FlipFit Enterprise

JEDI BOOT CAMP 2026 CAPSTONE PROJECT

PRESENTED BY

Team Edith

The Ask – Where We Started

Understanding the initial problem statement and the core expectations from the system.

- Design and develop a **backend system** for the FlipFit Enterprise application from scratch.
- Partner with multiple gyms across **Bangalore** and bring them onto a single unified platform.
- Support management of **multiple gym centers**, each operating with fixed 1-hour workout slots.
- Implement core functionality including **user registration, login, and role-based access**.
- Enable users to **view availability** and **book workout slots** without conflicts or overbooking.

Project Scope & Objectives

The Goal: Transform a manual gym booking process into a **robust, digital enterprise backend**.

Multi-Center Management

Scalable architecture designed to seamlessly handle management and discovery for multiple gym centers across Bangalore.

Slot Logistics

Implementation of rigid 1-hour slots with strict fixed seat capacity enforcement to prevent overbooking scenarios.

Persistence

Full migration from volatile in-memory data structures to a robust, persistent MySQL relational database system.

Execution Timeline

Delivery Roadmap

Phase 1

FOUNDATION PHASE

- **Requirement Analysis** & Database Design
- Core Java Implementation
- SME Consultations & Logic Planning
- Setting up Development Environment

Phase 2

REFINEMENT PHASE

- Integration of Backend with **MySQL (JDBC)**
- Comprehensive Testing & Bug Fixes
- Code Refactoring & Optimization
- Final Demo Preparation

The Squad

Project Stakeholders & Engineering Team

Deepanshu Singh (Leader)

Namita Bhatt (SubGroup Leader)

Shivam Singh Parihar

Ritik Kumar Gupta

Shivaramakrishna Karthikeyan

Dhruti Bhadoria

Shounak Kulkarni

Shubham Jha

Sambhav Singla

Parasagoni Vani

Technical Blueprint

Architecture & Development Tools

The Stack

CORE TECHNOLOGIES

Java 17 (JDK)
Core Backend Language

MySQL
Relational Persistence Database

Maven
Dependency & Build Management

MVC + DAO
Architectural Design Pattern

Dev Tools

ENVIRONMENT & QUALITY

IntelliJ / Eclipse / STS
Integrated Development Environment

Git & Github
Version Control & Collaboration

Draw.io
Documentation & Design

Core Functionalities

System Capabilities & Logic

User Ops

CLIENT SIDE

- **Registration**
  Secure onboarding with validation
- **Center Discovery**
  Browse locations & facilities
- **Availability Checks**
  Real-time slot status view

Conflict Resolution

Intelligent logic that automatically detects overlaps. System triggers **auto-cancellation** of existing conflicting slots if a user books a new gym for the same time window.

Capacity Guard

Strict enforcement of seat limits. Atomic checks prevent **overbooking** beyond defined *n* slots per session, ensuring compliance with safety protocols.

Waitlist Algo

BONUS FEATURE

Automated queue management system. When a confirmed booking is cancelled, the algorithm instantaneously **promotes** the next user in the FIFO (First-In-First-Out) queue to a confirmed status and triggers notifications.

Code Quality & Practices

Engineering Standards & Workflow

FLIPFIT STANDARD

Modularity

ARCHITECTURE

Strict separation of concerns ensures maintainable and scalable code.

- Use of `Interfaces` for abstraction
- Decoupled implementation logic
- Component-based design pattern

Version Control

WORKFLOW

Structured Git workflow to manage collaboration effectively.

- Feature-branch strategy
- Pull Request (PR) reviews
- Meaningful commit messages

Defensive Coding

RELIABILITY

Proactive handling of edge cases and potential system failures.

- Robust `try-catch` blocks
- Input validation & sanitization
- Graceful SQL failure handling

Analysis

QUALITY ASSURANCE

Continuous inspection of code quality to detect bugs early.

- Static code analysis
- Automated bug detection

Roadblocks & Resolutions

Technical Hurdles & Engineering Solutions

CHALLENGE 01

Data Volatility

"Moving from In-Memory to Storage"

Initial prototype relied on volatile array lists, risking **critical data loss** during system restarts or crashes.

RESOLUTION

JDBC Persistence

"Robust DAO Pattern Implementation"

Implemented a DAO layer with **JDBC transactions** connecting to MySQL. Ensures ACID properties and permanent storage.

CHALLENGE 02

Race Conditions

"Simultaneous Booking Conflicts"

Concurrent users attempting to book the last slot resulted in **overbooking** and logical state inconsistencies.

RESOLUTION

Atomic Synchronization

"Thread-Safe Booking Logic"

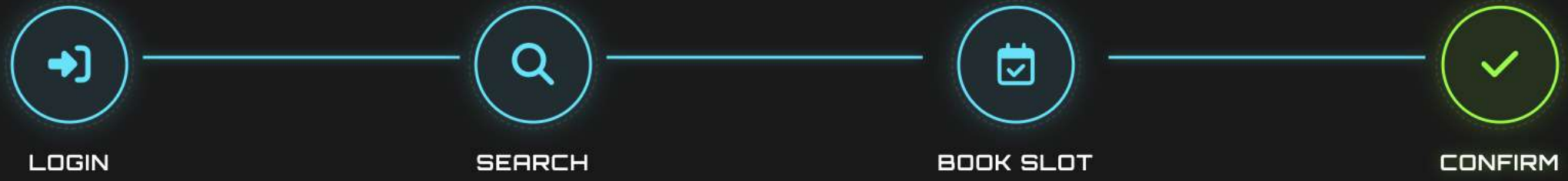
Applied **synchronized methods** and atomic transaction logic to serialize critical write operations and prevent conflicts.

Application Walkthrough

User Journey & Future Scope

LIVE DEMO READY

User Flow



Next Steps

Frontend Integration
Development of SPA using React/Angular

Payment Gateway
Secure processing for paid subscriptions

THANK YOU

Questions?

JEDI BOOT CAMP 2026