**CODECAMP**



Submitted by

Sanjeev Kumar 13000114086

Saquib Ul Hassan 13000114087

Shaikh Md Ashif Iqbal 13000114091

Shiv Kumar 13000114093

**Under the guidance of Ms. Mousumi Bhattacharyya**

Submitted for the partial fulfillment for the degree of Bachelor of Technology in Computer Science and Engineering

****

Techno India

EM 4/1, SaltLake, Sector – V, Kolkata – 700 091.

**ACKNOWLEDGEMENT**

We would like to express our sincere gratitude to Ms. Mousumi Bhattacharyya of the department of Computer Science and Engineering, whose role as project guide was invaluable for the project. We are extremely thankful for the keen interest she took in advising us, for the books and reference materials provided for the moral support extended to us.

Last but not the least we convey our gratitude to all the teachers for providing us the technical skill that will always remain as our asset and to all non-teaching staffs for the gracious hospitality they offered us.

Place: Techno India, Salt Lake

Date:

….…………………………………

…………………………………….

…………………………………….

…………………………………….

Department of Computer Science and Engineering

Techno India, Salt Lake

Kolkata – 700 091

West Bengal, India.

## Approval

This is to certify that the project report entitled “CODECAMP” prepared under my supervision by *(Sanjeev Kumar 13000114086), (Saquib Ul Hassan 13000114087), (Shaikh Md Ashif Iqbal 13000114091), (Shiv Kumar 13000114093),* be accepted in partial fulfillment for the degree of Bachelor of Technology in Computer Science and Engineering

It is to be understood that by this approval, the undersigned does not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn thereof, but approves the report only for the purpose for which it has been submitted.

……………………………………….. … ………………………………………

Name & Designation of Internal Guide Name & Designation of the HOD

Table of Contents

[Approval 4](#_Toc499502138)

[1. Introduction 3](#_Toc499502139)

[1.1 Briefing 3](#_Toc499502140)

[1.2 Problem Domain 3](#_Toc499502141)

[1.3 Related Study 4](#_Toc499502142)

[1.4 Glossary 4](#_Toc499502143)

[2. Problem Definition 5](#_Toc499502144)

[2.1 Scope 5](#_Toc499502145)

[2.2 Exclusions 5](#_Toc499502146)

[2.3 Assumptions 5](#_Toc499502147)

[3. Project Planning 6](#_Toc499502148)

[3.1 Software Life Cycle Model 6](#_Toc499502149)

[3.2 Scheduling 7](#_Toc499502150)

[3.3 Cost Analysis 11](#_Toc499502151)

[3.3.1 COCOMO 11](#_Toc499502152)

[3.3.2 Estimation of Effort, Development time and Productivity 11](#_Toc499502153)

[4. Requirement Analysis 12](#_Toc499502154)

[4.1 Requirement Matrix 12](#_Toc499502155)

[4.2 Requirement Elaboration 14](#_Toc499502156)

[4.2.1 JUDGE SETUP 14](#_Toc499502157)

[4.2.2 CLIENT SIDE 15](#_Toc499502158)

[4.2.3 MODERATOR 19](#_Toc499502159)

[5. Design 21](#_Toc499502160)

[5.1 Technical Environment 21](#_Toc499502161)

[5.2 Hierarchy of Modules 21](#_Toc499502162)

[5.3 Detailed Design 23](#_Toc499502163)

[5.3.1 User Management 23](#_Toc499502164)

[5.3.2 Submission Control 23](#_Toc499502165)

[5.3.3 Judge 23](#_Toc499502166)

[5.3.4 Profile Viewer and Editor 23](#_Toc499502167)

[5.3.5 Code Editor and Syntax Highlighter 23](#_Toc499502168)

[5.3.6 Sign in and Sign up page(SISU) 24](#_Toc499502169)

[5.3.7 Problem Statement Viewer 24](#_Toc499502170)

[5.4 Data Flow Diagrams 24](#_Toc499502171)

[5.4.1 Level -0 24](#_Toc499502172)

[5.4.2 Level -1 24](#_Toc499502173)

[5.4.3 Level -2 24](#_Toc499502174)

[5.5 Test Planning 24](#_Toc499502175)

[6. Implementation 30](#_Toc499502176)

[6.1 Implementation Details 30](#_Toc499502177)

[6.2 System Installation Steps 31](#_Toc499502178)

[6.3 System Usage Instructions 31](#_Toc499502179)

[7. Conclusion 32](#_Toc499502180)

[7.1 Project Benefits 32](#_Toc499502181)

[7.2 Future Scope of Improvements 32](#_Toc499502182)

[8. References 33](#_Toc499502183)

**List of Tables**

[Table 1: Glossary Table 4](#_Toc483112477)

[Table 2: Constants for Cost Analysis 11](file:///C:\Users\Puja%20Singh\Desktop\CODERUDITE\8th%20sem\Project%20Report%20Document_8th.docx#_Toc483112478)

[Table 3: Test Plan Table 3](#_Toc483112479)2

**List of Figures**

[Figure 1: Iterative Waterfall Model 6](file:///C:\Users\SANJEEV%20KUMAR\Desktop\Project%20Report%20Document_8th.docx#_Toc499427163)

[Figure 2: Gantt chart (1) 7](#_Toc499427164)

[Figure 3: Gantt chart (2) 8](#_Toc499427165)

[Figure 4: Gantt chart (3) 9](#_Toc499427166)

[Figure 5: Gantt Chart (4) 10](#_Toc499427167)

[Figure 6: Requirement Matrix (1) 12](#_Toc499427168)

[Figure 7: Requirement Matrix (2) 13](#_Toc499427169)

[Figure 8: Hierarchy Model of Client 21](file:///C:\Users\SANJEEV%20KUMAR\Desktop\Project%20Report%20Document_8th.docx#_Toc499427170)

[Figure 9 : Hierarchy Model of Moderator 2](file:///C:\Users\Puja%20Singh\Desktop\CODERUDITE\8th%20sem\Project%20Report%20Document_8th.docx#_Toc483112460)2

Figure 10: Hierarchy Model of judge API……………………………………………………………22  
Figure 11: DFD diagram (Level 0)……………………………………………………………………24  
Figure 12: DFD diagram of Judge(Level 0)…………………………………………………………..25  
Figure 13: DFD diagram (Level 1)………………………………………………………………….26  
Figure 14: DFD diagram of Judge(Level 1)…………………………………………………………..26  
Figure 13:DFD diagram(Level 2)……………………………………………………………………..27

# Introduction

## Briefing

The following project aims at developing a next generation platform which is more than a coding judge. Merging the competitive coding environment with the academics will provide a new dimension when it comes to learning and students will evolve with a balanced set of skills at the end of 4 years.

The two main pillars around which the project is going to revolve are as follows: -

Open Source: - The project is open source so that anyone can contribute to the project for sustaining the advancing technologies in future.

Modular: - The project is Modular so that any component of it can change without affecting the functionality of other.

Single Page Web Application: - The project designed this way to make it as responsive as possible.

The motivational study sources enabled us to focus on the primary problems which include building a lag free environment, irreversible loss of contest data due to crash and a perfect merger for coding and academics by means of a platform because the sources only possess maybe one or two features mentioned above. Merging them into a single platform will be a huge success.

Apart from the problems mentioned above, we will also be introducing features such as dynamic leaderboards, graphs and pie charts. We also introduce the feature of integrated multifunctional behavior or features expected from a coding platform which includes on-the-go problem statement editor and previewer and a code editor with intellisense, syntax highlighting and beautifier.

Academic curriculum will be managed in such a way that a student gets the best out of him while he writes a code for particular assignment. Needless to say, that any college or university will appreciate the product because it will cover all the aspects of erudite colosseum as well as coder circuit.

## Problem Domain

* Coding Judge.
* Single Page Web Application
* MEAN Stack , HTML5 , BOOTSTRAP , JAVASCRIPT.
* Networking.

## Related Study

There are three different sources of motivation, which underwent a thorough study, conducted by our team.

* Software Applications which are meant for serving the purpose of competitive programming environment having some simple UI features or rather static contents like percentage of successful submissions and individual progress by testing the program against given test cases. [E.g. 1.)PC2JUDGE [[**1**]](http://pc2.ecs.csus.edu/pc2desc.html)2.)DOMJUDGE[[2]](https://www.domjudge.org/development)]
* Web based competitive programming environment which requires a certain level of algorithms to be known, before one can start competing. [E.g. 1.)Hackerrank[[1]](http://www.hackerrank.com)2.)SPOJ [[2]](http://www.spoj.com)]

So after this extensive study, we came to a conclusion that we need to build a platform which will combine all the features provided by the above mentioned products and also improvise our project by introducing new features of data analysis and easy to go access at any point of time with a simple and lucid UI so that it can be easily interpreted or used by all Coding Event Hosting teams.

We made the project with the help of [MEAN](https://www.codeschool.com/mean)

## Glossary

|  |  |
| --- | --- |
| TCP | Transfer Control Protocol |
| MEAN | MongoDB , ExpressJS , Angular , NodeJS |
| HTML | Hyper Text Markup Language |
| IDE | Integrated Development Environment |
| UI | User Interface |
| CHS | Contest Hosting Server |
| SS | Submission Server |

Table 1: Glossary Table

# Problem Definition

## Scope

The project is intended to serve the purpose of a single page web application platform which will help to organize coding competitions in a lucid manner just by a few clicks with our simple and efficient UI. The user end is going to interact by means of a web based interface compatible with the desktop browsers available on the machine. The administrator end or the instructor end is also going to be web based.

## Exclusions

* We should try to avoid, hosting two moderator on the same contest because the running time for each individual submission id or judgement will be affected by one another.

## Assumptions

* We assume that we have the latest browser versions installed in all the participating as well as the client machines.
* We assume that we have the latest NodeJS version installed in the host machines.
* The machines which are meant to run the judges should have proper compilation tools to execute the submitted code and produce the outputs accordingly.
* We assume that all host machines have Linux as Operating System.

# Project Planning

## Software Life Cycle Model

The first SDLC to be developed and implemented was the classic Waterfall Model. In "The Waterfall" approach, the whole process of software development is divided into separate phases. In Waterfall model, typically, the outcome of one phase acts as the input for the next phase sequentially. In this model phases do not overlap. The next phase is started only after the defined set of goals are achieved for previous phase and it is signed off.

It does not allow for much reflection or revision. Once an application is in the testing stage, it is very difficult to go back and change something that was not well-documented or thought upon in the concept stage.

In this Project we have used a modified and more comprehensive form of the Waterfall Model called the Iterative Waterfall Model.

Iterative process starts with a simple implementation of a subset of the software requirements and iteratively enhances the evolving versions until the full system is implemented. At each iteration, design modifications are made and new functional capabilities are added. During each iteration, the development module goes through the requirements, design, implementation and testing phases. Each subsequent release of the module adds function to the previous release. Parallel development can be planned. Testing and debugging during smaller iteration is easy.

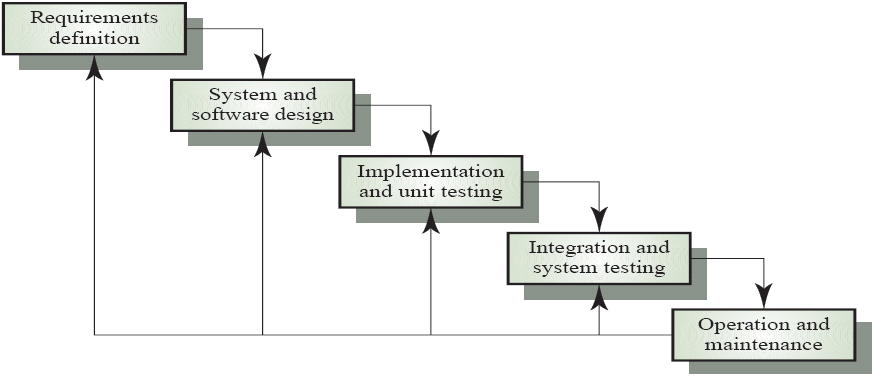


Figure 1: Iterative Waterfall Model

## Scheduling

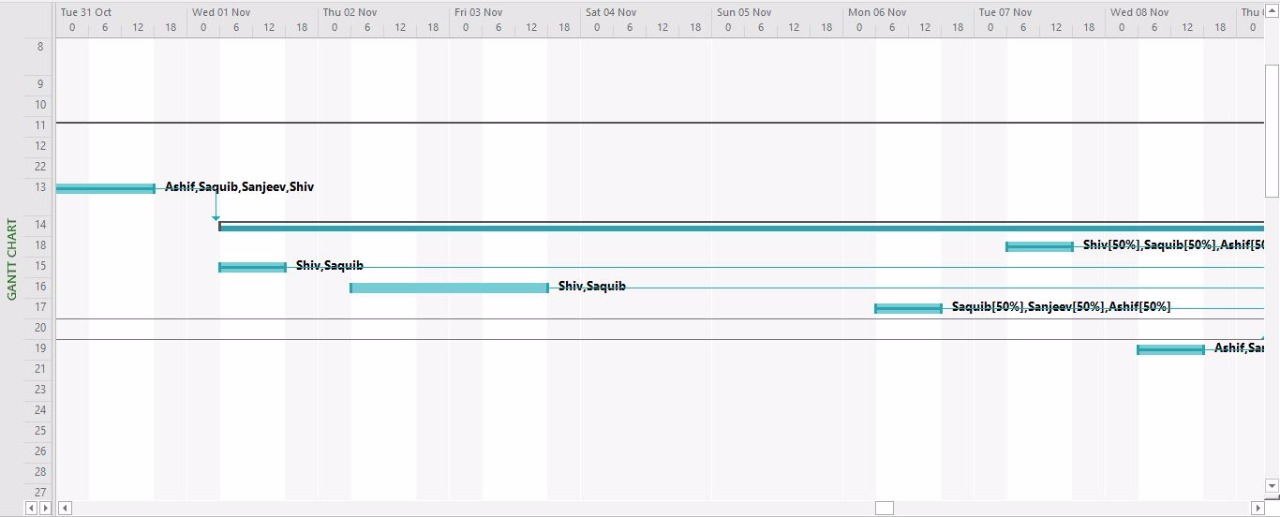


Figure 2: Gantt chart (1)

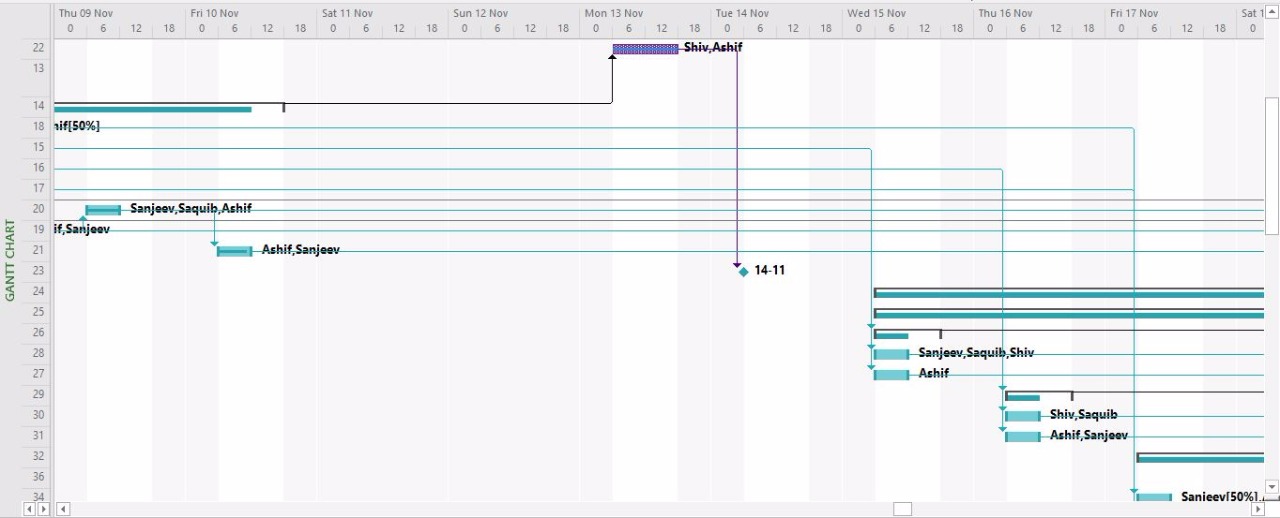


Figure 3: Gantt chart (2)

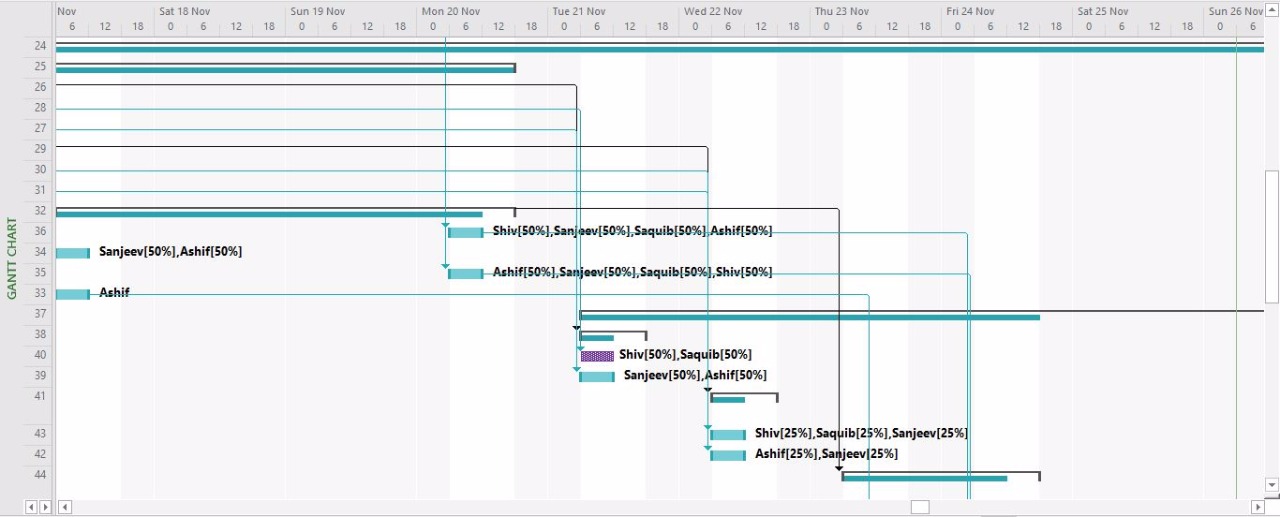


Figure 4: Gantt chart (3)

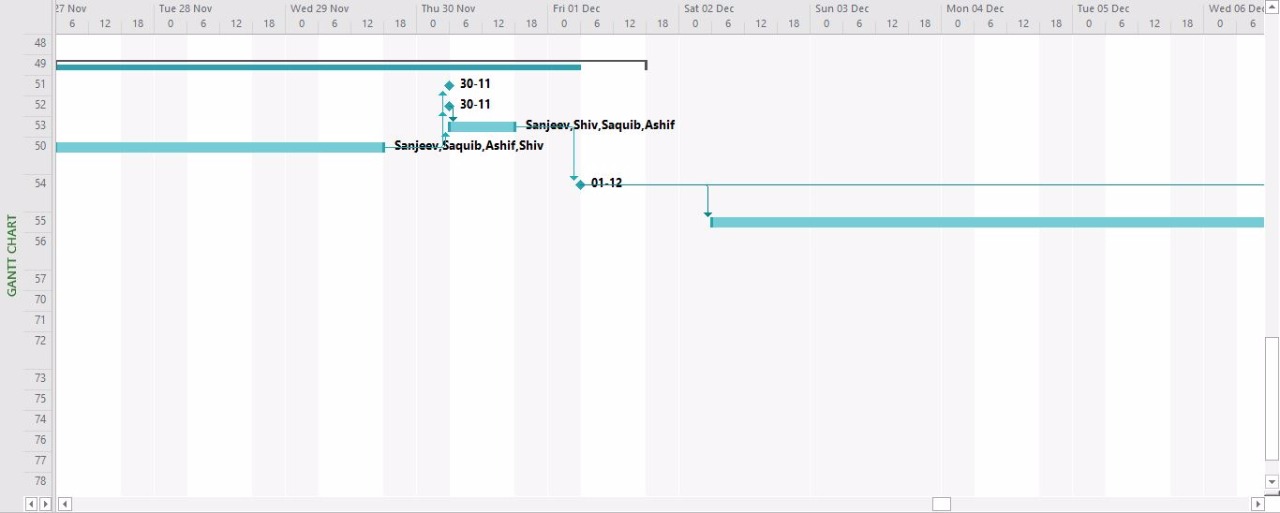


Figure 5: Gantt Chart (4)

## Cost Analysis

A project is feasible if it is possible to finish it within fixed budget constraints. Cost Analysis, sometimes also referred to as Cost/Benefit Analysis (CBA), is a systematic approach used to calculate and compare the benefits and costs of a project.

For the purpose of this project, we are using a heuristic estimation technique called Constructive Cost Estimation Model (COCOMO).

### COCOMO

The basic COCOMO gives an approximate estimate of the project parameters. This estimation model is given by the following expressions:

Where:

1. KLOC is the estimated size of the software product expressed in Kilo Lines of Code.
2. Effort Applied is the total effort required to develop the software product, expressed in person-months (PMs).
3. Development Time is the estimated time to develop the software, expressed in months.
4. are constants as given in the following table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Software Project** |  |  |  |  |
| Organic | 2.4 | 1.05 | 2.5 | 0.38 |
| Semi-detached | 3.0 | 1.12 | 2.5 | 0.35 |
| Embedded | 3.6 | 1.20 | 2.5 | 0.32 |

Table 2: Constants for Cost Analysis

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Software Project** |  |  |  |  |
| Organic | 2.4 | 1.05 | 2.5 | 0.38 |
| Semi-detached | 3.0 | 1.12 | 2.5 | 0.35 |
| Embedded | 3.6 | 1.20 | 2.5 | 0.32 |

### Estimation of Effort, Development time and Productivity

Lines of code = 3,500 (estimated)

Effort = 3.0 × (3.5) ^1.12 = 12 man-months (estimated)

Development Time = 2.5 × (12) ^0.35 = 5.9 months

Productivity = KLOC/Effort = 6/12 KLOC/MM (approx.)

# Requirement Analysis

## Requirement Matrix

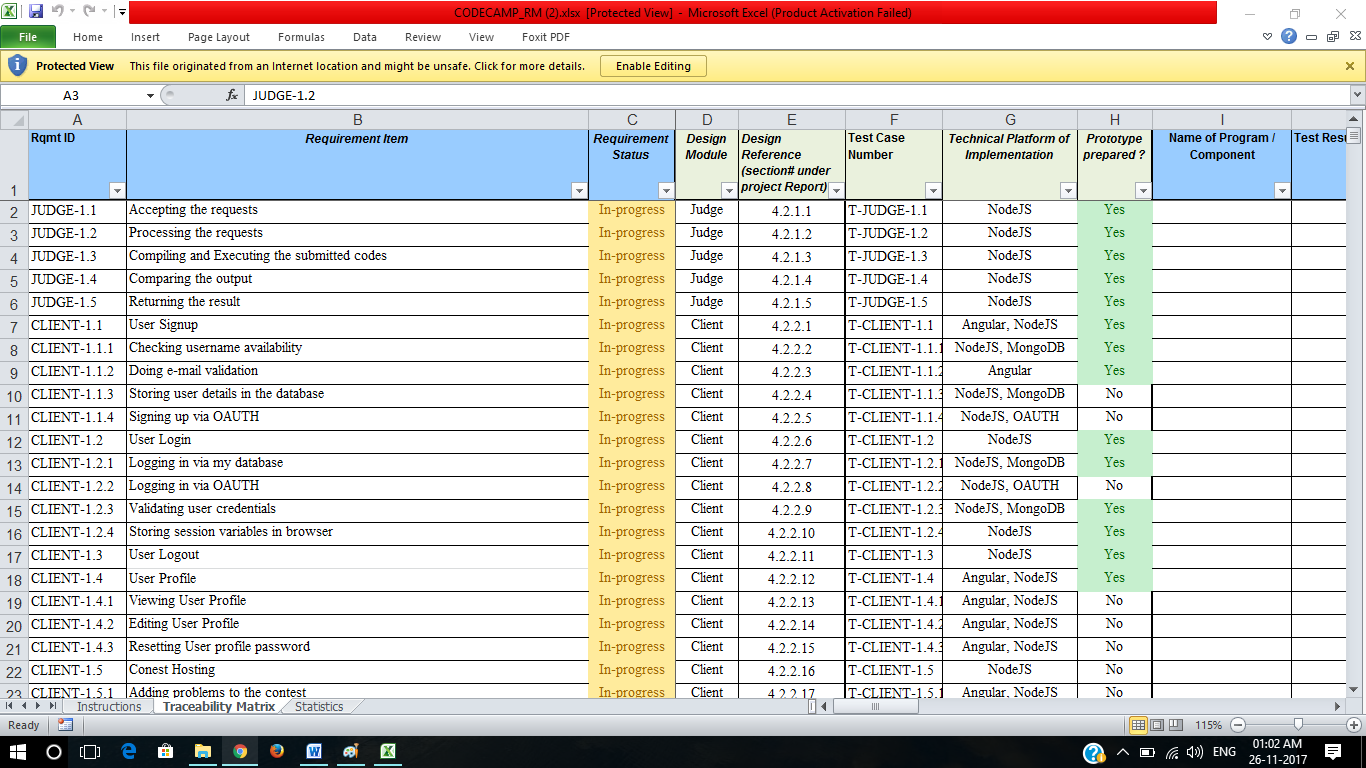


Figure 6: Requirement Matrix (1)

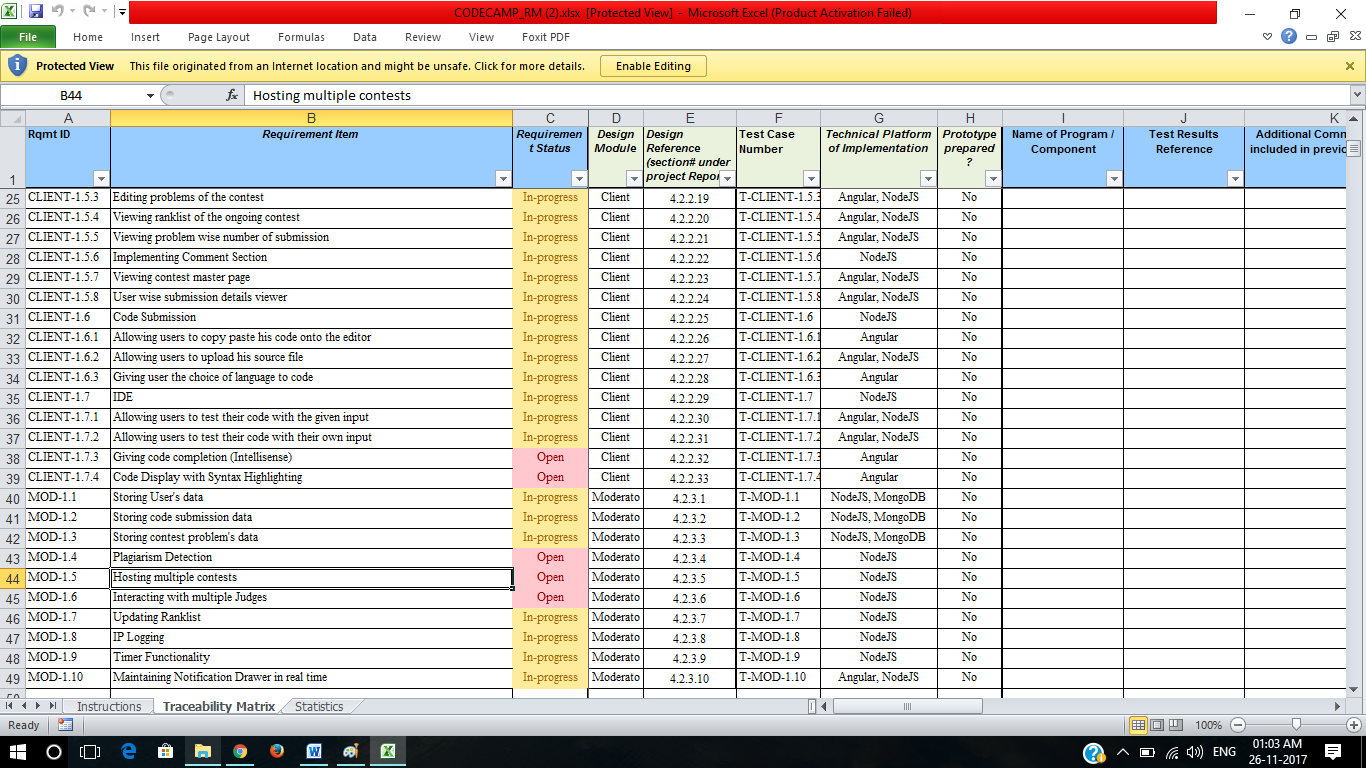


Figure 7: Requirement Matrix (2)

## Requirement Elaboration

The functional requirements of the project can be broadly summarized into two parts. The first one includes the launch of all the server programs needed to start the contest. The second part consisting of client side code which is responsible for uploading the questions, manipulating them (The admin) and submitting a solution to a problem in different languages and will be accepted from multiple client machines in the form of semi structured objects containing the submission id, the contest id and the code string submitted in which language. The details are provided below.

### JUDGE SETUP

#### Accepting the Requests

ID: JUDGE 1.1

**Description:** This module will accept the incoming request from other sources to the judge api.

#### Processing the Requests

ID: JUDGE 1.2

**Description:** This module will process and parse the request from other sources to the judge api.

#### Compiling and Executing the Submitted Codes

ID: JUDGE 1.3

**Description:** This module compiles and executes the submitted code on the judges host machine.

#### Comparing the Output

ID: JUDGE 1.4

**Description:** Judge will be compare the output of the submitted code with the given output in the accepted incoming request.

#### Returning the Judge Results

ID: JUDGE 1.5

**Description:** The judge computes the result for a corresponding submission id and returns the result accordingly.

### CLIENT SIDE

#### User SignUp

ID: CLIENT-1.1

**Description:** This Module will help client to sign up in codecamp .

#### Checking Username Availability

ID: CLIENT-1.1.1

**Description:** This module checks the availability of the username in the database of users.

#### Doing e-Mail Validation

ID: CLIENT-1.1.2

**Description:** This module checks for the validity of the entered e-mail

#### Storing user details in the Database

ID: CLIENT-1.1.3

**Description:** This will stores the Users details in the Database

#### Signing Up via OAUTH

ID: CLIENT-1.1.4

**Description:** This module will let users signup using their credentials of another account like Google, facebook , Github

#### User Login

ID: CLIENT-1.2

**Description:** This Module deals with Login Details

#### Logging In via my Database

ID: CLIENT-1.2.1

**Description:** This module will let users to signin their account by the username and password entered while signup.

#### Logging in via OAUTH

ID: CLIENT-1.2.2

**Description:** This module will let users to login their account using their credentials of another account like Google, facebook ,Github

#### Validation user Credentials

ID: CLIENT-1.2.3

**Description:** Checking whether the entries done from the client side for the signup or sign-in page are consistent with the database and the constraints properly.

#### Storing Session variables in Browser

ID: CLIENT-1.2.4

**Description:** This module will store the session variable in the Browser of the client

#### User Logout

ID: CLIENT-1.3

**Description:** This module will help to logout the user from their account.

#### User Profile

ID: CLIENT-1.4

#### Viewing User Profile

ID: CLIENT-1.4.1

**Description:** By this option the user can view his/her profile details.

#### Editing User Profile

ID: CLIENT-1.4.2

**Description:** On the go Profile Edit option with detail filling for a particular user profile.

#### Resetting User Profile Password

ID: CLIENT-1.4.3

**Description:** This will give user permission to reset their Password .

#### Contest Hosting

ID: CLIENT-1.5

**Description:** This will give option to user to host contest on the web appliaction.

#### Adding Problems to the Contest

ID: CLIENT-1.5.1

**Description:**  By this Option host user can add problems in the contest.

#### Adding test cases for respective Problems

ID: CLIENT-1.5.2

**Description:** By this Option host user can add test cases for respective problems of the contest.

#### Editing Problems in the Contest

ID: CLIENT-1.5.3

**Description:** By this Option host user can edit the problems of the contest.

#### Viewing ranklist of the Ongoing Contest

ID: CLIENT-1.5.4

**Description:** User can view the Rank List of the ongoing Contest.

#### Viewing problem wise number of submission

ID: CLIENT-1.5.5

**Description:** Users can view the problem wise submission to give them an understanding of the complexity of the problem in reference to the others

#### Implementing Comment Section

ID: CLIENT-1.5.6

**Description:** This module will implement Comment section for each problem in each contest for queries which can be resolved by other users or the admin himself. The comments will be approved by the admin so that there is moderation about what a user can post in the comment section.

#### Viewing Contest Master Page

ID: CLIENT-1.5.7

**Description:** A basic prototype page whose outline and theme will be followed in every content display. Every other page will be displayed by overwriting some of the contents of Master Page.

#### User wise submission details viewer

ID: CLIENT-1.5.7

**Description:** This will help the host to view the total submission details of each user

#### Code Submission

ID: CLIENT-1.5.7

**Description:** This module facilitates the user to submit the code.

#### Allowing users to copy paste his code onto the editor

ID: CLIENT-1.5.7

**Description:** It allows the user to copy and paste

#### Allowing users to upload his source file

ID: CLIENT-1.5.7

**Description:** This module allows the front end to accept the source code as a file input.

#### Giving user the choice of language to code

ID: CLIENT-1.5.7

**Description:** This module allows the user to upload his source code in different allowed languages.

#### IDE

ID: CLIENT-1.5.7

**Description:** This module allows us to create an Integrated Development Environment for the user to run his code and also write it.

#### Allowing users to test their code with the given input

ID: CLIENT-1.5.7

**Description:** This allows users to test their code in the IDE with the given test case in the questions.

#### Allowing users to test their code with their own input

ID: CLIENT-1.5.7

**Description:** This allows users to test their code in the IDE with their own test case in the questions.

#### Giving code completion (Intellisense)

ID: CLIENT-1.5.7

**Description:** Using Intellisense open-source technology to provide with code completion.

#### Code Display with Syntax Highlighting

ID: CLIENT-1.5.7

**Description:** Using limiting open source technology to provide syntax highlighting.

### MODERATOR

#### Storing User’s Data

ID: MOD 1.1

**Description:** This will store the User’s Data in the [nosql](https://www.mongodb.com/nosql-explained) database.

#### Storing Code Submission Data

ID: MOD 1.2

**Description:** This will store the submitted codes of the users so that they can revert back to any one of their submitted code and make changes if needed to perfect their code.

#### Storing contest Problem’s Data

ID: MOD 1.3

**Description:** This will store the problem data for the contests to be held so that in the event of a crash of the moderator server, we can restart it and the data will still be there.

#### Plagiarism Detection

ID: MOD 1.4

**Description:** For online contests we need to implement an algorithm which will prevent multiple users to submit the same code and pass the testing phase. This will be implemented by the plagiarism tester module.

#### Hosting Multiple Contests

ID: MOD 1.5

**Description:** We are planning to host multiple contests side by side and as our judge is an API, we can just use it for the multiple contests.

#### Interaction with Multiple Judges

ID: MOD 1.6

**Description:** We are also planning to incorporate multiple judges so that the execution of the many thousands codes are distributed among the different judges for a faster system.

#### Updating Ranklist

ID: MOD 1.7

**Description:** A ranklist will also be maintained and updated in real-time in accordance to the ACM style of ranklist. The user can view the ranklist at anytime

#### IP Logging

ID: MOD 1.8

**Description:** A user can log in from only one machine and not multiple machines. This will prevent fraud and prevent multiple users to use the same id and crack the contest in a distributed manner.

#### Timer Functionality

ID: MOD 1.9

**Description:** A timer will be present in the backend to show the amount of time left in the contest.

#### Maintaining Notification Drawer RealTime

ID: MOD 1.10

**Description:** There will be notification functionality in case there is any notification that needs to be provided to the users participating in the contests.

# Design

## Technical Environment

The interface development involved use of the following software and programming languages mentioned below

1. VSCode, for the development of this web application.
2. HTML5,CSS 3, Angular and TypeScript for the client side.
3. NodeJs and its libraries are used for building server side.
4. BootStrap and D3js for adding graphical assistance features in the platform.
5. MongoDB for database management system.

Thus the project can be summarized as a mixture of JavaScript and its Frameworks and APIs used as tools to build this whole interface or platform.

## Hierarchy of Modules

Figure 8: Hierarchy Model of Client

Figure 9 : Hierarchy Model of Moderator

Figure 10 : Hierarchy Model of JUDGE API

## Detailed Design

The detailed design for the project consists of the following diagrams that determine the design analysis for each of the modules in the requirement section. To summarize the whole design, there are specifically 11 modules to be described as per the requirement matrix. Let us have a in-depth look in the modules

### User Management

User Management module is meant for registration and login procedures of the user base and is responsible for entry of data in the User database. The client side of the same module contains form validation for the login and signup page where it is checked whether the user already exists or the it satisfies the constraints for being an appropriate entry. It also involves storing of session ids such that it involves smooth login and site visiting experience without multiple time login. Obviously the span of these cookies will be limited to certain period of time, supposedly 4-6 months.

### Submission Control

Submission Server Format uses JSON file format which is semi structured object type and is meant for storing key value pair of submission id corresponding to a particular user and obviously shard database acting as a repository of submission with appropriate time stamps and language stamps so that it can be extracted from the waiting queue accordingly.

### Judge

The judge here is an API to which we will be posting the incoming codes from the client and the judge will then parse the incoming request to its required format. After that, it will compile the code if necessary and then execute it with input test cases to produce the respective output. It then checks the produced output with that of the given expected output. It then returns the result in response to the incoming request.

### Profile Viewer and Editor

A client side module used for providing a first person view to the profile and obviously an editor to fill up the personal details corresponding to the user profile along with interests so a user can constantly focus to learn what he wants to after solving one problem.

### Code Editor and Syntax Highlighter

Online Code Editor with proper syntax highlights, Intellisense and code completion features with proper indentation measures so that a prettified code is available for reading and debugging. This will be achieved using ACES application programming interface.

### Sign in and Sign up page(SISU)

Sign up and Sign in pages will obviously be derived from the master pages along with proper validations, duplicate entry checking and appropriate transfer or entry of data in the user database as the normal sign up and sign in page procedure follows.

### Problem Statement Viewer

Problem Statement Viewer module will work with JSON string received from PPMGMT module and put it in presentable format overwriting on the master page format and help to view the problem statement accordingly along with submission statistics display.

## Data Flow Diagrams

### Level -0

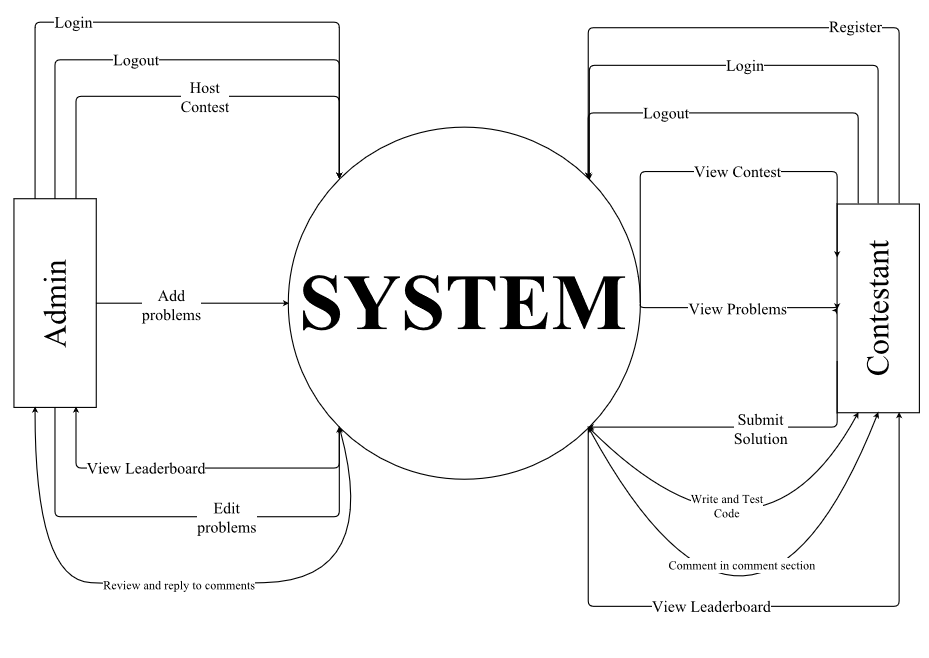


Figure 11 : DFD diagram (level 0)

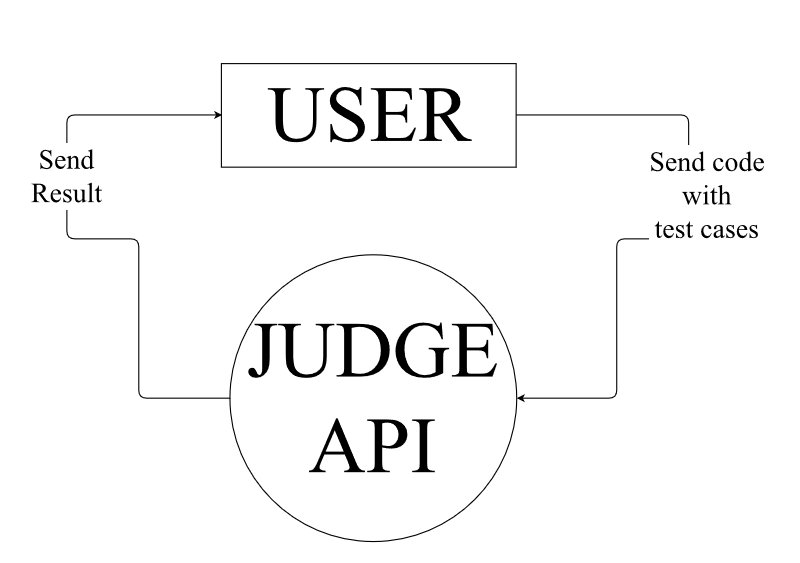


Figure 12 : DFD diagram of Judge(level 0)

### Level -1

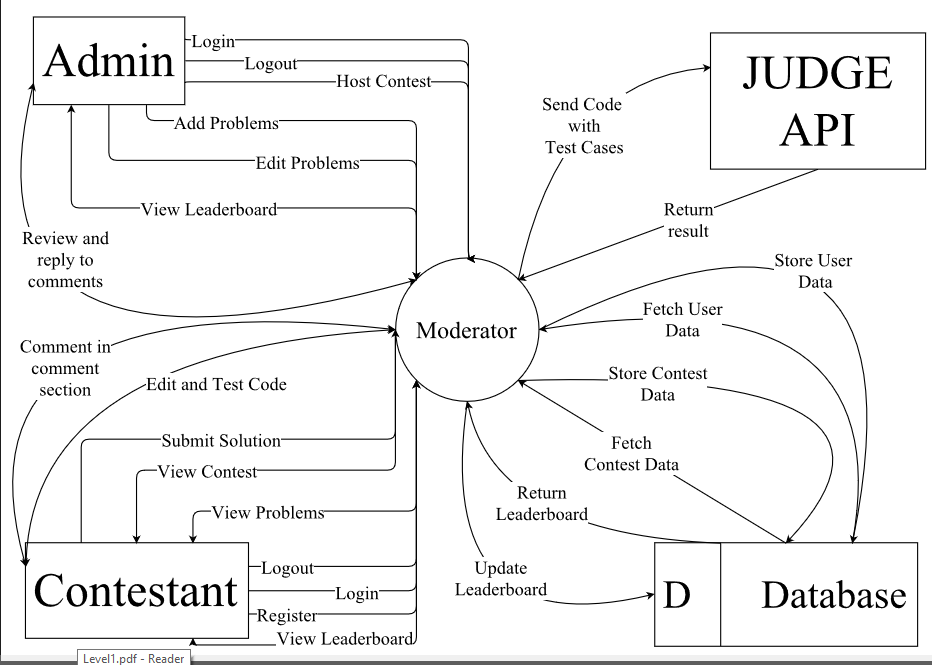


Figure 13 : DFD diagram (level 1)

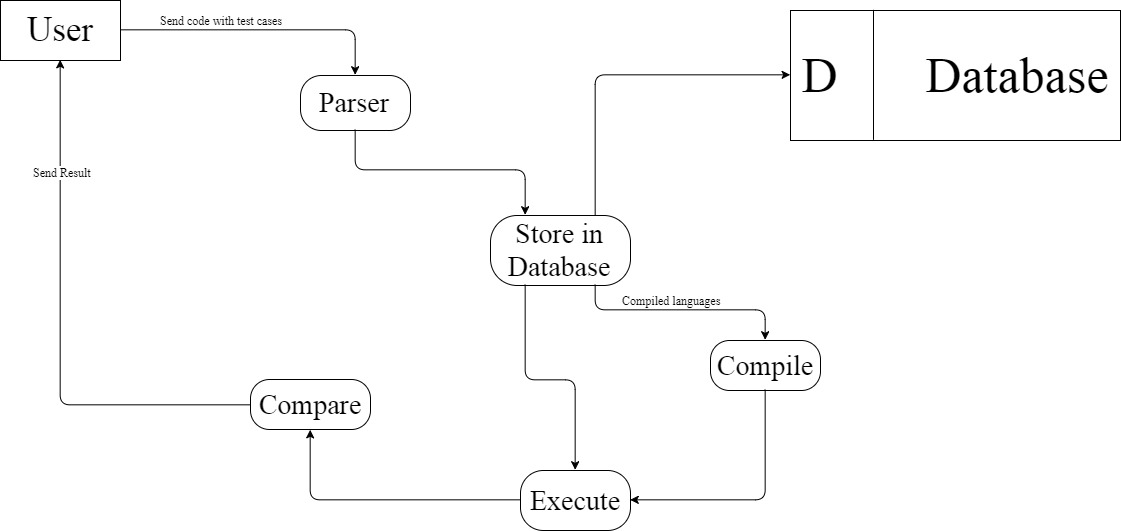


Figure 14 : DFD diagram of Judge (level 1)

### Level -2

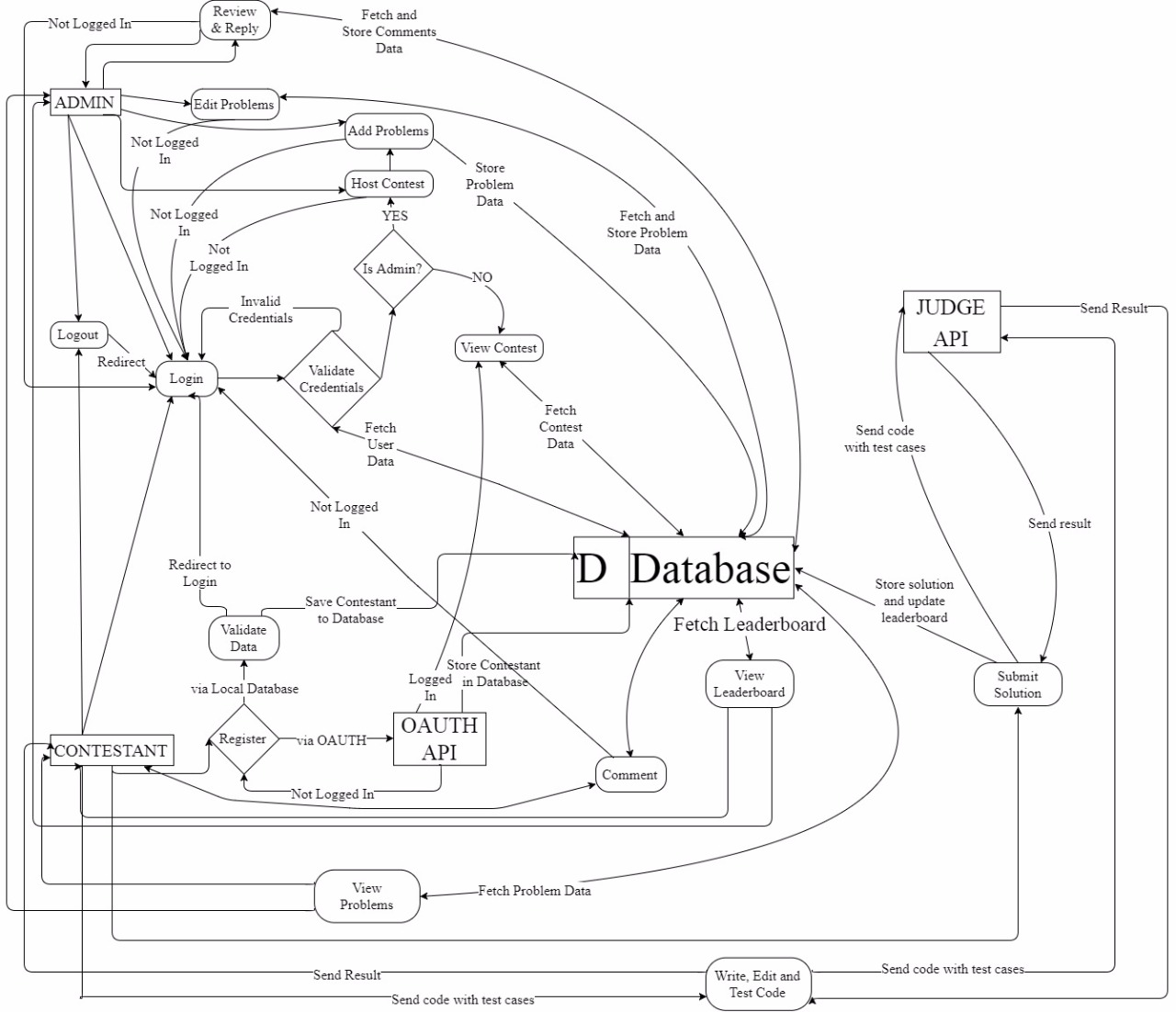


Figure 15 : DFD diagram (level 2)

## Test Planning

| **Test Case Number** | **Test Case** | **Expected Output** | **Result** | **Remarks** | **Result Case Number** |
| --- | --- | --- | --- | --- | --- |
| T-JUDGE -1.1 | Accepting the requests | Request Accepted |  |  |  |
| T-JUDGE -1.2 | Processing the requests | Request Processed |  |  |  |
| T-JUDGE -1.3 | Compiling and executing the submitted codes | Compilation Successful and Output sent |  |  |  |
| T-JUDGE -1.4 | Comparing the output | Output is compared with the given output |  |  |  |
| T-JUDGE -1.5 | Uploading Judge result | Result uploaded to the Judge |  |  |  |
| T-CLIENT -1.1 | User SignUp |  |  |  |  |
| T-CLIENT-1.1.1 | Checking username availability | Username Avialable / Not available |  |  |  |
| T-CLIENT -1.1.2 | Doing e-mail validation | Email validated |  |  |  |
| T-CLIENT -1.1.3 | Storing user details in the database | Details Is Stored in DataBase |  |  |  |
| T-CLIENT -1.1.4 | Signing up via OAUTH | Sign in successful |  |  |  |
| T-CLIENT -1.2 | User Login |  |  |  |  |
| T-CLIENT -1.2.1 | Logging in via my database | UserName and password valid . User Logged in |  |  |  |
| T-CLIENT -1.2.2 | Logging in via OAUTH | User Verified via oauth |  |  |  |
| T-CLIENT -1.2.3 | Validating user credential | User Credetial Valid |  |  |  |
| T-CLIENT -1.2.4 | Storing session variables in browser | Session Variables stored |  |  |  |
| T-CLIENT -1.3 | User Logout | User got Logout |  |  |  |
| T-CLIENT -1.4 | User Profile | Open the page in html containing user details |  |  |  |
| T-CLIENT -1.4.1 | Viewing User Profile | User Profile Opened in |  |  |  |
| T-CLIENT -1.4.2 | Editing User Profile | Access granted to eidt profile |  |  |  |
| T-CLIENT -1.4.3 | Resetting User Profile password | Password Successfully resetted |  |  |  |
| T-CLIENT -1.5 | Contest Hosting | Contest Hosted on the Codecamp |  |  |  |
| T-CLIENT -1.5.1 | Adding problems to the contest | Problem Added in the contest |  |  |  |
| T-CLIENT -1.5.2 | Adding test case for respective problems | Test Case Added for respective problems |  |  |  |
| T-CLIENT -1.5.3 | Editing Problem in the contest | Access granted to edit the problem |  |  |  |
| T-CLIENT -1.5.4 | Viewing rank list of the on going contest | Ranklist published in the page |  |  |  |
| T-CLIENT -1.5.5 | Viewing problem wise nuber of submissions | Prined number of submission |  |  |  |
| T-CLIENT -1.5.6 | Implementing Comment Section | Comment Section is present having comments |  |  |  |
| T-CLIENT -1.5.7 | Viewing Contest master page | Contest master page Created |  |  |  |
| T-CLIENT -1.5.8 | User wise submission details viewer | Able to view submission details of each viewer |  |  |  |
| T-CLIENT -1.6 | Code submission | Code Submitted |  |  |  |
| T-CLIENT -1.6.1 | Allow user to copy paste his code onto the editor | Copy paste Worked on the editor |  |  |  |
| T-CLIENT -1.6.2 | Allowing user to upload his source file | Source File Uploaded |  |  |  |
| T-CLIENT -1.6.3 | Giving user the choice of language to code | Choices present for different languages |  |  |  |
| T-CLIENT -1.7 | IDE | IDE Working Correctly |  |  |  |
| T-CLIENT -1.7.1 | Allowing users to test their code with given test case | User able to test their code with given test case |  |  |  |
| T-CLIENT -1.7.2 | Allowing users to test their code with their own input | User able to test their code with own test case |  |  |  |
| T-CLIENT -1.7.3 | Giving code completion(Intellisense) | Intellisense working |  |  |  |
| T-CLIENT -1.7.4 | Code Display with Syntax Highlighting | Code Displayed |  |  |  |
| T-MOD -1.1 | Storing User’s Data | User Data Stored |  |  |  |
| T-MOD -1.2 | Storing Code Submission Data | Code Submisson Data Stored in the database |  |  |  |
| T-MOD -1.3 | Storing Contest Problem’s Data | Problem’s Data Stored in the database |  |  |  |
| T-MOD -1.4 | Plagiarism Detection | Plagiarism is working |  |  |  |
| T-MOD -1.5 | Hosting multiple Contests | Multiple Contest Hosted |  |  |  |
| T-MOD -1.6 | Interacting with Multiple Judges | Multiple judges are interacting |  |  |  |
| T-MOD -1.7 | Updating Rank list | Rank List Updated |  |  |  |
| T-MOD -1.8 | IP Logging | IP Logging working properly |  |  |  |
| T-MOD -1.9 | Timer functionality | Timer is working properly |  |  |  |
| T-MOD -1.10 | Maintaining Notification Drawer Up to date | Notification Drawer maintained |  |  |  |

Table 3: Test Plan Table

# Implementation

## Implementation Details

The languages and libraries used were:

* **HTML, CSS , Angular and JavaScript** for designing the Front End or client side
* **NodeJS and their Frameworks** for designing the backend or the server side
* **MongoDB** for storing submissions, user information and problems information.
* **Compilers and Environments** for required languages to execute the submissions on judge.
* **Bootstrap D3js** for graphical and tabular representation of data.

The JUDGE API is explained as follows:

* The incoming request will be in the form of JSON and will contain the source code encoded in base64 along with some metadata.
* The metadata contains the details of the test cases, the language used in the source code and also the expected output.
* It then parses the request into the required formats.
* It the compiles the source code if necessary according to the language of the submitted code.
* It then executes the code with given input test cases and produce the output files
* It then compares the produced output files with that of expected output files that is present with the incoming request.
* It then responds back with the result in the form of JSON

The databases used are as follows:

* Submission database containing all submissions.
* Problems database to store data of all problems along with the input test cases and the expected the output.
* Users database to store information of all registered users.

The front-end contained

* Contest Page showing all problems.
* Login and register page for users.
* Profile page for users.
* Inbuilt code editor with Intellisense and syntax highlighting.
* Problem page showing description of problem and allowing user to submit the problem.
* Problem page also shows a comment section for doubt clearance.
* Ranklist page

The control is as below:

* The moderator and judge is setup.
* The admin logs in from the front end and creates contest.
* He then adds the problem definition along with the test cases.
* He then starts the contest.
* The participants register themselves and then logs in.
* They then solve the problems and submit the code.
* The frontend sends the code to the moderator.
* The moderator saves the code to the database and then sends a request to the JUDGE with the code and the test cases and output files.
* The JUDGE then parses the request, compiles the code if needed and then executes it with the given input test cases.
* It then compares the respective outputs with that of the produced outputs.
* It then responds back to the moderator with the result.
* The server saves the status, updates the ranklist and sends the status back to the client.

Apart from that, we have session ids on client side so that the client does not need to login multiple times, if he sits on the same desktop with the browser being used is the same.

## System Installation Steps

* Insert the code and run it on NodeJS for both the moderator server and judge server.
* The client can now access it on the network from their machines.

## System Usage Instructions

* Type the http address on to the address bar to launch the client side interface.
* Now the client can use the application from their own web browser.

# Conclusion

## Project Benefits

Our product might be the one of those, which are already available in the market but there are enormous number of features which will prove that we are way better than what exists in the market right now. Some of them are listed below: -

1. Our first oath and the most important feature of this product, the competitions conducted will be 99% lag free.
2. The environment will help the coders to realize their status and position without disturbing them amidst of the competition
3. The complete analysed data set will be presented with the help of dynamic leader boards, graphs and pie charts which will make us the first online coding judge to do so.
4. Integrated Code Editor and Beautifier along with intellisense.
5. Partial Code Modification assignments which will help the students to learn only that part of the code, according to the demand of the assignments.
6. Suggestions on what problems are needed to be solved next after we are done with a problem.
7. Administrator or Problem end software is portable, cross platform and “easy to use and modify” environment.
8. Automatic crash (application crash) recovery.

In terms of Market performance, demand for such a product will be huge which can meet the demands of erudite colosseum as well as the coder circuit. Colleges and Universities will be keen to try out this stuff and make the most out of it.

## Future Scope of Improvements

* Chat Bots to assist the clients to explore and exploit the website to the fullest
* Chat options to communicate with peers
* Introducing the IP Logging Feature
* Merging the profile with google accounts so that the code save feature can also be brought into existence. But up till, this point of time, the whole process works as mentioned above.
* Video Tutorials from professor to learn and explore more.

# References

[1] "https://university.mongodb.com/courses/MongoDB/M101JS/2017\_August/courseware" [Online].

[2] "https://nodejs.org/api/index.html" [Online].

[3] "https://www.udemy.com/the-complete-guide-to-angular-2/" [Online].

[4] "https://getbootstrap.com/docs/4.0/getting-started/introduction/" [Online].

[5] "http://expressjs.com/api.html" [Online].

[6] "https://docs.npmjs.com/" [Online].

[7] "http://mongoosejs.com/docs/guide.html" [Online].

[8] "https://github.com/d3/d3/wiki" [Online].