionals and hobbyists need tools to balance meetings, deadlines, and personal commitments.

Gap in Existing Solutions: While Google Calendar offers a general-purpose calendar, it lacks academic-specific features like room and instructor assignments, bulk event creation, or conflict detection. Specialized systems (ascTimeTables, EduCloud) are powerful but often require extensive training and are not easily customizable or open source.

Market Demand: The shift to hybrid learning models and remote work has increased reliance on digital scheduling tools. A 2024 survey by EdTech Research Group indicated that 78% of institutions plan to invest in scheduling and resource management platforms in the next two years. Furthermore, the global market for educational software is projected to grow at a CAGR of 20% through 2028.

Versatility: TTMS is not limited to academic contexts. By providing a generic scheduling core, it can serve as a personal task planner, a team project scheduler, or a booking system for small businesses. This versatility increases its real-world relevance and potential user base.

**Target Audience**

1. Primary and Secondary School Students & Educators: Need intuitive interfaces to handle weekly lesson plans, substitute teachers, and classroom changes.
2. University Students (Undergraduate to Doctoral): Manage complex timetables including lectures, seminars, labs, and study groups.
3. Academic Administrators and Timetable Coordinators: Require bulk scheduling, conflict resolution, and exportable reports for accreditation and resource planning.
4. Personal and Professional Users: Freelancers, remote workers, and hobbyists seeking structured time management with advanced features like conflict detection and resource booking.

By designing flexible role-based permissions and a modular interface, TTMS will cater to both individual users and large organizations, supporting diverse workflows without overwhelming casual users.

**Competitor Analysis**

|  |  |  |
| --- | --- | --- |
| Competitor | Pros | Cons |
| Google Calendar | Clean, familiar UI  Cross-device sync  Free | No academic templates  Lacks resource assignment |
| ascTimeTables | Comprehensive academic features | Overly complex UI  Desktop-centric, not cloud-ready |
| EduCloud | Institutional-grade suite  Reporting modules | Closed ecosystem  Expensive, steep learning curve |
| Digii | Modern, student focused UI | Limited extensibility  No open API |
| Rajat19’s TTMS | Open-source, simple structure | Minimal security  Lacks documentation & polish |

Our Advantage: TTMS unites the intuitive design of general-purpose calendars with academic-specific scheduling logic and a microservice-based, open-source architecture - offering both simplicity for end users and flexibility for developers and administrators.

**Unique Value Proposition**

* Microservice Architecture: Each service can be developed, tested, and deployed independently, enabling rapid iteration and scaling of individual components.
* Role-Based Access Control: Fine-grained permissions ensure that users see only relevant data and actions, improving security and user experience.
* High Performance with Caching: Redis integration will accelerate data retrieval for frequently accessed endpoints, providing sub-100ms response times under typical loads.
* API-First Design: Comprehensive RESTful APIs and OpenAPI/Swagger documentation facilitate integration with external tools, mobile apps, or third-party services.
* Future-Ready Extensions: Built-in hooks for Telegram/Discord bots and AI-driven schedule optimization (e.g., suggesting ideal class times based on user preferences).
* Open Source & Community-Oriented: Transparent codebase encourages contributions, peer review, and rapid feature development driven by real user needs.

These differentiators position TTMS as more than a calendar—it’s a full-fledged scheduling platform designed for modern educational and personal use.

**Development Phases**

**Phase 1 (Weeks 1–2):**

* Set up GitHub repositories and CI/CD pipelines (GitHub Actions).
* Implement Authentication Service (JWT) and User Management Service.
* Provision PostgreSQL and Redis instances via Docker Compose.

**Phase 2 (Weeks 3–4):**

* Build Schedule Service with CRUD operations, conflict detection, and basic unit tests.
* Develop initial frontend pages (login, dashboard, schedule viewer) in HTML/CSS/JS.
* Conduct integration testing and document APIs.

**Phase 3 (Weeks 5–6):**

* Introduce Role-Based Access Control and admin interfaces.
* Refactor frontend with React components and state management.
* Optimize performance with Redis caching; add logging and monitoring (Prometheus & Grafana).

**Phase 4 (Weeks 7–8):**

* Implement Notification Service (email, in-app).
* Prototype Telegram bot integration for schedule queries.
* Conduct end-to-end testing, security audits, and user feedback sessions.

**Phase 5 (Weeks 9–10):**

* Polish UI/UX, add accessibility features and dark mode.
* Finalize documentation, tutorials, and deployment guides.
* Prepare demo, gather community feedback, and plan v1.1 enhancements.

**7. Success Criteria & Evaluation**

* Functional Completeness: All core features (authentication, scheduling, resource assignment, role control) implemented and tested.
* Performance Metrics: Average API response time under 200ms; 95th percentile under 500ms under simulated load.
* Usability: Achieve a System Usability Scale (SUS) score of 80+ in user testing with at least 10 participants.
* Stability: Zero critical bugs in production; 99.9% uptime over a two-week pilot.

**Conclusion**

The Universal Timetable Management System (TTMS) represents a concerted effort to deliver a robust, scalable, and user-centric scheduling platform. By leveraging microservices, modern frameworks, and best practices in DevOps, we will not only meet academic scheduling needs but also extend the system’s applicability to personal and professional contexts.

Our structured development roadmap, clear success metrics, and commitment to open-source collaboration ensure that TTMS can evolve beyond a class project into a widely adopted tool for efficient time management.