FLIP FIT GYM

OUR TEAM

Kumar Sourav

Samprati Vishnoi

Mohit Kumar

Sakshi Goyal

Firdaus Javed

Anushka Vijay

FRAMEWORK FOR 6 DAYS

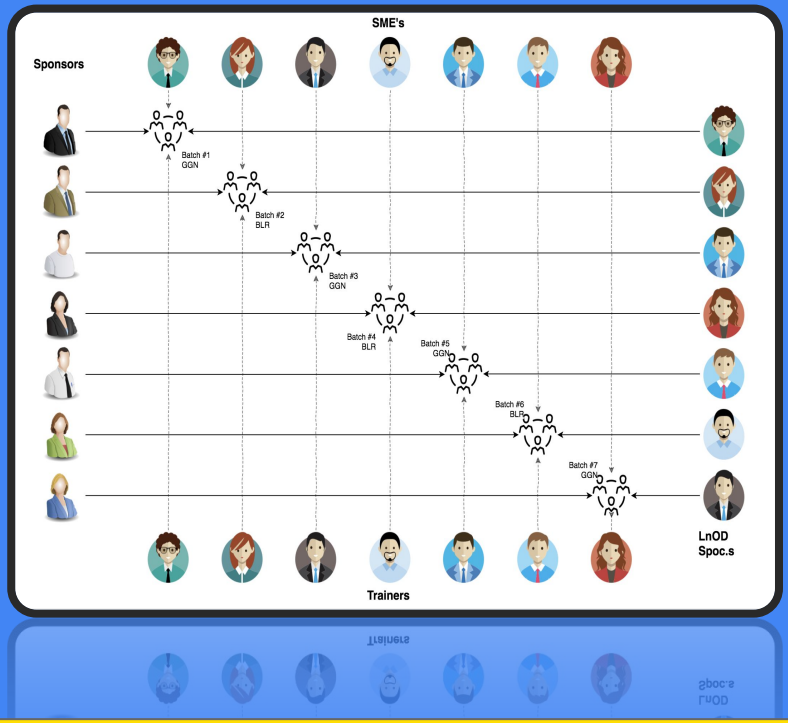
- Talks about various subjects and technologies.
- Educate us on the Problem Statement and Tech Stacks.
- Workflow diagram for the problem statement.
- Overview of Git and Github.
- Worked on Project Skeleton and UML Artifacts.
- Developed the POS application.
- Gained knowledge of MySQL and linked the database to the POS application.
- Converted the POS Application to Dropwizard.

Stakeholders

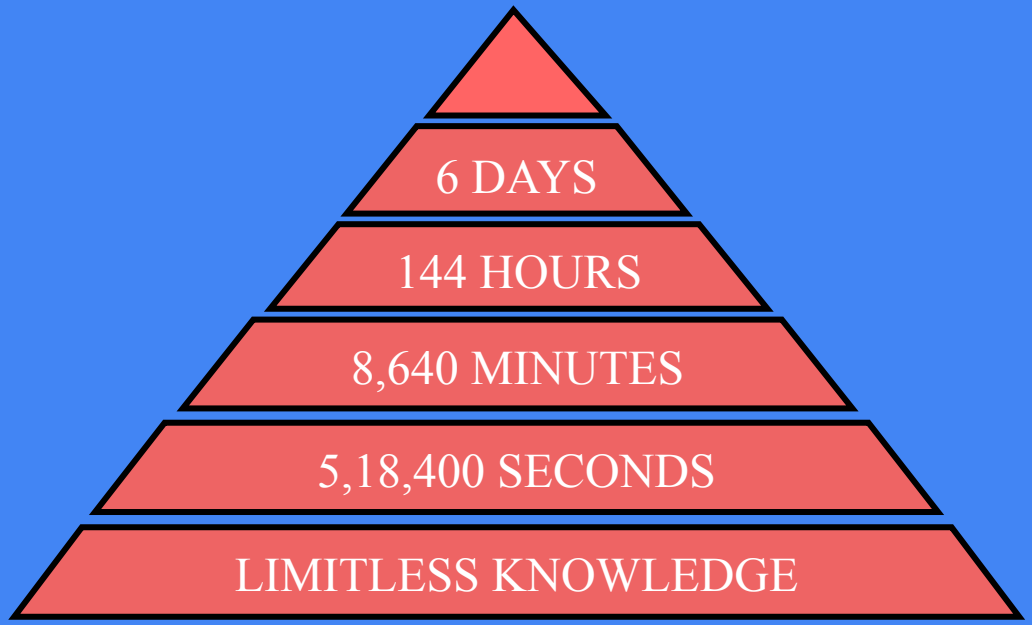
1. **Sponsors**
   Flipkart

2. **SME's**
   Mr. Amit Balyan

3. **Trainers**
   Ms Anushka Khanna



6 DAYS OF TRAINING



Agenda

01 Our Journey

02 Our Team

03 Team Structure

04 Project

05 Tech Stack

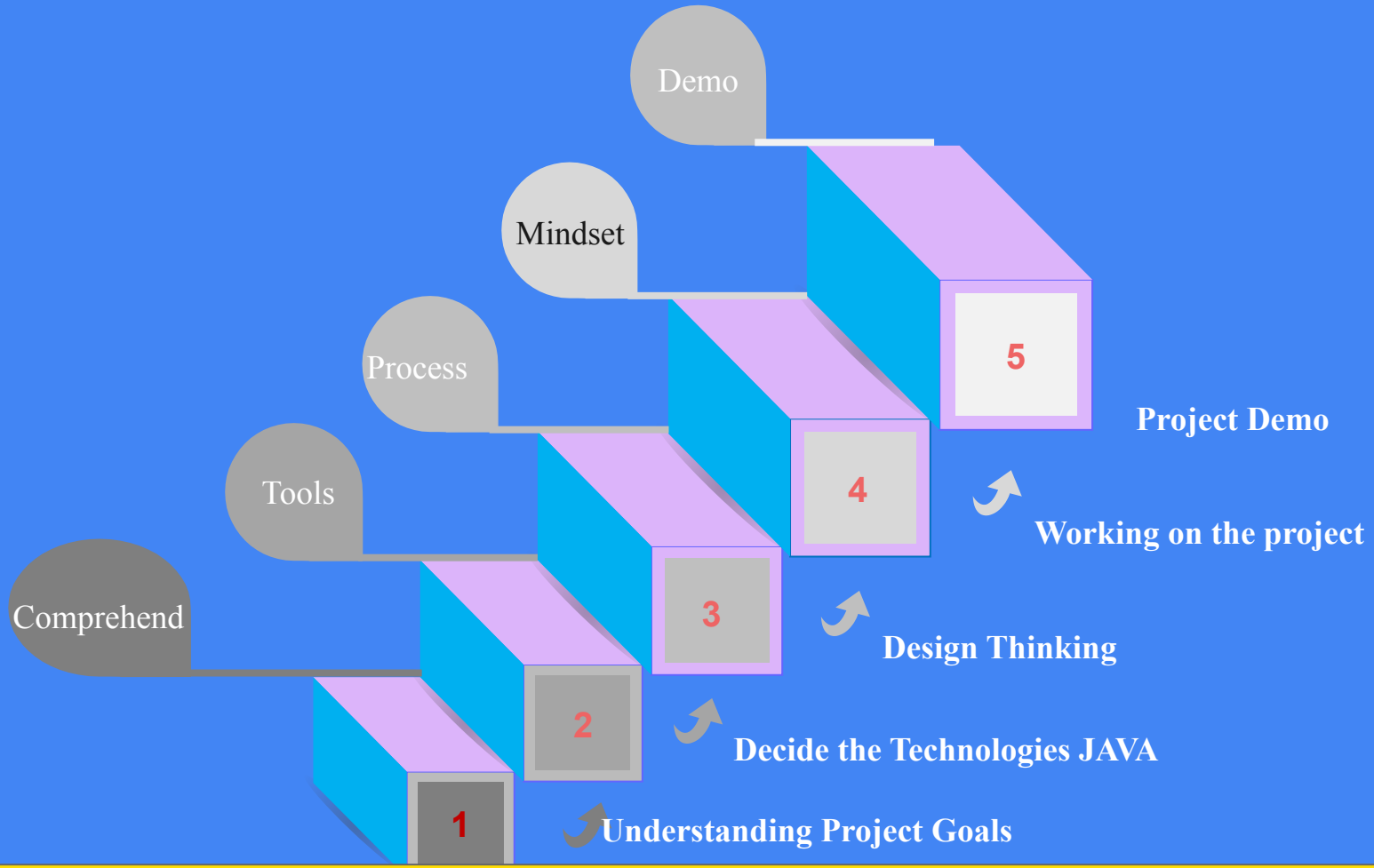
06 Development

07 Challenges

08 Demo

09 Questions

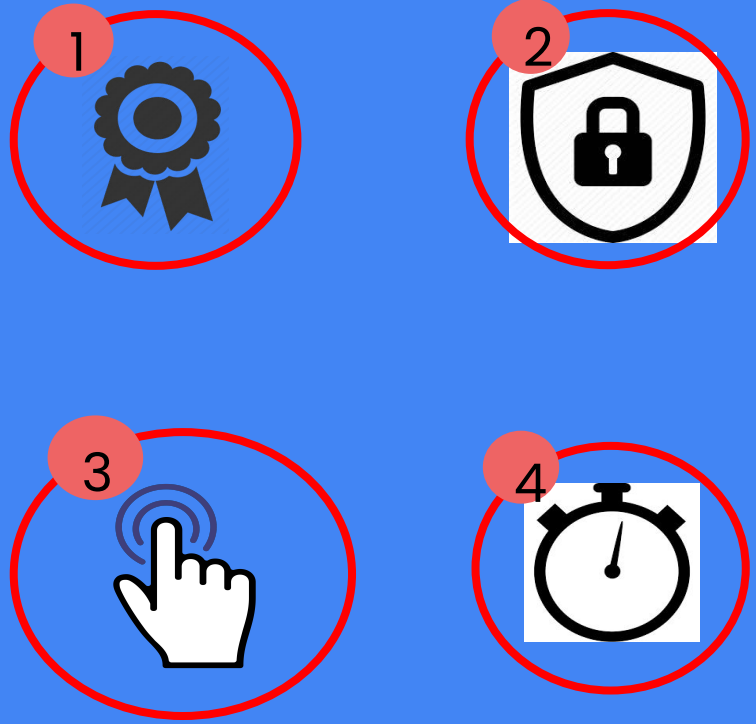
Our Journey


Demo
Mindset
Process
Tools
Comprehend
5
4
3
2
1
Project Demo
Working on the project
Design Thinking
Decide the Technologies JAVA
Understanding Project Goals

Project Goals

OUR VISION

To design and implement **FlipFit Application**(Gym Management System) where user can view gym centres and slots available, book and cancel slots. Gym owner can register his gym and add slots for a particular gym, view bookings etc. Admin can approve register requests for gym owner, gym centres and users.

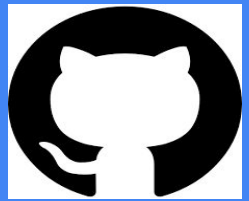


Tech Stack

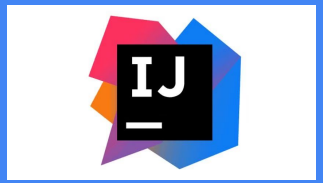
Backend

Data

Tools



IDE



Engineering Practices - We used

1. **Single Responsibility Principle (SRP):**
   Each part of your code should do one specific thing. For example, if you have a class that handles user data, it shouldn't also handle GymOwner data.

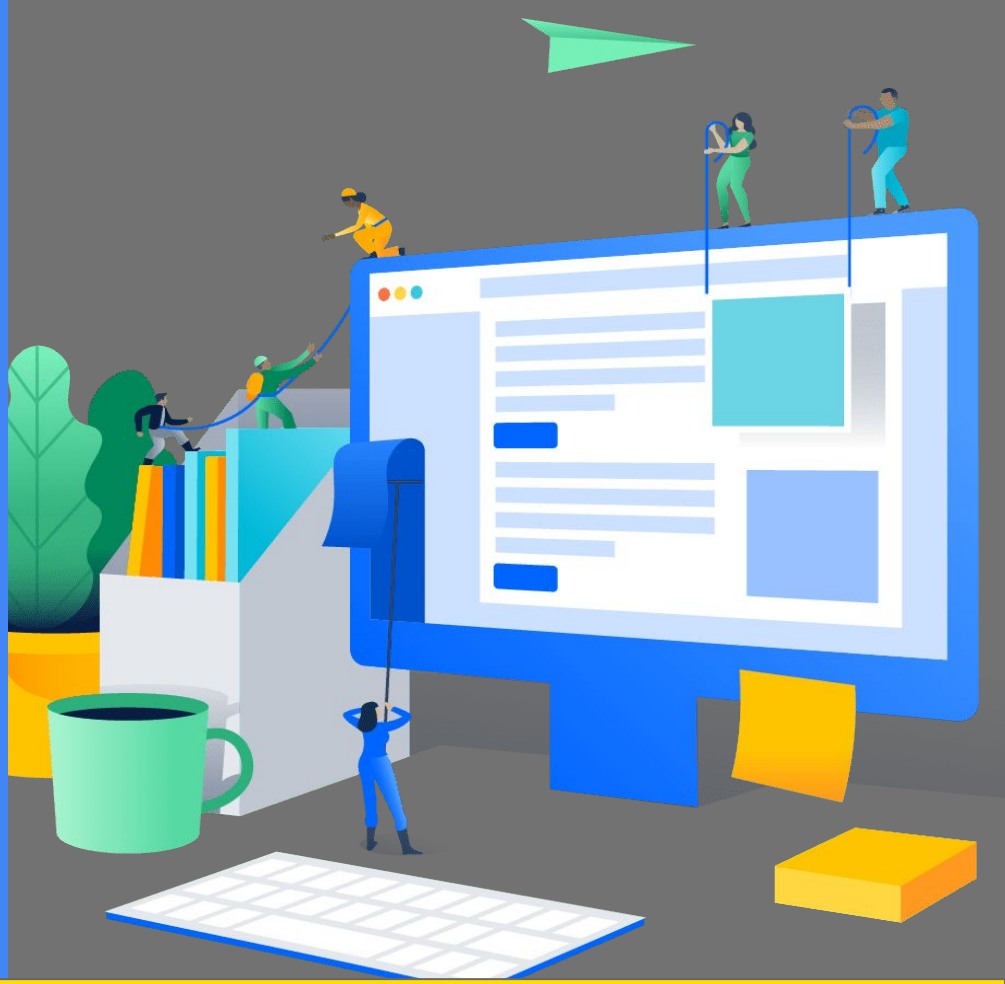
2. **Open Closed Principle (OCP)**:
   Your code should be easy to extend with new features without changing existing code. Think of it like adding a new app to your phone without needing to update the phone's operating system.

3. **Liskov Substitution Principle (LSP)**:
   If you have a class that works in your code, you should be able to replace it with a subclass without breaking anything. Like using a universal remote that works with any TV.

4. **Interface Segregation Principle (ISP):**
   Don't force parts of your code to depend on things they don't need.

5. **Dependency Inversion Principle (DIP)**:
   High-level code (like the main logic of your app) shouldn't depend on low-level code (like database details).It's like a boss giving orders to a manager who then coordinates with the workers.

Development




Requirements Gathering
Feasibility Study
Project Charter
Risk Assessment
Timeline and Budget
Planning
Backlog Creation
Prioritization
Task Breakdown
Estimation
Sprint Planning
Grooming
Code Review
Commit
Go Live
Delivery
Development
Testing
Continuous Integration
Code Reviews
Documentation
Implementation

Challenges & Learnings

- JDK 17
- Git
- JDBC Connection
- Dropwizard
- Team Collaboration
- Tight Schedule

Demo

[Github]

Questions

THANK YOU

STAY FIT with FlipFit :)