Groofy Code

Competitive programming platform

Under the supervision of:

- *Dr. Mohamad Abdelwahab*
- *TA. ?*

Agenda

01
Overview

02
Supported Functionalities

03
Architecture

04
Technologies and Libraries used

05
Challenges

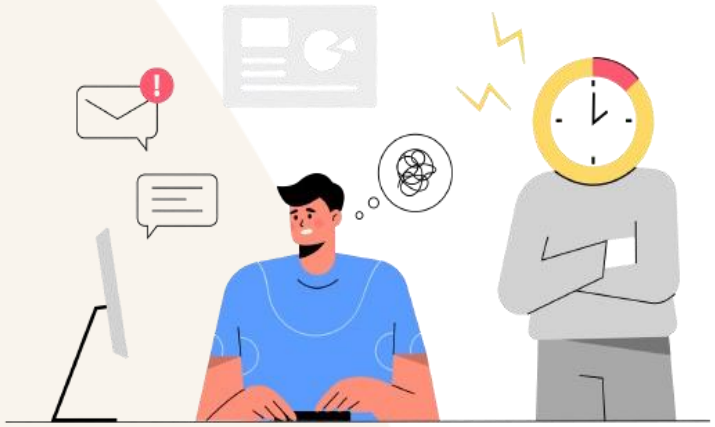
06
Traditional VS Machine Learning

07
Our Model

08
Demo

Overview

Introduction & Problem Description



Tasks

Deadlines

Why don’t I know how to implement it ?

Thinking of the solution

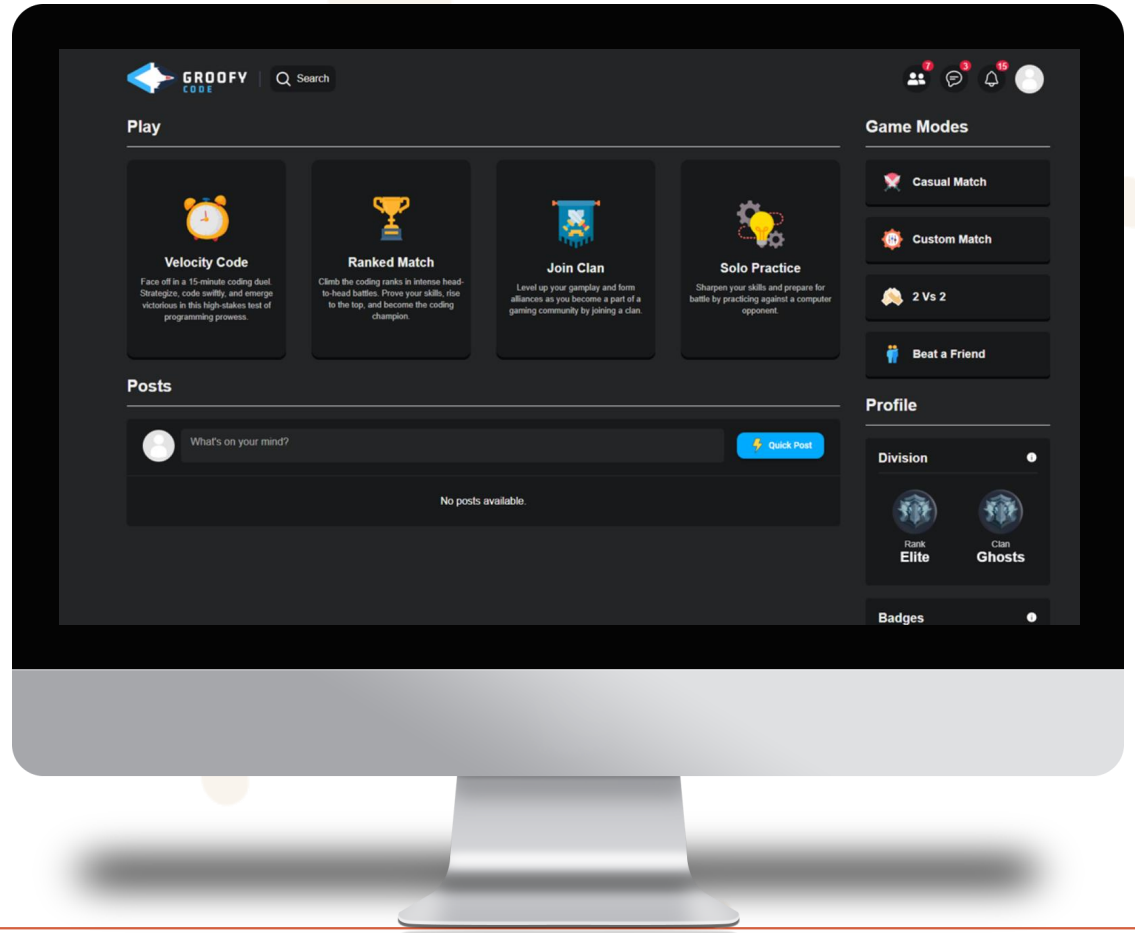
What should I do

1. Should I solve a lot of problems ?
2. Should I solve hard problems ?
3. What time should I spend in solving ?
4. What about challenging people ?
5. Leetcode ? Codeforces ? ….. etc

Motivation

- Competitive Programming Library
- FCAI-ICPC Community
- Discord Bot Challenging
- Website -> Gaming Platform

Our Solution: Groofy Code



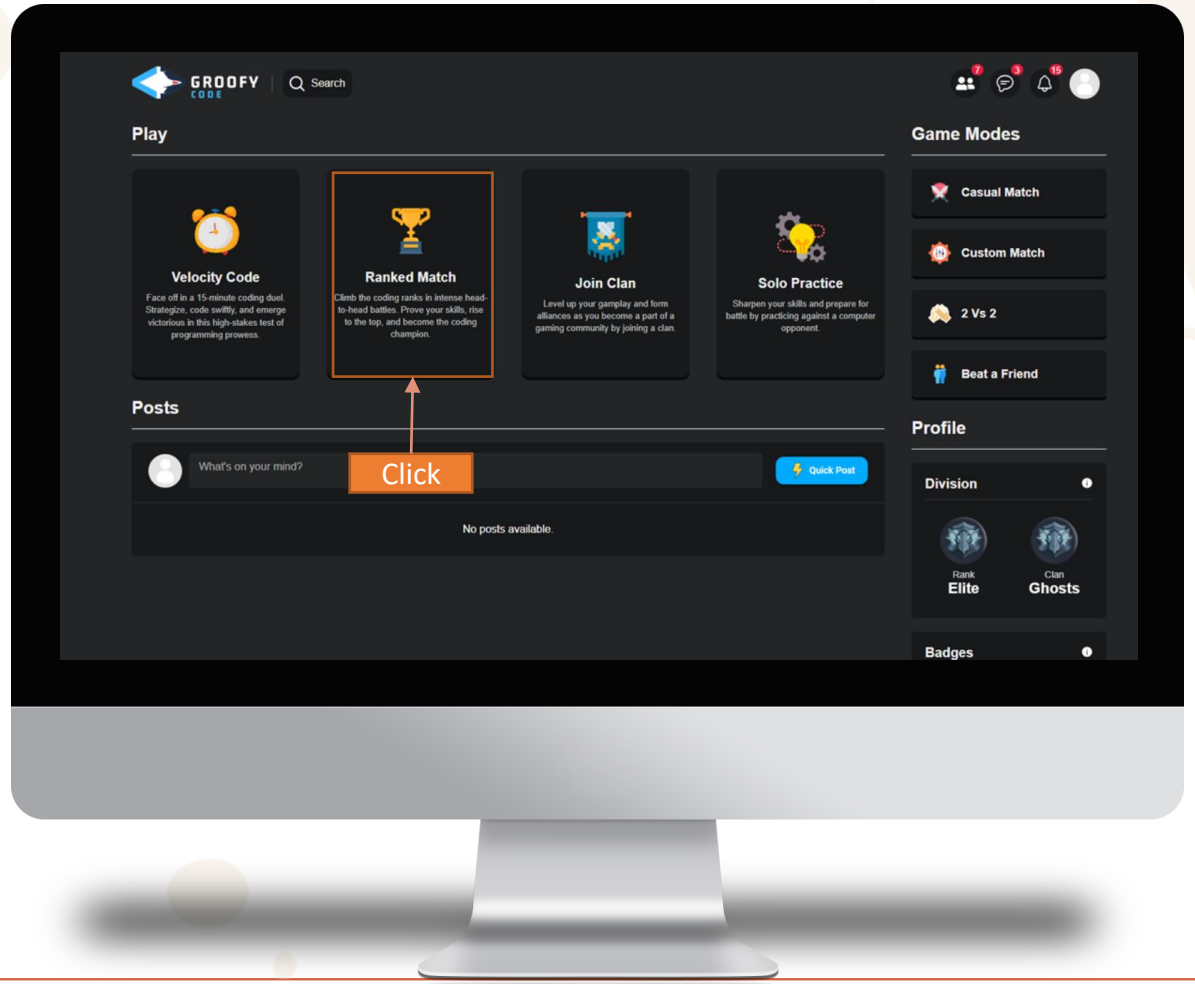
Supported Functionalities

Highlight Features

1. Challenge Mechanism
2. Comprehensive User System
3. Clan Features
4. Interactive Chat System

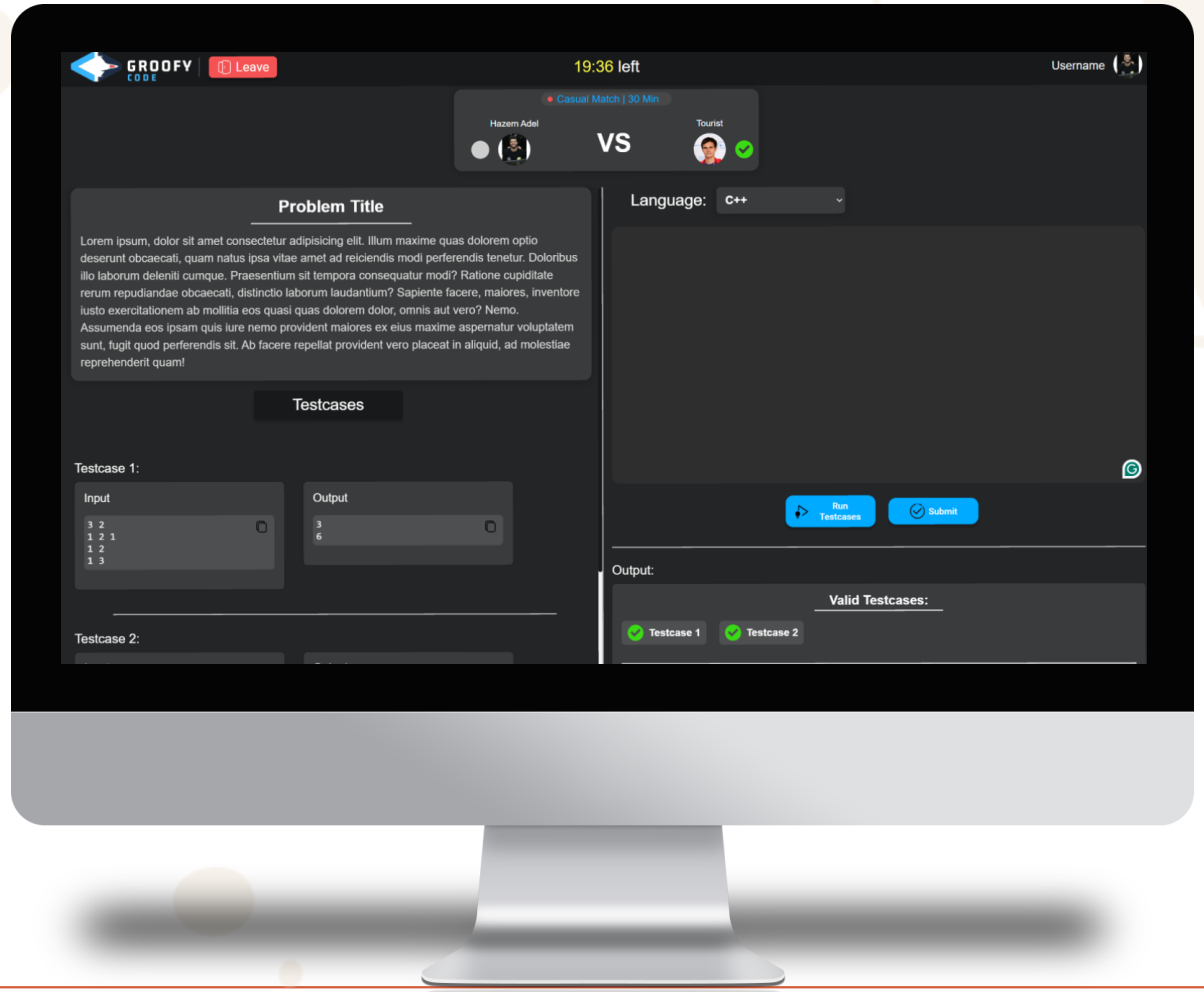
Match Modes

- Various Coding Challenges
- Difficulty Levels



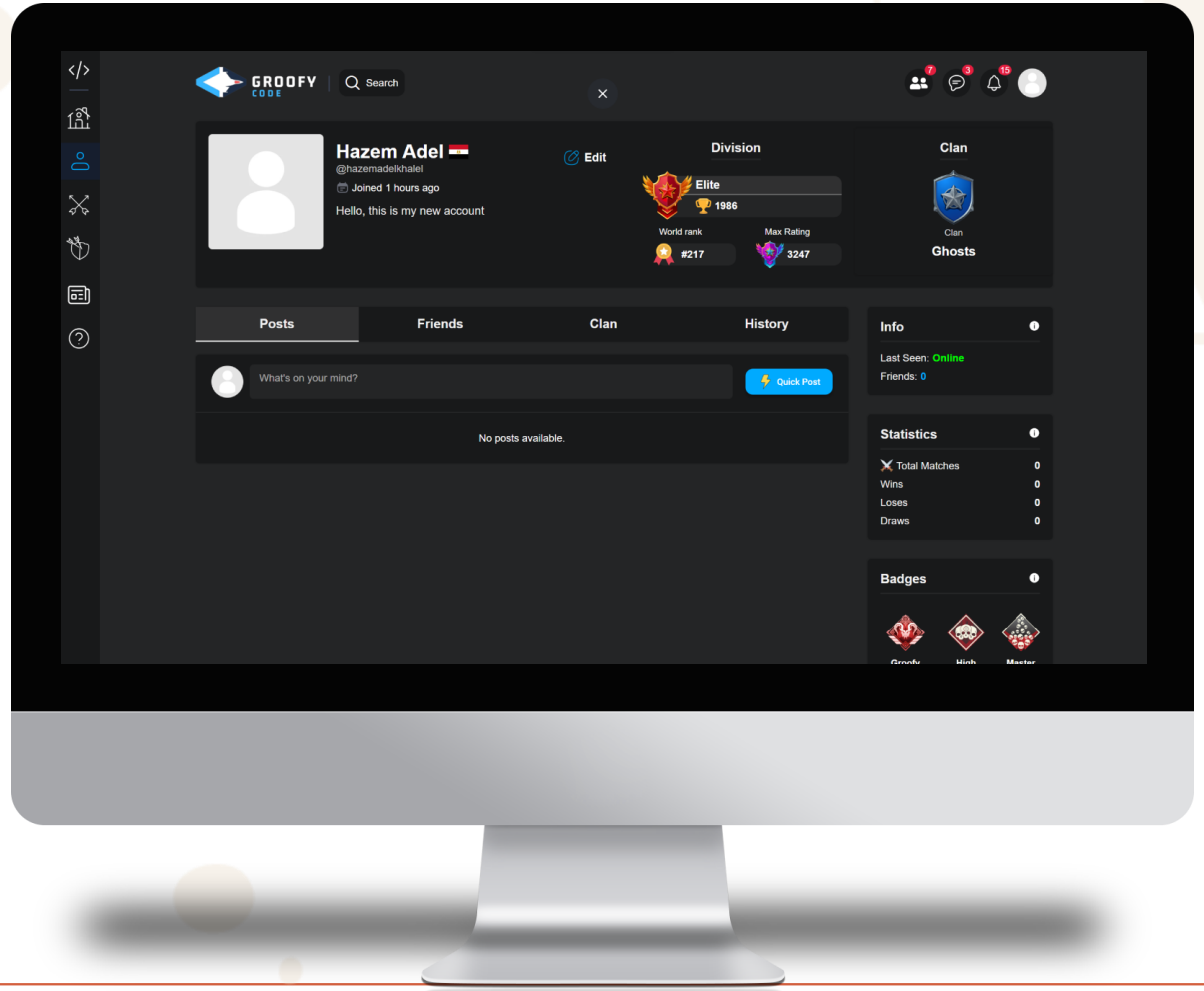
Match Challenge

- Real-time Code Editor
- Submission and Evaluation



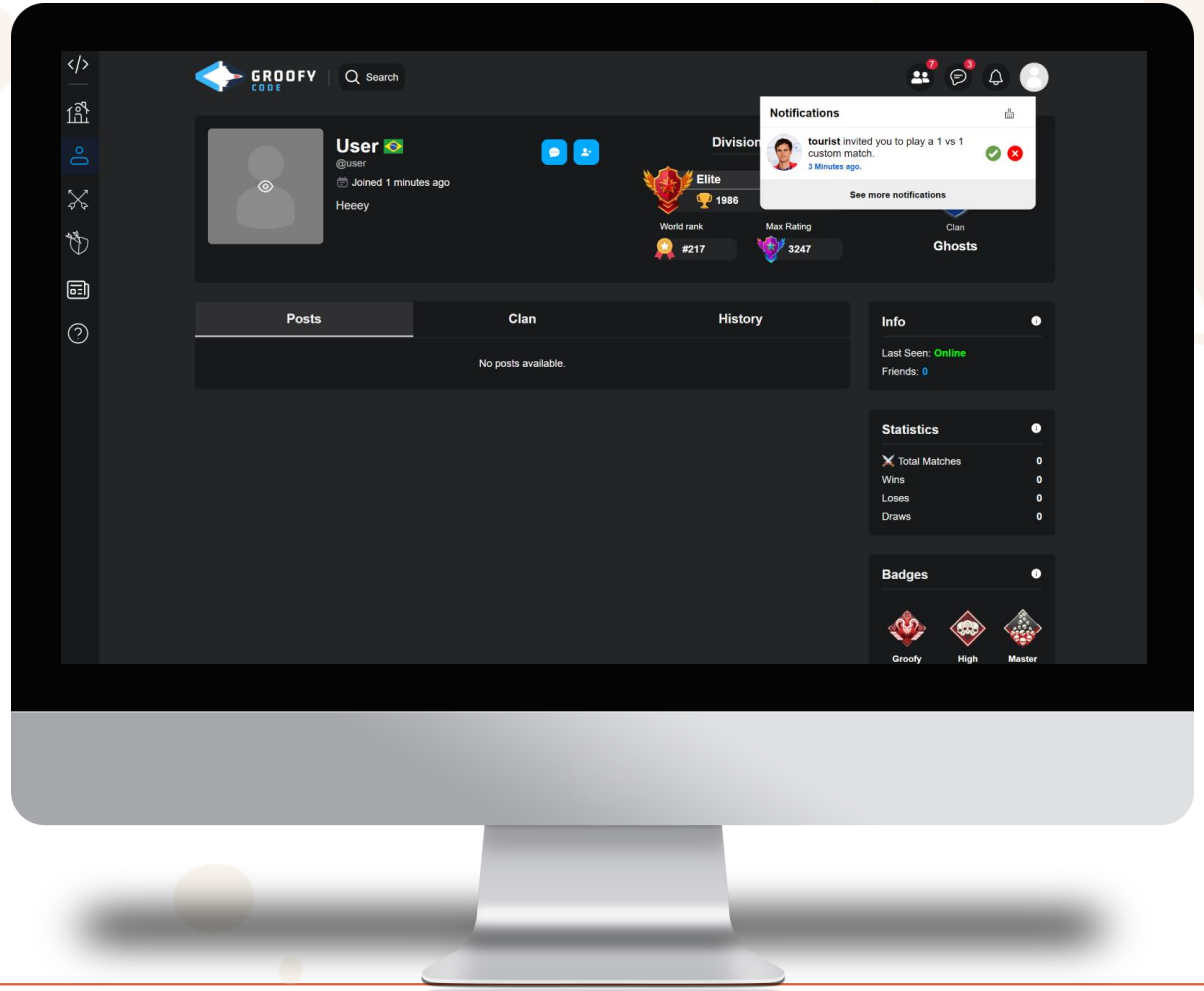
User Profile

- User Registration / Login
- Profile Creation
- Personalized Dashboard
- Insightful User Statistics



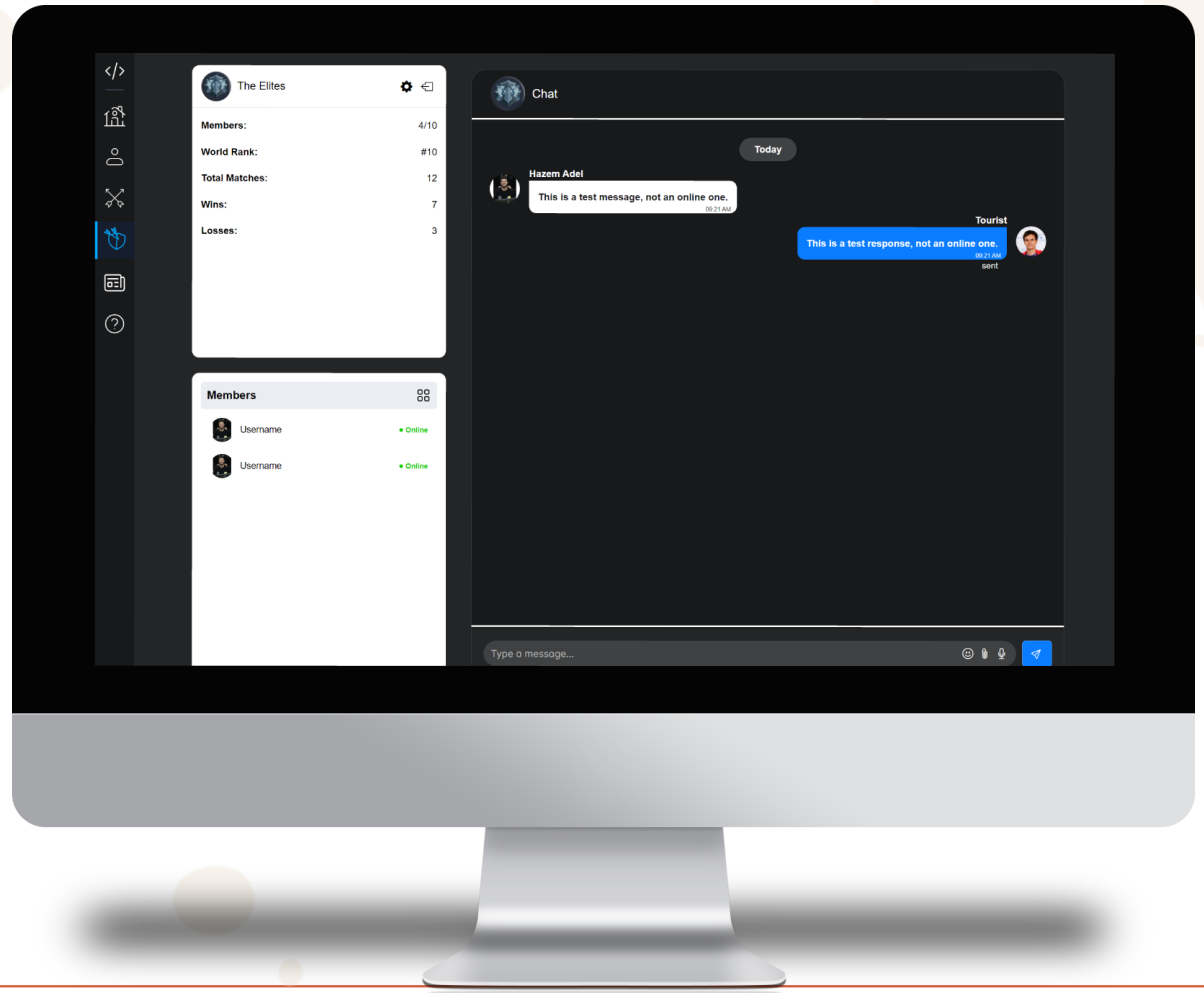
User Interactions

- View users’ profile
- Friend Requests
- Notification system
- Invite a Friend to Match/Clan
- Posts



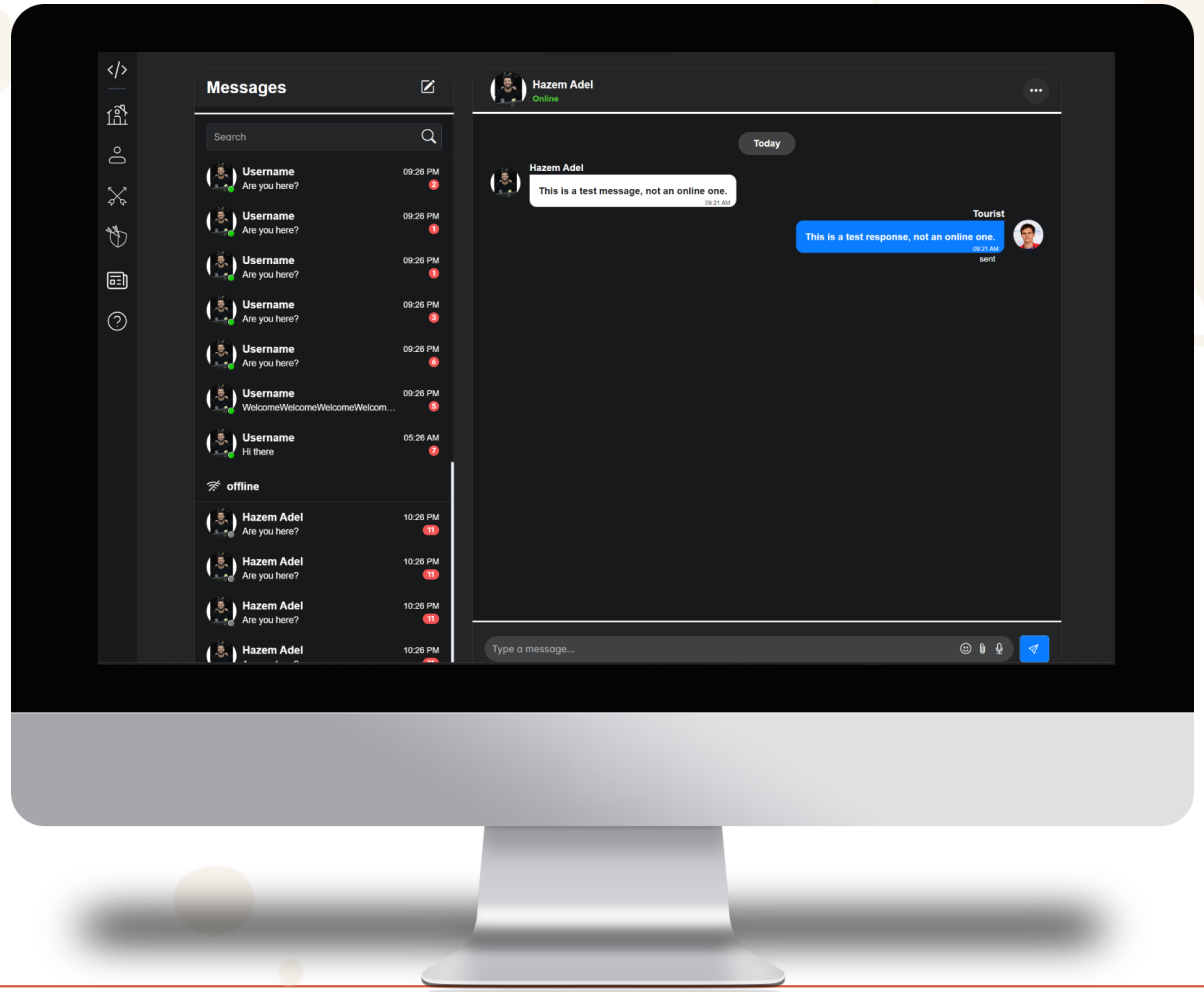
Clan Feature

- Viewing Clan members
- Collaborate with people
- Messaging members
- Play with them



Chat System

- Viewing Friends
- Message a user
- Search for a friend



Technologies and Libraries used

Technologies and Libraries Used

- Backend

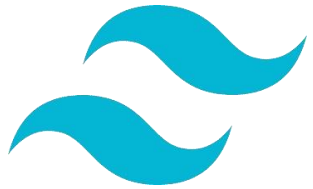





- Frontend





Agile Methodology

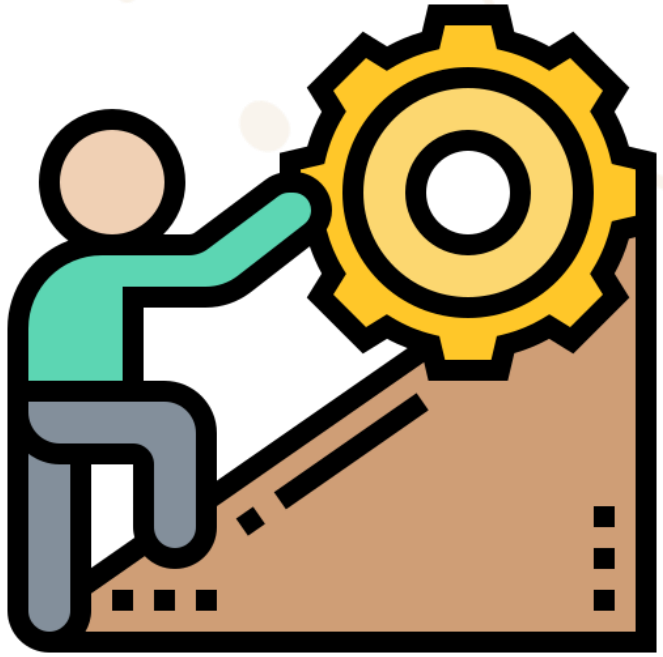
Agile Methodology

- Why Agile?
- Scrum Master
- Our Sprints

Challenges

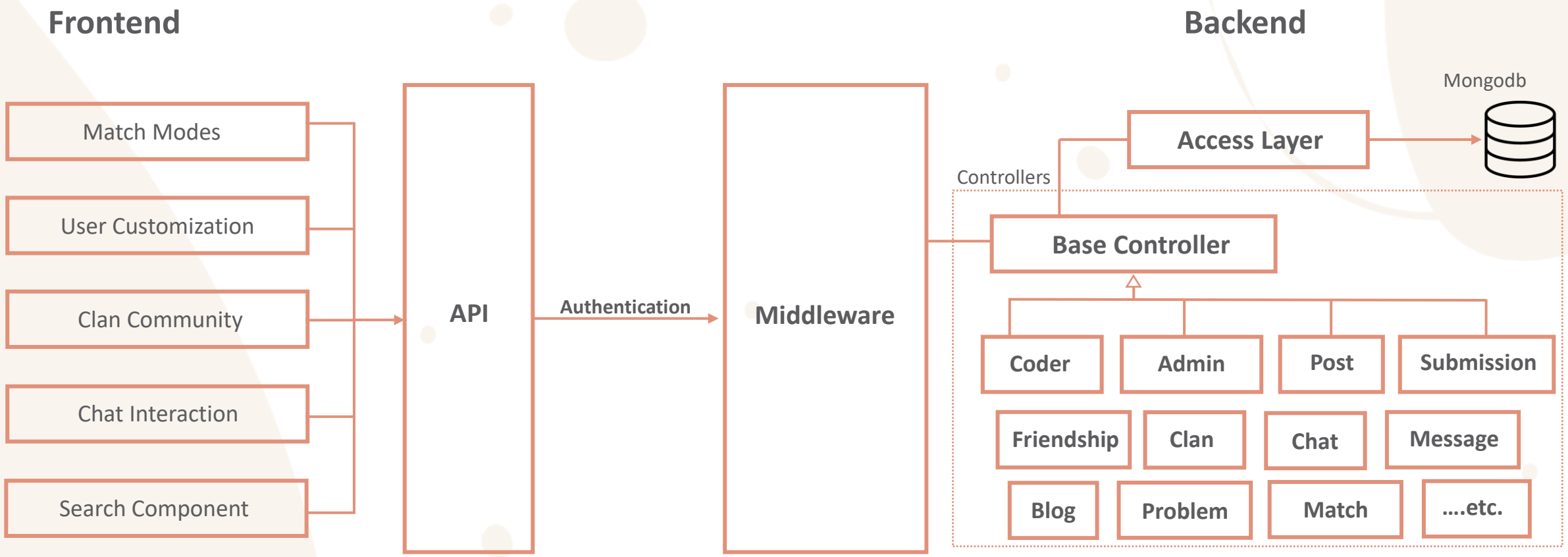
Challenges

- Lack of hands-on experience with used technologies.
- Integration with already existing website such as Codeforces, Codemirror and judge0
- Time & Collaboration constraints due to studies and work commitments
- Nodejs VS Java Spring boot



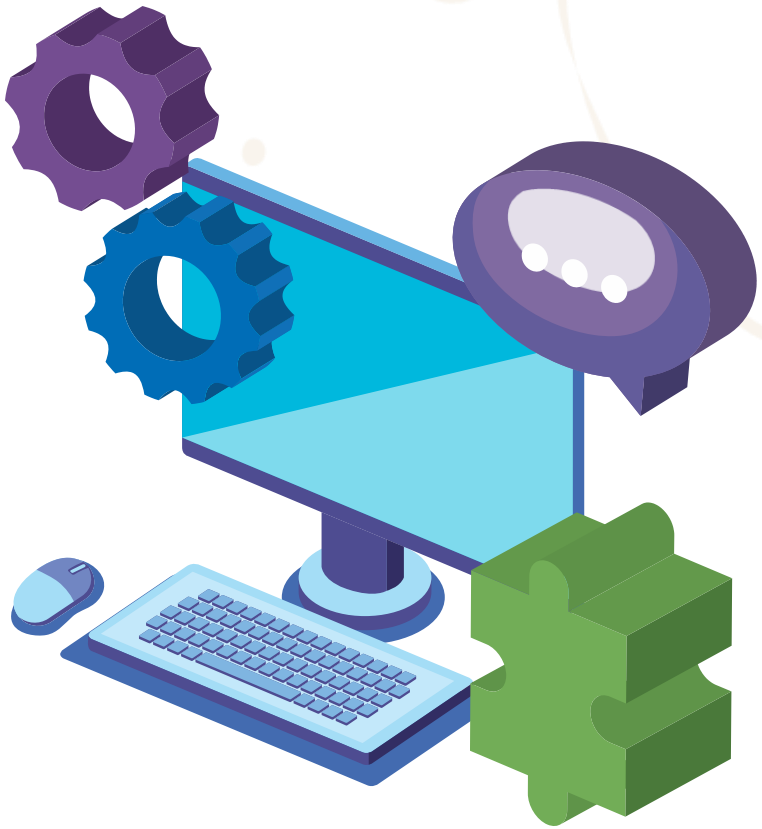
Architecture

Architecture



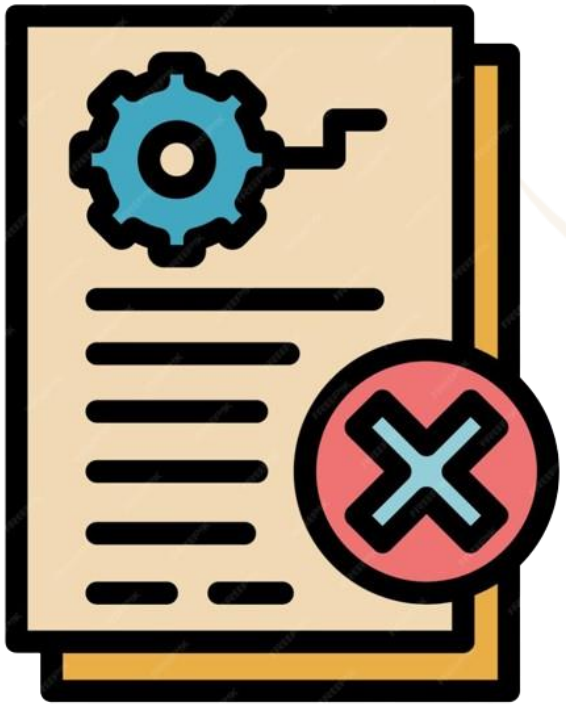
Design Patterns & SOLID Principles

- Strategy Pattern
  - Match Controller
- Singleton
  - Controllers
- Abstract Factory
  - Match with Controller



Non-Functional Requirements

- Scalability
  - Stable Server
  - Event-Driven & Non-Blocking I/O Backend
- Useability
  - Friendly User Interface
- Portability
  - Media Queries
- Performance
  - Single Page Application
  - Caching



Traditional
VS
Machine Learning

Why Machine Learning ?

- The Slow Improvement Problem
- Quantity vs Quality
- Reason for Slow Improvement
- Examples in Computer Science & Problem solving
- Finding Suitable Problems for Individual Abilities
- Leveraging Machine Learning
- Matchmaking Problem

Our Model

Our Model

- Problem Nature
- Recommendation System Type
- Similarity Scores Metrics and Implementation
- Solo Practicing Considerations
- Data Collection Methodology
- Feature and Target Variables
- Model Selection

Demo

Questions ?

Thank you!