TickPulse, Architecture and Implementat ion

Wu Chun Hei (1155194804), Li Chak Man (1155194613), Yang Yufeng (1155194222)

1. Introduction and goals

What is TickPulse?

This document outlines the design, architecture and implementation strategy for TickPulse, a web-based task management system aimed at helping users or ganize their tasks effectively.

Main features

- User Management: Supports sign-up, login, logout, and license key va lidation.
- Task Management: Allows users to create, edit, delete tasks with pri orities, categories and deadlines.
- User interface: Requires a clear UI with menus and buttons, supporti ng List, Board and Calender view.
- Data Synchronization: Implements offline-first functionality with re al-time sync using IndexedDB and a Last-Write-Win policy for conflic ts.

Quality Goals

Nr.	Quality	Motivation
1	Security	Ensure data encryption in transit, hashed password storage, and integration with Firebase for authentication.
2	Performance	UI must load within 3 seconds, with task reminders sent 5 m inutes before deadlines, aims for 95% uptime.
3	Adaptability	Supports major browsers and multiple languages(English, Sim plified Chinese, Traditional Chinese)

Stakeholders

Role/Name	Goal/Boundaries
Developers	To get familiar with the architecture a

	nd ways to implement the project
Instructors	To evaluate the project
End Users	Efficient task management, deadline rem inder

2. System Scope and Architecture Constraint

System boundaries:

TickPulse is a web application accessible via modern browsers, focusing on desktop and laptop usage. Mobile app development is explicitly out of scop e.

Technical Constraints:

Next.JS based on React for frontend, Python for backend.
Socket.io for real-time sync. Prioritized local operations and sync conflic ts via Last-Write-Win policy

3. Solution Strategy

In terms of technological stack of TickPulse, Next JS is decided to be adap ted, so as to provide a graphical user interface and ensure a seamless user experience.
Building the back-end of the application by Python is an obvious choice. Fl ask or Django are likely to be used for server-side logic, API handling and database interactions.
In terms of database system, MySQL is decided to be the persistent storage due to its relational structure and ACID compliance, which is suitable for task and user data.For the real-time communication, we will adapt the Socke t.io for websocket-based real-time updates so as to support features like l ive task synchronization.

Besides the basic components of the application, the usage of some additional tools is expected. For instance, i18next for language support, Git Flow for version control.

In terms of implementation approach, we plan to adopt a client-server architecture where the client handles UI and user interactions, and the server processes requests, validates logic, and manages database operations.
A Last-Write-Win policy will be adapted for conflict resolution during data synchronization, simplifying data consistency management. Moreover, due to security concerns, HTTPS will be used for data in transit, hashed passwords, and input validation so as to prevent injection attacks.

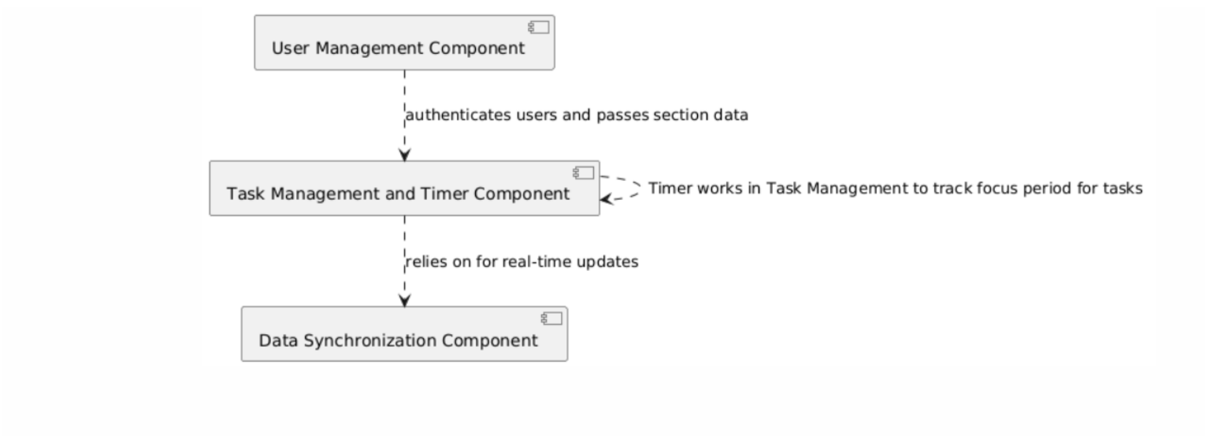
4. Building Block View

Major Components and Their Responsibilities:

The system is divided into several key components, each with specific roles:

Component	Responsibilities	Interfaces
User Management Component	Handles interaction with users include registration, login and logout	Interacts with pass authentication system, UI
Task Management Component	Manage affairs related to tasks such as edit, delete, create.	Communicates with Database, UI
Timer Component	managing focus and rest times, with visualized alert	Integrates with the Task Management Component

Data Synchronization Component	Ensure real-time data sync between client and server	Interfaces with client, server and database


User Management Component
authenticates users and passes section data
Task Management and Timer Component
Timer works in Task Management to track focus period for tasks
relies on for real-time updates
Data Synchronization Component

5. Runtime View

5.1 User login:

User enter credentials or use Google login on the UI. The User Management Component will then validates via authentication system, checks license key, and grant access.

Sequence: User → UI → User Management → Authentication → UI

5.2 Task Creation:

User first add a task with details such as name, priority, category and deadline via the UI. Task Management Component will then saves to local Indexed DB and syncs with server via Data Synchronization Component.

Sequence: User → UI → Task Management → Data Synchronization → Server → Database

5.3 Task Reminder:

System will check task deadlines, and notification will be sent 5 minutes before deadline.

Sequence: Task Management → UI → Notification → User

6. Deployment View

Deployment Architecture:

The application is deployed on a Linux server, with the front-end served by the next.js, accessible via a web server like Google Chrome.
The back-end runs as a Python application on a web server, communicating wi th MySQL for database operations.
All the real-time communication is facilitated by Socket.io, potentially on the same server or a dedicated instance for scalability.

Software Requirement:

Server: Linux based, with sufficient RAM and CPU for user load.
Client: Modern Browsers such as Google Chrome or Firefox, with resolution 1920x1080, supporting Indexed DB.

Scalability Considerations:

For high traffic, load balancing front-end servers and database replication for MySQL will be considered, ensuring 95% uptime.

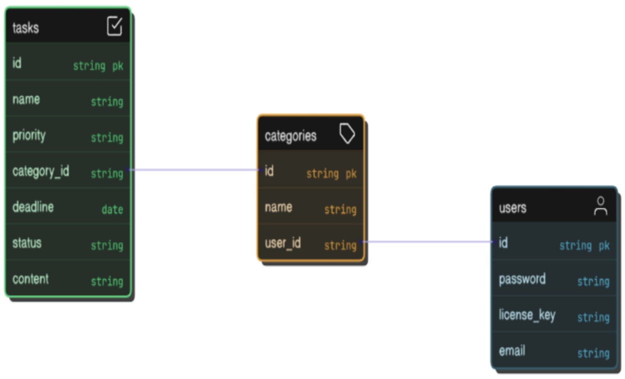
7. Concepts

Domain Models

User: Represents an individual with attributes like user id, password, license key, email

Task: Encapsulates tasks details like name, priority, category, deadline, status and content

Category: User-defined labels for tasks, linked to users for personalization.



User Interface:

The default user interface for TickPulse is expected to be written in Next.JS, within the application on the popular browsers such as Google Chrome and Firefox

Database Schema:

Following SQL schema outlines the data model in database:

PK = Primary Key, F = Foreign

Table Name	Attributes	Notes
Users	user_id(INT, PK), password(VARCHAR), license_key(VARCHAR) email(VARCHAR)	Stores user credentials and information
Categories	cat_id(INT, PK), user_id(INT, F to Users), category_name(VARCHAR)	Manages user-defined task categories
Tasks	task_id(INT, PK), user_id(INT, F to Users), cat_id(INT, F to Categories), task_name(VARCHAR)	Stores tasks details

8. Design Decisions

8.1 Front-end Language(Next.JS):

Chosen due to the great ability to handle routing, data fetching and other essential tasks out of the box, allowing developers to focus on building the core functionality of the application.

8.2 Back-end Language(Python):

Python offers lots of libraries for web development and database integration, suitable for the back-end server logic.

8.3 Database Choice(MySQL):

Selected for the relational structure. fitting the need of structured data storage.

8.4 Real-time Communicatio(Socket.io):

Used for websocket-based real-time updates, ensuring efficient synchronization.

9. Quality Scenarios

Quality Attributes and Corresponding Scenarios:
Performance: GUI must load within 3 seconds, achieved by optimizing asset loading. Notifications sent 5 minutes before deadlines, implemented by integrating timer into Task Management system.

Security: Data in transit encrypted via HTTPS, passwords will be hashed, inputs to be validated to prevent SQL injection

Reliability: Aim for 95% uptime, mitigated by server monitoring and failover mechanisms.

Adaptability: Supports major browsers

10. Technical Risks

Technical Risks and Corresponding Mitigations:
Risk: Data synchronization conflicts
Mitigation: Apply Last-Write-Win policy, with user notifications for conflicts.

Risk: H2 database corruption
Mitigation: Regular backups of the database, use MySQL for product stability.

11. Citation

Throughout the documentation of the design and implementation of our project, various generative AI tools including deepseek and Grok3 are applied to help us finish such document.