**CSN-254**

**SOFTWARE ENGINEERING**

**Project: Recommendation System for CodeForces**

**Group no. 15**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** | **NAME** | **ENROLLMENT NO.** | **EMAIL** |
| 1 | BALNE NITEESHA | 18114017 | bniteesha@cs.iitr.ac.in |
| 2 | DIKSHA | 18114019 | diksha@cs.iitr.ac.in  9929155847 |
| 3 | KARAN SINGH | 18114035 | ksingh1@cs.iitr.ac.in |
| 4 | KHUSHI | 18114040 | khushi@cs.iitr.ac.in |
| 5 | MANASVI | 18114046 | mpatidar@cs.iitr.ac.in |

**Proposal**

**Background**

Competitive Coding can be a bit infuriating at times, imagine yourself in the shoes of a beginner you would either get bored by those easy problems if you sort them by descending number of solves or get discouraged by jumping straight into the difficult sections. Finding the right problem-set was always a hassle for me. Adding to these problems, CodeForces the most popular platform provides a very minimalistic GUI and a difficulty tag that makes little to no sense. This is a problem faced by all students preparing for a Computer Science oriented Job interview, irrespective of their college, branch or year.

This means that every year, thousands of students start Competitive Coding but only a select few excel at it, this is what we are aiming to cure. The lack of a proper recommendation system can demotivate a lot of these young and fragile minds, but with a proper guide, the students would really be empowered to explore this field to its true depth.

**Objectives**

The system will deliver the following:-

1. Requests information from users to **optimize their suggestions**. This information includes:-

* User’s CodeForces username.
* User’s Friend’s CodeForces usernames.

1. Accesses problems solved and other details of the user’s friends and the seniors using CodeForces’s API.
2. Display the accessed information in a raw yet well-formatted form.
3. Process the accessed information to generate the aforementioned lists.
4. Display these lists in an exhaustive manner.
5. Requesting user response for the problems they solved recently, on random occasions.

**Impact**

This application will not only act as a guide to CodeForces it will also compensate for the weak GUI they provide by enabling users to read the problems in-app. The user will use this application alongside their regular coding environment, the problems suggested by the application can be accessed swiftly on the browser by typing in the 2-5 character problem code and hitting enter. The user can also search for other problems while they are sitting with fingers crossed on the submissions page waiting for the “Accepted” to pop-up.

**Justification**

The project involves a fair share of various different technologies that are currently being used in the market. It involves a Machine Learning based algorithm for generating the recommendations and a flutter based front end for mobile devices(both android and iOS). Both of these are fairly new technologies for our team and the project will require a good understanding of these technologies for its successful completion. They also have good documentation and community that will be of great help while the development of the project. Hence in our opinion, the project is neither too easy nor too ambitious for a group of 5.

**Feasibility Report**

**Customer:-**

As a customer, I need an android app. The main purpose of this app will be to help me track my progress on a competitive platform like codeforces. This app can be implemented using different features

1) The app will show my already solved problems.

2)The app will recommend a similar problem that was solved by me and my friends. Recommendation type will be

1.user base

2.item base

3.hybrid.

I need a flutter for frontend development.

**Visibility Plan:-**

The team intends to be in regular touch with the customer through time to time video conferences, interview sessions and be updated about the changing demands of the customer. Within the team, the team can communicate via effective group discussions and survey forms, to get the better of the ideas among the team, thus maintaining the alignment of the team too.

The communication might happen through a chatbot such as slack with two channels, one for communication of project head with other team members, and other for communication of project head with the client, discussing the continuous progress taking place in every phase while project development.

**Task and Deliverables:-**

A complementary application to CodeForces that eases beginners way into the world of competitive coding. The application provides suggestions that include three lists:-

* Made by User-Based Collaborative Filtering
* Made by Item-Based Collaborative Filtering
* Made by a dynamic hybrid of Item and User-Based Collaborative Filtering

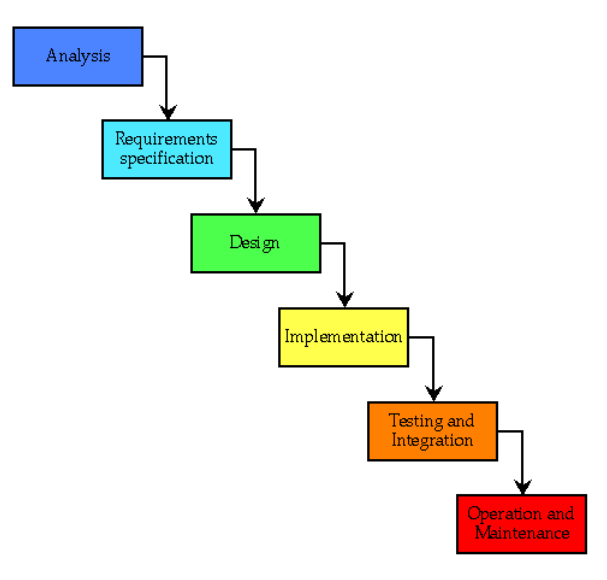
Deliverables:-

* An android application released on PlayStore and Github.
* An iOS application released on Github.
* Backend Deployed on Firebase Cloud Functions.

**The Life Cycle Model:**

Depending on the type of our project, we have decided to go with the **Classical Waterfall Model**.

The reason for the choice of this model is the clarity of the requirements and feasibility of our chosen project. We have a clear picture of what the customer wants, and how to deploy those features. The technology we intend to use is also quite well understood.



**Outline Plan:-**

**Milestones to be achieved and why should the customer use it?**

* **Find support and motivation**

Some people do coding for fun, it’s their passion but for others, they want to do it as they want to pursue a career in related fields. If you fall in the latter category, being surrounded by people who are on the same boat provides you with the incentive to make it a daily habit. Seeing your buddies around indulging in it can spark an interest in you and help to excel in the field.

* **Improve skills**

Looking at the problems your peers have solved, the response they have provided to certain problems, their recommendations whether certain things are useful or not will provide you with a better understanding of the kind of problems you need to do. Thus you can gradually proceed towards your goals.

* **Help to make proper decisions**

Once you are good at your skills or you have done some sort of problems and you are in a position to present your view about it then you can rate and tag those problems as you find suitable so that others can see it and do if they find useful.

* **Learn from each other**

Give and take is the principle of life. Help others by providing reviews about what you have done and looking at other reviews you can decide for yourself. Thus helping each other will improve everyone’s productivity.

**Principle activities:-**

1. Discussion about the various needs of the customer and the functionalities to be included in the application.
2. Explore the technology and platforms to be used to provide the customer with good user experience.
3. Designing application based on our proposal

**Risk Analysis:-**

It’s important to create a recommendation system that will scale with the amount of data you have. If it’s built for a limited dataset and that the dataset grows, computation costs grow exponentially, and the system will be unable to handle the amount of data. To avoid having to rebuild your recommendation system, later on, we might need to ensure from the beginning it is built to scale to expected data volumes.

It’s also possible that after spending time, energy, and resources on building a recommendation system (and even after having enough data and good initial results) that the recommendation system only makes very obvious recommendations. The crux of avoiding this pitfall really harkens back to the first of the seven steps: understand the business need. If there isn’t enough of a content long-tail or no need for the system, perhaps you need to reconsider the need to build a recommendation system in the first place.

Finally, people’s tastes don’t stay static over time, and if a recommendation system isn’t built to consider this fact, it may never be as accurate as it could be. Similarly, there is a risk of building a recommendation system that doesn’t get better over time. As users continue to consume content and more data is available, our recommendation system needs to learn more about users and adapt to their tastes. A recommendation system not agile enough to continue to adapt can quickly become obsolete and won’t serve its purpose.

**Probable Technical Requirements:-**

* **Flutter:-** The front end of our application will be made using Google’s Flutter framework, which will also involve some level of native coding in Java and Swift (for android and iOS).
* **TypeScript/JavaScript:**- The backend will be written using TypeScript which will then be transpiled into JavaScript and deployed on the Firebase Cloud Function.
* **Dart**:- This is the language used by Flutter and is required for frontend.
* **Machine Learning Algorithms:-** There exist many recommendations for algorithms, but the best results are obtained where we use Machine Learning to generate them.
* **ML Libraries(TensorFlow):-** A good understanding of these would allow us to write efficient and effective code.