
FlipFit – Gym Management System
JEDI BOOT CAMP 2026
BY TEAM SIGMA

The Ask – Where we started

- Design UI & Backend for FlipFit enterprise app
- Partner with gyms across Bangalore
- Manage multiple centers with fixed hourly slots
- Core functionality: Register, View Availability, and Book Workouts

AGENDA

01 Our Journey

02 Our Team

03 Project Goals

04 Engineering Practices

05 Tech Stack

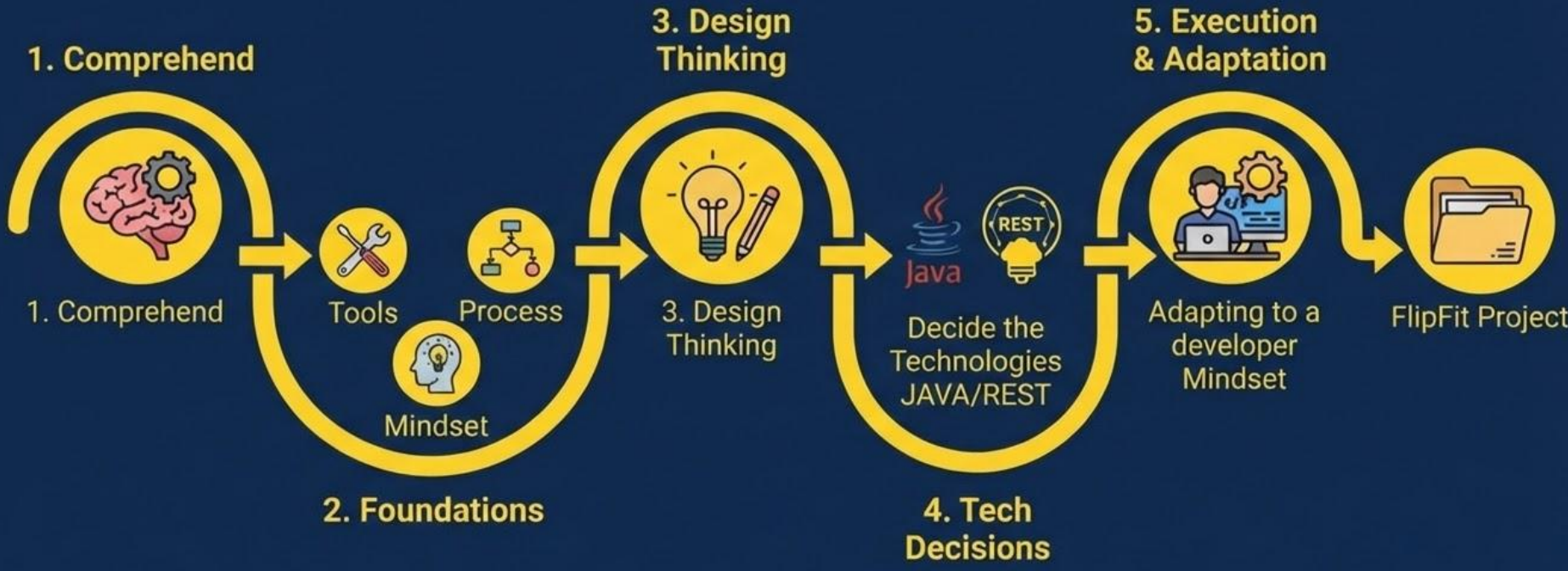
06 Development

07 Challenges & Learnings

08 Demo

09 Questions

Our Journey: Understanding Project Goals



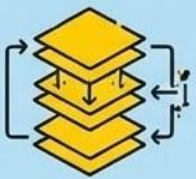
OUR TEAM - SIGMA

- Rishit Parija
- Ankush
- Ayushi Sinha
- Deergha Kulkarni
- Kannika Bhat
- Rajul Chaudhary
- Rohit Shah
- Greeshma Chowdary
- Shailendra Sahu
- Swayam Agrawal

Project Goals

- **Vision:** To build a robust, persistent **Gym Management System** (FlipFit) that bridges the gap between Bangalore fitness centers and users.
- **Comprehend:** Implementing the ability to view all centers in a city and check **real-time availability** for specific 1-hour slots.
- **Process Logic:**
  - Ensuring **zero overbooking** through synchronized seat management.
  - **Waitlist & Promotion:** Implementing logic to manage a waiting list once seats are filled and automatically promoting users when vacancies occur.
  - **Notification System:** Developing a mechanism to notify waitlisted candidates immediately upon their promotion to a confirmed booking.
- **Tools & Quality:** Transitioning from in-memory arrays to a **persistent MySQL database** while maintaining modular interfaces (DAOs) for center and booking data.

ENGINEERING PRACTICES



MVC ARCHITECTURAL PATTERN

Separated concerns into **Models (POJOs)**, **DAOs (Data Access)**, and **Services** for modularity.

Used a **Menu-driven Console UI** to handle interactions.

DATABASE PERSISTENCE (JDBC)

Strictly moved from in-memory to **Persistent MySQL Database** integration.

Implemented **DAO (Data Access Object)** patterns for SQL queries.

OBJECT-ORIENTED DESIGN (OOD)

Leveraged **Encapsulation** and **Interfaces** for user types.

Utilized **Java Collections** (Lists/Sets) to process data.

DEFENSIVE PROGRAMMING

Implemented **Exception Handling** for SQL errors and invalid inputs.

Used **Input Validation** to ensure correct data formats.

PROFESSIONAL SDLC WORKFLOW

Utilized **Git** for version control with clear commit histories.

Followed a **Documentation-first approach** for requirements.

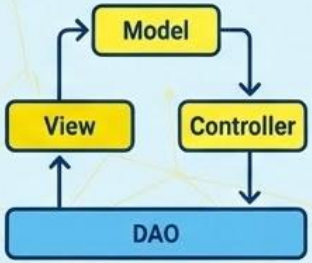
Tech Stack

Backend & Core Language

- **Java (JDK 17+):** Used for the core application logic, utilizing Object-Oriented Programming (OOP) principles.
- **Collections Framework:** Extensive use of List, Map, and Set for data manipulation within the service layer.

Data Persistence

- **MySQL:** Relational database used for permanent storage of center and booking information.
- **JDBC (Java Database Connectivity):** The bridge used to execute SQL queries and manage database transactions from the Java code.



Architecture & Design

- **MVC Pattern:** Separation into Models (POJOs), Views (Console UI), and Controllers/Services.
- **DAO Pattern:** Data Access Objects used to abstract and encapsulate all access to the data source.

Development Tools & Version Control

- Development Tools:
  - **Maven:** Project management and comprehension tool for managing dependencies like the MySQL Connector.
  - **IntelliJ IDEA / Eclipse:** Integrated Development Environments used for coding and debugging.
- Project Management & Version Control:
  - **Git:** Distributed version control system for tracking changes and collaborative development.

Development

Development Environment

- Utilized **Spring Tool Suite (STS)** as the primary IDE for its integrated Maven support and specialized tools for Java enterprise development.

Core Feature Development & Sprint Lifecycle

- **Core Feature Development:**
  - **Business Logic:** Implemented services for center management, slot allocation, and seat concurrency checks.
  - **User Interaction:** Built a menu-driven console interface to process user inputs.
- **Sprint Lifecycle:**
  - **Requirement Analysis:** Mapping the FlipFit Beta launch requirements into functional modules.
  - **Implementation:** Incremental coding of user registration, center discovery, and booking logic.
  - **Code Quality:** Regular peer reviews and local testing within the STS environment before final commits.

Architectural Implementation

- **Layered MVC Structure:** Organized the code into specific packages for Models (POJOs), DAOs (Data Access), and Services to ensure modularity.
- **DAO Pattern:** Developed specialized Data Access Objects to handle the interaction between the Java application and the MySQL database via JDBC.

Continuous Integration

- Managed project dependencies and builds using Maven, ensuring consistent environments across the team.

Challenges & Learnings

Key Challenges

Concurrency & Overbooking Logic

Designing a robust mechanism to ensure no center exceeded its fixed seat capacity during simultaneous booking attempts.

Architecture Integration

Connecting the layered backend logic to a menu-driven console interface for a seamless user experience within the STS environment.

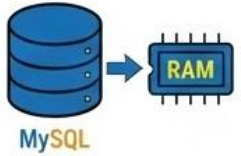
JDBC Data Mapping

Transitioning from simple Java objects to a persistent MySQL schema while ensuring complex relationships between Users, Centers, and Slots remained intact.

Waiting List Management

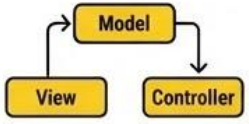
Developing a promotion system that accurately tracks the queue and notifies waitlisted candidates when a seat becomes available.

Key Learnings



Persistence over In-Memory

Gained hands on experience in why enterprise applications require persistent databases (MySQL) rather than temporary in memory storage



MVC Modularity

Learned the importance of separating business logic (Services) from data access (DAOs) to make the code maintainable and testable.

Developer Mindset

Transitioned from writing simple scripts to building a structured, menu-driven enterprise application in the Spring Tool Suite (STS) environment.

Requirement Comprehension

Learned to break down complex user stories, such as "nearest time slot" recommendations, into executable Java methods.

DEMO


?
Questions?
Any questions from the audience?