Dissertation on

"Intelligent Teaching Assistant for Programming Concepts"

Submitted in partial fulfilment of the requirements for the award of degree of

Bachelor of Technology
in
Computer Science & Engineering

Submitted by:

Michelle Maria Roy	**01FB15ECS171**
Milan J S	**01FB15ECS172**

Under the guidance of

Internal Guide
Prof. Channa Bankapur
Assistant Professor,
PES University

External Guide
Dr. Viraj Kumar
Visiting Professor,
Indian Institute of Science

January – May 2019

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
FACULTY OF ENGINEERING
PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)

100ft Ring Road, Bengaluru – 560 085, Karnataka, India

PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)
100ft Ring Road, Bengaluru – 560 085, Karnataka, India

FACULTY OF ENGINEERING

CERTIFICATE

This is to certify that the dissertation entitled

'Intelligent Teaching Assistant for Programming Concepts'

is a bonafide work carried out by

Michelle Maria Roy	**01FB15ECS171**
Milan J S	**01FB15ECS172**

In partial fulfilment for the completion of eighth semester project work in the Program of Study Bachelor of Technology in Computer Science and Engineering under rules and regulations of PES University, Bengaluru during the period Jan. 2019 – May. 2019. It is certified that all corrections / suggestions indicated for internal assessment have been incorporated in the report. The dissertation has been approved as it satisfies the 8th semester academic requirements in respect of project work.

Signature Prof. Channa Bankapur Designation	Signature Dr. Shylaja S S Chairperson	Signature Dr. B K Keshavan Dean of Faculty

External Viva

Name of the Examiners	**Signature with Date**
1. _____	_____
2. _____	_____

DECLARATION

We hereby declare that the project entitled **"Intelligent Teaching Assistant For Programming Concepts"** has been carried out by us under the guidance of Prof. Channa Bankapur and Dr. Viraj Kumar submitted in partial fulfilment of the course requirements for the award of degree of **Bachelor of Technology** in **Computer Science and Engineering** of **PES University, Bengaluru** during the academic semester January – May 2019. The matter embodied in this report has not been submitted to any other university or institution for the award of any degree.

01FB15ECS171 Michelle Maria Roy

01FB15ECS172 Milan J S

ACKNOWLEDGEMENT

We take this opportunity to thank the people who have been invaluable to us during the course of this project

We express our most sincere gratitude to Dr. Viraj Kumar and Prof. Channa Bankapur, without whose guidance, advice and support this project would not have been possible. We thank our guides for the timely advice, shaping the path of the project, and defining the concept in a manner that highlights and expresses its utility.

We thank the project coordinators, Prof. Preet Kanwal and Prof. Sangeeta V I for the clear and precise manner in which the project was coordinated. We thank you for the excellent organization, the schedule, and the clear understanding of the expectations from the project.

We would like to thank the Chairperson of the department of CSE, Dr. Shylaja SS, for providing us with the amazing opportunity to undertake a project in college, and providing us with the resources to complete the project.

We would also like to thank the Dean of Faculty, Dr. B K Keshavan, for his help during the project

We extend our heartfelt gratitude to the Vice Chancellor, PES University, Dr. KNB Murthy, the Chancellor and Chairman of PES University, Prof. M R Doreswamy, and the Pro Chancellor and CEO of PES University, Prof. D Jawahar for providing the opportunities and resources during the course of our study.

This project could not have been completed without the outstanding and unconditional support of our parents, who have helped us in every way possible to complete this project. We express our deepest gratitude for all the help provided by them.

We also thank our friends for their keen insights, friendly advice, and moral support through the course of the project

ABSTRACT

With the growing interest in Computer Programming in today's world, there has arisen a need for an effective solution for teachers to be able to effectively teach programming concepts in an intuitive manner. Concerns have been raised about the quality of examinations that predominantly have questions requiring a simple memory recall, claiming that such questions do not ensure deep, meaningful learning. This project aims to solve the non-trivial problem faced by teachers to create, and evaluate questions on codes that require logical thinking and understanding. This project focuses on the task of creating Multiple Choice Questions, in the domain of Introductory Programming, a course taken by lakhs of University level students. The project serves as a tool for teachers to leverage intelligent algorithms, which generate tags for code snippets to facilitate search, recommend the appropriate code snippets based on the user's interests, predict the likely perception of difficulty of a code to a user, suggest comments for un-commented functions in the code, and most importantly, generate multiple choice questions, with high quality distractors based on the functionality of the code.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER - 1

INTRODUCTION

Computer Science Engineering has become an integral part of several industries today. It has been stated that programming skills will soon be an essential basic skill to be acquired, just as much as writing and reading. Due to the rising popularity of computer science and the ubiquity of its applications, there has been a steep increase in demand for skilled labour in the field. The abundance of job opportunities has encouraged a large number of the students to take up courses in computer programming at the University level. It has been estimated by a study by in the USA that the size of the workforce employed in jobs involving Computer Science is 1.97 million, growing at a rate of 6.33% annually. The same study shows that the number of degrees awarded in the field were 35,342 in the year 2016, with a staggering year-on-year growth of 24.5%[1]. These statistics serve as an insight into the sheer popularity and ubiquity of the field of Computer Science in today's world. The meteoric rise in popularity has been made possible by the large strides in research and development of computer based technology, which in turn is a result of the effort of a highly skilled workforce that has worked tirelessly since the advent of computers.

The field of computer science is particularly popular in India, with a substantial part of the educated population opting to pursue courses in this field. In the year 2017, there were an estimated quarter million Indians in the workforce employed in Computer Science related occupations. This is the largest population of workers from a single country in the global workforce for Computer Science. This data underscores the popularity of this field in the country, and hence the need to have skilled workers graduate year after year.

While it has been established that the workforce, both nationally and globally is very large, there has been a slew of concerns about the quality of the workforce. A study conducted by an assessment company called Aspiring Minds[2] created a stir with their claim that 95% software engineers were

not fit for their job. The Aspiring Minds study claimed that only 4.77 per cent candidates could write the correct logic for a program, which should be a minimum requirement for any basic programming job. Out of the 36000+ students that took up a certain course related to computer science in over 500 colleges all over India, it was found that more than 60% wrote code that did not even compile properly and only 1.4% could write functionally correct and efficient code. Such figures are worrisome indicators of the quality and skill of the workers emerging from colleges in recent times. An article published in the Economic Times has stated that India's problem of substandard engineering education is widely known. According to the article, most engineering colleges in the country are unable to provide education to engineering student that would help them get suitable jobs. As students from colleges fail to get suitable jobs, they face decline in enrolment. Now a large number of these colleges are being closed down. Mr. Narayan Murthy, the co-founder of Infosys and a veteran of the industry, has stated that 80% of the youth of India are not suitable for any kind of jobs, placing the blame on the emphasis on rote learning in the educational system in our country, rather than problem solving.

Wikipedia defines Computer Science as “the study of processes that interact with data and that can be represented as data in the form of programs. It enables the use of algorithms to manipulate, store, and communicate digital information.” Computer Science is thus a study that is based on understanding, and not rote learning. It is a subject that must be taught in a way that stimulates the student’s logical reasoning and must not rely on his/her memorisation skills. The students must be tested on the basis of their ability to understand and infer algorithms. Computer Science is introduced to most student in the form of an “Introduction to Programming” class that covers topics in data structures and algorithms. Students are taught to familiarise themselves with the syntax of a common programming language like C, Java or Python. They are taught the basic data structures and workings of commonly used algorithms. These concepts are then applied by the students to solve problems programmatically.

The All India Council for Technical Education (AICTE) has expressed concern about the quality of exams where a large proportion of questions only require “simple memory recall”. It has been stated by the AICTE is that such questions do not ensure deep, meaningful thinking in the examinations. With the growing concern about rote learning prevailing over problem solving abilities as highlighted above, the lack of good quality skill testing is adversely affecting the field of Computer Science at the University level. One of the most popular types of questions in this fields asks: “What does the given code snippet do?”, with a small code snippet, with sufficient context, provided to the test taker in any one programming language. Analysing the code and deciphering the functionality of the code snippet involves an understanding of the programming language, understanding of the concepts, and logical and analytical reasoning. Thus, this form of question is very effective in testing the problem solving and thinking skills of an individual, rather than questions based on the intricacies of a programming language that involve rote learning.

Multiple Choice Questions is a commonly accepted way of testing students that is employed in several institutions worldwide. It consists of a stem which identifies the problem and a set of alternate solutions to the problem. Each option will contain either a key to the best answer to the question or a “distractor”. Distractors are defined as plausible but incorrect answers to the questions. Multiple Choice Questions (MCQs) owe their popularity to the ease with which they can be evaluated. As there is no subjective aspect to the answers of the question, it is very easy to administer an MCQ based test to a large number of students, with very quick evaluation and scoring, that requires little to no human intervention or skill. As the Multiple Choice Questions do away with a verbal answer, they limit the number of possible answers that are deemed correct for a given question from an infinite number, to a small finite number. The most popular value for the number of options for a given question is 4. The absence of verbal answers limits the difficulty of the question to an extent, as there are only 4 options to choose from. Hence, the efficacy of MCQs largely depends on the quality of the options. The incorrect options, also known as distractors are most effective when they serve to confound the user enough that the real answer isn’t obvious from a given set of options.

This requires that the options not be completely randomized but follow a pattern that places them in the vicinity of the real answer in the terms of semantics of the sentences.

It is important to understand and follow good coding practices. Among the principles followed, one that is often neglected is the practice of providing comments for the code one has written. Documentation comments are intended for anyone who consumes the code, beyond simply reading it. Although it has been said that good code is self-documenting, understanding the code by tracing the code will work only to a certain extent. Good code comments and documentation provide information about the working of a code snippet in a concise and precise manner. Code comments can vary with respect to coverage. They can provide intricate details about the code regarding the use and need of variables and various loops, to significantly abstract details such as use of the functions defined.

CHAPTER – 2

PROBLEM DEFINITION

There is a need for teachers to be able to create and evaluate deep, meaningful questions around programming concepts, in a manner that promotes analytical thinking and tests for logical reasoning and understanding, rather than a simple memory recall. The teachers require a solution that can provide the functionality with minimal efforts for the creation and evaluation of such questions. This project aims to create a tool that aids the teachers in creating Multiple Choice Questions, with good quality distractors based on given snippets of code, focussing on the domain of Introductory Programming Concepts.

The key problems the project aims to solve are:

1. Creating “deep, meaningful” questions – we create questions of the type “What does <code> do?”
2. Creating good “distractors” (incorrect choices) – MCQs in every domain have this challenge
3. Limiting teacher burden to create new questions based on topic/concept (where consensus is likely) and difficulty (where consensus is less likely)

CHAPTER – 3

LITERATURE SURVEY

There has been some work done in the form of research in the fields that this project aims to solve current problems. There are also a few existing solutions that provide some portion of the functionality that this project aims to provide

3.1 Existing Solutions

3.1.1 GeeksForGeeks

GeeksForGeeks is a web based platform that provides well written, well thought, and well explained solutions to select questions. It is a well known platform widely used to learn the basic concepts of Computer Programming, data structures, and algorithms in a variety of programming languages. The concepts are searchable by tags, and by concept name, and the search functionality is used by the user to find the most appropriate code as required for his/her purpose. The codes on the platform are of varying difficulties, ranging from the most basic introductory concepts, to complex algorithms that consist of multiple functions. GeeksForGeeks has a large focus on interview preparation, focussing on the most popular algorithms, leading to a diverse but highly limited set of codes on the platform. Some of the codes that are explained on the platform have a difficulty level associated, which is a general level assigned to the code or concept, irrespective of the perception of the user. GeeksForGeeks contains codes that are manually described and tagged to facilitate search. There are no questions provided on the codes, with an explanation being the only information associated with the code.

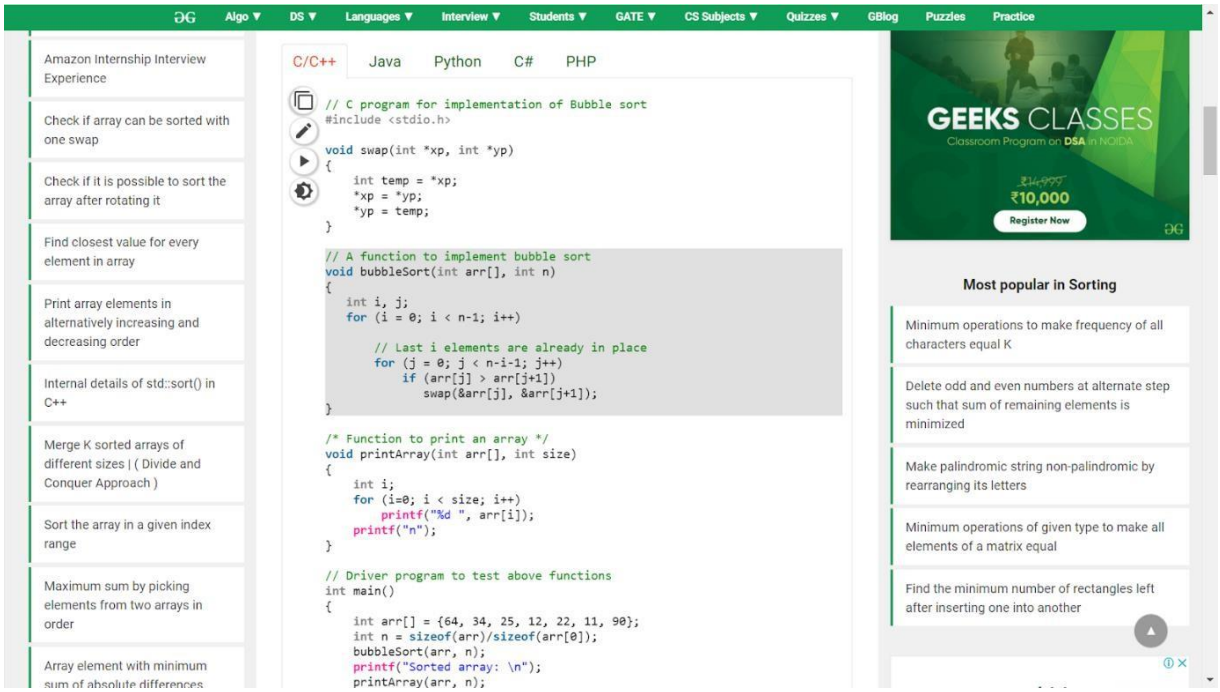

Fig 3.1 Geeks for Geeks code sample

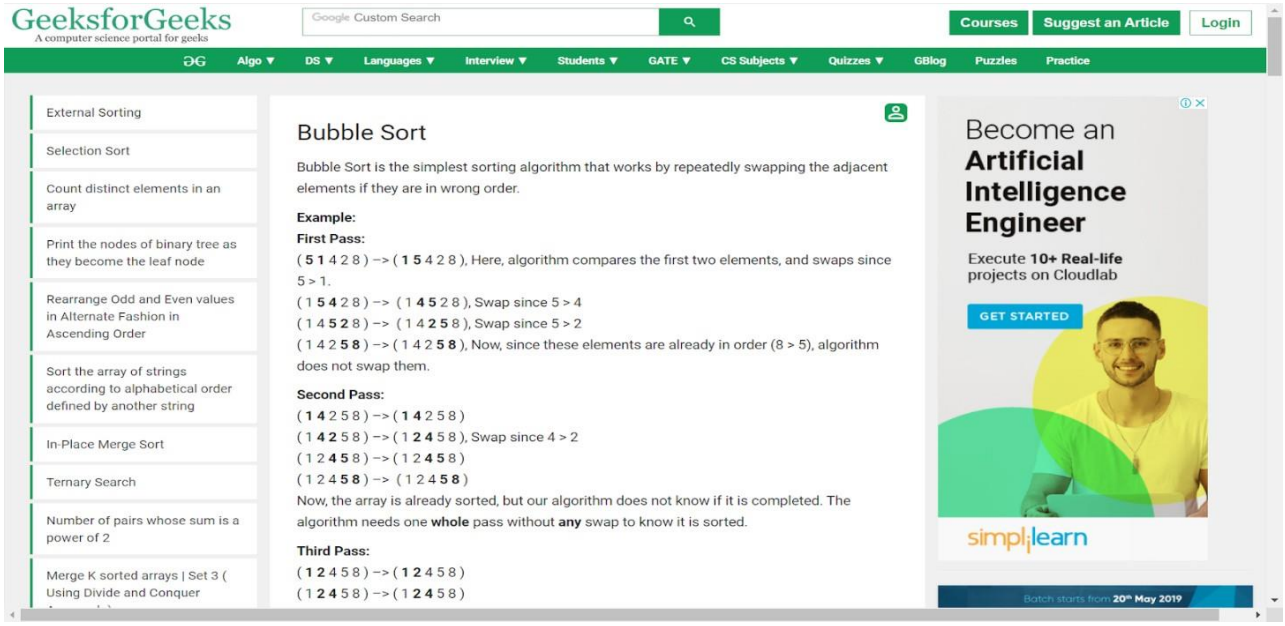

Fig 3.2 Geeks for Geeks explanation sample

3.1.1 StackOverflow

StackOverflow is a web based platform for crowdsourced codes, with user interactions. StackOverflow serves as a database for codes, mostly erroneous to be stored, which can be solved, or have solutions suggested by other members of the community. The codes are tagged for searchability and relevance, and the tags are manually provided by the uploader. Due to its nature as a platform to solve errors in code, most of the code provided on the platform are erroneous or are codes that do not have the complete context. These codes are very effective for finding solutions to isolated problems but are not very useful in the context of university level teaching. StackOverflow also does not provide any intuitive questions or difficulty rating for the codes that reside on the platform.

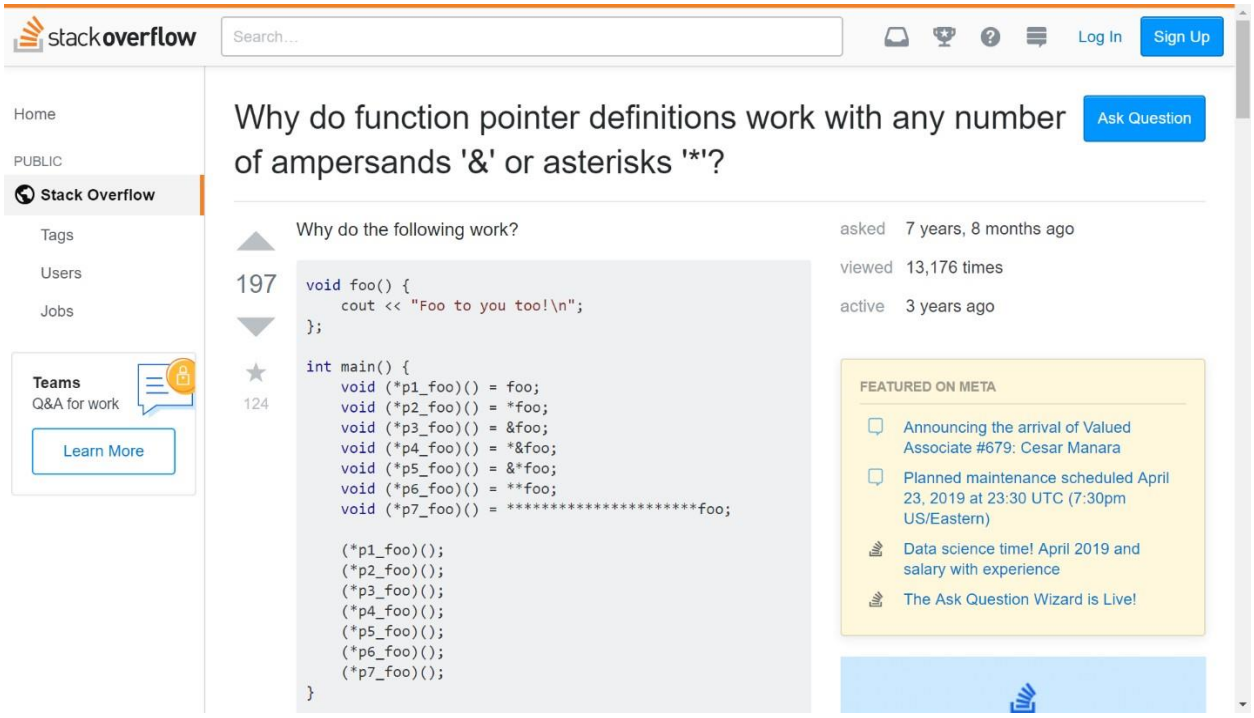

Fig 3.3 Stack Overflow code example

3.1.3 Salon (John Barr, Ananda Gunawardena)

Classroom Salon is an online collaboration tool that allows instructors to create, manage, and analyse social networks (called Salons) to enhance student learning. Salons are used by students to cooperatively create, comment on, and modify documents. Classroom Salon allows the instructor to monitor the social networks through various tools. They can gauge both student participation and individual effectiveness. This paper describes Classroom Salon and provides some use cases based on introductory computer science classes and presents some preliminary observations of using this tool. These observations are from several classes being conducted at Carnegie Mellon University.

Classroom Salon is used to annotate documents and videos. All course study material can be annotated by the students using the various available tools. Documents and other study material are directly uploaded into classroom salon. Several pedagogical activities can be created to engage students. Apart from encouraging effective student-student and student-teacher interaction, Salon also provides a variety of analytics and insights into student thinking.

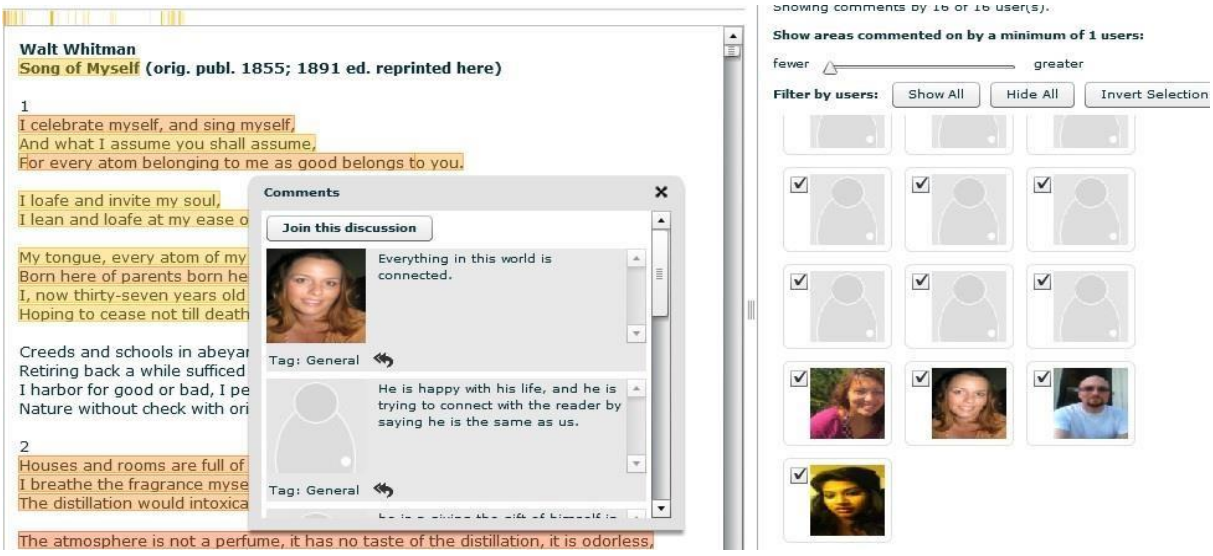

Fig 3.4 Salon example

3.1.4 Piazza (Pooja Sankar)

Piazza is a Q&A web service. It can be described as a "mixture between a wiki and a forum" that can be used with learning management systems. This tool was developed to encourage classroom discussion about assignments among students. Students can post questions, reply and answer other questions and post their comments. It even allows for users to post anonymously to encourage shy students to take part in classroom discussion. The service also displays updates as they happen thus preventing the site to become like any other dormant college site. This helps encourage activity on the site.

3.2 Suggesting description & distractors

3.2.1 Deep Code Comment Generation

3.2.1.1 **Deep Code Comment Generation by Xing Hu et al. (2018) for automatic generation of comments in Java Code from Abstract Syntax Tree using Natural Language Techniques**.

The paper illustrates how comments for functions are generated by feeding in the Abstract Syntax Tree to a Sequential Model to generate comments.
This paper was referenced by us to implement the comment suggestion feature for codes in the C language. The paper describes the procedure followed successfully to create a deep attention based neural network model that automatically generates comments for Java codes, capturing the semantics. The paper has specified the methodology of the research, the approach taken by the team to achieve the results, the concepts used for the machine learning model, and the quality of the results. The research specifies the conversion of a code into its Abstract Syntax Tree (AST) form so that there are no ambiguities in the code structure before training the neural network model.

The inherent structure of the code, being different from that of natural languages, poses the challenge of semantic capture in neural machine translation, which has been solved by the team by using the AST representation, deleting the irrelevant and redundant parts of the code, and creating a custom neural unit. The model has been trained and validated on a very large dataset. The results achieved are better than previous efforts, with a nearly 30% Bilingual Evaluation Understudy (BLEU) score on the testing set, which is better than the previous state-of-the-art code summarization model (Code-NN) at 20%. The industry-wide BLEU score of state-of-the-art neural machine translation models is about 40%.

3.2.2 Automatic Generation of Multiple Choice Questions

3.2.2.1 Automatic Generation of Multiple Choice Questions using Surface-based Semantic Relations by Naveed Afzal (2015)

This paper uses unsupervised relation extraction to identify the most popular named entities and the related semantic relationships between them. This relation extraction patterns help in the development of Multiple Choice Question generating systems. This is a popular technique used in several Natural Language Processing problems. The authors have investigated several relation extraction patterns. They also describe several assumptions made in the linguistic expression of the semantic relationship between the named entities.

3.2.2.2 Automatic Generation of Multiple Choice Questions Using Wikipedia by Arjun Bhatia et al. (2013)

This paper presents a methodology for the generation of Multiple Choice Questions (MCQs) from resources that already exist on the internet, and specifically from Wikipedia. The questions are created from text that is available by using a set of patterns, which are themselves acquired from existing questions. The paper also presents a new method of generating distractors, or incorrect

choices for the questions, using named entity and the relations to find similar named entities by searching Wikipedia. The questions and the generated distractors, which are focussed on the sports domain, have been evaluated by humans, following a set of parameters, and the results have been found to be reasonably satisfactory.

CHAPTER – 4

PROJECT REQUIREMENTS SPECIFICATION

4.1 System Architecture and Flow

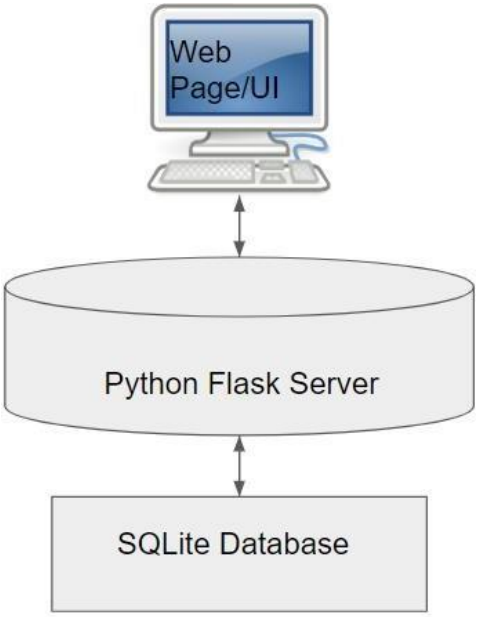

Fig 4.1 Basic System Architecture

The system follows a client server architecture, with the server at the back-end connecting to multiple clients, which have the front-end code to render on a browser. The database is connected to the back-end or server, and all the database accesses are performed from the server. The server performs most of the processing on the code snippets, thus ensuring that the client has a very light load.

The interface between the Client and Server modules is a HTTP interface via XHR calls. The interface between the server and database is SQL queries. The communication occurs over a standard internet connection. A HTTP connection with an underlying TCP connection allows the clients to interact with the server. The connections can be made from any device around the globe that has access to the internet. HTTP ensures that the communications follow a global standard that is widely used in applications and does not require any additional hardware or setup. The Python-Flask server accepts connections on the port 5000, and clients can make a connection to the server by making a request to the port 5000 on the IP address of the server.

4.2 Modules

4.2.1 Module 1: Client Module - The Web Page

A Client web application that runs on browsers. This consists of the HTML Browser view which the user uses to interact with the tool. The user uploads code onto a text area that uses CodeMirror to appropriately represent uploaded code according to the language it was written in. The conversations around the code uploaded are stored on our local database.. We use HTML and Bootstrap as the technologies for this module. It is a web-based GUI for ease of use. This also allows our product to be platform independent as a web page can be easily accessed over a range of devices starting from PCs to Smartphones. Bootstrap allows us to scale to a variety of screen sizes and orientations with ease.

Requirement #	Requirement
1	An easily accessible interface that can work on a variety of devices

2	Provision to upload code along with relevant comments and tags in a convenient manner
3	Provision to view all available code resources
4	Ability to easily search for code resources based on certain topics (tags)
6	Provision to communicate with other users and have discussion about the code
7	Ability to rate code and view code based on a difficulty metric
8	The front end must be able to send back the code along with all relevant details like comments for the code, code difficulty, conversations around code and so on. It must also receive and render relevant information sent back by the server

Table 4.1 Web Page Module Requirements

4.2.2 Module 2: Server Module

This consists of our server and Machine Learning models for processing the code comments to generate multiple choice questions and other intelligent functionality, along with serving the files to the browser. We use Python Flask as our back end. We run our tool on a local instance of Flask. The server also takes care of login and authentication. We opted for Python as it offers a range of libraries that would add functionality to our platform. It also allows easy access to APIs of various social media platforms. There is also a lot of support for applying Machine Learning techniques in the form of Tensorflow and Keras libraries.

Requirement #	Requirement
1	Receive all data (code, related comments and code difficulty) and store it in the database

2	Return all the required data such and serve up the correct web pages to the client as requested
3	Perform Processing on data for the intelligent modules

Table 4.2. Server Module Requirements

4.2.3 Module 3: Database Module

The Database module consists of the connections to, and operations on the database to add, modify and access data. The database stores the relations that are to be captured in the system, and hence, a relational database is being used. SQLite 3 has been chosen as the database for its easy integration with Python and the convenience of having a local instance without requiring a separate configuration. The details that are required by the system for the operations and functionality are accessed from the database, and hence the database module forms a key part of the project.

The database stores the mapping between descriptions and other metadata associated with a snippet of code, such as the language, and the filename with which it is stored in the filesystem. The database also stores the important data associated with the code, including the tags that are associated with each code, the comments for each function, the number of times a certain user has viewed the code, and the difficulty associated with the code for each person. The database also stores the authentication details of the user. This data is used by the other modules for providing functionality such as code recommendation for users, tag recommendation, and difficulty estimation and prediction.

Requirement #	Requirement
1	Write data into a certain table of a database

2	Access the data from a certain table of the database
3	Access data joined from multiple tables of the database

Table 4.3. Database Module Requirement

4.2.4 Module 4: Tag Generation Module

The tag generation module is one of the intelligent modules of the system. This module performs the task of generating tags automatically for a given snippet of code. The tags are generated based on the structure of the code snippet. The module leverages the Doc2Vec framework provided by Gensim to gather code structure similarity. The code is cleaned to remove all irrelevant and redundant parts, and converted into an Abstract Syntax Tree to capture the structure unambiguously. The Doc2Vec based model is used to find codes that are similar in structure and semantics to the given snippet of code, and the tags associated with such similar snippets of code are consolidated, thresholded based on similarity, and recommended for the given snippet of code.

The module reduces the load on the uploader to provide tags for a given snippet of code, instead making each snippet of C code searchable based on the tags which are auto-generated. Thus, all C codes that are entered into the database are rendered searchable with minimum requirements from the uploader to appropriately tag or describe the snippet of code

Requirement #	Requirement
1	Recommend a set of tags for the code
2	Capture the structure similarity between snippets of C code

Table 4.4. Tag Generation module requirements

4.2.5 Module 5: Multiple Choice Question Generation Module

The Multiple Choice Question (MCQ) module is one of the intelligent modules of the system. This module generates MCQs from code snippets and the comments associated with the functions, providing the key functionality of the system to serve as a tool to generate high quality questions that elicit logical reasoning. This module is responsible for generating questions of the type “What does the <code> do?”, on the given code snippet, along with the choices for the MCQ. The distractors, or incorrect options, are generated by this module in a manner that serves to confound the user enough to require logical thinking.

The distractors are generated using sentence similarity to find choices that are close to the correct answer. Thus, the options are not randomized, which would lead to an ease of answering and a large variance in the difficulty level of questions, but are intelligently specified such that they are optimal. As the module is working with the data that is constantly updated in the database, the quality of the distractors will increase with time and data. The distractors are generated using a model based on the Doc2Vec framework by Gensim. The Doc2Vec model is trained to capture the sentence similarity for all the comments that exist for every function in the database. Once the model has trained, a vector can be inferred for any new sentence. This vector is gathered for the new comment of a new code snippet, which is the right answer to the question as mentioned above. Cosine similarity between vectors is used to determine the closeness of the sentences. The 3 closest sentences or comments that exist in the database are picked for a new comment, taking care to ensure that the similar sentences are not the same as the new sentence, or the same as each other. Summarily, the 3 nearest distinct comments are chosen as distractors for each comment, and are provided as options for the given function

Requirement #	Requirement
1	Generate distractors for an MCQ
2	Find the closest distinct sentences
3	Shuffle the order of the choices

Table 4.5. MCQ Generation module requirements

4.2.6 Module 6: Comment Suggestion Module

The comment suggestion module is one of the intelligent modules of the system. This module performs the task of generating and suggesting a comment for a given snippet of code, in the event that the uploader has not provided a comment for it. The module uses an attention based sequence-to-sequence neural model to generate comments for each function of a snippet of code. The model serves to suggest short and concise comments for snippets of code based on the code structure and semantics.

The attention based model takes the code as the input, and generates a sequence of words as the output. The input code is encoded to a vector that can be used by the neural network. The code is not used as it is, but is converted to its Abstract Syntax Tree (AST) representation to capture the structure of the code. PyCParser, a Python library is used to convert the code to its AST form, which is represented in the JavaScript Object Notation (JSON) format. Th output codes are split into words and are converted into one hot vectors. The neural network is then trained to learn on the dataset and generate comments. These comments are suggested to the user, who must then correct it as required, and choose whether to accept or reject a comment for a function.

Requirement #	Requirement
1	Generate a comment for a given snippet

Table 4.6. Comment suggestion module requirements

4.2.7 Module 7: Difficulty Prediction Module

The difficulty prediction module is one of the intelligent modules of the system. This module performs the task of predicting the perceived difficulty of a given snippet of code for a particular user. As the perception of difficulty differs from individual to individual, the score is required to be predicted in a manner customized to the user. This module uses collaborative filtering, using user similarity to find the likely perception of difficulty of a code snippet.

The prediction of difficulty of unseen code snippets for a user helps in providing a mechanism to find the most appropriate and relevant codes for a user's purpose. The user can select to view the codes that are most appropriate according to his/her perception of difficulty, without having to manually rate every code for difficulty. The adaptive difficulty metric offers the advantage of dynamic difficulty ratings and personalization, compared to a global metric where the difficulty rating is associated only with the code and not the viewer. The predictive scoring creates patterns, with difficulty ratings on certain topics or areas or even languages being different for different users based on their personal skills and capabilities. Finally, this module is used to sort the codes based on the difficulty for a user while recommending and searching for codes by topic. The user may update the difficulty with his/her actual perceived difficulty while viewing the code

Requirement #	Requirement
1	Find the most similar users based on difficulty ratings

2	Predict a difficulty level based on the similar users
3	Update the difficulty level if the user provides a rating

Table 4.7. Difficulty Prediction module requirements

4.2.8 Module 8: Code Recommendation Module

The code recommendation module is one of the intelligent modules of the system. This module performs the task of recommending the most appropriate code snippets to a user. Users view codes on the platform, which are recorded and saved in the database, and the pattern used to recommend codes that they would likely be interested in. The code recommendation works on codes across all languages, and works on the user similarity rather than code structure similarity to transcend languages. The module uses collaborative filtering, with a custom algorithm to find the nearest users, and the codes that have been viewed by that user. These codes are than ranked using a custom metric that involves the similarity score between users.

As the platform will host a large number of codes, the users will only want to be able to see the most suitable and appropriate ones. The system aims to provide this functionality to better the user experience and an adaptive and intelligent system that customizes the list of codes shown for the user.

Requirement #	Requirement
1	Find the closest users
2	Rank the codes based on custom metric

Table 4.8. Code recommendation module requirements

4.2.9 Module 9: Search Module

The search module provides the functionality to search the database that exists in the system, and stores the codes. The module performs search by tags, which is an efficient way to search for codes, or concepts as required by the user. Multiple tags can be specified by the user, and the search algorithm uses all of them to query the database. A custom ranking system orders the search results in the order of highest to lowest relevance based on the number of keywords or tags that were hit by the query for each snippet of code. The search module also provides functionality to filter the code snippets based on language and difficulty level. Thus, the user can use this module to get the code snippets that are required about a particular concept, in the language of their choice, and with a difficulty level that is most appropriate for their use.

Requirement #	Requirement
1	Search codebase based on tags
2	Filter on language
3	Filter on difficulty

Table 4.9. Search module requirements

4.3 Product Perspective

The platform is not a component of a larger system or product. It is an independent entity. The software technologies used for development would be HTML, CSS/Bootstrap and other related web technologies for the user interface. A Python-flask server is used as the backend for the platform. The development of the product an happen on a Windows or Linux based system. There are no OS dependencies for the development of the project. The project is being developed on our personal computers. The development requires no additional hardware resources.

4.4 User Characteristics

Creators:

These users add code to the collaborative platform. The creator must annotate each function with a description of what it does (this description is the correct answer for the MCQ – our tool limits the teacher burden by providing automatic suggestions) and tags to facilitate search (the creator can specify tags, or simply accept/reject auto-generated tags). The creators contribute to our database and help improve the tool by providing data to improve our tagging system, our recommendation system, our MCQ distractor generation and our comment suggestion system. The creators can also participate in conversations around a code resource. They will gain points for uploading code and can keep track of this through their profile.

Consumers:

These users use the platform to find questions based on tags and keywords (our tool automatically suggests distractors for each MCQ). The platform asks users to rate the difficulty of each question they view, and uses these ratings to automatically order the other problems based on this user's conception of difficulty. They answer the MCQs that are found while viewing code resources and gain points for the correct answer. They can view all their cumulative scores in their profile. They can also view other user's scores in the scoreboard. They too participate in discussions around the code.

CHAPTER – 5

SYSTEM REQUIREMENTS SPECIFICATION

5.1 Hardware and Software platform for Development

HARDWARE:

The system uses a client-server architecture to interface between the backend and the frontend of the platform. The database is connected to the backend of the system, either as a local instance or as a cloud instance. The client and server interact with each other over a HTTP interface.

The server is a Python-Flask instance that runs on the hardware. Python-Flask is relatively lightweight and does not require any special hardware requirements. The server allows for connections on port 5000 of the system. The HyperText Transfer Protocol(HTTP) is used for transfer of data between the server and client.

The database management system being used is SQLite3. This is a relational database that runs an instance locally alongside the python-flask server instance. There is no special hardware requirement of the DBMS, but its storage capacity is limited by the disk capacity of the local system that it is running on. A cloud based database will be used for scalability when the local instance proves to be insufficient to hold all the data. Hence, the hardware requirement for a local DB is to have a large amount of storage, and the same is required of a cloud storage system. However, since the cloud storage solutions are provided by a third party with expandable storage and scalability, we are not limited by the hardware resources on our local system

Any device that can connect to the internet and has a Web Browser that can work with HTML will be able to access our system. There are no other hardware requirements for any of the devices that access the application. The system will be supported across a diverse range of devices, ranging from laptops and desktops to mobile phones for ease of access and availability. The devices are

supported over an internet connection and the HyperText Transfer Protocol (HTTP) is used for transfer of data.

Hardware Platform for Development

Computer System	
System	Intel Core i7
Hard Disk	256 GB
RAM	8 GB

Table 5.1 Hardware Platform for Development

SOFTWARE:

Front End : We will be using HTML and Bootstrap. It will be web based GUI for ease of use. This also allows our product to be platform independent as a web page can be easily accessed over a range of devices starting from PCs to Smartphones. Bootstrap allows us to scale to a variety of screen sizes and orientations with ease.

Back End : Our back end server is a Python Flask Server. We opted for Python as it offers a range of libraries that would add functionality to our platform. It also allows easy access to APIs of various social media platforms. There is also a lot of support for applying Machine Learning techniques in the form of Tensorflow and Keras libraries.

Database : The database we are using is SQLite. This is a relational database that is well supported by Python. We are using a relational database since we would like to keep track of the relations between users and tags, questions etc. It is well suited for the data to be stored for the problem.

The database runs locally with a python server instance and does not require an external instance. If we require to scale our platform, we would have to migrate to a cloud server as the capacity of this is limited by the hardware used.

Software Platform for Development

Operating System	
Name	Windows
Version	10
Source	https://www.microsoft.com/en-in/software-download/windows10
Application Coding Language	
Name	Python
Version	3.6
Source	https://www.python.org/downloads/
Name	Python Flask
Version	1.02
Source	http://flask.pocoo.org/
Name	HTML
Version	5
Name	JavaScript
Machine Learning Modules	
Name	Keras
Version	2.24
Source	https://keras.io/
Name	Tensorflow
Version	2.0
Source	https://www.tensorflow.org/

Name	Gensim
Version	3.60
Database	
Name	SQLite3
Version	3.28.0
Source	https://www.sqlite.org/index.html

Table 5.2 Software Platform for Development

5.2 Hardware and Software platform for Deployment

Hardware Platform for Deployment

Server	
System	Any
Hard Disk	Any
RAM	Any
Mobile Phone	
OS	Any
Memory	Any

Table 5.3. Hardware platform for Deployment

Software Platform for Deployment

Software	Web Browser (Chrome/Firefox/Microsoft Edge etc.)

Table 5.4. Software Platform for deployment

5.3 General Constraints, Assumptions and Dependencies

Hardware Limitations

1. We have assumed that the current expected number of users will be small and the data generated can be stored in a local database. In case of increased use and traffic to our tool we will migrate to a cloud server.

Software Limitations:

1. Application requires a mobile phone or computer with a web browser
2. Requires internet connectivity to connect to server
3. The intelligent design features of automatic multiple choice question generation, and comment and tag suggestion are restricted to only C language for optimum results. However, users can still upload code and have conversations around them in any language (both programming and vernacular languages).

5.3 Risks

The project risks include the following:

1. As the project involves a large community as its users, it is possible that the requirements of the entire diverse community are not fully captured in the Customer Requirements Survey. Any later changes to the requirements will be accommodated as additional features or patches to the system
2. The system stores textual data, which is largely the code snippets that are uploaded and the conversations that take place. It is possible that the size of the data becomes very large with continued usage of the system and exceeds the capacity of the local storage. In this situation, the data storage will be migrated to the cloud.

5.4 Performance Requirements

1. The system supports a large number of users at a time. Each user has his/her own credentials, and can access content separately and independently from other users.
2. The code snippets or code files that form the core of the contents displayed on the system will not be too large. Hence, the system supports file transfers to and from servers of sizes in the order of a few kilobytes (kB)
3. A large number of code snippets uploaded by various users will be available on the system. We expect the total size to continuously expand, as content is always being added.
4. The system is not time sensitive, and hence there is no limit imposed on the latency for usability of the software. As the data to be transferred is primarily textual, the expected latency is small, in the order of a few hundred milliseconds, or a few seconds, even over low speed networks
5. The amount of data to be processed can be quite large if a lot of activity is to take place at the same time. However, since the data can be processed independently of any other data, and the application does not have a real time use case, the time taken to process the data at the backend will not affect the usability of the application, allowing users to interact with the other parts of the application and already existing data without any delay. The amount of processing to be performed on the data can be quite large depending on the size and complexity of the code that is being processed. No significant data processing, other than simple formatting, is occurring at the front-end.
6. The intelligent components of the system can add to latency in terms of time to initially load the models and predicting the desired results from the loaded models. The initial training of the Doc2Vec models will take up some time the first time the server starts up..

7. The system extends compatibility to a wide variety of devices all around the globe. There are no restrictions or requirements to access the system except having a working internet connection and a standard web browser

5.5 Special Characteristics

1. The data that we store is mainly the code uploaded and relationships between code and tags. The data is not sensitive and need not be stored with cryptographic techniques. Our site will however be protected from malicious access by SQL injection by ensuring the data we take from forms will have to adhere to a format we set through regular expressions. The code that we upload is not to be executed at any time on the system and is stored as plain text. We will try and take precautions to prevent the execution of malicious code uploaded as legitimate course material.
2. We store all the questions, comments and code contributed by every user thus being able to keep track and log activity. This will also help us encourage participation by providing users with points based on activity.
3. A notable special feature would be automatic MCQ generation for the code. The Multiple Choice Question (MCQ) module is one of the intelligent components of the system. We generate MCQs from code snippets and the comments associated with the functions. The tool generates questions of the type “What does the <code> do?”, on the given code snippet, along with the choices for the MCQ. The distractors, or incorrect options, are generated by this module in a manner that serves to confound the user enough to require logical thinking. This is a feature unique to our platform compared to all other existing platforms.
4. A special feature unique to our tool is automatic tagging for code uploaded. This is shown as suggestions to the user and reduces the load on the uploader to provide tags for a given snippet of code, instead making each snippet of C code searchable based on

the tags which are auto-generated. Thus, all C codes that are entered into the database are rendered searchable with minimum requirements from the uploader to appropriately tag or describe the snippet of code

5. In order to encourage good coding practices, users are suggested comments for each of the function that is present in the code uploaded. The tool performs the task of generating and suggesting a comment for a given snippet of code, in the event that the uploader has not provided a comment for it. In case comments have been provided for the function it is automatically picked up by the tool and displayed to the user to verify before submission. These comments are suggested to the user, who must then correct it as required, and choose whether to accept or reject a comment for a function.
6. Another unique feature of our code is the ability to provide a perceived difficulty level for all code, even those that haven't been viewed by the user. As the perception of difficulty differs from individual to individual, the score is required to be predicted in a manner customized to the user. The user can select to view the codes that are most appropriate according to his/her perception of difficulty, without having to manually rate every code for difficulty. The difficulty level assigned is thus not a global difficulty and is not user interdependent. It is based on an individual's perception and not the code itself and is thus highly personalized. Finally, this difficulty is used to sort the codes for a user while recommending and searching for codes by topic. The user can also update the difficulty of a code upon viewing it.

5.6 Other Requirements

5.6.1 Site Adaptation Requirements

1. System that must run the application must have a good internet connection
2. System can be any device eg. Personal Computer, Mobile Phone, Tablet etc.
3. System must be equipped with a web browser.

5.6.2 Safety Requirements

There are no safety requirements for the system, as there is no sensitive data that is being used or processed by the system, nor does the operation of the system affect any external applications that are sensitive in nature.

5.7 Help

The system will be fairly simple to use for all users as it is a simple web based GUI. The buttons will be marked with descriptive statements that follow standard and uniform guidelines to minimize the learning curve. As the primary users of our system are University level teachers, who would be fairly familiar with the use of computer applications, we do not believe there would be a need to provide guidelines on the basic use of the system.

5.8 Packaging

The system will be packaged as the following:

- The system will not need to be packaged and distributed to the users as it is a web based system.
- The backend of the system will be packaged along with the dependencies to be able to run on a server/host.
- The clients, or users, will be provided with a URL that can be used to access the system

CHAPTER - 6

SYSTEM DESIGN

6.1 Design Constraints, Assumptions and Dependencies

6.1.1 Design Constraints:

1. The intelligent design features of automatic multiple choice generation is restricted to only C language for optimum results. However, users can still upload code and have conversations around them in any language (both programming and vernacular languages).
2. We have assumed that the current expected number of users will be small and the data generated can be stored in a local database. In case of increased use and traffic to our tool we will migrate to a cloud server.
3. The comment suggestion feature only works on properly formed, syntactically accurate code in the C language. The constraint arises from the need to convert the code to its Abstract Syntax Tree representation to remove ambiguities, which is only possible from correct code.
4. The suggestion performs best on codes in the domain of Introductory Programming, due to the limited nature of the dataset it was trained on

6.1.2 Assumptions

1. The user is using a system with internet connection and a web browser.
2. The code uploaded by the user is syntactically correct
3. The code uploaded is in the same language as specified by the user
4. The final comments submitted by the user are relevant to the function in the code uploaded
5. The final tags submitted are relevant to the code uploaded

6. As the comments sent in are legitimate the MCQs generated will have relevant right answers and distractors

6.1.3 Dependencies:

The system does not currently does not have any external dependencies. A future task, of attaching a social media platform, which is pending due to lack of approval by Facebook, will have a dependency on the Facebook API, and the platform. A failure of any of the API will lead to loss of functionality of the system

6.2 Architecture

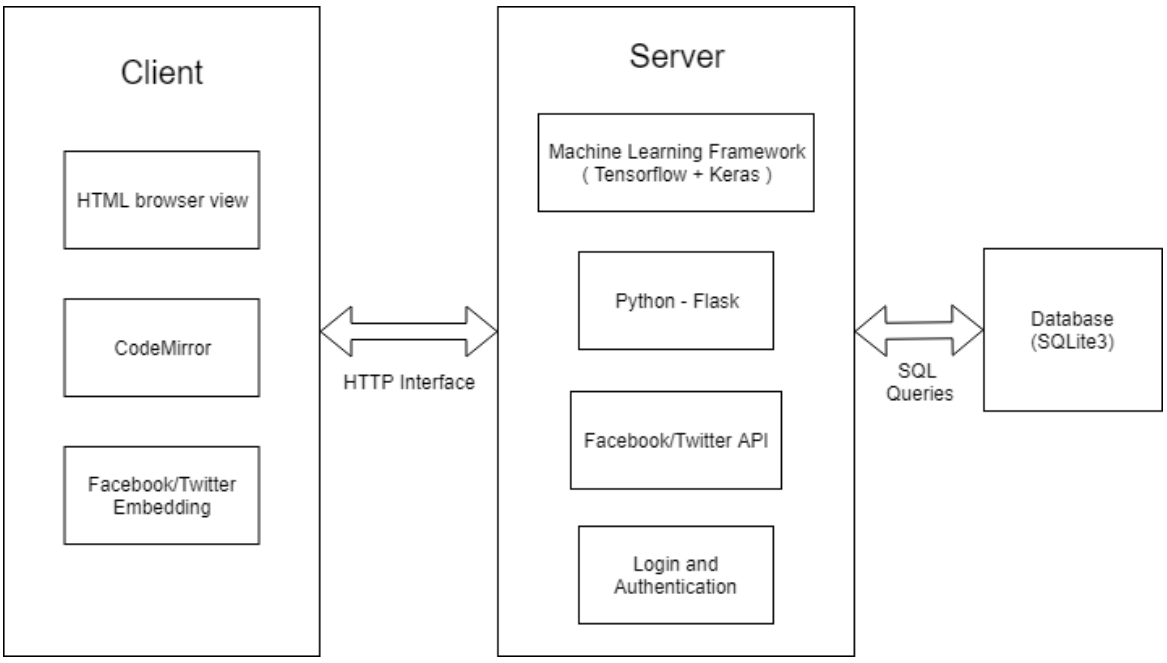

Fig. 6.1 Architecture diagram

CHAPTER - 7

DETAILED DESIGN

7.1 Design Description

7.1.1 Master Class Diagram

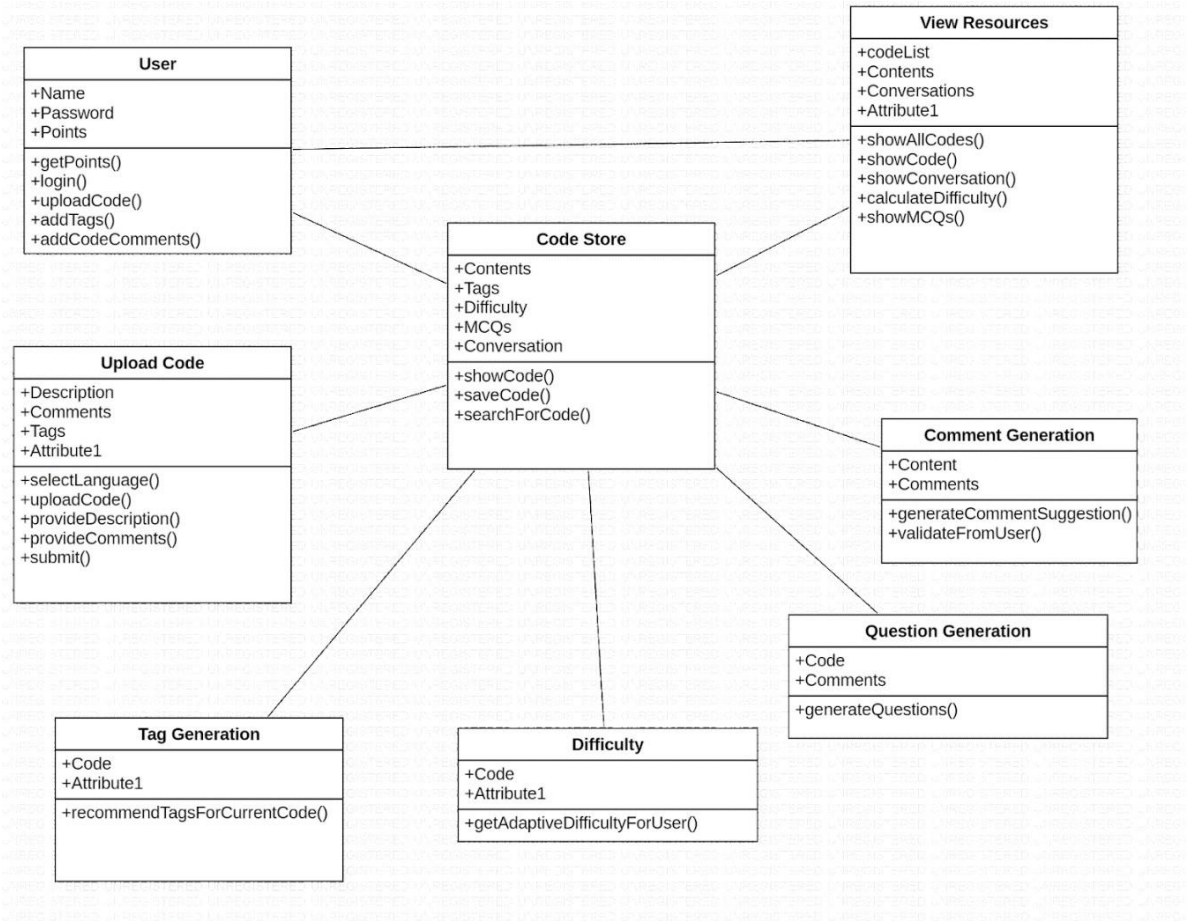

Fig 7.1. Master Class Diagram

Web Page

- The application has a sign-up and login module which enables the users to be authenticated.
- The user can upload code resources. While uploading code resources the user uses the comments and tags suggestions to properly annotate and tag the code
- The user can view all available code resources, have a discussion with other users around it and assign a difficulty level for each code
- The user can easily search all the tagged database
- The user is scored for his activities and can view his cumulative score in his profile

Server

- Receive all data (code, related comments and code difficulty) and store it in the database
- Return all the required data such and serve up the correct web pages to the client as requested
- Use a machine learning model to suggest comments to the user
- Suggest tags to the user upon code upload
- Provision to search database with tags
- Provide and maintain code difficulty levels
- Generate and serve up Multiple Choice Questions for functions in a code resource

3. Database

- The database stores the relations that are to be captured in the system, and hence, a relational database is being used.
- The details that are required by the system for the operations and functionality are accessed from the database, and hence the database module forms a key part of the project.

- The database stores the mapping between descriptions and other metadata associated with a snippet of code, such as the language, and the filename with which it is stored in the filesystem.
- The database also stores the important data associated with the code, including the tags that are associated with each code, the comments for each function, the number of times a certain user has viewed the code, and the difficulty associated with the code for each person. The database also stores the authentication details of the user.

4. Tag Generation

- Performs the task of generating tags automatically for a given snippet of code.
- The tags are generated based on the structure of the code snippet.

5. Multiple Choice Question Generation

- Generates MCQs from code snippets and the comments associated with the functions, providing the key functionality of the system to serve as a tool to generate high quality questions that elicit logical reasoning. .

6. Comment Suggestion

- This module performs the task of generating and suggesting a comment for a given snippet of code, in the event that the uploader has not provided a comment for it.
- These comments are suggested to the user, who must then correct it as required, and choose whether to accept or reject a comment for a function

7. Difficulty Prediction

- Performs the task of predicting the perceived difficulty of a given snippet of code for a particular user. As the perception of difficulty differs from individual to

individual, the score is required to be predicted in a manner customized to the user. This module uses collaborative filtering, using user similarity to find the likely perception of difficulty of a code snippet.

- Used to sort the codes based on the difficulty for a user while recommending and searching for codes by topic. The user may update the difficulty with his/her actual perceived difficulty while viewing the code

7.1.2 Use Case Diagram

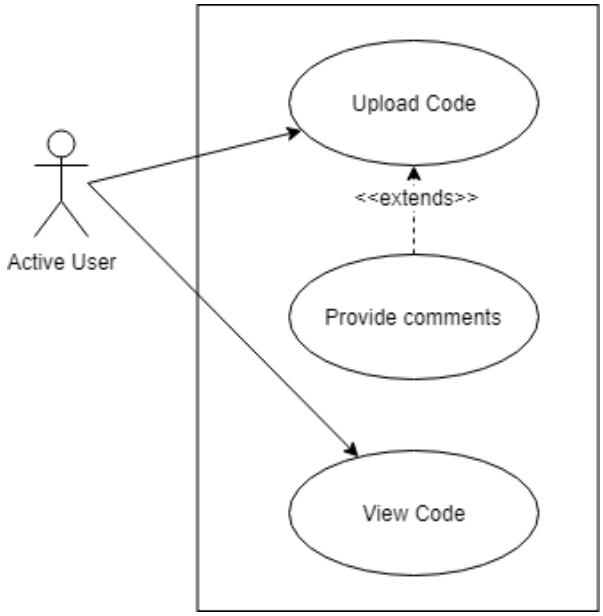

Fig 7.2 Creator use case diagram

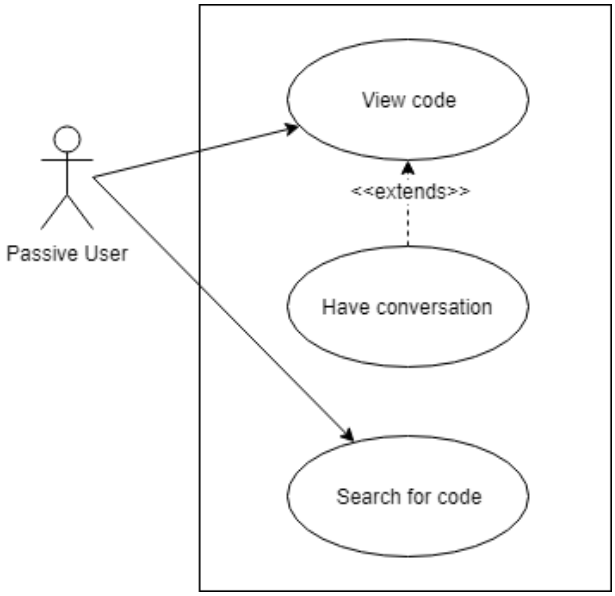

Fig 7.3 Consumer use case diagram

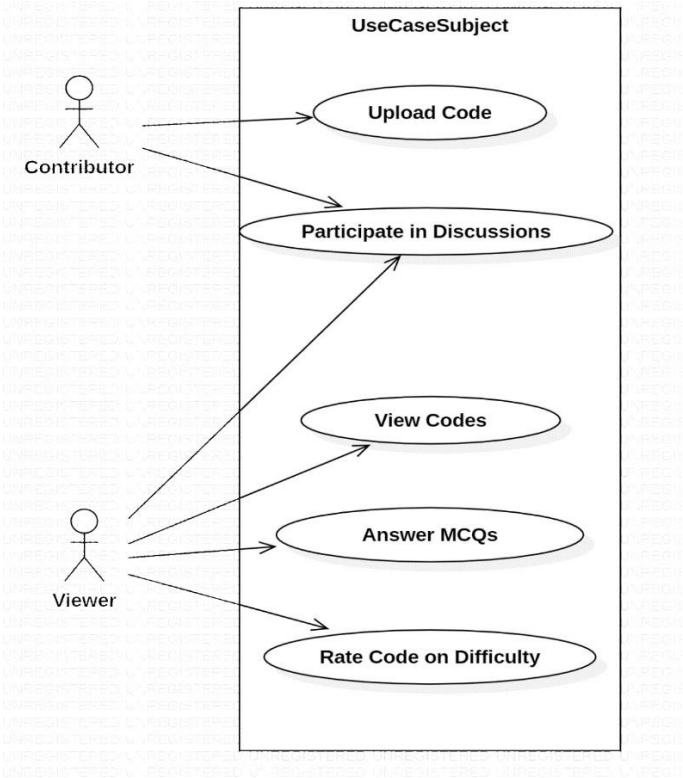

Fig 7.4 Master Use Case Diagram

Use Case Item	Description
Contribution	The user contributes a snippet of code to the system. This snippet of code is accessible to all the users. The user has the option to provide comments to the code in this use case
Viewing and Use	The user can view the code that is present on the system. This use case is an interaction between a single user and the contents on the system.
Searching and use in Assignments	The user can search for particular snippets of code or concepts to use, primarily in the classroom. The user can use the questions that are generated from the code by the machine learning module in the system
Conversations	The user can have conversations around a snippet of code uploaded by the contributor. The conversation is also made available on a social media platform for availability

Table 7.1 Use Cases

7.1.3 Class Diagram

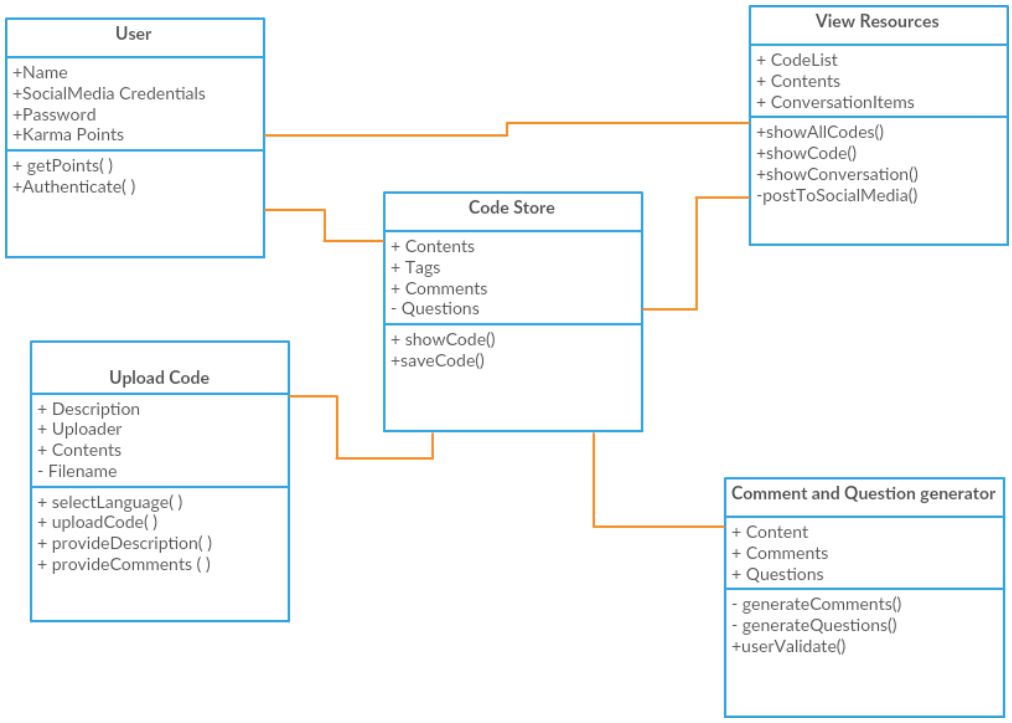

Fig 7.5. Class Diagram

Class Descriptions:

The system does not follow an Object Oriented Model. The classes represented in this section are the main partitions of functionality of the system

The main classes are :

- User : The entity that is using the application. The user has his/her credentials, and logs into the application. The user is also assigned points and recommended appropriate codes based on their use

- Upload Code : The user uploads code about a particular concept onto the tool in a language of his choice

- Generate Comments on Code : The system generates comments on the code, and has the user validate them

- View all Resources : The user can view all uploaded code that has been submitted by other users

- View Uploaded Code : The user can view his/her uploaded code as well as code uploaded by other users. He can also view related conversations and participate in them.

- Question Generation : Questions are generated based on code uploaded that will help with learning

- Find Resources (based on keywords) : Users can search for code or resources on certain concepts based on keywords or tags

7.1.4 Sequence Diagram

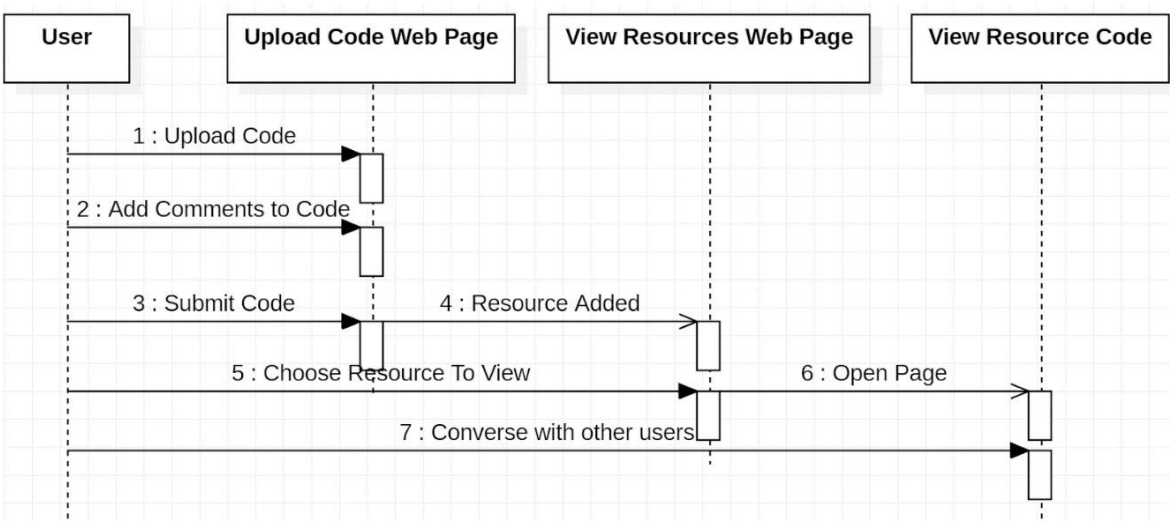

Fig 7.6. Interaction of User With Client Module Sequence Diagram

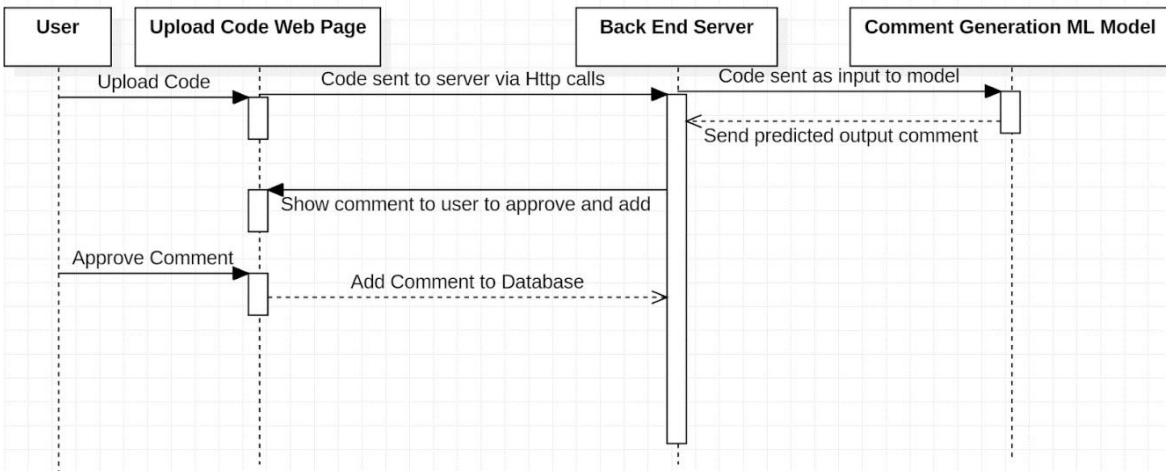

Fig 7.7. .Comment Generation Sequence Diagram

The above figure depicts the Sequence Diagram of the system. The user uploads code with the appropriate comments and tags. After submission the code is added to the list of resources. The

user can then view the various available code resources. On selecting a code resource, the user can then answer various MCQs on the code. The user can also have discussion with other users about the code.

7.2 ER Diagram

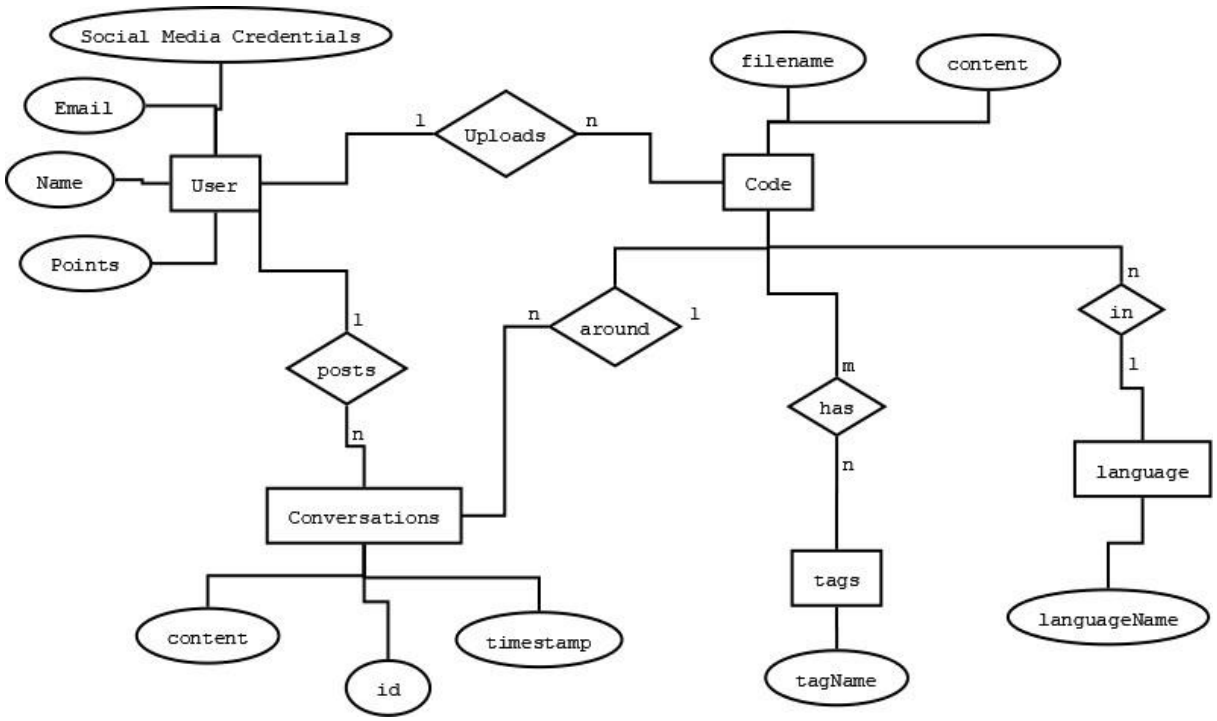

Fig 7.8. ER Diagram

#	Entity	Name	Definition	Type
ENTITIES				
1.	User	User	The person who uses the system	
2.	Code snippets	Code	The code snippets that are stored in the system	
	Conversation	Conversation	The conversations that happen around the code snippets	
	Tags	Tag	A finite set of tags that are associated with the code snippets	Finite set, for repeated values
	Code Language	Language	The language the code snippet is written in	Finite set ,for repeated values
#	Attribute	Name	Definition	Type (size)
DATA ELEMENTS				
	name	name	name of the user	Text
	email	email	email ID of the user	Text
	social media credentials	social media credentials	social media credentials of the user	Text(0 or 1)
	points	points	points awarded to the user	Numeric

	filename	filename	filename associated with file, where it is stored	Text
	content description	content	description of the file content	Text
	conversation content	content	post by user	Text
	id	id	unique ID for post	Text
	timestamp	timestamp	time at which post was made	Numeric
	tag name	tagName	tag associated with code	Text
	language name	languageName	language code is written in	Text

Table 7.2. Entity Attributes

7.3 User Interface Diagrams

The primary user interface for interacting with the system is a web based GUI. The GUI will be accessible by a browser. The user interacts with the GUI using a mouse based interface. The GUI contains buttons that are clickable to perform actions and text fields for textual inputs that can be provided via a keyboard. The usage will be very similar to other web applications and will not require a steep learning curve for interacting with the system

The web application will consist of multiple pages. Navigation between pages is done through the toolbar at the top of the page, which lists all the pages of the system, and as a result of certain actions taken on a currently open page.

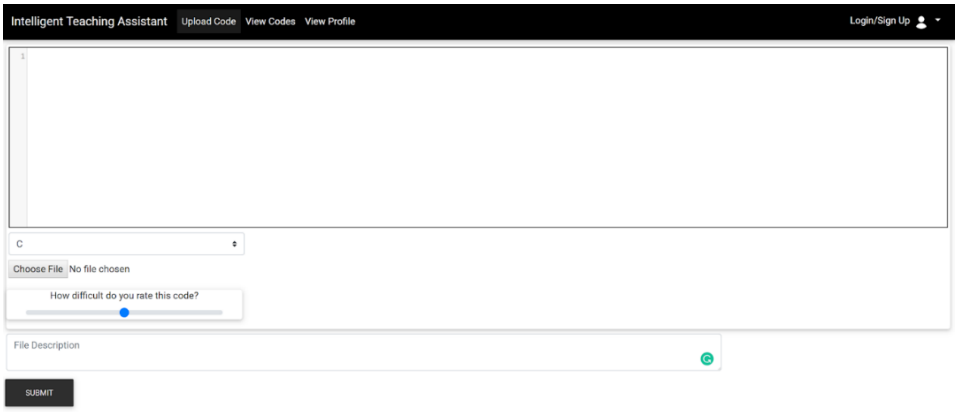

Fig 7.9 User Interface diagram

7.4 Packaging and Deployment Diagram

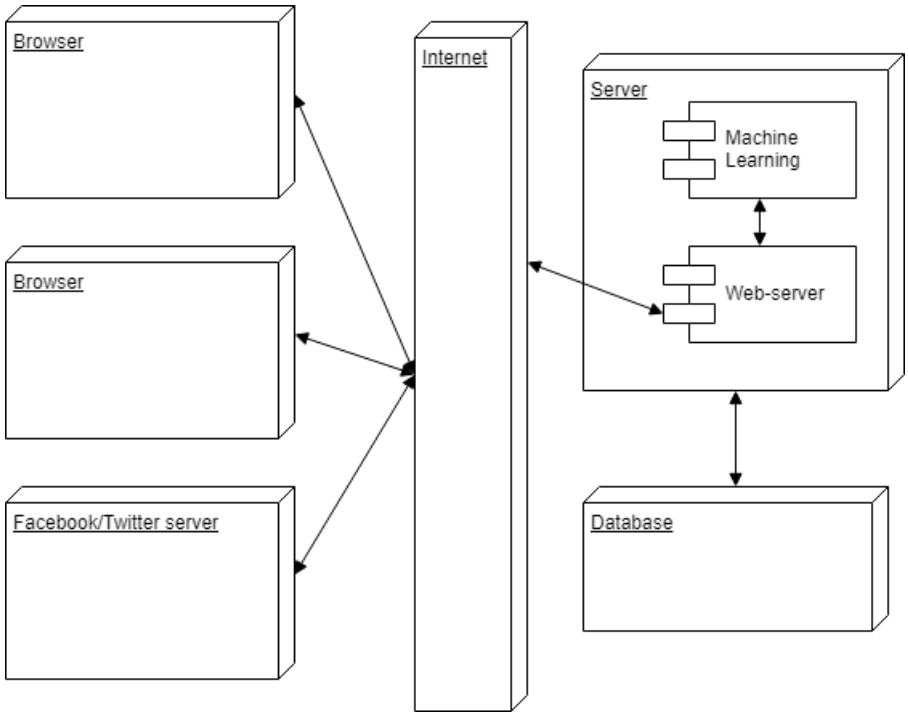

Fig. 7.10 Packaging and Deployment Diagram

The system is packaged as a web application and is served to the users as a web page. The users are provided with a Universal Resource Locator (URL) which provides the route to connect to the server. The deployment of the application is through a server at the back-end, which resides in the developer's environment. The HTML files for the web app, and the codes that the user wishes to view are served up through this server. The database also resides in the developer's environment and is connected to the server.

As the web-app has no dependencies on the system Operating System(OS) or the underlying hardware, there is no requirement to package different versions of the system

Updates, revisions, and patches to the system are made at the server side, and will reflect to the users on any connection to the server thereafter.

7.5 Alternate Design Approach

Some of the alternate design approaches considered were:

- Making a mobile application: One of the design approaches to the system is to make a mobile application that is accessible on the popular platforms (Android, iOS). The downside of this design approach is that the app has to be coded separately for each platform. This takes up unnecessary resources and time and would thus not be a good approach. Web applications, on the other hand, can be accessed on a wide variety of applications and are hence more suited for accessibility and availability
- Making a standard HTML page: A design approach is to make a standard HTML page for the application without using any external frameworks. This allows full control over the page and allows for custom styling. The disadvantage with this approach is that it leads to a lot of code being re-written from scratch. This is redundant as most are already implemented in popular frameworks. Thus this design approach discourages reuse of code, and hence is not a good design approach

for this case. Frameworks like Bootstrap also have components that scale automatically given the size of the window, allowing the system to be used seamlessly on mobile devices as well as personal computers

7.6 Reusability Considerations

The project has been designed as a stand-alone project and reusability has not been a priority in the design of the system. The components of the system that could be reused are:

- The machine learning model: The machine learning model to generate comments for C code can be reused to train on other programming languages using transfer learning
- The conversations: The conversations module of the system can be reused in many other applications that have this functionality
- The different features for tag generation, comment generation, searching, adaptive difficulty for collaborative filtering can be used by various applications that want to adopt this functionality.

CHAPTER – 8

IMPLEMENTATION AND PSEUDO CODE

This section of the document gives insights on the coding guidelines which were followed. It also gives information regarding the coding languages that were used and provides the list of all modules implemented along with the pseudo-code of the main functions that is present within each module

8.1 CODING GUIDELINES

The standard coding guidelines associated with the languages that were used for the project were followed. The code was linted and formatted for easy readability and understandability. The collaboration on the project was done using GitLab, a git based web platform that allows for collaboration between users on a project. A protected master branch was use, with development occurring on separate branches which were then merged to the master branch

8.2 CODING LANGUAGES

The main programming languages that were used in the development of this project are Python, HTML and JavaScript. The proportions of the languages used, as shown by Gitlab statistics for the project repository, are Python - 66.78% and HTML/JS - 28.47%

8.3 CODE AND PSEUDOCODE

8.3.1. Front End Module

The front-end module consists of HTML code, with bootstrap as a framework, and JavaScript code to perform the actions. The HTML code renders the page that is displayed on the browser to the

user. The code of some of the important JavaScript functions that have been implemented are provided below:

8.3.1.1 Login

```
function login()
{
  email = document.getElementById("email").value;
  passw = document.getElementById("passw").value;
  var xhr = new XMLHttpRequest();
  xhr.open("POST","/login");
  xhr.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
  xhr.onreadystatechange = function()
  {
        if(xhr.readyState == 4 && xhr.status == 200 )
        {
           if(JSON.parse(xhr.response).auth==true)
           {
                    document.location.reload();
           }
           else{
           document.getElementById("passw").style.background = "red";
           }
        }
  }
  postvars = "email="+email+"&password="+passw+"&operation="+operation;
  xhr.send(postvars);
}
```

8.3.1.2 Search

```
function searchCont()
{
 inp = document.getElementById('search').value;
 lang = document.getElementById('lang_op').value;
 diff = document.getElementById('diff_op').value;

 var xhr = new XMLHttpRequest();
 xhr.open("POST", "http://localhost:5000/search");
 xhr.setRequestHeader("Content-type","application/x-www-form-urlencoded");

 xhr.onreadystatechange = function()
 {
  if(xhr.readyState == 4 && xhr.status == 200 )
        {
           results = JSON.parse(xhr.response).files;
           resources = document.createElement("div");
           resources.setAttribute("id","searchresources");

           for( var i=0; i<results.length ; i++)
           {
                 // add a new div with content
           }
           document.body.appendChild(resources);

               }
 }
```

```
 xhr.send("searchTerms="+inp+"&lang="+lang+"&difficulty="+diff);
}
```

8.3.2 Server Module

The server module serves the code from the back-end to the client or browser. It also acts as an interface between the browser and the database.

The generic pseudocode for the functionality of the server is:

```
@app.route("/path", methods=["GET"])
def function ():
  #perform some processing
   return render_template('template.html')
```

8.3.3 MCQ generation module

```
doc2vec_model = Doc2Vec.load("d2v.model")
v1 = doc2vec_model.infer_vector(list(filter(lambda x: x not in
stop,word_tokenize(row[0].lower()))), steps=1000)
similar_docs = doc2vec_model.docvecs.most_similar([v1], topn=len(doc2vec_model.docvecs))
options.append((row[0],1))
options_actual = [row[0]]
i = 1
while(len(options)!=4):
   if(comments[int(similar_docs[i][0])] not in options):
      options.append((comments[int(similar_docs[i][0])], 0))
      options_actual.append(comments[int(similar_docs[i][0])])
```

```
  i+=1
shuffle(options)
options_per_func.append((row[1], options, function))
```

8.3.4 Difficulty prediction module

```
for i in range(len(new_codes_dict)):
  if (difficulty_matrix[new_users_dict[username]][i] == 0):
     adaptive_score = 0
     count = 0
     for index in indices:
        if (difficulty_matrix[index][i] > 0):
           count += 1
        adaptive_score += difficulty_matrix[index][i]
     if (count > 0):
        code_difficulties.append(adaptive_score / count)
     else:
        code_difficulties.append(0)
  else:
     code_difficulties.append(difficulty_matrix[new_users_dict[username]][i])
  adaptive_difficulty_matrix[new_users_dict[username]][i] = code_difficulties[-1]
```

8.3.4 Code Recommendation module

```
for j in range(len(new_users_dict)):
  user_stats_dict[j] = 0
  for item in user_has_viewed:
    user_stats_dict[j] += user_codes_matrix[j][item]
users_ordered = sorted(user_stats_dict.items(), key=lambda kv: kv[1], reverse=True)[1:4]
for i in range(len(new_codes_dict)):
  if (i in user_has_viewed):
    continue
  code_to_recommend[i] = 0
  for user in users_ordered:
    code_to_recommend[i] += user_codes_matrix[user[0]][i]
codes_ordered = sorted(code_to_recommend.items(), key=lambda kv: kv[1], reverse=True)[1:]
```

8.3.5 Search module

```
for searchTerm in searchTerms:
  if not lang == '':
    cur.execute("SELECT code from CodeTags JOIN Codes WHERE tag like (?) AND lang =
(?) and code=filename", (searchTerm+'%',lang))
  else:
    cur.execute("SELECT code from CodeTags WHERE tag like (?)", (searchTerm+'%',))
  rows = cur.fetchall()
  for row in rows:
    if (row[0] not in files_that_match):
      files_that_match[row[0]] = 0
```

```
    files_that_match[row[0]] += 1
files_ordered = sorted(files_that_match.items(), key=lambda kv: kv[1], reverse=True)
```

8.3.6 Comment suggestion module

The comment suggestion module was implemented using deep artificial neural networks, which form a sequence to sequence attention model. Several models were created during the course of the project. The implementation was done on the Keras framework, which works with a Tensorflow backend, and is available in Python. The steps for the process of preparing data, training, and prediction is specified below:

1. Gather the data of functions of C code and comments associated by reading the text files.
2. Clean the code data by removing redundant parts
3. Convert the code into AST, split per function
4. Convert the AST into a vector representation
5. Convert the comments into a one-hot representation based on the vocabulary. Denote start and end of sentence with special tokens
6. Train the deep attention based sequence to sequence model with the code vectors as inputs, and the comment vectors as output
7. Save the model, the weights, and the mapping
8. Predict for new code by converting to vector, and calling the function predict() on the neural network model
9. Use the mapping to convert the one-hot vector mapping to the associated word to get the comment suggestion sentence

CHAPTER – 9

TESTING

Testing this project has been performed at many stages of the development and has been performed in a different form for each of the modules. The front-end has been evaluated by preparing a table of expected and observed actions and responses for a set of scenarios. The backend functions have been tested to see if they return the correct type and expected output.
The intelligent modules have been tested by manually examining the results, and by using scores on test data.

9.1 Front end

Scenario	Expected Outcome	Observed Outcome
1. Login		
Enter an email address that exists in the database	Response from checkUser() call	200 OK from checkUser() call
Enter an email address that does not exist in the database	Unsuccessful response from checkUser() call, change to sign-up	200 OK with unsuccessful from checkUser() call, add new field for confirm password, change button to SignUp
Login with correct email and password	Login success, reflect in navbar	Login success, navbar shows username
Login with correct email and incorrect password	Login fail, notify user	Login fail, turn password field red

2.Navbar		
Click on current item	No change	No change
Click on other item	Navigate to appropriate page	Navigate to appropriate page
Click on login button	Dropdown	Dropdown

Table 9.1 Front end test cases

All buttons were tested and found to be working correctly.

9.2 Comment Suggestion

The comment generation neural networks were tested with the test data by splitting the existing data into training and testing data in a ratio of 9:1. The BLEU score metric was used to evaluate it and achieved a score of 15% on test data and 70% on training data.

9.3 Other Intelligent Models

Tagging, Difficulty Prediction and Code Recommendation were tested manually and were found to reasonable results. Difficulty Prediction and Code Recommendation were verified by testing with different mock users.

9.4 Server

Scenario	Expected Outcome	Observed Outcome
Upload code in C language	Tags and Comments are populated	Tags and comments are populated
C function with comments	Comments are same as those in code	Comments are same as those in code
C function without comments	Comments generated and suggested	Comments generated and suggested
Code in any other language except C	Empty Tags and Comments Lists	Tags = [] Comments = []

Table. 9.2 Server test cases

CHAPTER – 10

RESULTS AND DISCUSSIONS

The project has successfully led to the development of a system that serves as an intelligent teaching assistant to teachers for teaching programming concepts. The results of the project are:

1. Multiple Choice questions are created from code snippets and the comments associated with each function. Good quality distractors are created in an intelligent manner and are found to be close to the given correct answer, while not being the same. Teachers can use the tool for assignments with MCQs of the form “What does the <code> do?”, with the correct answer extracted from the comment, and the distractors generated dynamically.
2. Tags are automatically generated for code in the C language. The tags are found to be reasonably accurate and usable. Tags enable search of code that has been uploaded, requiring little to no manual intervention. This allows users to upload code which become searchable by others with minimal effort by the uploader
3. Codes are recommended to each user based on his/her interests. Users are able to view the codes that are the most relevant to them on visiting the application
4. Difficulty is predicted for each code per user, which is dynamically generated as a prediction of the likely perception of the difficulty of a particular code per user. The users can find the most appropriate code based on the difficulty levels associated with them. The difficulty level prediction is personalized to each user and adapts to their perception of difficulty.
5. Codes are searchable through tags, and ranked in order of relevance, allowing teachers to find the most relevant code in a particular concept or topic. The search results can be filtered on language and difficulty level
6. Comments are suggested for functions in the C language that do not have comments provided. The comments generated are found to be reasonably accurate in the domain of Introductory Programming. Due to the limited dataset available, the model performs best

on a certain set of codes. The preparation of a larger and more diverse dataset is out of the scope of this project, as it is a product-based project. The BLEU score has been used as a metric for evaluating the performance, and a score of 15% has been obtained on test data, and a score of 70% on training data. Some examples were found where the expected comment was semantically the same as the generated comment, but was different in the terms of a distance measure between words of a sentence . State-of-the-art systems, which predict comments for Java code, have demonstrated a score of 20-30%. As our project is not a research-oriented project, and does not have a large dataset, and due to the structural differences in C code, the observed results have been lower. As users use the system, the model should display better results

7. The project provides a platform for teachers in diverse geographic locations to collaborate, contribute and gather resources. The conversations associated with the code snippets are used by teachers to have a clearer understanding of the codes.

CHAPTER – 11

SNAPSHOTS

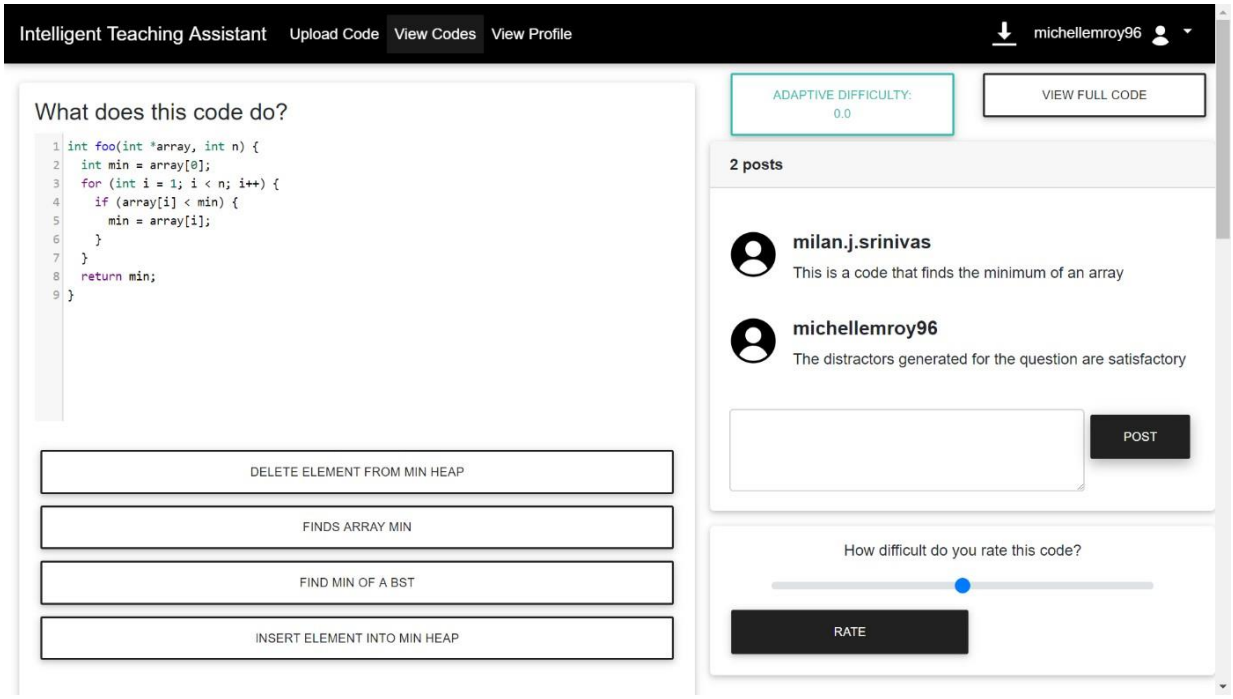

Fig 11.1 Code View Page

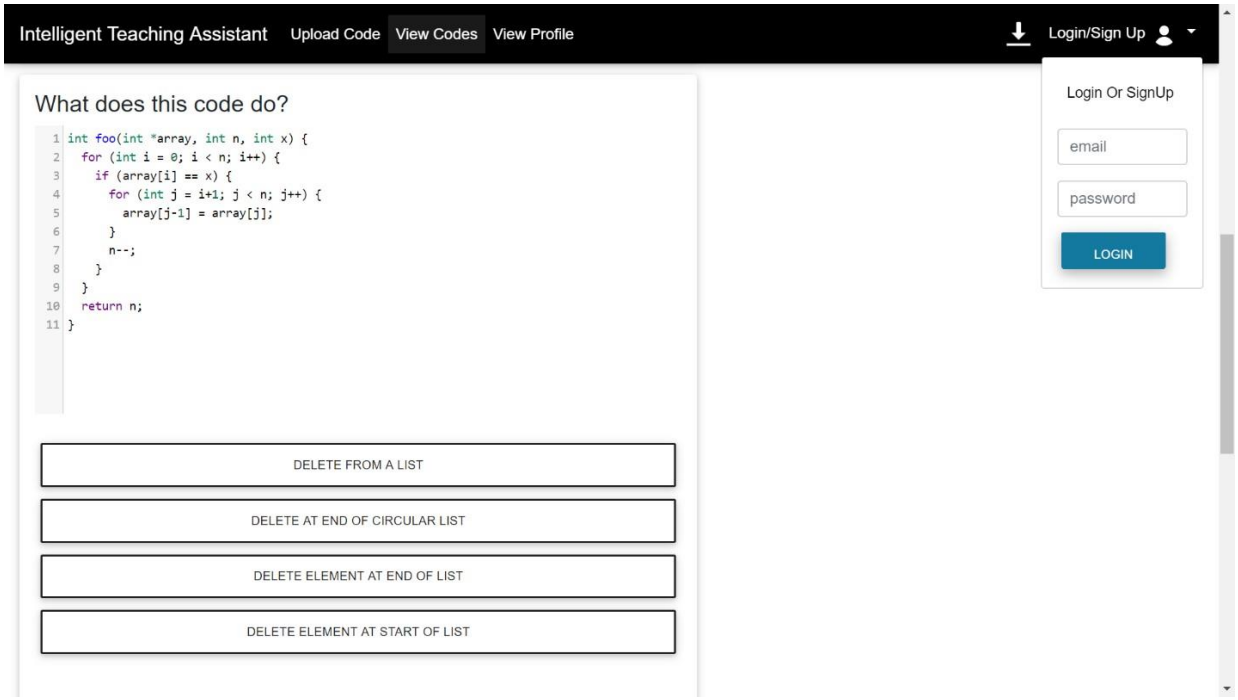

Fig 11.2 Login View

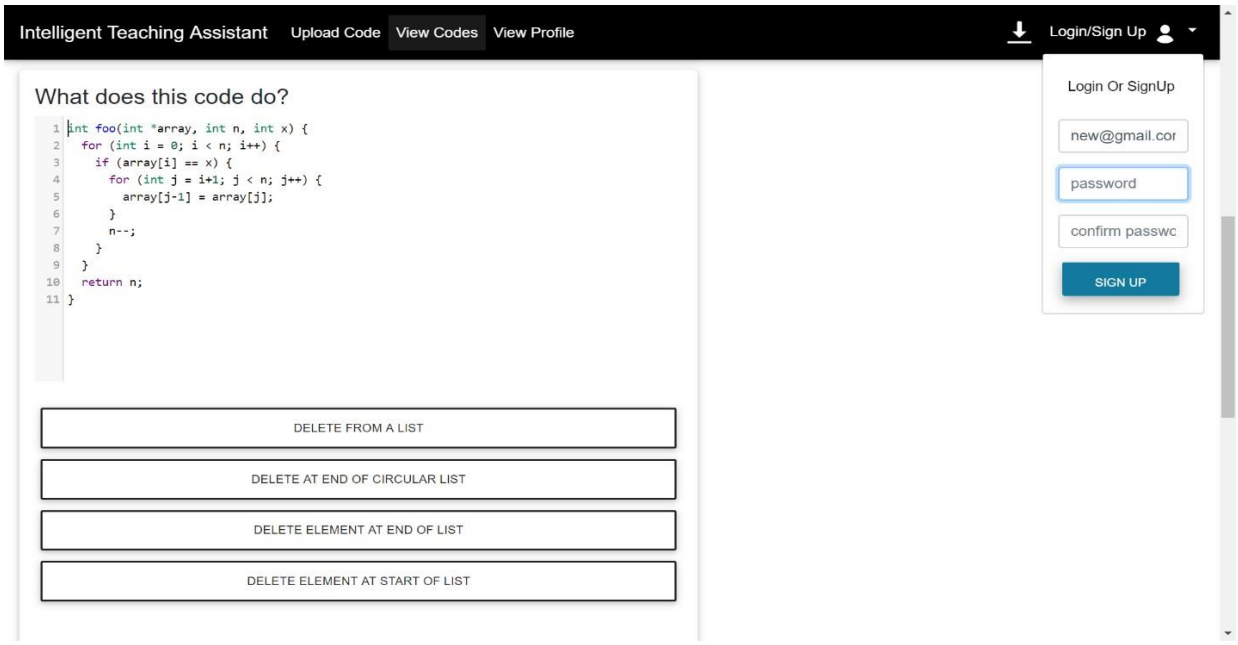

Fig 11.3 Sign up View

Fig 11.4 Code upload editor view

Fig 11.5 Code Upload tags and comments suggestion example view

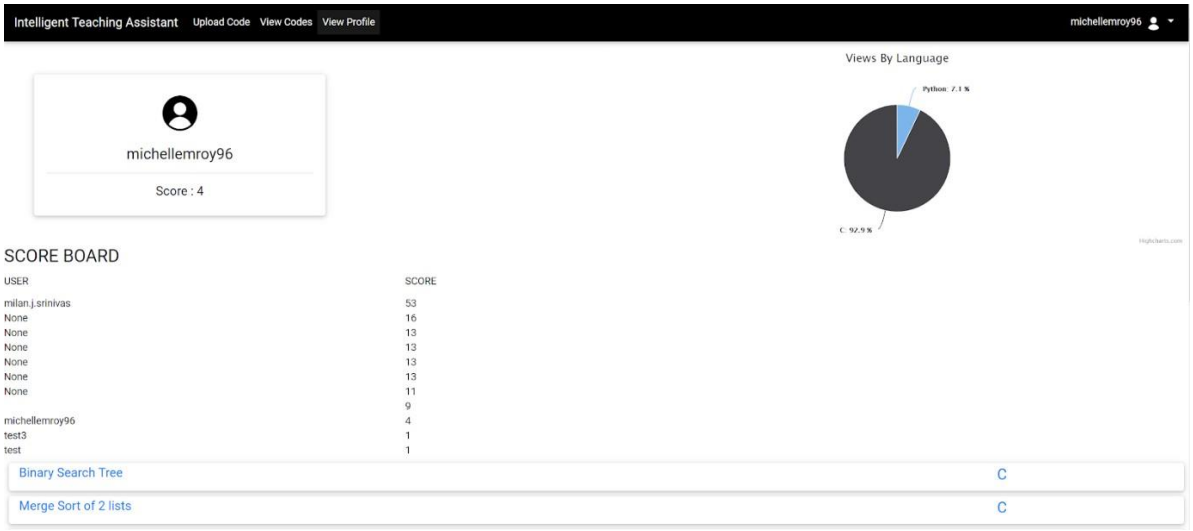

Fig 11.6 User Profile example view

CHAPTER – 12

CONCLUSIONS

The outcome of this project - Intelligent Teaching Assistant for Programming Concepts - is a web application that helps teachers collaborate on resources, and use these resources for administering tests, in an intelligent manner.

The application lets users upload code to the database, which is shared among users. If the code is in the C language, tags are generated automatically, and comments are suggested for the functions in the code that do not have one provided by the uploader. The codes in the database are searchable by tags, and are recommended to each user based on his/her viewing patterns. The perceived difficulty of a given code by a user, both seen and unseen, are predicted by the application. Finally, Multiple Choice Questions are formed on each function of C code, and one per code for other languages. The correct answer is the associated comment, and the incorrect options are generated intelligently to be close to the actual answer.

CHAPTER – 13

FURTHER IMPROVEMENTS

There are some improvements that we would like to implement in the future for this project.

The first improvement we suggest is to integrate a social media platform into our platform tool so that it is more accessible to users. This would allow users to login via their social media credentials. Our platform should also be able to post questions on our social media page and users can interact on the page. These interactions will be reflected on our website. A suitable application to integrate it with would be Facebook. In order to do this our website must be verified and approved after it is hosted.

Another improvement is an additional feature for providing assignments. Allowing for an assignment mode for teachers will allow them to conduct tests and evaluate student son our platform. This assignment mode will allow teachers to choose MCQs or some code resource to be assigned to a set of users. The students will then answer the questions and be rewarded points accordingly. The teacher will be able to view the performance of his/her student groups in the form of collected scores. This would help integrate our tool into classroom teaching better.

We also hope to be able to conduct surveys of our generated MCQs with human validators as a measure of accuracy. We will test out the quality of our distractors and improve it based on feedback.

REFERENCES

[1] DataUSA https://datausa.io/profile/cip/110701/#employment 2019

[2] Aspiring Minds, “National Employability Report Engineers Annual Report”, 2019

[3] Salon (John Barr, Ananda Gunawardena) https://dl.acm.org/citation.cfm?id=2157196 (SIGCSE'12 conference, February 27th - March 2nd 2019, Minneapolis, Minnesota, USA)

[4] Piazza (Pooja Sankar) https://piazza.com/product/overview

[5] Deep Code Comment Generation by Xing Hu et al. (2018) for automatic generation of comments in Java Code from Abstract Syntax Tree using Natural Language Techniques.

[6] Automatic Generation of Multiple Choice Questions using Surface-based Semantic Relations by Naveed Afzal

[7] Automatic Generation of Multiple Choice Questions Using Wikipedia by Arjun Bhatia et al.

[8] Python Flask : http://flask.pocoo.org/

[9] Keras : https://keras.io/

[10] Facebook API : https://developers.facebook.com/docs/apis-and-sdks/

APPENDIX A: DEFINITIONS, ACRONYMS AND ABBREVIATION

- HTML : HyperText Markup Language
- Code Snippets : The pieces of code that a user uploads onto the platform
- Tags : Keywords associated with a concept or a topic used for classification and ease of searching
- MCQ : Multiple Choice Questions
- Comments : Code comments, provided to explain the code
- Conversations: Discussions that happen among users around a snippet of code
- Web-based : A system that is viewed using a web browser and is connected to via the World Wide Web
- XHR : XML Http Request
- BLEU score : Bilingual Evaluation Understudy score