CS-257
DBIS-Project

CODE MANAGEMENT

Submitted by: Sundesh Gupta (180001057)
Aniket Sangwan (180001005)

Introduction

Computer Science students have a number of files which include hundreds of lines of code. Our aim is to develop a database where students collaborate and share various codes related to their curriculum or any other code that a student wants to share with his/her fellow students. This will increase discussion about practical work between professors, instructors and students leading to better application of theoretical knowledge gained from lectures. For example, implementation of a data structure in a particular programming language such as C++, Python, Lua, Clojure, java or a code for a webpage (or template) in HTML, javascript, PHP that can be reused by other programmers/students. If a student studying an Algorithms course wants to know the implementation of an algorithm in a certain language, he will be able to find it in the database. Students can also share their course projects which can be used by their classmates and juniors to get deeper insights into the project. If a student wants to send an HTTP request using javascript, he can find the required function and can easily incorporate in his/her project. Professors could be provided with special permissions to edit or delete any code. Moreover, coders can search, filter and find any other required code that is available in the database. The rating system will help rate and sort the code. This will help the students to check the quality of their code and improve their code readability and reusability and will also be a really helpful tool for beginners.

Each user will have to login in order to view any code or add/delete an existing code that he/she added previously. Each article will be associated with tags and course name. Tags will be used for searching and ratings will be used for sorting. Course name will help in direct navigation to the required code. The articles will provide users with an option to add comments to eliminate errors and promote discussion among students and professors.

Purpose

Our database is a storehouse of codes written in different programming languages and the various attributes encapsulated in it. To implement the same idea in a large project or in a different language, it is time-consuming to type it every time. Our database management project aims to provide a platform to store pre-written codes for easy access and modification and to gather reviews.

This tool provides for:

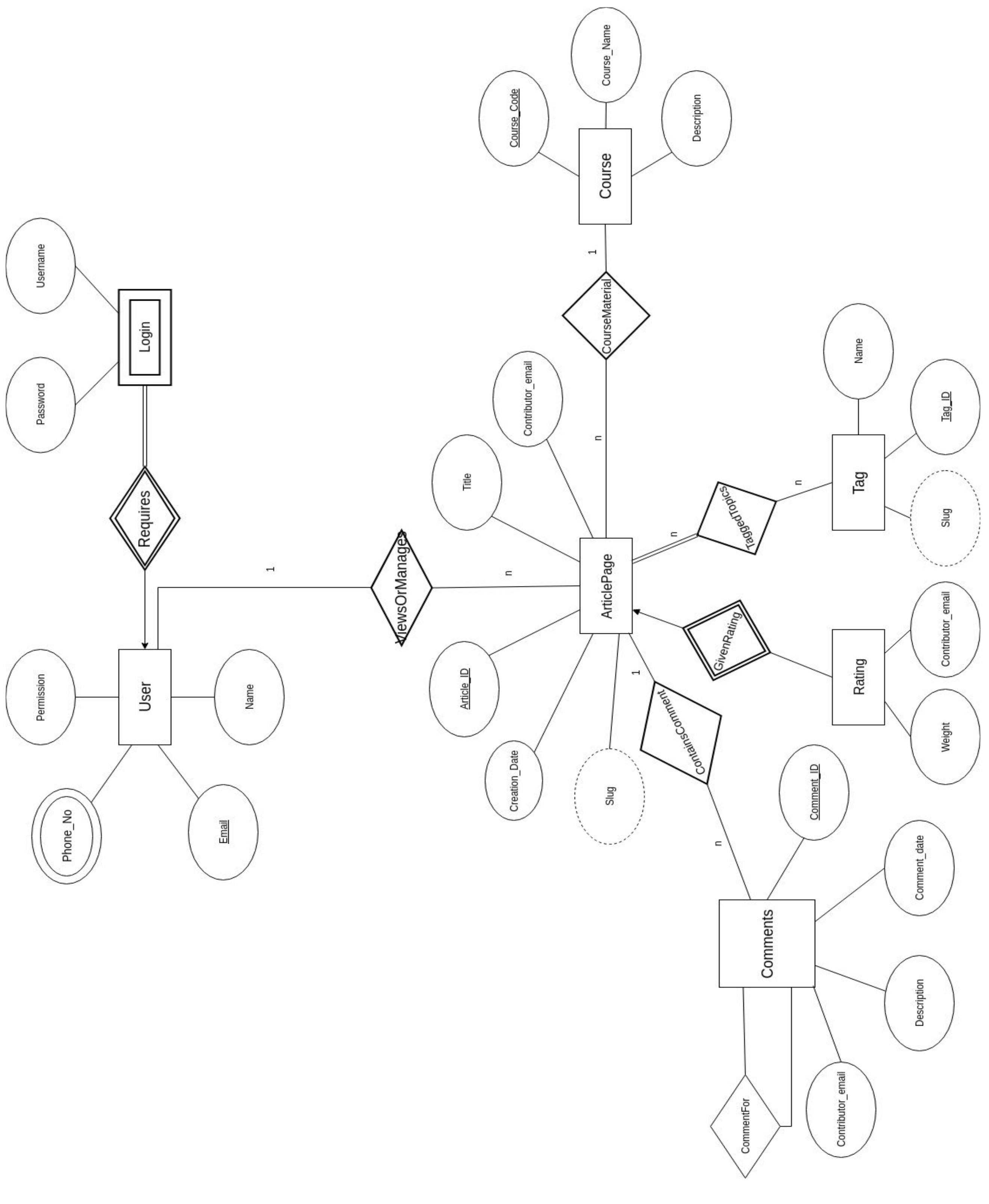
1. Storage of codes, and its maintenance in a consistent manner.
2. Easy to use interface, with menus, for clearer navigability.
3. Rating of every article with the ability to filter using tags.
4. A simple platform for code sharing and related discussion.
5. Increased gain of practical knowledge as it leads to increased interaction between faculty and students.

The tool differentiates between two types of users: One category is of viewers who can view and review every article and add comments to them. The administrative user is another category who can modify, add or delete codes from their own accounts.

<u>ER Analysis (Identifying Entity Sets and Relationship Sets)</u>

1. User (Entity)
2. Login (Entity)
3. ArticlePage (Entity)
4. Comment (Entity)
5. Rating (Entity)
6. Tag (Entity)
7. Course (Entity)
8. Requires (Relation between User and Login)
9. ViewsOrManages (Relation between User and ArticlePage)
10. ContainsComment (Relation between ArticlePage and Comments)
11. GivenRating (Relation between ArticlePage and Rating)
12. TaggedTopics (Relation between ArticlePage and Tag)
13. CourseMaterial (Relation between ArticlePage and Course)
14. CommentFor (Recursive Relationship in Comments)

ER-Diagram


Course
Course_Code
Course_Name
Description
CourseMaterial
1
n
Login
Username
Password
Requires
User
Permission
Name
Phone_No
Email
ViewsOrManages
1
n
ArticlePage
Title
Contributor_email
Article_ID
Creation_Date
Slug
TaggedTopics
n
n
Tag
Name
Tag_ID
Slug
GivenRating
Rating
Contributor_email
Weight
ContainsComment
1
n
Comments
Comment_ID
Comment_date
Description
Contributor_email
CommentFor

<u>Transformation of ER diagrams into set of Tables:</u>

1. **User**

```
CREATE TABLE User
(
        Name        VARCHAR(25),
        Email       VARCHAR(50),
        Permission INT,
        PRIMARY KEY (Email)
);
```

2. **PhoneNoDetails**

```
CREATE TABLE PhoneNoDetails
(
        Email         VARCHAR(50),
        Phone_no      VARCHAR(10),
        PRIMARY KEY (Email, Phone_no),
        FOREIGN KEY (Email) REFERENCES User(Email)
);
```

3. **Login**

```
CREATE TABLE Login
(
        Email      VARCHAR(50),
        Password  VARCHAR(25),
        Username VARCHAR(10),
        PRIMARY KEY (Email, Password, Username),
        FOREIGN KEY (Email) REFERENCES User(Email)
);
```

4. **ArticlePage**

```
CREATE TABLE ArticlePage
(
        Article_id             INT AUTO_INCREMENT ,
        Title                  VARCHAR(255),
        Creation_date          TIMESTAMP,
        Contributor_email VARCHAR(50),
        PRIMARY KEY (Article_id),
        FOREIGN KEY (Contributor_email) REFERENCES User(Email)
);
```

5. **Comment**

```
CREATE TABLE Comment
(
        Comment_id           INT,
        Contributor_email VARCHAR(50),
        Comment_date         TIMESTAMP,
        Description          VARCHAR(255),
        PRIMARY KEY (Comment_id)
);
```

6. **Rating**

```
CREATE TABLE Rating
(
        Article_id           INT,
        Weight               INT,
        Contributor_email VARCHAR(50),
        PRIMARY KEY (Article_id,  Weight, Contributor_email),
        FOREIGN KEY (Article_id)  REFERENCES ArticlePage(Article_id)
);
```

7. **Tag**

```
CREATE TABLE Tag
(
        Tag_id INT AUTO_INCREMENT,
        Name  VARCHAR(25),
        PRIMARY KEY (Tag_id)
);
```

8. **Course**

```
CREATE TABLE Course
(
        Course_code  VARCHAR(10),
        Description   VARCHAR(255),
        Course_name VARCHAR(50),
        PRIMARY KEY (Course_code)
);
```

9. ViewsOrManages

```
CREATE TABLE ViewsOrManages
(
        Email      VARCHAR(50),
        Article_id INT,
        PRIMARY KEY (Article_id),
        FOREIGN KEY (Email)  REFERENCES User(Email),
        FOREIGN KEY (Article_id)  REFERENCES ArticlePage(Article_id)
);
```

10. ContainsComment

```
CREATE TABLE ContainsCommment
(
        Comment_id  INT,
        Article_id     INT,
        PRIMARY KEY (Comment_id),
        FOREIGN KEY (Comment_id)  REFERENCES Comment(Comment_id),
        FOREIGN KEY (Article_id)  REFERENCES ArticlePage(Article_id)
);
```

11. TaggedTopics

```
CREATE TABLE TaggedTopics
(
        Tag_id     INT,
        Article_id INT,
        PRIMARY KEY (Tag_id, Article_id),
        FOREIGN KEY (Tag_id) REFERENCES Tag(Tag_id),
        FOREIGN KEY (Article_id)  REFERENCES ArticlePage(Article_id)
);
```

12. CourseMaterial

```
CREATE TABLE CourseMaterial
(
        Course_code VARCHAR(10),
        Article_id      INT,
        PRIMARY KEY (Article_id),
        FOREIGN KEY (Course_code)  REFERENCES Course(Course_code),
        FOREIGN KEY (Article_id)  REFERENCES ArticlePage(Article_id)
);
```

13. CommentFor

```
CREATE TABLE CommentFor
(
        Comment_id       INT,
        CommentFor_id INT,
        PRIMARY KEY (CommentFor_id,Comment_id),
        FOREIGN KEY (Comment_id)  REFERENCES Comment(Comment_id),
        FOREIGN KEY (CommentFor_id) REFERENCESComment(Comment_id)
);
```

<u>**TRIGGERS:**</u>

```
CREATE TRIGGER After_Article_Insertion_ViewsOrManages
AFTER INSERT ON ArticlePage
FOR EACH ROW
BEGIN
INSERT INTO ViewsOrManages VALUES
(NEW.Contributor_email,NEW.article_id);
END$$
```

```
CREATE TRIGGER After_Article_Insertion_DATE
BEFORE INSERT ON ArticlePage
FOR EACH ROW
BEGIN
SET NEW.Creation_date = CURRENT_TIMESTAMP();
END$$

CREATE TRIGGER After_Article_Insertion_Rating
AFTER INSERT ON ArticlePage
FOR EACH ROW
BEGIN
INSERT INTO Rating VALUES (NEW.Article_id, 0, NEW.Contributor_email);
END$$

CREATE TRIGGER COMMENT_INSERT
BEFORE INSERT ON Comment
FOR EACH ROW
BEGIN
SET NEW.Comment_date=CURRENT_TIMESTAMP();
END$$

CREATE TRIGGER ARTICLE_DELETE
BEFORE DELETE ON ArticlePage
FOR EACH ROW
BEGIN
DELETE FROM Rating where Rating.Article_id=OLD.Article_id;
DELETE FROM ViewsOrManages where
ViewsOrManages.Article_id=Old.Article_id;
DELETE FROM TaggedTopics where TaggedTopics.Article_id=Old.Article_id;
DELETE FROM CourseMaterial where
CourseMaterial.Article_id=Old.Article_id;
```

```
DELETE FROM ContainsComment where
ContainsComment.Article_id=Old.Article_id;
END$$

CREATE TRIGGER COMMENT_DELETE
BEFORE DELETE ON Comment
FOR EACH ROW
BEGIN
DELETE FROM CommentFor where
CommentFor.CommentFor_id=Old.Comment_id;
DELETE FROM ContainsComment where
ContainsComment.Comment_id=OLD.Comment_id;
END$$
```

PROCEDURES:

```
CREATE PROCEDURE get_email_from_username(uname varchar(25))
BEGIN
SELECT Email FROM Login where Login.Username=uname;
END$$

CREATE PROCEDURE get_user_data(usersname varchar(25))
BEGIN
SELECT PhoneNoDetails.Phone_no,T.Email_add,T.uname FROM
PhoneNoDetails inner join (SELECT Login.Email as Email_add,User.Name as
uname from Login inner join User on User.Email=Login.Email where
Login.Username=usersname) T on Email_add=Email;
END$$

CREATE PROCEDURE get_max_article_id()
BEGIN
SELECT MAX(Article_id) FROM ArticlePage;
END$$
```

```
CREATE PROCEDURE get_max_comment_id()
BEGIN
SELECT MAX(Comment_id) FROM Comment;
END$$

CREATE PROCEDURE get_tag_id_from_tag_name(tag_name varchar(25))
BEGIN
SELECT Tag_id FROM Tag where Name = tag_name;
END$$
```

DEPENDENCIES AND STEPS TO RUN:

The Project Uses:

1. MySql (8.0.18)
2. HTML 5
3. Flask Framework (Python)
4. CSS
5. Javascript

How to Run the Project:

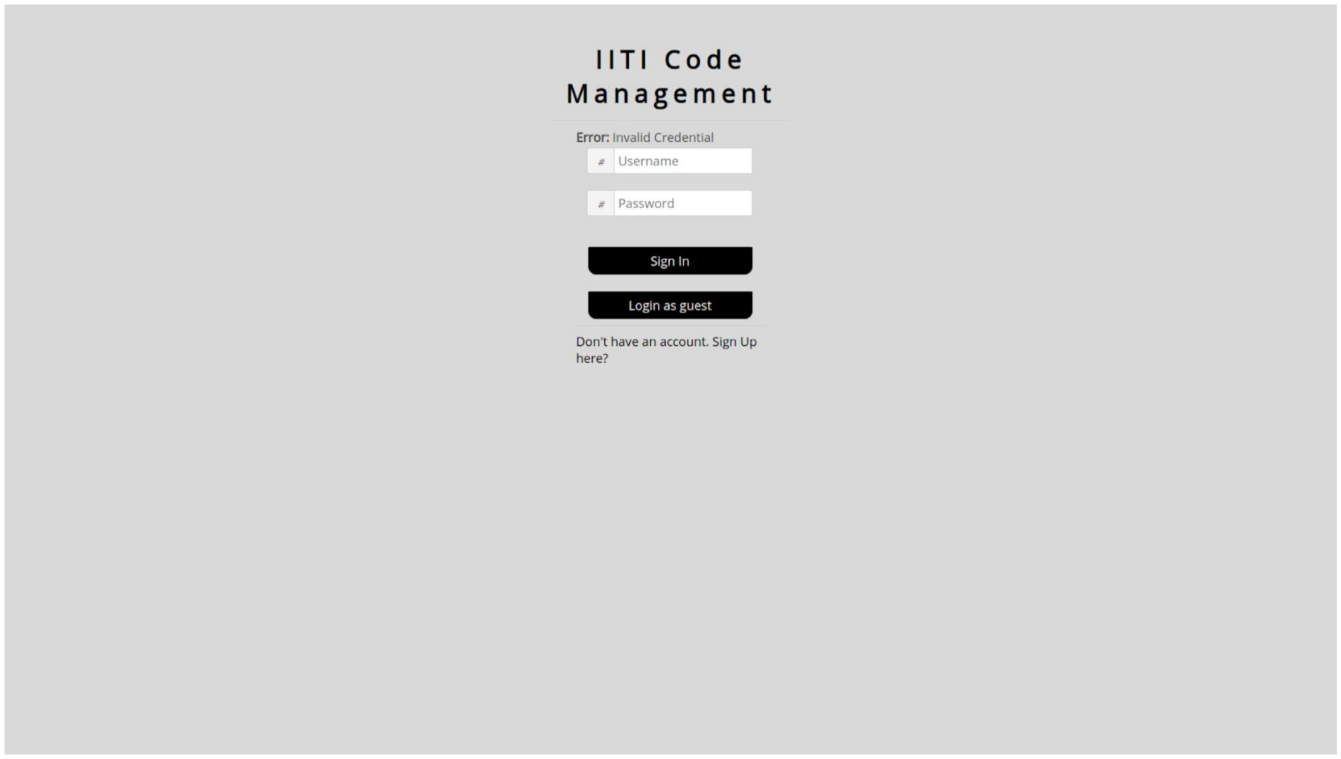
Step 1. Making the database.
Import Tables.sql and Tables_data.sql in database "dbms_project" and grant permission to user “aniket”.

Step 2. Run mysql server.

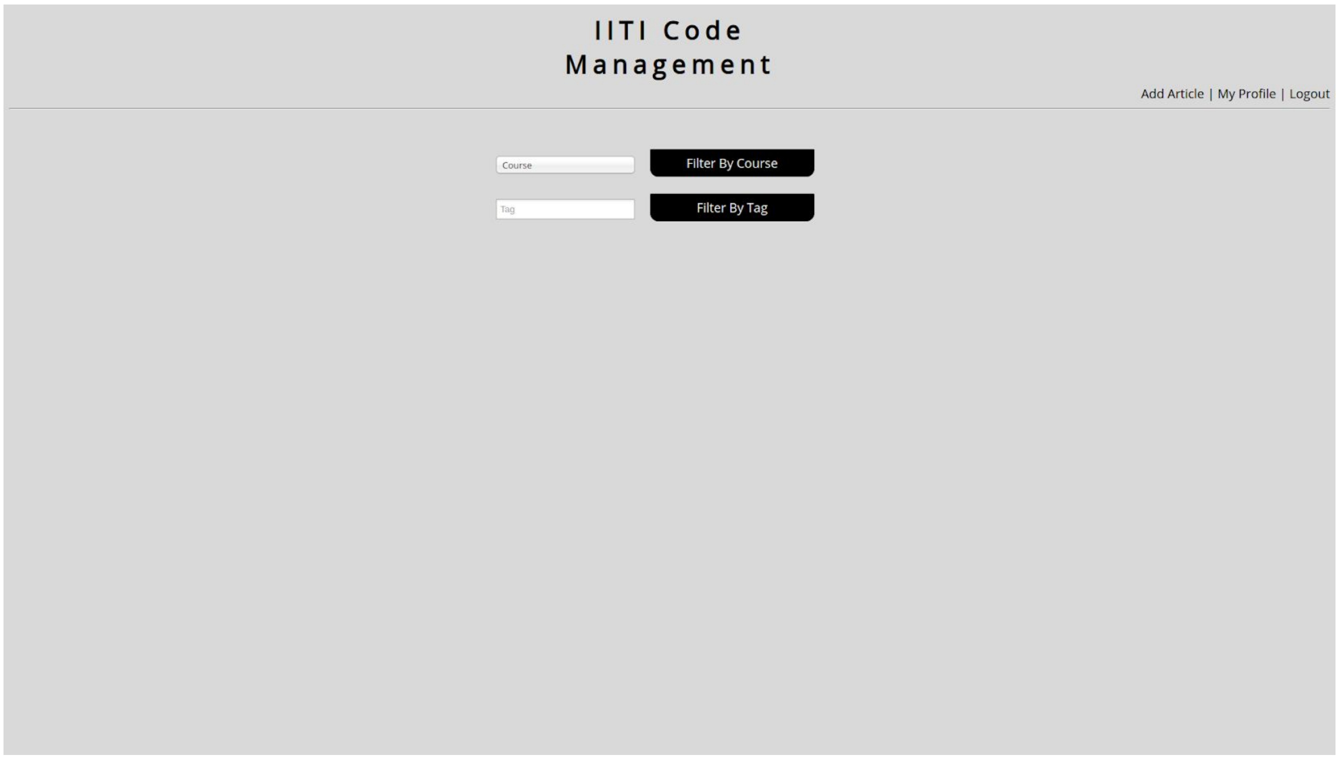
Step 3. Run app.py.

SCREENSHOTS:

Login Page:


IITI Code Management
Error: Invalid Credential
Username
Password
Sign In
Login as guest
Don't have an account. Sign Up here?

Homepage:

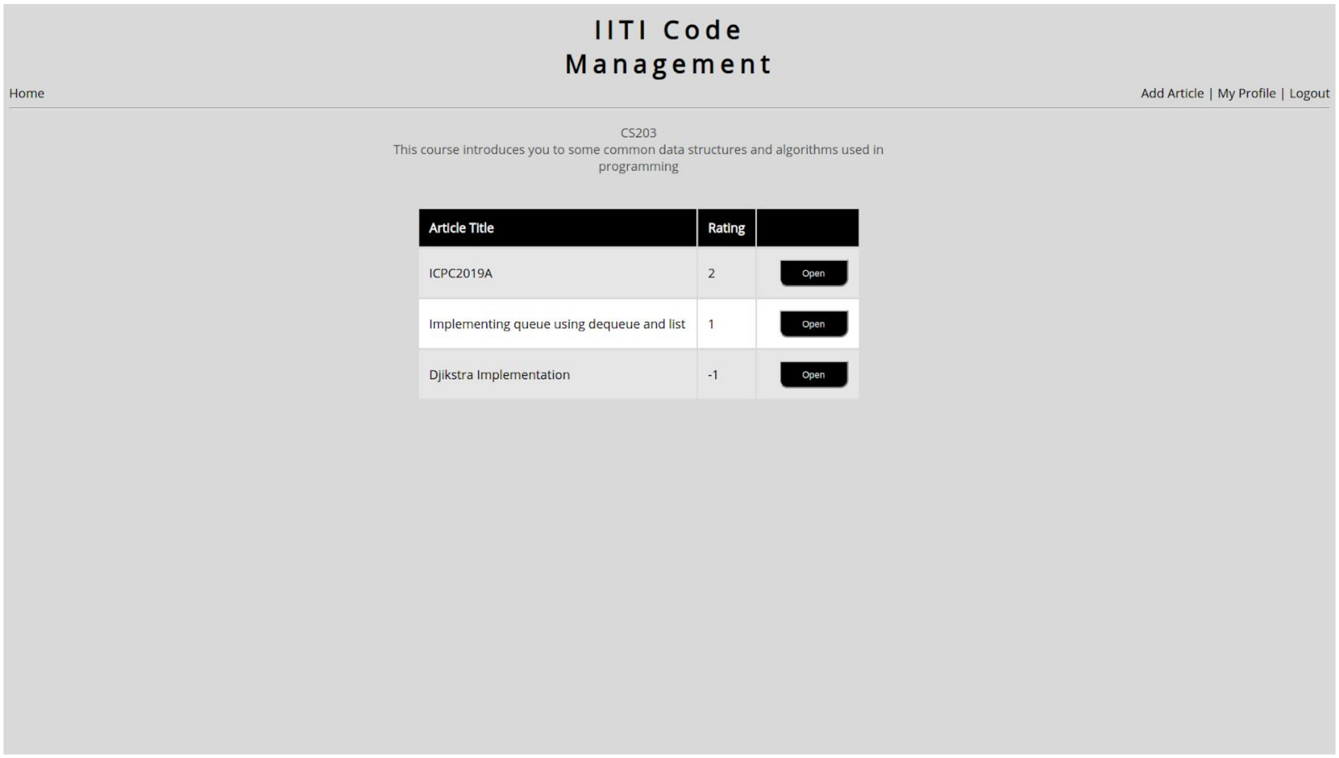

IITI Code Management

Add Article | My Profile | Logout

Course — Filter By Course

Tag — Filter By Tag

Filter By Course or Tag Order By decreasing rating:


IITI Code Management

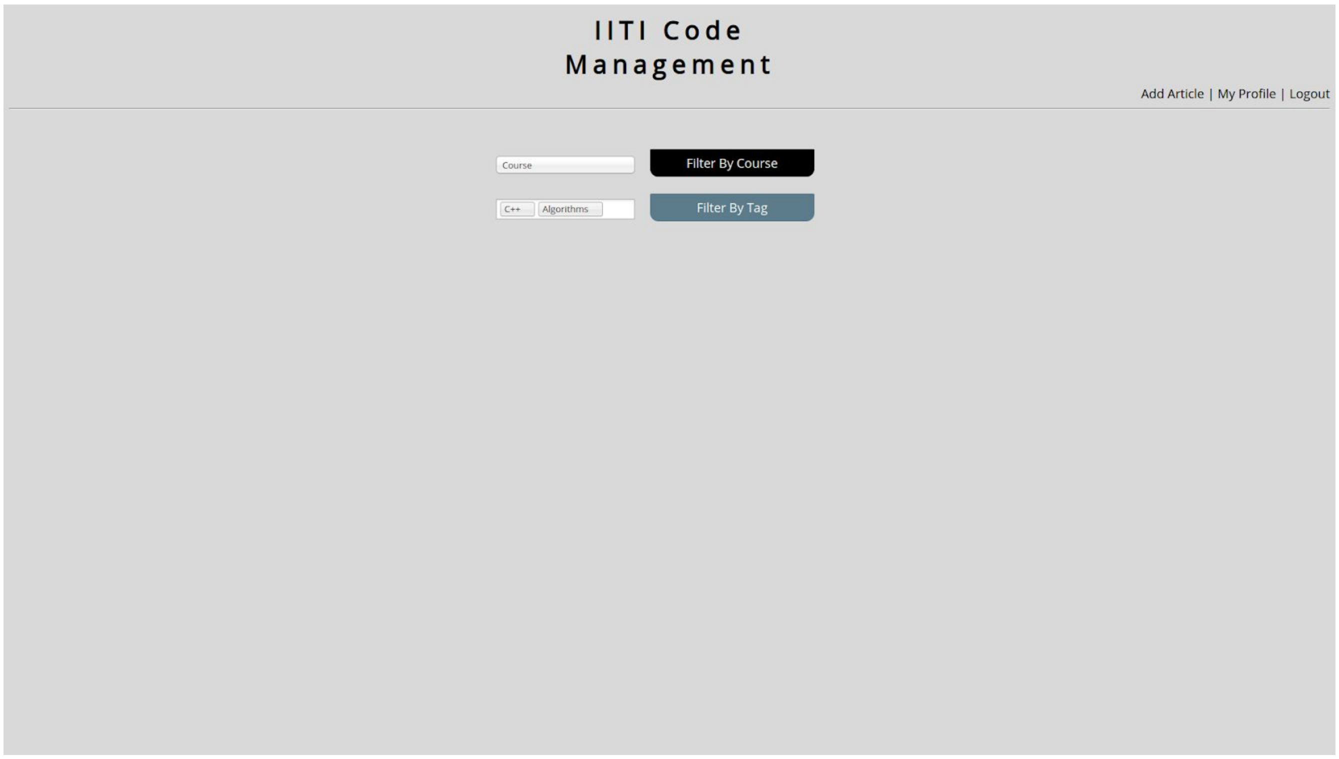
Home

Add Article | My Profile | Logout

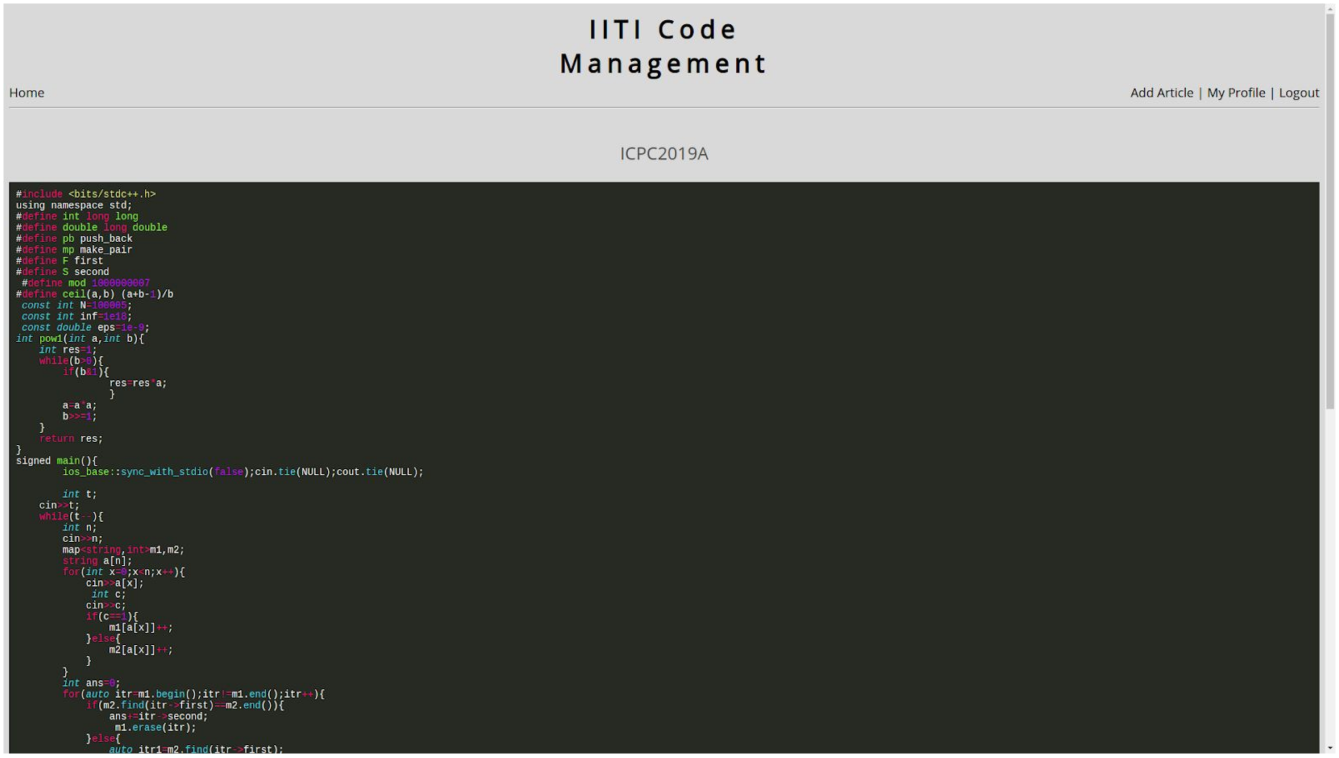
CS203
This course introduces you to some common data structures and algorithms used in programming

Article Title	Rating	
ICPC2019A	2	Open
Implementing queue using dequeue and list	1	Open
Djikstra Implementation	-1	Open

Optional Filtering by Multiple Tags:



View Article Page with Comments and Rating:



IITI Code Management

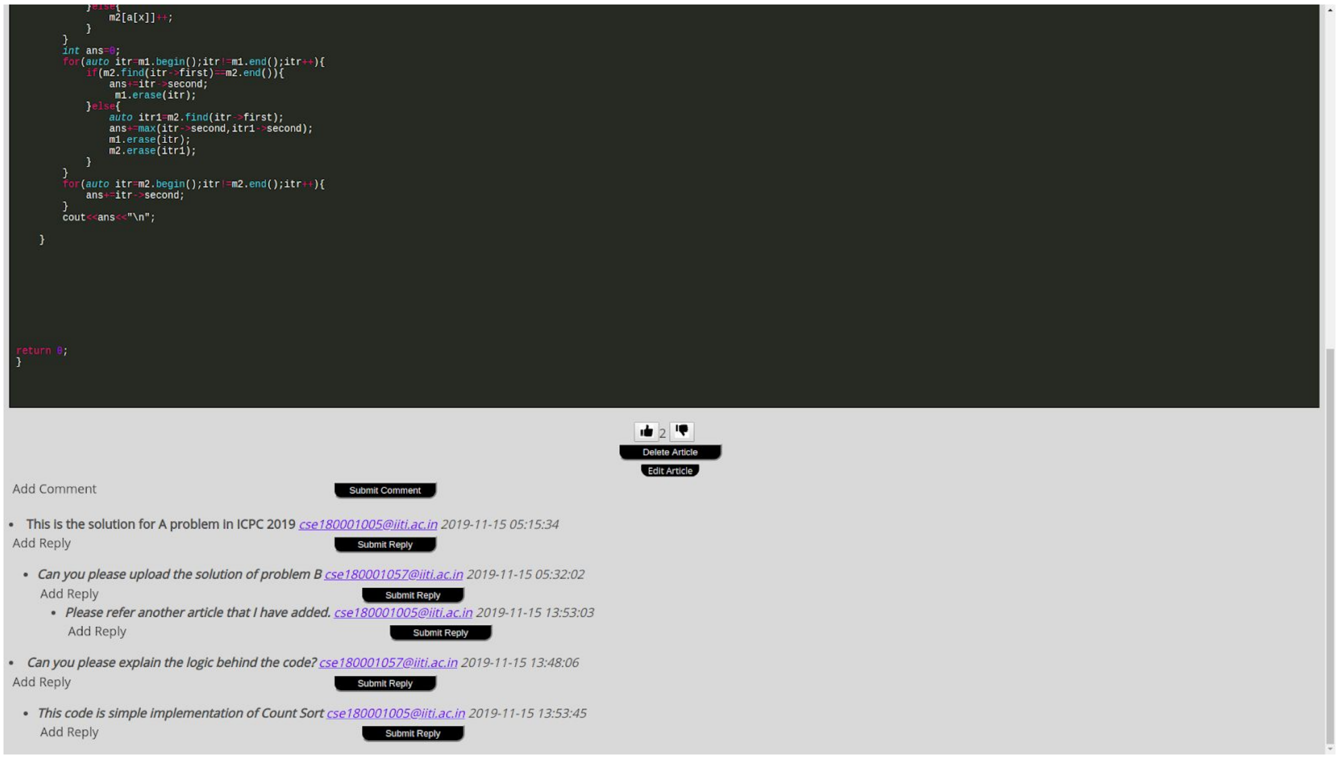
Home

Add Article | My Profile | Logout

ICPC2019A

```
#include <bits/stdc++.h>
using namespace std;
#define int long long
#define double long double
#define pb push_back
#define mp make_pair
#define F first
#define S second
 #define mod 1000000007
#define ceil(a,b) (a+b-1)/b
 const int N=100005;
 const int inf=1e18;
 const double eps=1e-9;
int pow1(int a,int b){
    int res=1;
    while(b>0){
        if(b&1){
                res=res*a;
                }
        a=a*a;
        b>>=1;
    }
    return res;
}
signed main(){
        ios_base::sync_with_stdio(false);cin.tie(NULL);cout.tie(NULL);

        int t;
    cin>>t;
    while(t--){
        int n;
        cin>>n;
        map<string,int>m1,m2;
        string a[n];
        for(int x=0;x<n;x++){
            cin>>a[x];
             int c;
            cin>>c;
            if(c==1){
                m1[a[x]]++;
            }else{
                m2[a[x]]++;
            }
        }
        int ans=0;
        for(auto itr=m1.begin();itr!=m1.end();itr++){
            if(m2.find(itr->first)==m2.end()){
                ans+=itr->second;
                 m1.erase(itr);
            }else{
                auto itr1=m2.find(itr->first);
```



```
            }else{
                m2[a[x]]++;
            }
        }
        int ans=0;
        for(auto itr=m1.begin();itr!=m1.end();itr++){
            if(m2.find(itr->first)==m2.end()){
                ans+=itr->second;
                 m1.erase(itr);
            }else{
                auto itr1=m2.find(itr->first);
                ans+=max(itr->second,itr1->second);
                m1.erase(itr);
                m2.erase(itr1);
            }
        }
        for(auto itr=m2.begin();itr!=m2.end();itr++){
            ans+=itr->second;
        }
        cout<<ans<<"\n";

    }







return 0;
}
```

- This is the solution for A problem in ICPC 2019 cse180001005@iiti.ac.in 2019-11-15 05:15:34

  Add Reply Submit Reply

  - *Can you please upload the solution of problem B* cse180001057@iiti.ac.in *2019-11-15 05:32:02*

    Add Reply Submit Reply

    - *Please refer another article that I have added.* cse180001005@iiti.ac.in *2019-11-15 13:53:03*

      Add Reply Submit Reply

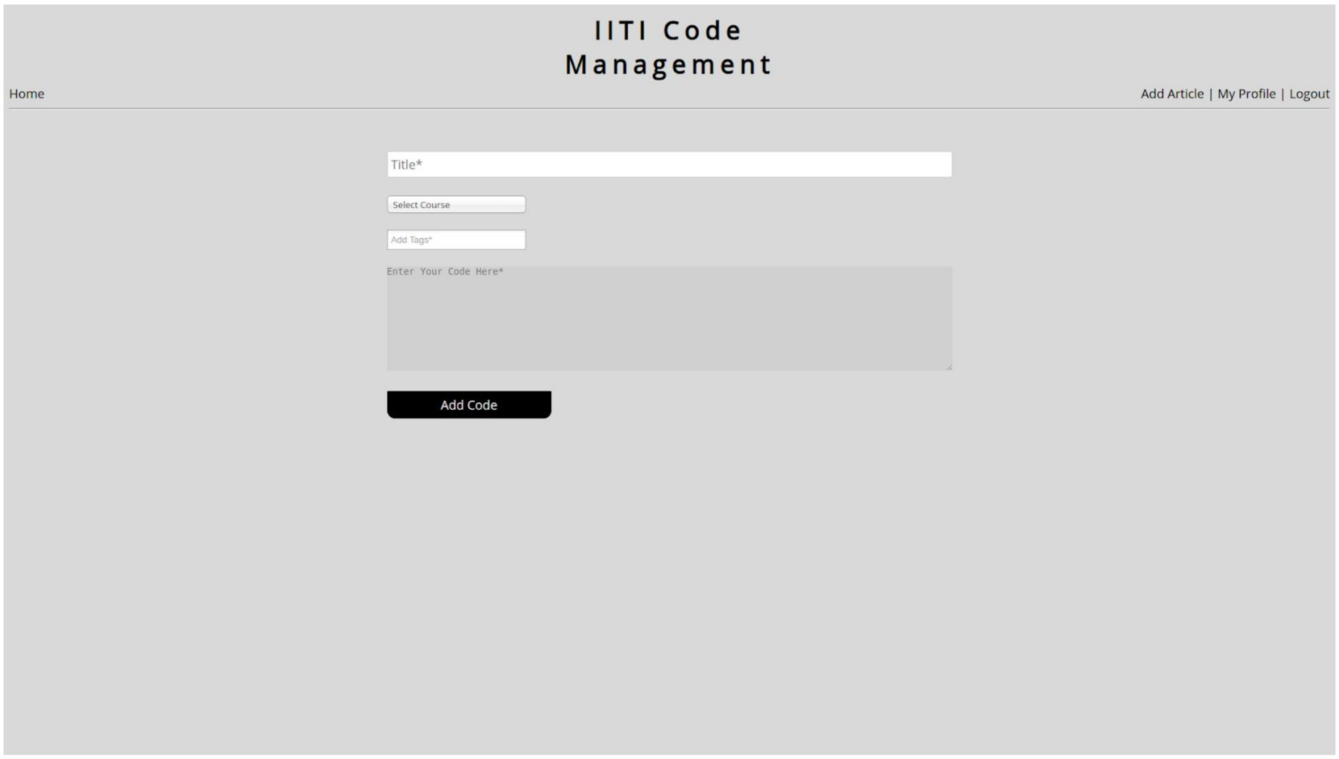
- *Can you please explain the logic behind the code?* cse180001057@iiti.ac.in *2019-11-15 13:48:06*

  Add Reply Submit Reply

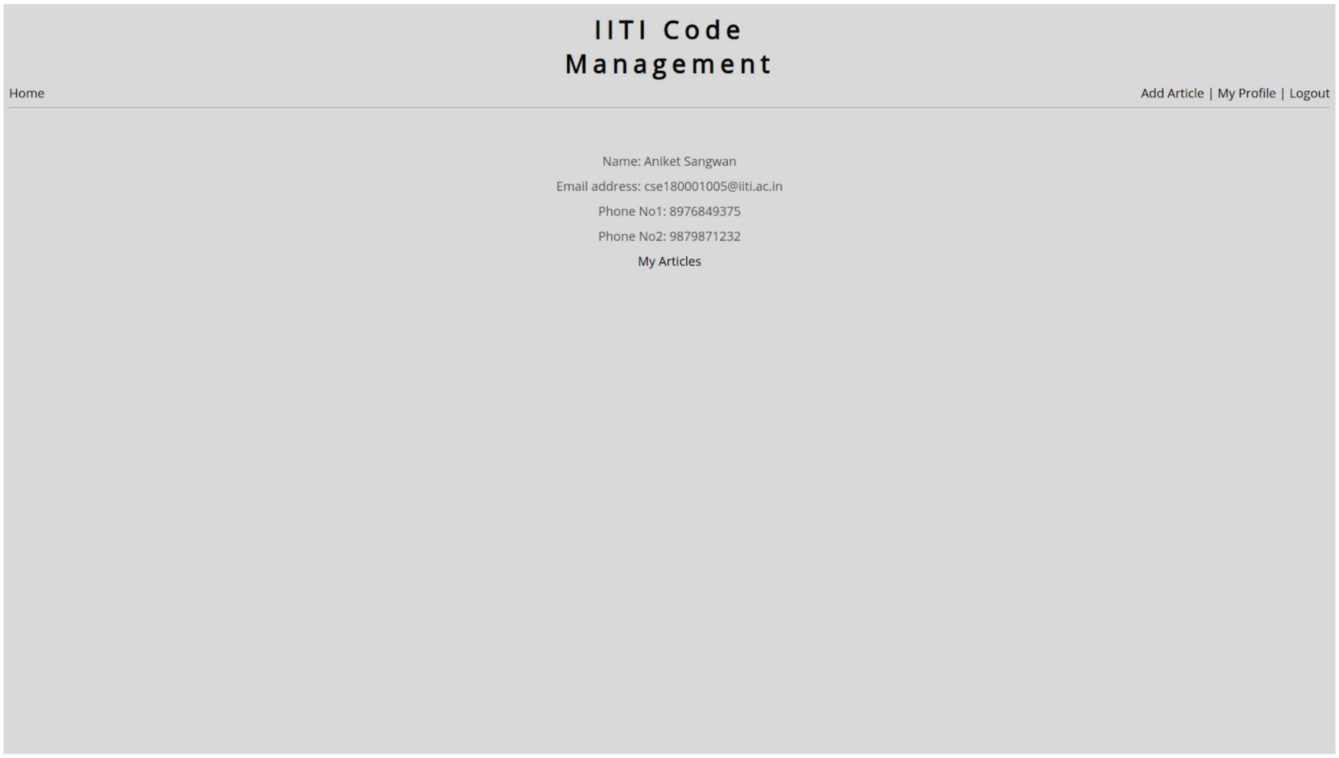
  - *This code is simple implementation of Count Sort* cse180001005@iiti.ac.in *2019-11-15 13:53:45*

    Add Reply Submit Reply

Add Article Page:



My Profile:



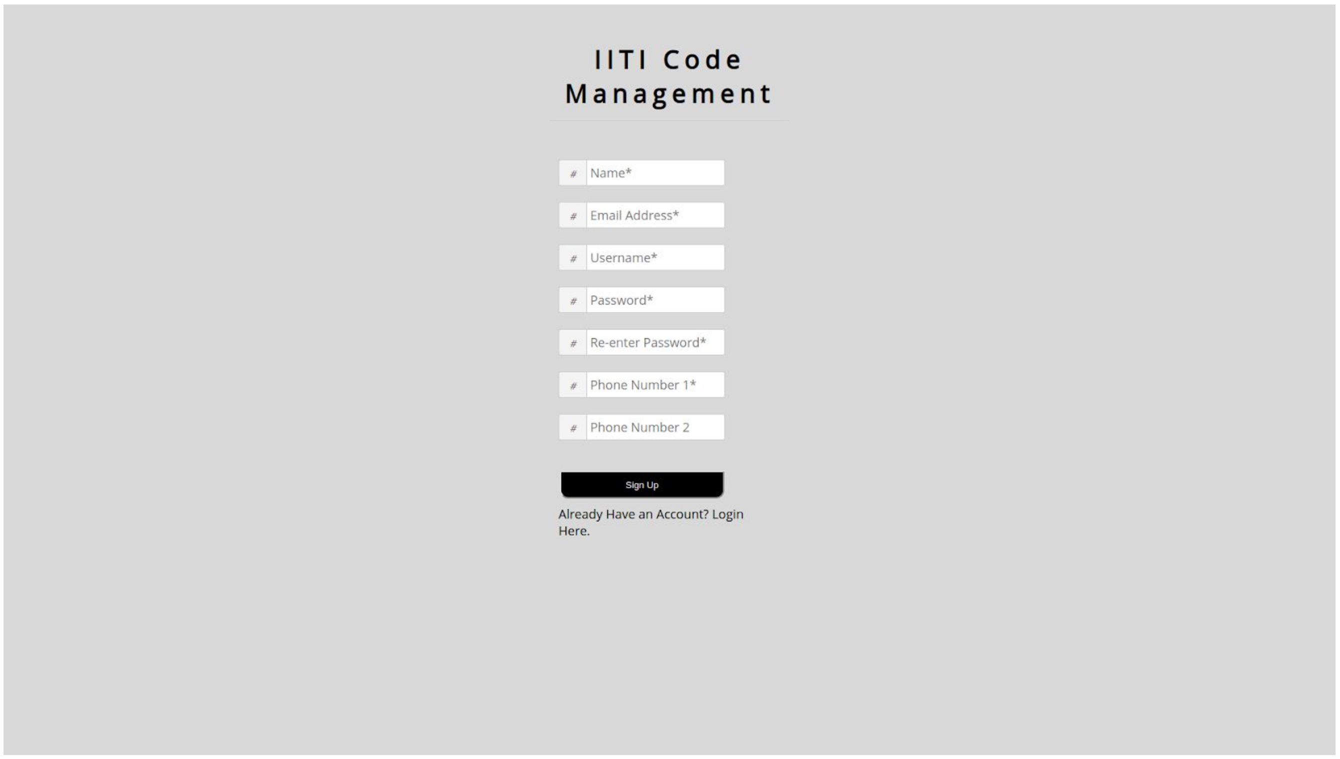
Name: Aniket Sangwan

Email address: cse180001005@iiti.ac.in

Phone No1: 8976849375

Phone No2: 9879871232

Signup Page:


IITI Code Management

Name*

Email Address*

Username*

Password*

Re-enter Password*

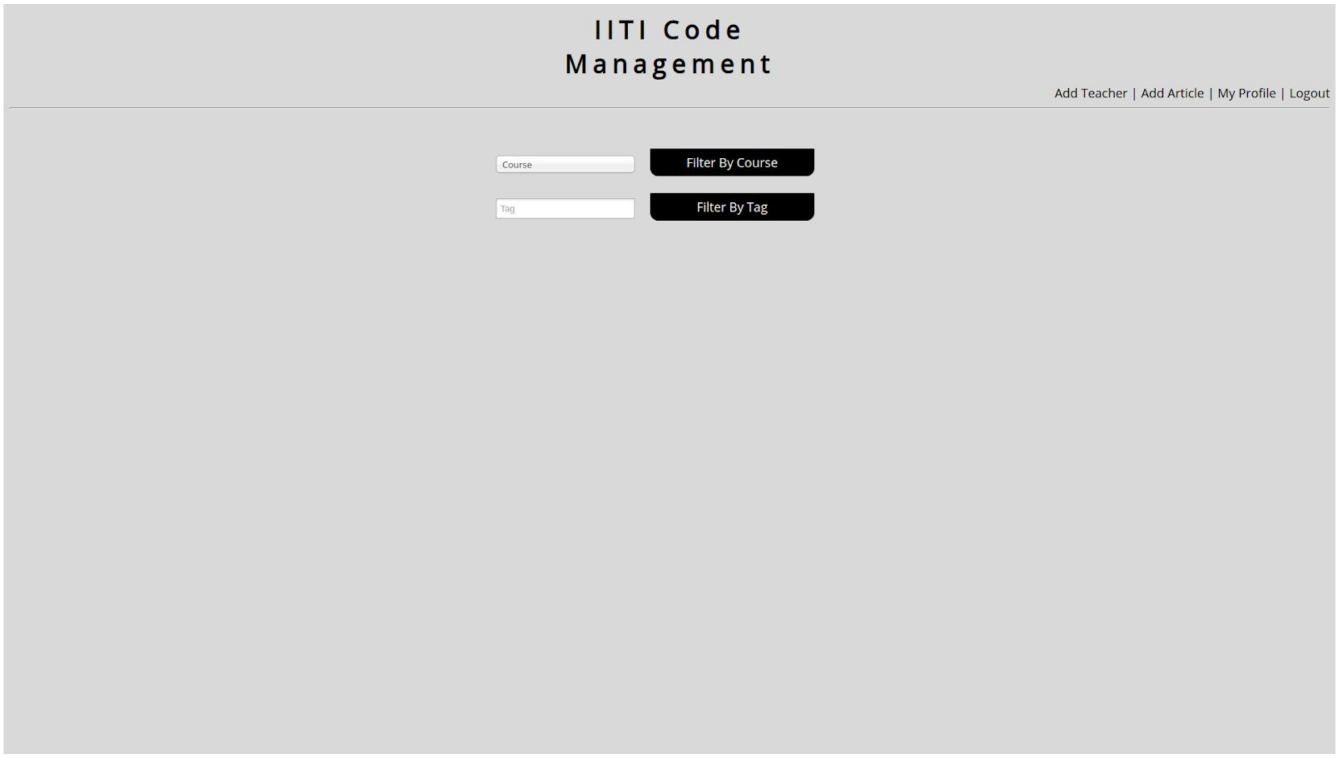
Phone Number 1*

Phone Number 2

Sign Up

Already Have an Account? Login Here.

Admin Login to Add Teacher:


IITI Code Management

Add Teacher | Add Article | My Profile | Logout

Course

Filter By Course

Tag

Filter By Tag