SOFTWARE DESIGN DOCUMENT
for
Brainstorming Application

Prepared by

Mehmet Nuri Başa

Umut Hüseyin Satır

Emre Durmaz

Mert Certel

Bedirhan Ömer Aksoy

TABLE OF CONTENTS

1 Introduction

The purpose of the Introduction is to provide a brief overview of the software architecture and the design goals. It also provides references to other documents and traceability information (e.g., related requirements analysis document, references to existing systems, constraints impacting the software architecture).

1.1 Purpose of the system

The purpose of the Brainstorming Application is to provide a structured, reliable, and real-time collaborative platform for ideathons and workshops that utilize the 6-3-5 brainstorming method.
The system is designed to address the limitations of traditional brainstorming practices that rely on fragmented tools such as spreadsheets, shared documents, and messaging applications.

The Brainstorming Application enables Event Managers, Team Leaders, and Team Members to collaboratively generate, manage, and analyze ideas within a controlled and observable environment.
By enforcing the rules of the 6-3-5 method and integrating AI-assisted ideation and summarization through OpenAI's ChatGPT, the system ensures balanced participation, improved creativity, and consistent documentation of brainstorming outcomes.

1.2 Design goals

The primary design goals of the Brainstorming Application are:

- Structured Ideation:
  To strictly enforce the constraints of the 6-3-5 brainstorming method, including participant count, idea limits, round timing, and deterministic idea passing.
- Real-Time Collaboration:
  To provide low-latency synchronization of session state, timers, and idea submissions across web and mobile clients.
- Role-Based Control and Visibility:
  To support fine-grained access control for Event Managers, Team Leaders, and Team Members, ensuring privacy, focus, and proper authority separation.
- AI-Assisted Creativity:
  To integrate ChatGPT for contextual idea generation and post-session summarization without disrupting the core brainstorming flow.
- Reliability and Auditability:
  To guarantee persistent storage of all ideas, session data, and logs, enabling post-event reporting, auditing, and export.

- Scalability:
  To support multiple teams running concurrent brainstorming sessions without performance degradation.

1.3 Definitions, acronyms, and abbreviations

6-3-5 Method:
A structured brainstorming technique where six participants each contribute three ideas across five timed rounds.

Event Manager:
A user responsible for managing events, topics, participants, teams, and overall system oversight.

Team Leader:
A user who moderates a single team's brainstorming sessions and may interact with AI assistance.

Team Member:
A participant who contributes ideas during active brainstorming rounds.

ChatGPT:
An AI model by OpenAI used for idea generation and automatic summarization.

CRUD:
Create, Read, Update, Delete operations supported by the system.

RDBMS:
Relational Database Management System used for persistent data storage.

API:
Application Programming Interface enabling communication between system components.

1.4 References

1. **Problem Statement:** Brainstorming Application for Ideathons and Workshops, Project Specification Document.
2. **6-3-5 Method Definition:** Rohrbach, B. (1969). "Creative by Rules - Method 635". Creativity and Innovation Management.
3. **External API Documentation:** OpenAI API Documentation (https://platform.openai.com/docs).
4. **Course Specifications:** CSE443 - Requirements Analysis Document Template, Gebze Technical University, 2025.

1.5 Overview

This Software Design Document (SDD) details the architectural and detailed design decisions for the Brainstorming Application. The system is designed to facilitate real-time, structured ideation sessions using the 6-3-5 method, augmented by AI capabilities .

This document provides a comprehensive view of:

- The transition from manual, fragmented tools to a unified, centralized software architecture.
- The decomposition of the system into functional subsystems (User Management, Session Orchestration, AI Integration).
- The hardware/software mapping, including client-side (Web/Mobile) and server-side components.
- Data management strategies utilizing RDBMS and access control mechanisms via JWT.

The intended audience for this document includes the software development team, project managers, and quality assurance testers.

2 Current software architecture

Currently, there is no unified legacy software system being replaced. Instead, the "current system" consists of a fragmented, manual workflow utilizing a combination of ad-hoc tools.

Existing "Architecture" Components:

- ***Data Storage:*** *Spreadsheets (Excel/Google Sheets) used for participant lists and idea tables.*
- ***Documentation:*** *Cloud documents (Google Docs/Word) used for free-form notes and idea passing.*
- ***Synchronization:*** *Messaging tools (Slack/WhatsApp/Email) used for coordination and file forwarding.*
- ***Timing****: Independent timer applications controlled manually by a facilitator.*

Architectural Deficiencies:

1. ***Lack of Centralization:*** *Ideas, participants, and timers live in separate environments, creating version conflicts .*
2. ***Concurrency Issues:*** *"Passing" ideas (files) breaks due to file locks or duplication errors, causing round desynchronization.*
3. ***Scalability Limits:*** *Monitoring multiple teams simultaneously is impossible for a single Event Manager, as there is no central dashboard.*
4. ***Security Gaps:*** *Authentication is informal (shared links), lacking role-based access control.*

3 Proposed software architecture

3.1 Overview

The proposed Brainstorming Application adopts an N-Tier Client–Server architecture to support structured, real-time, AI-assisted ideation sessions using the 6-3-5 method. The system is delivered through two client types (Web and Mobile) connected to a backend service layer via secure APIs. The backend decomposes core features into cooperating subsystems: event and directory management (topics/participants/teams), a 6-3-5 Session Orchestrator that enforces rounds, timing, and idea passing, a Real-Time Synchronization layer for low-latency updates across devices, AI Integration (ChatGPT) for idea generation and session summarization, reporting/export and audit logging, and role-based security controls using authentication and authorization mechanisms.

Architecturally, the system separates concerns across tiers:

- **Presentation tier:** web UI + mobile UI for dashboards and brainstorming workspace.
- **Application tier:** REST endpoints for CRUD/session management and a real-time transport channel (e.g., WebSocket).
- **Domain/service tier:** session orchestration logic, AI prompting/summarization, export/report services.
- **Data tier:** relational persistence for event entities, sessions, rounds, ideas, summaries, and logs.

This separation ensures the solution meets key requirements such as enforcing “six participants / three ideas / five-minute rounds”, supporting multiple concurrent teams, ensuring automatic saving and auditability, and providing AI assistance while keeping the core session functional even when AI is unavailable.

3.2 Subsystem decomposition

1) Client Applications (Web + Mobile)

- Provides role-based UI flows for Event Manager, Team Leader, and Team Member.
- Web focuses on administration dashboards + facilitation; Mobile focuses on quick idea entry, session participation, and notifications.
- Communicates with backend through HTTPS (REST) and subscribes to real-time session updates via WebSocket (or equivalent).

2) API Layer (Controller / Gateway)

- Exposes RESTful endpoints for:

- CRUD (topics, participants, teams)
- Session control (start/pause/resume/end)
- Round transitions and idea submission
- Reporting/export requests
- AI request endpoints (idea generation & summarization)

- Validates JWT tokens and forwards requests to internal services based on role permissions.

3) Event & Directory Management Service

- Manages Topics, Participants, Teams, team membership, and role assignment.
- Enforces setup constraints (e.g., team composition completeness before session start).
- Provides filtered views (Event Manager sees all; Team Leaders see assigned team; Team Members see own team).

4) 6-3-5 Session Orchestrator (Core Engine)

- Implements deterministic session lifecycle:
  - Creates session, initializes Round 1, starts 5-minute timer
  - Enforces “exactly three ideas per member per round”
  - Ends round on timer expiration or when all have submitted
  - Performs **idea passing** to the next participant and advances to next round
- Owns global session state machine: **`Planned`** → **`Running`** → **`Paused`** → **`Completed/Ended`.**
- Publishes state changes to Real-Time Sync Layer.

5) Real-Time Sync Layer

- Maintains low-latency synchronization for:
  - Countdown timer updates
  - Member submission status
  - Round transitions

- New idea availability (including “previous teammate’s ideas”)

- Provides resilience features such as replay/refresh on reconnect to prevent data loss.
- Keeps all team clients consistent during multi-round sessions.

6) AI Integration Service (ChatGPT)

- Provides two core workflows:
  - **Idea generation:** prompt ChatGPT using current topic + current/previous ideas.
  - **Summarization:** send session ideas and receive structured summary highlighting themes/trends.
- Includes guardrails: rate-limiting, timeout handling, and fallback messaging if external AI is unavailable.
- Stores AI outputs as part of session artifacts for later viewing/export.

7) Reporting, Export, and Audit Service

- Generates event/team dashboards and completed session views.
- Exports summaries and idea reports to PDF/CSV.
- Maintains immutable logs of session actions, timestamps, and status changes for auditing.

8) Security & Access Control Service

- Authenticates users (JWT) and authorizes actions based on roles:
  - Event Manager: full control across events/teams/sessions
  - Team Leader: controls assigned team sessions, AI requests for their team
  - Team Member: submit/view within their team scope
- Enforces privacy boundaries so teams cannot view other teams’ in-progress ideas.

3.3 Hardware/software mapping

Client Tier

- ***Web App:*** *Runs on desktop/tablet browsers; recommended deployment as a SPA (or server-rendered web) served via CDN or web server.*

- ***Mobile App:*** *Runs on iOS/Android devices; uses push notifications for session start and round transitions; optimized for touch idea entry.*

Server Tier

- ***Backend Application Server(s):***
    - *Hosts REST APIs, session orchestration, auth logic, reporting/export, and AI integration endpoints.*
    - *Horizontally scalable to support many concurrent teams and sessions.*
- ***Real-Time Messaging Component:***
    - *WebSocket server (or equivalent) that broadcasts session state and timer updates to connected clients.*
- ***Background Worker:***
    - *Handles long-running tasks: PDF generation, bulk exports, AI summarization jobs, email/notification dispatch, log aggregation.*

Data Tier

- ***Relational Database (RDBMS):*** *Central persistence for event entities and session artifacts (e.g., PostgreSQL/MySQL).*
- ***Object Storage (optional):*** *Stores generated export files (PDF/CSV) and returns download links.*
- ***Cache (optional):*** *Improves performance for hot session state reads (e.g., Redis) while database remains source of truth.*

External Services

- ***OpenAI API:*** *Used by AI Integration Service for idea generation and summarization.*
- ***Push Notification Provider:*** *Used by mobile app for reminders and round transition notifications.*

3.4 Persistent data management

The application uses an **RDBMS as the system of record** to ensure reliability, auditability, and consistent access control for structured brainstorming artifacts.

Core persisted entities (suggested tables)

- **Event**: event metadata, status.

- **Topic**: title, description, status (open/closed/archived), event_id.
- **Participant/User**: identity, contact info, hashed credentials, role.
- **Team**: team name, leader_id, event_id.
- **TeamMembership**: participant_id, team_id (enforce one team per event if required).
- **Session**: team_id, topic_id, start/end timestamps, status.
- **Round**: session_id, round_number (1..5), start/end time, timer configuration.
- **Idea**: round_id, participant_id, content, timestamp, “passed_from” reference (optional) to support traceability of idea passing.
- **AIRequest / AISummary**: session_id, prompt metadata, model/version, generated text, created_at.
- **AuditLog**: who did what and when (login, start/pause/resume/end, submission events, AI calls, export generation).

Integrity constraints

- Enforce **6-3-5 rules** at the data level where possible:
  - Round_number bounded 1..5.
  - Ideas per (participant_id, round_id) ≤ 3; “exactly 3” enforced by application workflow before round completion.
  - Session cannot start unless team has 6 active members.
- Foreign keys for referential integrity (ideas → rounds → sessions → teams/events).
- Visibility rules enforced by queries filtered via team_id and role (prevent cross-team leaks).

Transaction strategy

- Use database transactions for critical transitions:
  - On round end: finalize round, persist all submissions, compute next participant mapping, create next round state.
  - Prevent partial transitions that could desynchronize clients.
- “Auto-save on submit” to avoid data loss during disconnects (each idea persisted immediately).

Indexing and performance

- Index common access paths: (session_id), (team_id), (round_id), (participant_id, round_id).
- Store derived aggregates (optional) for dashboards (e.g., number of submissions per round, completion rates) while keeping raw logs for auditing.

Retention & compliance

- Store session artifacts and summaries for post-event review/export.
- Apply retention rules (e.g., archive completed events; configurable deletion policy) and keep audit logs for compliance needs.

3.5 Access control and security

The Brainstorming Application defines a structured user model with clear authorization rules and security mechanisms. Access to all API operations and real-time session actions is governed by role-based access control (RBAC), which is aligned with the three primary system roles: Event Manager, Team Leader, and Team Member.

3.5.1 Authentication mechanism

The system authenticates users via a JWT-based login flow, where the backend validates credentials and issues a signed token used on subsequent API calls. Tokens include user identity and role claims to support authorization checks.

3.5.2 Authorization model (access matrix)

Authorization is enforced at the service boundary (API gateway/controller layer) by verifying role claims before executing any protected operation and returning an immediate “Access Denied” response for unauthorized actions.

Access matrix (high-level):

Operation / Resource	**Event Manager**	**Team Leader**	**Team Member**
Create/Edit/Delete Topics	Allowed	Not allowed	Not allowed
Register/Edit/Remove Participants; assign roles	Allowed	Not allowed	Not allowed

Create/Modify Teams; assign membership	Allowed	Limited (own team creation if enabled)	Not allowed
Start/Pause/Resume/End session	Allowed (any team)	Allowed (own team)	Not allowed
Submit ideas in active round	Not typical	Allowed (as participant if included)	Allowed
View live session ideas	Allowed (monitoring, scoped by policy)	Allowed (own team)	Allowed (own team only)
Export reports / view completed sessions / audit logs	Allowed	Limited (own team views)	Not allowed

Key visibility constraints:

- Ongoing ideas are visible only to the active team participants, preventing cross-team leakage.
- Team Leaders have restricted participant views (team-scoped), while Event Managers can view the full roster.

3.5.3 Data protection

- Transport security: All client-server traffic (REST + WebSocket) must use TLS to protect credentials, JWTs, and session data in transit.
- Sensitive data at rest: Participant personal/contact information and role/permission records are stored in the database with appropriate protection controls.
- Session integrity: Ideas are persisted immediately on submission to prevent loss from crashes/disconnects.

3.5.4 Key management and secrets

- JWT signing keys and external service keys (e.g., OpenAI API key) are stored in a secure secrets mechanism (environment secrets / vault) and rotated periodically.

- Least-privilege access is applied to backend services accessing secrets and the database.

3.5.5 Abuse prevention and auditing

- Rate limiting is applied to OpenAI/ChatGPT requests to control costs and availability risk.
- Audit logs capture session events (round transitions, submissions, start/stop actions, exports) for traceability and investigation.
- Login attempts and authorization denials are logged for security monitoring.

3.6 Global software control

Client requests are initiated through defined service interfaces, while internal subsystems coordinate and synchronize their operations to ensure consistent system behavior. Concurrency is managed across REST-based services and real-time collaboration mechanisms to maintain data integrity and reliable execution.

3.6.1 Control style and request initiation

The system uses a service-oriented backend with:

- RESTful APIs for CRUD operations (topics, teams, participants) and administrative actions.
- A real-time channel (e.g., WebSocket) to synchronize time-critical session state such as timers, round transitions, and submissions.

Clients initiate actions (e.g., submit ideas, pause session). The backend validates JWT, checks authorization, executes business logic, persists state, and then emits real-time updates to affected clients.

3.6.2 Session orchestration control (6-3-5 engine)

A dedicated Session Controller/Orchestrator implements the 6-3-5 lifecycle:

- Enforces prerequisites: exactly six participants required to start a session round.
- Maintains a round state machine: `Planned → Active → Transitioning → Completed` with explicit session statuses (`Active`, `Paused`, `Completed`).
- Runs the 5-minute timer per round and triggers automatic round transitions on expiry.

- Executes deterministic idea passing at each transition (participant i → participant i+1), then opens the next round.

3.6.3 Synchronization and concurrency strategy

Primary concurrency risks include simultaneous submissions, timer expiry races, and concurrent admin actions (Event Manager overriding Team Leader).

Controls:

- Single-writer principle for session state: Only the Session Orchestrator commits authoritative round transitions and status changes.
- Optimistic concurrency for updates (e.g., session version/ETag or updatedAt checks) to prevent lost updates when multiple controllers act concurrently.
- Atomic persistence: idea submissions are written as atomic transactions; a round is marked “ready to transition” only when the timer expires or all six submissions are complete.
- Idempotent commands: start/pause/resume/end endpoints are idempotent (repeating the same command yields the same final state), preventing duplicate transitions due to retries.

3.6.4 Real-time propagation

When state changes occur (timer ticks, new ideas submitted, round transitions), the system broadcasts minimal state diffs to relevant clients:

- Team-scoped channels prevent data leakage (only the team receives team events).
- The Event Manager may subscribe to aggregated monitoring channels for multi-team oversight.

3.6.5 Integration control: ChatGPT services

AI requests are mediated through an AI service/controller:

- Requests are rate limited and retried with bounded backoff.
- AI is non-blocking for core session flow: session progression continues even if AI is unavailable, ensuring the 6-3-5 engine remains deterministic.

3.6 Boundary conditions

Boundary conditions describes the start-up, shutdown, and error behavior of the system.

3.7 Boundary conditions

The system defines the expected behavior during startup and shutdown phases, as well as error handling and recovery mechanisms, in accordance with the Software Design Document (SDD) template.

3.7.1 Startup behavior

On startup:

1. Backend initializes configuration (DB connection, JWT keys, external API keys, rate limit policies).
2. Database migrations/health checks run (fail-fast if critical dependencies are unavailable).
3. Real-time server initializes channels for active sessions; no session timer runs until a session is explicitly started.
4. Clients load dashboards based on role and fetch initial state via REST; live updates begin after WebSocket connection.

3.7.2 Shutdown behavior

On graceful shutdown:

- The system stops accepting new requests, completes in-flight writes, and closes WebSocket sessions cleanly.
- Active sessions are persisted with the latest known state; if a session is mid-round, it transitions to a recoverable “Paused/Interrupted” state so that Team Leader/Event Manager can resume manually after restart.
- Any queued exports are finalized or safely aborted with a recoverable job status.

3.7.3 Error behavior and recovery

Authentication/authorization errors

- Invalid credentials → login rejected; invalid/expired JWT → 401; insufficient role → immediate “Access Denied.”

6-3-5 rule violations

- If a team has fewer than six active participants when attempting to start → reject with a clear validation error.
- If a user attempts to submit more/less than exactly three ideas → reject submission; UI remains in input state until corrected.

Network disconnects during sessions

- Client-side temporary disconnect: upon reconnection, client re-fetches authoritative session state and continues from the current round.
- Autosave ensures submitted ideas are not lost.

Timer expiry and submission races

- If the timer expires before some users submit, the system transitions automatically; late submissions are rejected or attached only if the round is still active (no post-transition writes allowed).

ChatGPT / external API failures

- If AI generation or summarization fails (timeout, rate limit, service error), the system:
    1. logs the incident,
    2. shows “AI service unavailable,”
    3. continues the core session flow without interruption.

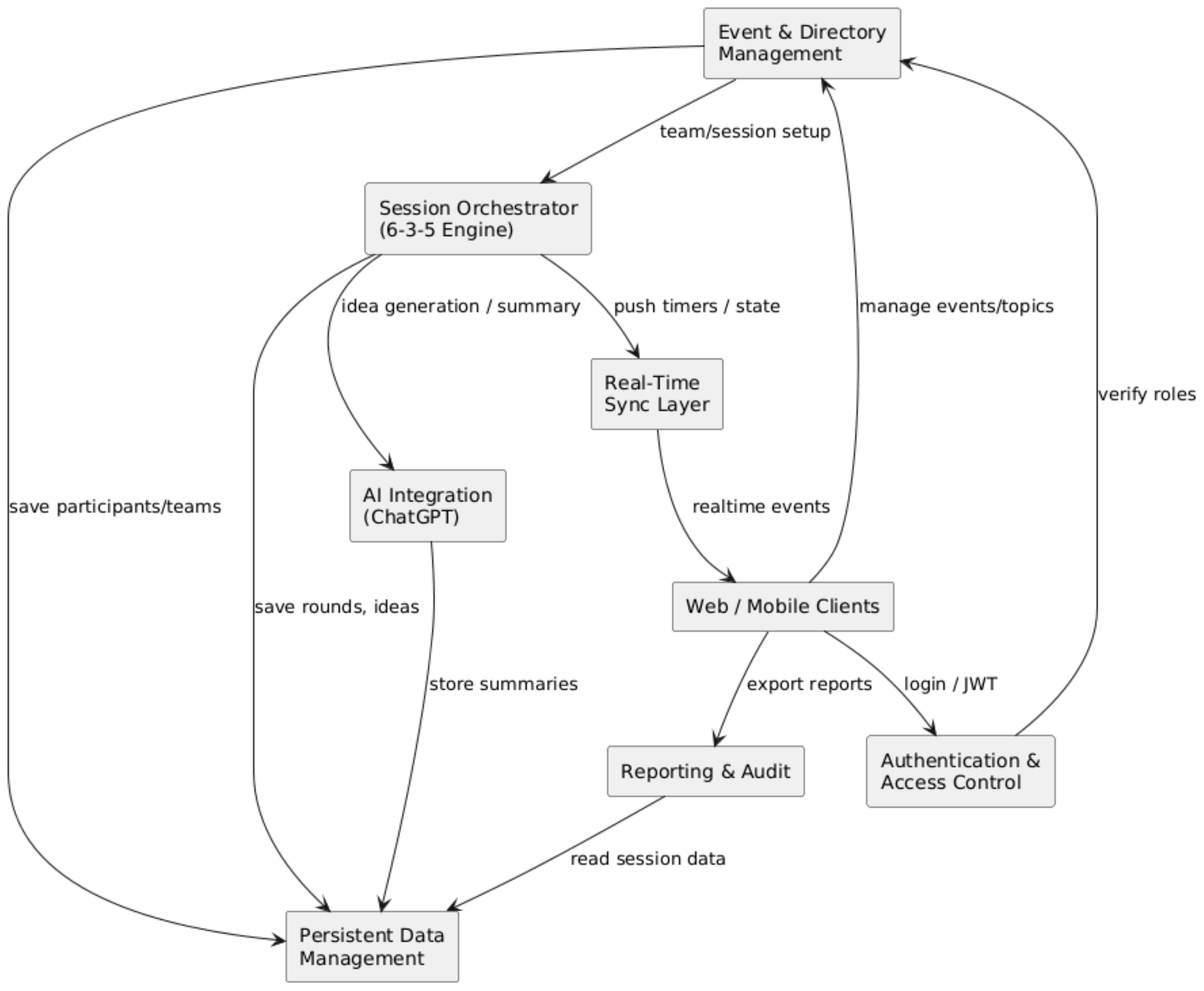
Database or persistence failures

- For write failures: return an error and do not acknowledge submission; clients display retry guidance.
- For transient failures: bounded retries may be applied for idempotent operations; otherwise fail-fast to prevent state corruption.

4 Subsystem services

This section describes the operations, responsibilities, and exposed interfaces of each subsystem in the Brainstorming Application.
Subsystem boundaries reflect the architecture defined in Section 3. Each subsystem exposes a set of services that other subsystems or user interfaces depend on.

A high-level subsystem architecture diagram is shown below:



4.1 Event & Directory Management Subsystem

Purpose

Manages event configuration, topics, participants, and team structures.

Provided Services

Service	Description	Inputs	Outputs
`createTopic()`	Creates a new brainstorming topic.	title, description	topicId
`updateTopic()`	Edits an existing topic.	topicId, newData	success/fail

`deleteTopic()`	Removes a topic from the system.	topicId	success/fail
`registerParticipant()`	Registers a user in the event.	userInfo	participantId
`assignTeam()`	Assigns participants to a team.	participantId, teamId	success
`createTeam()`	Creates a new team and sets leader.	teamName, leaderId	teamId
`updateTeam()`	Changes leader/members.	teamId, updateData	success
`listParticipants()`	Returns participant roster.	eventId	participantList

Dependencies

- Authentication Subsystem (for verifying manager role)
- Persistent Data Management Subsystem

4.2 Session Orchestrator (6-3-5 Engine)

Purpose

Controls the brainstorming session lifecycle, timers, idea submission, idea passing, and round transitions.

Provided Services

Service	Description	Inputs	Outputs
startSession()	Starts 6-3-5 session for a team.	teamId	sessionId
startRound()	Initiates a new round with timer.	sessionId	roundId

pauseRound()	Pauses current round.	sessionId	success
resumeRound()	Resumes paused round.	sessionId	success
endRound()	Ends the round early.	sessionId	success
submitIdeas()	Saves user’s 3 ideas.	participantId, ideas	success
passIdeasNext()	Passes idea sets to next participant.	roundId	success
finalizeSession()	Ends the full 6-3-5 workflow.	sessionId	success

Dependencies

- Real-Time Sync Subsystem (push updates)
- Persistent Data Subsystem
- AI Integration Subsystem (optional idea assist)

4.3 Real-Time Synchronization Subsystem

Purpose

Maintains low-latency, multi-device synchronization for timers, ideas, and state changes.

Provided Services

Service	Description	Inputs	Outputs
`broadcastTimer()`	Sends countdown timer updates to all clients.	teamId, timeLeft	realtime event
`broadcastIdeas()`	Sends new ideas to relevant participants.	roundId	realtime event

`syncState()`	Syncs full session state on reconnect.	sessionId	stateSnapshot
`notifyRoundChange()`	Alerts participants when a round starts/ends.	roundId	push event

Dependencies

- Session Orchestrator
- Mobile/Web Client UIs
- Push Notification Provider

4.4 ChatGPT AI Integration Subsystem

Purpose

Provides AI-based idea generation and post-session summarization for team leaders and event managers.

Provided Services

Service	Description	Inputs	Outputs
`generateAIideas()`	AI generates new ideas based on context.	topic, previousIdeas	aiIdeaList
`summarizeSession()`	Summaries all session ideas.	sessionId	summaryText
`chatInterface()`	Conversational refinement messages.	prompt	aiResponse

Dependencies

- OpenAI API
- Session Orchestrator
- Persistent Data Subsystem (for idea storage)

4.5 Reporting & Audit Subsystem

Purpose

Creates downloadable reports, exports, and maintains logs.

Provided Services

Service	Description	Inputs	Outputs
`generateReport()`	Creates PDF/CSV session report.	sessionId	reportFile
`exportIdeas()`	Exports all ideas.	teamId/sessionId	CSV
`logEvent()`	Saves system activities.	eventData	logId
`viewHistory()`	Retrieves past session data.	eventId	historyList

Dependencies

- Persistent Data Management
- AI Subsystem (for summaries)

4.6 Authentication & Access Control Subsystem

Purpose

Ensures secure login and role-based permissions.

Provided Services

Service	Description	Inputs	Outputs
`login()`	Authenticates user via JWT.	email, password	token
`verifyRole()`	Checks user authorization.	userId, requiredRole	allowed/denied

`refreshToken()`	Refreshes expired tokens.	oldToken	newToken
`encryptData()`	Encrypts sensitive fields.	data	encryptedData

Dependencies

- Persistent User Data
- All other subsystems enforce access control

4.7 Persistent Data Management Subsystem

Purpose

Stores all persistent entities (topics, teams, rounds, ideas, users).

Provided Services

Service	Description
`saveEntity()`	Generic CRUD persistence layer.
`loadEntity()`	Retrieves a record by ID.
`listEntities()`	Returns collections with filters.
`saveHistory()`	Stores completed sessions.
`transactionalWrite()`	Ensures atomic updates across services.

Dependencies

- All subsystems
- RDBMS (PostgreSQL/MySQL)

4.8 Client Applications (Web & Mobile)

Purpose

Provide UI screens for event managers, team leaders, and participants.

Provided Services

Service	Description
`renderDashboard()`	UI rendering service for role dashboards.
`submitIdeasUI()`	Three-idea input interface.
`receiveRealtimeUpdates()`	WebSocket event listeners.
`runTimersUI()`	Shows countdown animation.

Dependencies

5 Real-Time Sync Layer

6 Authentication

7 Session Orchestrator

8 Glossary of Terms

6-3-5 Method

A structured brainstorming approach where **6 participants** each contribute **3 ideas** during **5 rounds**, typically with rotating idea sheets or digital equivalents.

Access Control

A security mechanism that ensures each user can only access operations permitted by their assigned role.

AI Integration Subsystem

The subsystem responsible for interacting with OpenAI’s API to perform idea generation, clustering, and summarization.

API (Application Programming Interface)

The interface through which subsystems, services, or external clients access functionality via defined operations.

Authentication

The process of verifying user identity using credentials (email/password, JWT token, etc.).

Authorization

Determines whether an authenticated user is allowed to perform a specific operation based on their assigned role.

Brainstorming Session

A 6-3-5 event where a team participates in timed idea creation and idea-passing rounds.

ChatGPT

The AI model integrated through the OpenAI API to generate creative ideas and produce session summaries.

Client Application

Refers to the web or mobile front-end clients that participants use to interact with the system.

Countdown Timer

The timer that tracks the 5-minute duration for each round in the 6-3-5 process.

CRUD Operations

Create, Read, Update, Delete operations for persistent data entities like topics, teams, ideas, and participants.

Data Persistence Layer

The subsystem responsible for storing and retrieving data (e.g., using PostgreSQL or MySQL).

Dynamic Model

Represents how the system behaves over time (e.g., sequence diagrams and state transitions).

Entity

A persistent data structure stored in the system (e.g., Topic, Team, User, Idea, Session).

Event Manager

A high-privileged user responsible for creating events, managing participants, organizing teams, and overseeing all brainstorming operations.

Frontend

User interface components running on web or mobile devices.

Idea Object

A unit of content generated by participants or AI during brainstorming, stored as part of a round.

JWT (JSON Web Token)

A secure token used in authentication to identify a user and validate session access.

Participant

A user taking part in a brainstorming session. Can be a Team Leader or Team Member.

Real-Time Sync Layer

The subsystem responsible for live updates, state synchronization, and WebSocket communication.

Repository

A logic component in the persistence layer responsible for performing database operations for a specific entity type.

RESTful API

A standardized architectural style for backend services using HTTP verbs to access resources.

Role-Based Access Control (RBAC)

Authorization method assigning specific permissions depending on user roles (Event Manager, Team Leader, Team Member).

Round

A 5-minute segment within the 6-3-5 methodology during which participants enter their three ideas.

Session Orchestrator

Core subsystem that manages round timing, idea passing, and ensures that 6-3-5 rules are enforced.

Subsystem

A logical group of components within the software architecture providing a cohesive set of services.

Team Leader

A participant responsible for moderating brainstorming sessions and managing team-specific operations.

Team Member

A participant who contributes ideas during each round of the brainstorming session.

Traceability Matrix

A mapping between system requirements and the design or implementation elements that satisfy those requirements.

Use Case

A defined interaction between a user and the system describing how a specific functional goal is accomplished.

WebSocket

A bi-directional communication protocol used for real-time updates and synchronization between clients and server.

9 Traceability

Traceability ensures that every architectural decision and subsystem service within the Software Design Document (SDD) can be directly linked to the requirements defined in the Requirements Analysis Document (RAD).
It also confirms that all functional and non-functional requirements are addressed by at least one subsystem and design element.

This section provides:

6.1 Traceability Approach

The project uses a **bidirectional traceability strategy**:

Requirement → Design Element

Ensures that every Functional Requirement (FR) and Nonfunctional Requirement (NFR) defined in the RAD is implemented by:

- A subsystem
- A subsystem service
- A design component
- A diagram

Design Element → Requirement

Ensures that every subsystem, service, and architectural component is present for a reason and is backed by at least one requirement.

Traceability is maintained at 4 levels:

1. **Project Objectives**
2. **Functional Requirements (FR)**
3. **Nonfunctional Requirements (NFR)**
4. **Subsystem Services in SDD**

6.2 Functional Requirements → Subsystems Traceability Matrix

Requirement ID	Description (Summary)	Related Subsystem(s)	Design Element(s)
FR-100–115	Topic/participant/team CRUD	Event & Directory Management	createTopic(), registerParticipant(), createTeam()
FR-200–210	6-3-5 method, rounds, idea passing	Session Orchestrator	startRound(), passIdeasNext(), submitIdeas()
FR-300–307	Role-based access, reports, team-level data	Authentication & Access Control, Reporting & Audit	verifyRole(), generateReport()
FR-400–403	AI idea generation & summarization	AI Integration Subsystem	generateAIideas(), summarizeSession()
FR-500–503	Real-time synchronization	Real-Time Sync Layer	broadcastTimer(), notifyRoundChange()
FR-600–602	Reporting, export, logs	Reporting & Audit	exportIdeas(), logEvent()
FR-700–702	Authentication, authorization, security	Authentication & Access Control	login(), refreshToken(), encryptData()

6.3 Nonfunctional Requirements → Architecture Mapping

NFR ID	Requirement Category	Satisfied By	Supporting Components
NFR-100–103	Usability	Client Applications	responsive UI, guided workflows
NFR-200–202	Reliability	Real-Time Sync Layer, Persistence	auto-save ideas, failover behavior
NFR-300–303	Performance	Sync Layer, Session Orchestrator	<500 ms sync target, load-handling architecture
NFR-400–401	Supportability	Reporting & Audit, Persistence Layer	audit logs, maintainable repositories

NFR-500 –504	Implementation	All subsystems	RESTful APIs, modular subsystem decomposition
NFR-600 –604	Interface	Client Apps, AI Integration	conversational UI, mobile optimization
NFR-700 –703	Packaging	Deployment / Client	web app, mobile app packaging, notification support
NFR-800 –802	Legal	Authentication, Data Persistence	encryption, privacy compliance

6.4 Objectives → Subsystems Mapping

Objective (From RAD Section 1.3)	***Supporting Subsystems***	***Supporting Services***
O1 – Streamline event management	*Event & Directory Management*	*CRUD operations, team creation*
O2 – Implement 6-3-5 method	*Session Orchestrator*	*round control, timers, idea passing*
O3 – AI-assisted creativity	*AI Integration Subsystem*	*generateAIideas(), summarizeSession()*
O4 – Real-time collaboration	*Real-Time Sync Layer*	*broadcastTimer(), realtime state sync*
O5 – Security & data integrity	*Authentication & Access Control, Persistence*	*JWT authentication, role verification, encrypted storage*