

**TOBB ETU**

**Economy & Technology University**

**BIL 495 / YAP 495**

**Software Design Description (SDD)**

**Reference:** IEEE 1016-2009

## Document Control Table

|  |  |
| --- | --- |
| Document Title | Project Closure Report |
| Document ID | (To be assigned) |
| Prepared By (Author) | Emir Yücedağ |
| Reviewed By | Aylin Doğan |
| Approved By |  |
| Preparation Date | 16.11.2025 |
| Approval Date |  |
| Version / Revision | 1.0 |
| Confidentiality Level | Internal / Restricted / Public |
| Reference Standards | IEEE 15288, IEEE 12207, INCOSE SE Handbook v5 |

## Change Record (Revision History Table)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Revision | Date | Prepared By | Reviewed/Approved By | Description of Change |
| 1.0 | (Date) | (Author) | (Approver) | Initial Release |
| 1.1 |  |  |  | Minor editorial updates |
| 1.2 |  |  |  | Added stakeholder feedback section |

[1. Introduction](#_heading=)

[1.1 Purpose](#_heading=)

[1.2 Scope](#_heading=)

[1.3 References](#_heading=)

[2. System Overview](#_heading=)

[2.1 System Architecture](#_heading=)

[2.2 Design Rationale](#_heading=)

[3. Detailed Design](#_heading=)

[3.1 Component Descriptions](#_heading=)

[3.2 Interface Descriptions](#_heading=)

[3.3 Data Design](#_heading=)

[3.4 Algorithm Design](#_heading=)

[4. Traceability](#_heading=)

## 1. Introduction

## 1.1 Purpose

The purpose of this Software Design Description (SDD) is to provide a detailed technical blueprint for the "Gather Up" platform. This document translates the architectural decisions defined in the System Architecture Document (SAD) into low-level design specifications, including component definitions, data structures, and interface logic, to guide the development phase.

## 1.2 Scope

This document covers the detailed design of the backend system and its interaction with the mobile application. It specifically details the internal logic of the Modular Monolithic architecture, including the Authentication, Chat, Event Recommendation, Notifications, and User Management modules. It also defines the database schema designs for MySQL and the API specifications

## 1.3 References

 IEEE Std 1016-2009, IEEE Standard for Information Technology—Systems Design—Software Design Descriptions.

 System Architecture Document (SAD) v1.0 - "Gather Up" Project.

 Software Requirements Specification (SRS) v1.0.

## 2. System Overview

## 2.1 System Architecture

The **Gather Up** platform employs a **Modular Monolithic Architecture** built on **Express.js (Node.js)**. This structure organizes the application into distinct, feature-based modules (Authentication, Chat, Matching Engine, User Management, etc.) that operate within a single, unified deployable unit. This approach provides a balance between organized code structure and simplified deployment.

* **Frontend:** A **Mobile App** developed using **React Native**. This choice ensures cross-platform compatibility (iOS and Android) and rapid development to meet the critical time constraint.
* **Backend:** **Express.js (Node.js)**, responsible for handling all API requests, executing the core business logic, and managing real-time connections. Node.js is optimized for the I/O intensive nature of real-time communication.
* **Database:** **MySQL**, selected for its reliability in managing the structured, relational data required for users, events, participation records, and message history.
* **External Services:** **Firebase Cloud Messaging (FCM)**, utilized exclusively for reliable and efficient handling of push notifications.

## 2.2 Design Rationale

The architectural decisions were driven by the project's specific constraints (small team, limited time/budget) and core requirements (real-time interaction).

* **Modular Monolith:** This architecture was strategically chosen to balance scalability with maintainability for the small team of 4 developers. While keeping the code organized and feature-based for long-term clarity, it drastically simplifies deployment and infrastructure compared to a complex microservices setup, which is too resource-intensive for an MVP.
* **WebSockets:** This protocol is selected for the Chat Module to guarantee low-latency, real-time message delivery. Using WebSockets over traditional REST polling is essential for a spontaneous social interaction platform, providing the instant feedback required for user engagement.
* **Firebase Cloud Messaging (FCM):** FCM is employed to ensure reliable notification delivery for critical events (new matches, event reminders) even when the mobile application is closed or running in the background. Its native optimization helps conserve users' device battery and network usage.
* **Express.js (Node.js):** The selection of Node.js is key due to its asynchronous, non-blocking nature, making it exceptionally efficient for managing the numerous concurrent connections required by both the REST API and the WebSocket-based Matching Engine/Chat Module.

## 3. Detailed Design

## 3.1 Component Descriptions

 **3.1.1 Authentication Module**

* **Type**: Software Module
* **Purpose**: To centrally manage user registration, secure login processes, and session management.
* **Functionality**: Securely hashes passwords using a strong algorithm (e.g., bcrypt) during registration. It validates user credentials and issues JSON Web Tokens (JWT) for secure API access, while also managing the tokens' validity throughout the session.

 **3.1.2 Chat Module**

* **Type:** Software Module / WebSocket Service
* **Purpose:** Facilitates real-time messaging between users.
* **Functionality:** Manages WebSocket connections, routes messages to recipients, and stores message history in the database.

 **3.1.3 Event Recommendation Module**

* **Type:** Logic Module
* **Purpose:** Suggests events to users based on preferences.
* **Functionality:** Analyzes user attributes (interests, location) and past event history to filter and rank available events.

 **3.1.4 User Management Module**

* **Type:** Software Module
* **Purpose:** Manages user profiles and social connections.
* **Functionality:** Handles profile updates, friend requests, and friend lists.

## 3.2 Interface Descriptions

* **API Interface (REST):**
  + Protocol: HTTP/HTTPS via Express.js Network Thread.
  + Format: JSON.
  + Examples: POST /api/auth/login, GET /api/events/recommendations.
* **Real-Time Interface:**
  + Protocol: WebSocket (Socket.io).
  + Usage: Bi-directional communication for instant chat messages.
* **Database Interface:**
  + Protocol: SQL (via ORM or raw queries).
  + Connection: Backend interacts with MySQL on the Database Thread.

## 3.3 Data Design

* **Database Management System:** MySQL.
* **Key Entities (Proposed Schema):**
  + Users: (ID, Username, PasswordHash, Email, Preferences).
  + Events: (ID, Title, Location, Time, CreatorID, Category).
  + Messages: (ID, SenderID, ReceiverID, Content, Timestamp).
  + Friendships: (UserID\_1, UserID\_2, Status).

## 3.4 Algorithm Design

* **Event Matching Algorithm:**
  1. **Input:** Current User ID, User Preferences (Tags), Location.
  2. **Process:**
     + Fetch active events from Events table.
     + Calculate similarity score between User Tags and Event Category.
     + Filter out events outside the user's radius.
     + Sort candidates by Score (descending) and Time (ascending).
  3. **Output:** List of top 10 recommended events.

## 4. Traceability

| **Design Component** | **Requirement ID (SRS)** | **Requirement Type** | **Description** |
| --- | --- | --- | --- |
| **Authentication Module** | REQ-001, REQ-002 | Functional | Implements User Login and Registration logic, along with secure session management (JWT). |
|  | REQ-SEC-01 | Security / Legal | Addresses the core data security constraint through password hashing and authorization. |
| **Chat Module** | REQ-005 | Functional | Implements the real-time messaging system for event participants and social connections. |
| **Real-Time Availability Module** | REQ-006, REQ-007 | Functional | Ensures secure input, storage, and availability of user availability data for matching. |
| **Matching Engine** | REQ-008, REQ-009 | Functional / Critical | Implements the **smart, real-time event matching algorithm** based on user availability and interests. |
|  | REQ-PERF-01 | Non-Functional | Delivers matching results within the defined performance threshold (e.g., less than 2 seconds). |
| **User Mgmt. & Feedback Module** | REQ-010 | Functional / Ethical | Implements user profiles, social connections, and the **post-event feedback (rating)** system. |
|  | REQ-SAFE-02 | Security / Ethical | Supports reporting and rating mechanisms to prevent misuse and harmful behavior. |
| **MySQL Database** | REQ-SYS-01 | System | Meets the requirement for **persistent and secure** data storage for user, event, and message history. |
|  | REQ-LEGAL-03 | Legal | Supports necessary data retention and access controls for compliance with GDPR and local data protection laws. |

## 5. Appendix

### 5.1 Glossary of Terms and Abbreviations

|  |  |
| --- | --- |
| **Abbreviation** | **Description** |
| **SAD** | System Architecture Document |
| **SDD** | Software Design Description |
| **SRS** | Software Requirements Specification |
| **PMP** | Project Management Plan |
| **FCM** | Firebase Cloud Messaging (External service used for push notifications) |
| **API** | Application Programming Interface |
| **JWT** | JSON Web Token (Token standard to be used in the Authentication module) |
| **DB** | Database |
| **KVKK** | Personal Data Protection Law (Turkish data protection regulation) |
| **GDPR** | General Data Protection Regulation |
| **ERD** | Entity-Relationship Diagram |

### 5.2 Data Dictionary

**Table: Users**

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** | **Description** |
| UserID | INT | PRIMARY KEY, AUTO\_INCREMENT | The unique identifier for the user |
| Username | VARCHAR(50) | NOT NULL, UNIQUE | The user's display name in the system |
| Email | VARCHAR(100) | NOT NULL, UNIQUE | The user's email address |
| PasswordHash | VARCHAR(255) | NOT NULL | The user's password, hashed with bcrypt |
| Preferences | JSON | NULL | The user's event preferences (e.g., {"categories": ["sports", "music"]}) |

**Table: Events**

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** | **Description** |
| EventID | INT | PRIMARY KEY, AUTO\_INCREMENT | The unique identifier for the event |
| CreatorID | INT | FOREIGN KEY (Users.UserID) | The ID of the user who created the event |
| Title | VARCHAR(100) | NOT NULL | The title of the event |
| Location | VARCHAR(255) | NOT NULL | The physical location of the event |
| EventTime | DATETIME | NOT NULL | The start time of the event |
| Category | VARCHAR(50) | NULL | The event category (e.g., "sports") |

### 5.3 API Endpoint List (Example)

**Authentication Module (/api/auth)**

* **POST /register**
  + **Description:** Creates a new user registration.
  + **Request Body:** { "username": "...", "email": "...", "password": "..." }
  + **Response (201):** { "message": "User registered successfully" }
* **POST /login**
  + **Description:** Logs in a user and returns a JWT.
  + **Request Body:** { "email": "...", "password": "..." }
  + **Response (200):** { "token": "jwt.token.string" }

**Event Recommendation Module (/api/events)**

* **GET /recommendations**
  + **Description:** Lists event recommendations for the authenticated user.
  + **Headers:** { "Authorization": "Bearer jwt.token.string" }
  + **Response (200):** [ { "EventID": 1, "Title": "...", "Location": "..." } ]

## 6. References

1. **[SAD-v1.0]** "Gather Up" Project Team, "System Architecture Document (SAD) v1.0", TOBB ETÜ, 14 November 2025.
2. **[SRS-v1.0]** "Gather Up" Project Team, "Software Requirements Specification (SRS) v1.0", TOBB ETÜ, 31 October 2025.
3. **[IEEE 1016]** IEEE Std 1016-2009, "IEEE Standard for Information Technology—Systems Design—Software Design Descriptions."
4. **[IEEE 42010]** IEEE Std 42010-2011, "IEEE Standard for Systems and Software Engineering — Architecture Description."
5. **[ExpressJS]** Express.js Official Documentation. Retrieved from: <https://expressjs.com/>
6. **[MySQL]** MySQL Official Documentation. Retrieved from: <https://dev.mysql.com/doc/>
7. **[FCM]** Firebase Cloud Messaging Official Documentation. Retrieved from: <https://firebase.google.com/docs/cloud-messaging>
8. **[KVKK]** Personal Data Protection Law (Law No. 6698). Retrieved from: [https://www.mevzuat.gov.tr/](https://www.google.com/search?q=https://www.mevzuat.gov.tr/)
9. **[GDPR]** Regulation (EU) 2016/679 (General Data Protection Regulation). Retrieved from: <https://eur-lex.europa.eu/>