VISVESVARAYA TECHNOLOGICAL UNIVERSITY

BELAGAVI, KARNATAKA – 590018

DBMS MINI PROJECT REPORT ON

"STUDENT PROJECT MANAGEMENT SYSTEM"

Submitted in partial fulfillment of the requirements as a part of the DBMS Lab for the V semester of degree of ***Bachelor of Engineering in Computer Science and Engineering*** *of Visvesvaraya Technological University, Belagavi*

Submitted by:

DEEPAK NAIDU	**1JS19CS049**
GIRISH KUMAR DV	**1JS19CS057**

Under the Guidance of

Mrs. Rajeshwari K.S,

Assistant Professor, Dept of CSE,

JSSATE, Bengaluru

JSS ACADEMY OF TECHNICAL EDUCATION

Dr.Vishnuvardhan Rd, Srinivasapura Post,

Bengaluru – 560060 (2021-2022)

JSS MAHA VIDYAPEETHA, MYSORE

JSS ACADEMY OF TECHNICAL EDUCATION

JSS Campus, Uttarahalli - Kengeri main road, Bengaluru – 560060

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project work entitled "RESTAURANT RESERVATION SYSTEM" is a bonafide work carried out by **Mr. Deepak Naidu (1JS19CS049)** and **Mr. Girish Kumar DV (1JS19CS057)** in partial fulfillments of the requirements for DBMS Laboratory with Mini Project (18CSL58) of V semester **Bachelor of Engineering in Computer Science and Engineering** of the academic year 2021-2022. It is certified that all the corrections and suggestions indicated for Internal Assessments have been incorporated in the report deposited in the department Library. The project report has been approved as it satisfies the academic requirements in respect of project work prescribed for the said degree.

Mrs. Rajeshwari K.S,
Assistant Professor, Dept of CSE
JSSATE, Bengaluru

Dr. Naveen N.C,
Professor and Head, Dept of CSE
JSSATE, Bengaluru

Name of the Examiners:	**Signature with Date:**
1. …………………………	**……………………………**
2. …………………………	**……………………………**

ABSTRACT

The World of data is constantly changing and evolving every second. This in turn has created a completely new dimension of growth and challenges for companies around the globe. By accurately recording data, storing, updating and tracking them on an efficient and regular basis. Companies can address their challenges on one hand and make use of the immense potentials offered by this sector on the other hand.

A database management system stores, organizes and manages a large amount of information within a single software application. The use of this system increases efficiency of business operations and reduce overall costs.

Educational Institutes across the globe are always incessantly striving to churn out the best skillful and academically and holistically sound individuals. The organizing and administration has to be impeccable. Here we have come up with a portal – Student Project Management System that connects teachers and students and allows teachers to upload and assign as well as update and delete projects and grade them. Students can view and answer to those assignments as well as update and delete their assignment.

Student Project Management System (SPMS) - is a web-based application.
It is mainly focused to be used in an educational institute and allows teachers to monitor student project submissions and generate reports of the performance of each class that the teacher engages.

It provides two isolated views – A Student View and Teacher View.
All CRUD (Create, Retrieve, Update, Delete) operations which are the most fundamental operations and core essence of using a Relational Database are demonstrated in this project.
It Requires a new user to be registered before they get access to any facilities provided by the portal, Hence, the administrator can log each users work and timestamp, thus providing a security feature.

ACKNOWLEDGEMENT

We express our humble pranams to His Holiness **Jagadguru Sri Sri Sri Shivaratri Deshikendra mahaswamiji** for showering his blessings on us to receive good education and have a successful career.

The completion of any project involves the effort of many people. We have been very lucky to have received support and guidance from all kinds of sources to complete our project. So, we take this opportunity to express our gratitude to all those who gave us guidance and encouragement to successfully complete this project.

We express a sincere thanks to our beloved principal **Dr. Mrityunjaya V Latte** for having supported us in all academic endeavours.
We are also forever grateful to **Dr. Naveen N.C** Head of Department, Computer Science and Engineering for his unending support, guidance and encouragement in all are ventures.

We are very thankful for the resourceful guidance and timely assistance and graceful gesture of our guide **Mrs. Rajeshwari K.S** assistant professor, Department of computer science and engineering who has helped us in every aspect of our project work.

Last but not the least, we would be immensely pleased to express our heartfelt thanks to all the teaching and non-teaching staff of the Department of CSE and our friends for the timely help support and guidance.

DEEPAK NAIDU

GIRISH KUMAR DV

Table of Contents

Chapter Title **Page No.**

Abstract i

Acknowledgement ii

Table of Contents iii

List of Figures iv

Chapter 1 Preamble

1.1 Introduction. 1

1.1.1 HTML. 2

1.1.2 CSS 2

1.1.3 Python 2

1.1.4 Database-MySQL 4

1.1.5 Django 4

1.1.6 Normalization 5

1.2 Objectives 6

1.3 Organization of Report 6

1.4 Summary 6

Chapter 2 Requirement Specifications

2.1 Software Specification 7

2.2 Hardware Specification 7

2.3 User Characteristics 7

Chapter 3 System Design and Implementation

3.1 Introduction 8

3.2 ER Diagram 9

3.3 Schema Diagram 11

3.4 Queries 12

3.4.1 Creating Tables.. 12

3.4.2 Insertion..19

3.4.3 Deletion & Updating.. 23

3.5 Logic…….. 27

3.5.1 Registration 27

3.5.2 Teacher Profile and Add Activity... 27

3.5.3 Student Profile and Hand In Activity……...............................29

3.5.4 Update Assigned Project 30

3.5.5 Delete Assigned Project.. 30

3.5.6 Generating Report For each activity……………………….31

3.6 URL Mappings.. 32

3.7 Database Connection to Django.. 32

Chapter 4 Result & Discussions..33

Conclusion & Future Enhancements..37

References……………………………………………………….....38

LIST OF FIGURES

Figure No	Name of Figure	Page No.
Chapter 3		**8**
Fig 3.2	ER Diagram	9
Fig 3.3	Relationship Schema Diagram	11
Fig 3.4.1a	Desc Table Department	13
Fig 3.4.1b	Desc Table Class_Section	13
Fig 3.4.1c	Desc Table Subject	14
Fig 3.4.1d	Desc Table Student	15
Fig 3.4.1e	Desc Table Teacher	16
Fig 3.4.1f	Desc Table Project	17
Fig 3.4.1g	Desc Table SubmitProject	18
Fig 3.4.2a	Frontend Data Insertion (registration)	19
Fig 3.4.2b	SQL-Student and teacher data	20
Fig 3.4.2c	Teacher View – Add activity	20
Fig 3.4.2d	SQL-Project Table data	21
Fig 3.4.2e	Student View – Hand in Project	21
Fig 3.4.2f	SQL-Submit Project data	21
Fig 3.4.2g	SQL-ClassSection,Department,Subject Data	22
Fig 3.4.3a	Teacher View – Before Update	23
Fig 3.4.3b	Teacher View – Update page	23
Fig 3.4.3c	Teacher View – After Update	24
Fig 3.4.3d	Student View – Before Update	24

Fig 3.4.3e Student View – Update page 24
Fig 3.4.3f Student View – After Update 25
Fig 3.4.3g Teacher View – Before Deletion 25
Fig 3.4.3h Teacher View –Submission page after Delete 26
Fig 3.4.3i Student View – Graded page 26
Fig 3.5 Django File Structure 27
Fig 3.5.1 Views.py – register 27
Fig 3.5.1a Views.py - registerStudent 28
Fig 3.5.1b Views.py – registerTeacher 28
Fig 3.5.1c Views.py – login 28
Fig 3.5.2 Views.py – Teacher profile Add Project 29
Fig 3.5.3 Views.py – Student profile Hand in Project 29
Fig 3.5.4 Views.py – Update Project (Teacher) 30
Fig 3.5.5 Views.py – Delete Project (Teacher) 30
Fig 3.5.6 Methods.py – Generate Report (.xls file) 31
Fig 3.6 Urls.py file 32
Fig 3.7 Django database connectivity 32

Chapter 4 **33**

Fig 4.1 Login Page 33
Fig 4.2 Authenticate Login 33
Fig 4.3 User Registration 34
Fig 4.4 Alert Message – User Validation 34
Fig 4.5 Submissions Page (Teacher Views) 35
Fig 4.6 Graded Page (Teacher View) 35

Fig 4.7	Download Report (Teacher View)	36
Fig 4.8	Excel file containing report of Student	36

Chapter 1

Preamble

1.1 Introduction

A database is an organized collection of data. A relational database, more restrictively, is a collection of schemas, tables, queries, reports, views, and other elements. A database management system (DBMS) is a computer-software application that interacts with end users, other applications, and the database itself to capture and analyze data. A general-purpose DBMS allows the definition, creation, querying, update, and administration of database. There is a need for an application to make it easy for educational institutes to monitor and maintain records of students work and performance. Student Project Management System is a one stop for teachers and students to trace and update their projects. It provides a centralized storage stop of all projects and their related data (metadata). It also provides additional security features as only privileged/registered users can access/modify information about the projects. There is only authorized access to the portal, hence for a new user, registration is an indispensable step.

Database Management System (DBMS)

Following the technology progress in the areas of processors, computer memory, computer storage, and computer networks, the sizes, capabilities, and performance of databases and their respective DBMSs have grown in orders of magnitude. The development of database technology can be divided into three eras based on data model or structure: navigational, SQL/relational, and post-relational. The two main early navigational data models were the hierarchical model, epitomized by IBM's IMS system, and the CODASYL model (network model), implemented in a number of products such as IDMS.

The relational model employs set of ledger-style tables, each used for a different type of entity. Only in the mid-1980s did computing hardware become powerful enough to allow the wide deployment of relational systems (DBMSs plus applications). By the early 1990s, however, relational systems dominated in all large-scale data processing applications, and as of 2015 they remain dominant: IBM DB2, Oracle, MySQL, and Microsoft SQL Server are the top DBMS. The dominant database language, standardized SQL for the relational model, has influenced database languages for other data models.

1.1.1 HTML (Hypertext Markup Language)

The Hypertext Markup Language, or HTML is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript.

Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

HTML elements are the building blocks of HTML pages. With HTML constructs, images and other objects such as interactive forms may be embedded into the rendered page. HTML provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. HTML elements are delineated by tags, written using angle brackets.

1.1.2 CSS (Cascading Style Sheets)

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language such as HTML. CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.

CSS is designed to enable the separation of presentation and content, including layout, colors, and fonts. This separation can improve content accessibility; provide more flexibility and control in the specification of presentation characteristics; enable multiple web pages to share formatting by specifying the relevant CSS in a separate .css file, which reduces complexity and repetition in the structural content; and enable the .css file to be cached to improve the page load speed between the pages that share the file and its formatting.

1.1.3 Python 3

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

Python is a MUST for students and working professionals to become a great Software Engineer specially when they are working in Web Development Domain. I will list down some of the key advantages of learning Python:

Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.

Python is Interactive − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python is Object-Oriented − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.

Characteristics of Python

Following are important characteristics of python –

- It supports functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte-code for building large applications.
- It provides very high-level dynamic data types and supports dynamic type checking.
- It supports automatic garbage collection.

1.1.4 Database MySQL

MySQL is free and open-source software under the terms of the GNU General Public License, and is also available under a variety of proprietary licenses. MySQL was owned and sponsored by the Swedish company MySQL AB, which was bought by Sun Microsystems (now Oracle Corporation). In 2010, when Oracle acquired Sun, Widenius forked the open-source MySQL project to create MariaDB.

MySQL has stand-alone clients that allow users to interact directly with a MySQL database using SQL, but more often, MySQL is used with other programs to implement applications that need relational database capability.

1.1.5 Django

Django's primary goal is to ease the creation of complex, database-driven websites. The framework emphasizes reusability and "pluggability" of components, less code, low coupling, rapid development, and the principle of don't repeat yourself. Python is used throughout, even for settings, files, and data models. Django also provides an optional administrative create, read, update and delete interface that is generated dynamically through introspection and configured via admin models.

Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel. It’s free and open source.

- Ridiculously fast.

Django was designed to help developers take applications from concept to completion as quickly as possible.

- Reassuringly secure.

Django takes security seriously and helps developers avoid many common security mistakes.

- Exceedingly scalable.

Some of the busiest sites on the web leverage Django’s ability to quickly and flexibly scale.

Django officially supports the following databases:

- PostgreSQL
- MariaDB
- MySQL
- Oracle
- SQLite

1.1.6 Normalization

Normalization is a process of organizing the data in database to avoid data redundancy, insertion anomaly, update anomaly & deletion anomaly. To overcome these anomalies, we need to normalize the data. There are 4 basic types of normalizations. They are:

- First normal form (INF)
- Second normal form(2NF)
- Third normal form(3NF)
- Boyce & Codd normal form (BCNF)

First normal form (INF) is defined as per rule as: an attribute (column) of a table cannot hold multiple values. It should hold only atomic values. This means that there shouldn't be repetition of data in the tables.

A table is said to be in 2NF if the two conditions stated are satisfied. The table is in First normal form and all the non-prime attribute are dependent on the proper subset of any candidate key of table. The attribute that is not part of any candidate key are known as non-prime attribute.

A table design is said to be in 3NF if the table is in 2NF and Transitive functional dependency of non-prime attribute on any super key are removed.

Boyce Codd normal form (BCNF) is the advance version of 3NF that's why it is also referred as 3.5NF. BCNF is stricter than 3NF. A table complies with BCNF if it is in 3NF and for every functional dependency X-BY, X should be the super key of the table.

1.2 Objectives

Objectives of "Student Project Management System" are

- To optimize and increase efficiency in Student-Teacher Transactions (Projects).
- To provide an online platform so that the teacher and assign from anywhere and student can view and upload their work from anywhere.
- Provide security by authorizing every user and transaction.
- Reduce/Minimize usage of hardcopies of submissions.
- Track and monitor assigned projects.
- Reducing offline hassle.
- Bringing together an entire institution at one place for this particular purpose of project management.
- Ease of Use.
- Good user experience.

1.3 Organization of Report

Chapter 1 provides the information about the basics of HTML, CSS, Python, Django, and MySQL. In **Chapter 2**, we discuss the software and hardware requirements to run the above applications. **Chapter 3** gives the idea of the project and its actual implementation. **Chapter 4** discusses about the results and discussions of the program. It concludes by giving the direction for future enhancement.

1.4 Summary

The chapter discussed before is an overview about the Django Application and MySQL Database DBMS. The scope of study and objectives of the project are mentioned clearly. The organization of the report has been pictured to increase the readability. Further, coming up chapters depicts the use of various queries to implement various changes like insert, update, delete and also triggers to perform various functions.

Chapter 2

Requirement Specifications

2.1 SOFTWARE SPECIFICATION

- Operating System: Windows 10
- Front End: HTML, CSS
- Rear End: Django (python)
- DBMS: MySQL

2.2 HARDWARE SPECIFICATION

- Processor: x86 compatible processor with 1.7 GHz Clock Speed
- RAM: 512 MB or greater
- Hard Disk: 20 GB or grater
- Monitor: VGA/SVGA
- Keyboard: 104 keys standard
- Mouse: 2/3 button. Optical/Mechanical.

2.3 USER CHARACTERISTICS

Every user:

- Should be comfortable with basic working of the computer
- Must have basic knowledge of English
- Must register if it’s their first visit to the site and remember their login ID (USN / Teacher ID) and password for authentication and to access facilities provided in the site.

Chapter 3

System Design and Implementation

3.1 Introduction

Systems design is the process or art of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. One could see it as the application of systems theory to product development.

This Project is implemented using Django – a python framework, which is proven to be a very efficient tool in the field of Python programming. It is done under Windows 10/11 OS platform. HTML, CSS, and Python programming language is used to implement the entire code.

Developer has to have python3 installed and can install Django by executing the following command in the command prompt:

- `pip install django`

To start a new Django project:

- `django-admin startproject project_name`

To create an app inside the Django project:

- `django-admin startapp app_name`

To run a Django project:

- `python manage.py runserver`

These are a few basic commands to start developing a Django application.

Interface to the program is provided with the help of MySQL Database.

3.2 ER Diagram

An entity–relationship model or the ER Diagram describes inter-related things of interest in a specific domain of knowledge. An ER model is composed of entity types and specifies relationships that can exist between instances of those entity types.

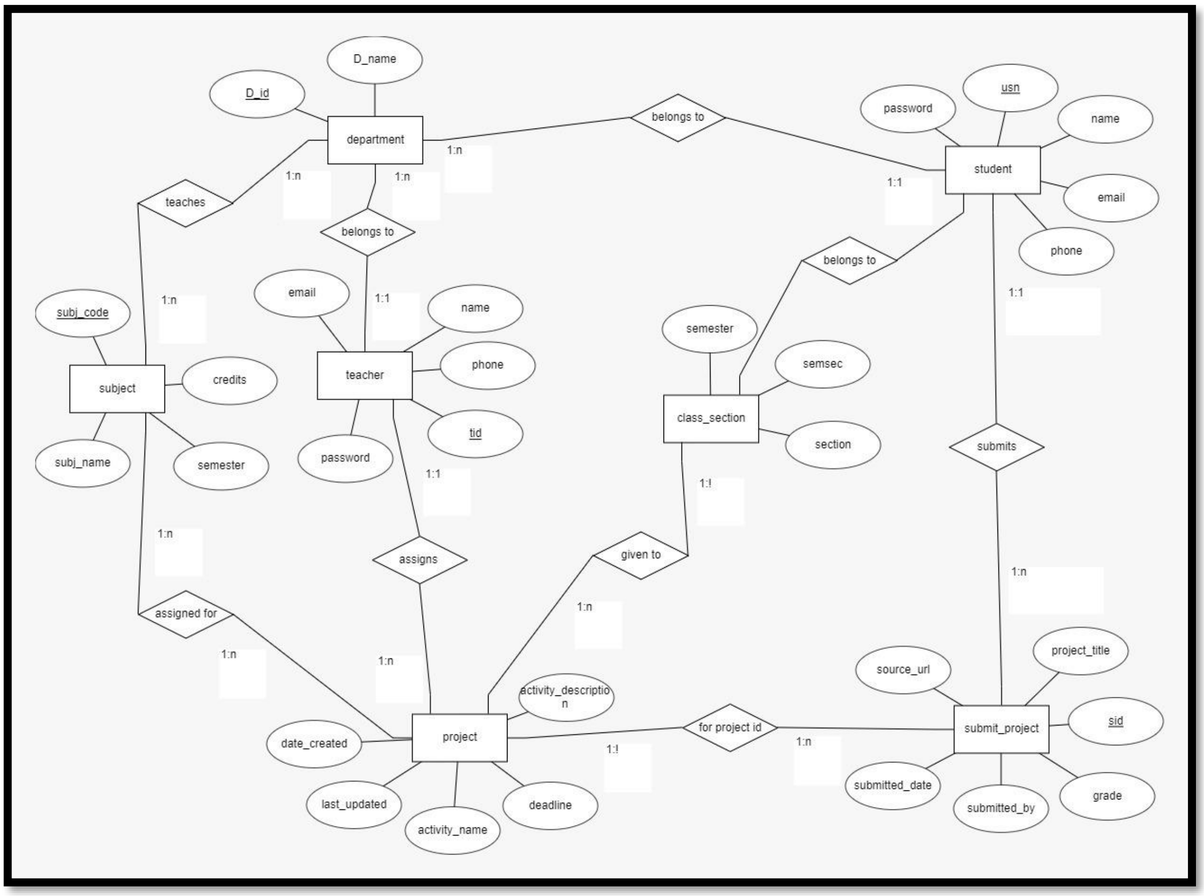


Fig 3.2 : ER Diagram

Fig 3.2 showing all the entities and their attributes along with the relationships which they share. There are 6 entities and 7 relations among them which are depicted above.

3.2 ENTITIES

1. Department – **D_id** , D_name.
2. Subject – **subj_code**, subj_name, semester, credits, (branc Foreign Key)
3. Class_Section – **semsec,** semester, section
4. Student – **usn,** name, email, phone, password, (branch, semsec Foreign Key)
5. Teacher – **tid**, name, email, phone , password, (branch Foreign Key)
6. Project – **pid,** activity_name, activity_description, date_created, last_updated, deadline, (for_semsec, for_subject, assigned_by Foreign Key)
7. SubmitProject – **sid**, project_title, source_url, grade, submission date , (submitted_by , for_project_id Foreign Key)

3.3 Schema Diagram

The schema diagram of a database system is its structure described in a formal language supported by the database management system (DBMS). The formal definition of a database schema is a set of formulas called integrity constraints imposed on a database.

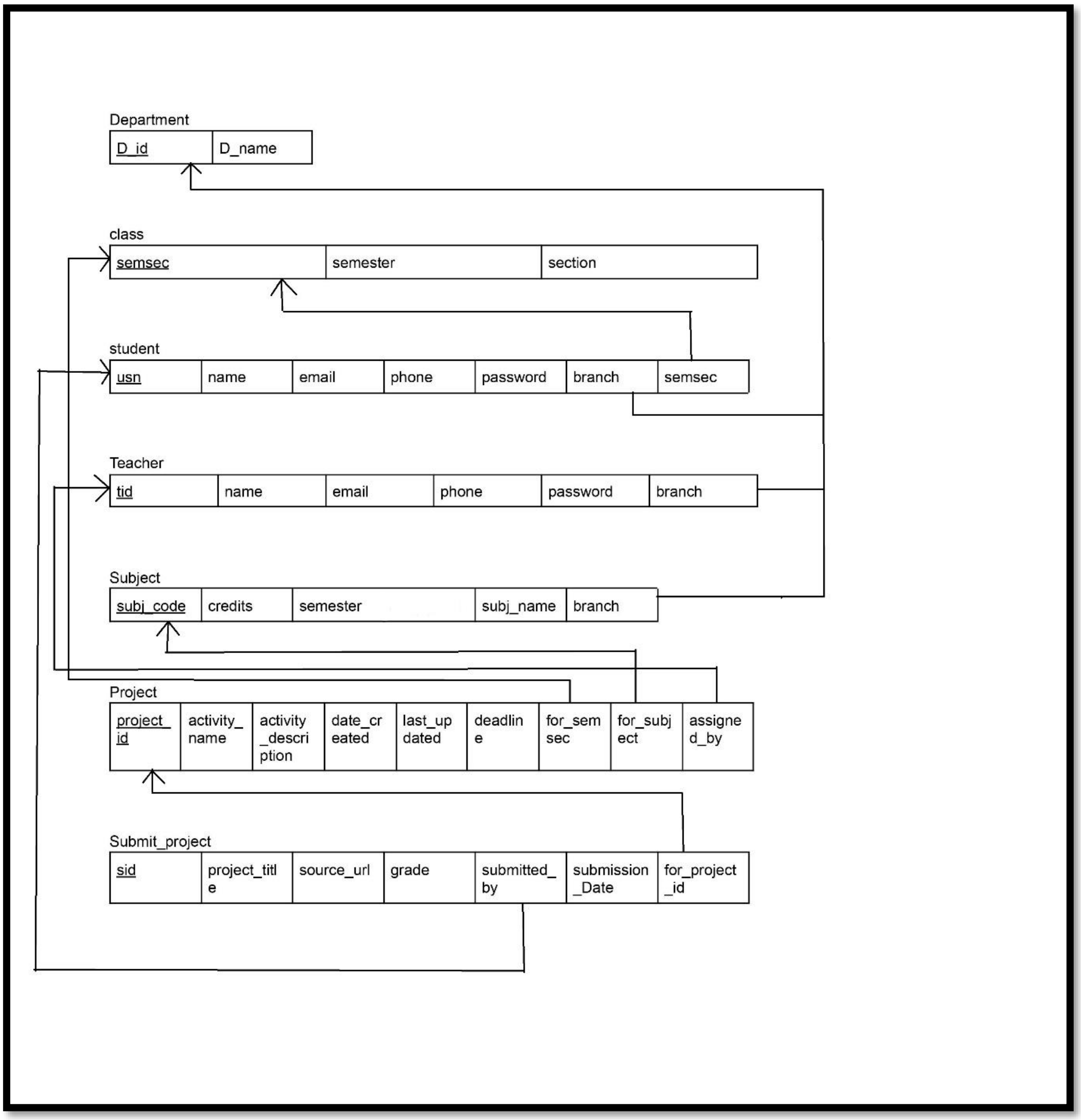


Fig 3.3 : Relationship Schema Diagram

Fig 3.3 shows the relationship schema diagram for Student Project management System with 7 relations (tables) and the mapping of the primary keys and foreign keys.

The term "schema" refers to the organization of data as a blueprint of how the database is constructed. These integrity constraints ensure compatibility between parts of the schema. All constraints are expressible in the same language. A database can be considered a structure in realization of the database language. The states of a created conceptual schema are transformed into an explicit mapping, the database schema. This describes how real-world entities are modeled in the database.

3.4 Queries

The below mentioned are all the queries used to perform various tasks in MySQL and DJANGO such as insert, delete, update. A short description of the query is also provided. Updating can also be done using the GUI provided in the admin page and the webapp.

3.4.1 Creating Tables

In Django tables are represented in the form of python classes which inherit the `models.Model` class. These classes are called models.

Department table.

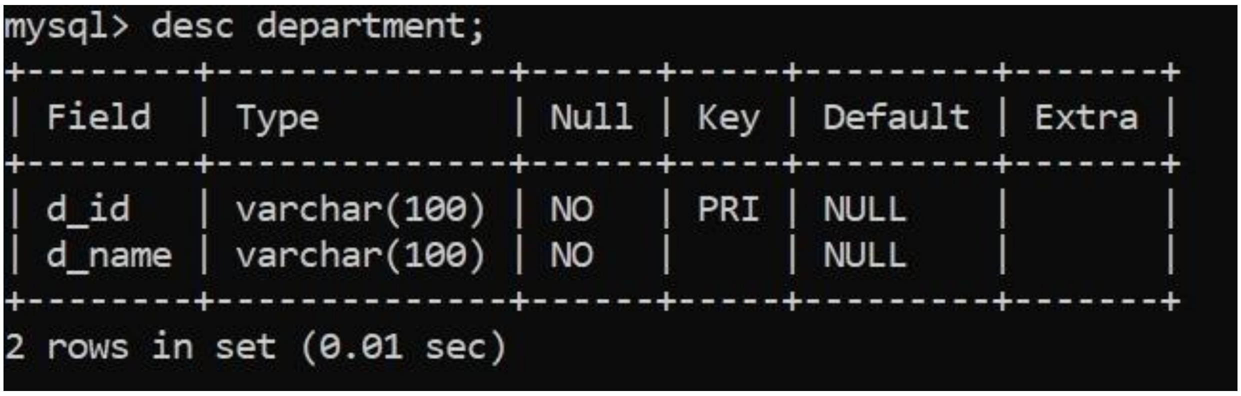
Django snippet –

```
class Department(models.Model):
     d_id = models.CharField(max_length=10, primary_key = True)
     d_name = models.CharField(max_length=100)
```

The above snipped is analogous to the following SQL DDL(Data Definition Language):

SQL command –

```
CREATE TABLE department{
d_id varchar(10) primary key,
d_name varchar(100) };
```



```
mysql> desc department;
+--------+--------------+------+-----+---------+-------+
| Field  | Type         | Null | Key | Default | Extra |
+--------+--------------+------+-----+---------+-------+
| d_id   | varchar(100) | NO   | PRI | NULL    |       |
| d_name | varchar(100) | NO   |     | NULL    |       |
+--------+--------------+------+-----+---------+-------+
2 rows in set (0.01 sec)
```

Fig 3.4.1a : Desc Table Department

Fig 3.4.1a shows a MySQL command line snapshot of table Department description with d_id as Primary Key and d_name as an attribute.

Class_Section Table

Django snippet –

```
class Class_Section(models.Model):
    semsec = models.CharField(max_length=2, primary_key = True)
    semester = models.IntegerField()
    section = models.CharField(max_length=1)
```

The above snipped is analogous to the following SQL DDL(Data Definition Language):

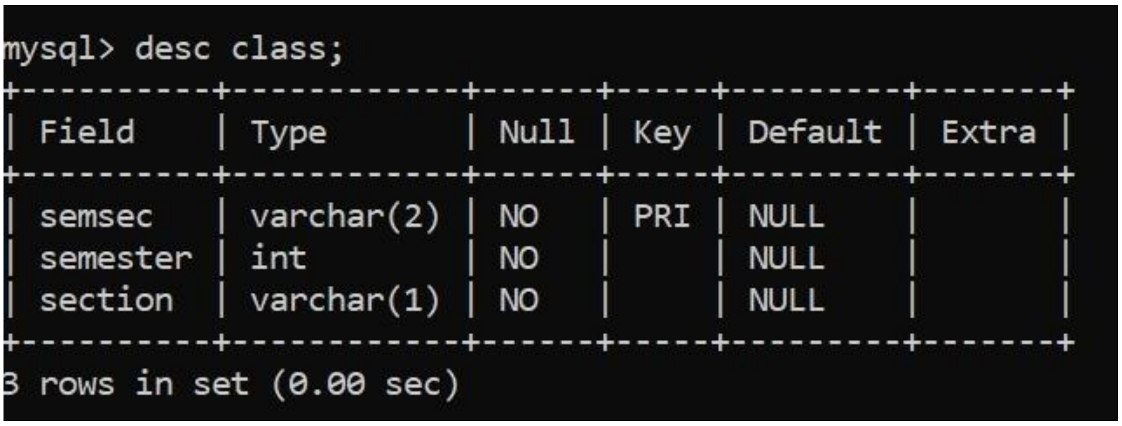
SQL command –

CREATE TABLE class_section{

semsec char(2) primary key,

semester INTEGER,

section char(1) };



```
mysql> desc class;
+----------+------------+------+-----+---------+-------+
| Field    | Type       | Null | Key | Default | Extra |
+----------+------------+------+-----+---------+-------+
| semsec   | varchar(2) | NO   | PRI | NULL    |       |
| semester | int        | NO   |     | NULL    |       |
| section  | varchar(1) | NO   |     | NULL    |       |
+----------+------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

Fig 3.4.1b : Desc Table Class_Section

Fig 3.4.1b shows a MySQL command line snapshot of table Class_Section description with semsec as Primary Key and semester and section as attributes.

Subjects Table

Django snippet –

```
class Subject(models.Model):
     subj_code = models.CharField(max_length=8,primary_key=True)
     subj_name = models.CharField(max_length=100)
     semester = models.IntegerField(default=1)
     credits = models.IntegerField(default=1)
     branch = models.ForeignKey(Department,on_delete=models.CASCADE)
```

The above snipped is analogous to the following SQL DDL(Data Definition Language):

SQL command –

CREATE TABLE subject{

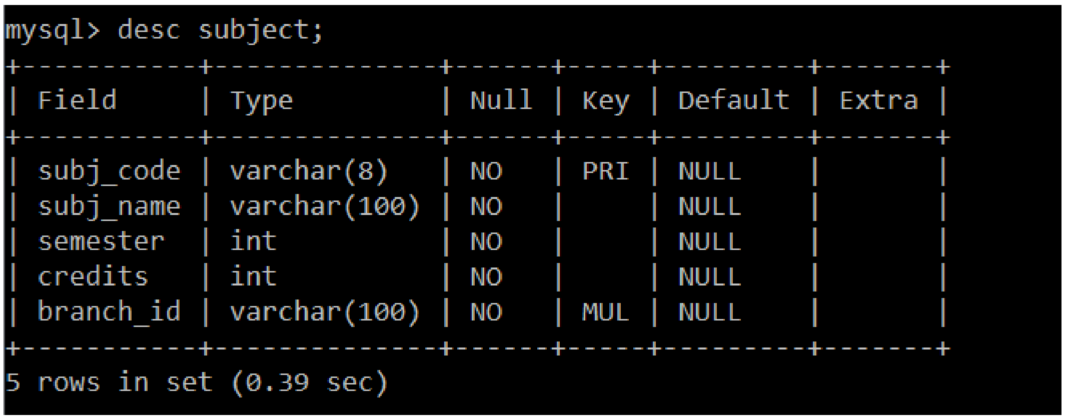
subj_Code varchar(8) primary key,

subj_name varchar(100),

semester INTEGER,

credits INTEGER,

branch references department(d_id) ON DELETE CASCADE };



```
mysql> desc subject;
+-----------+--------------+------+-----+---------+-------+
| Field     | Type         | Null | Key | Default | Extra |
+-----------+--------------+------+-----+---------+-------+
| subj_code | varchar(8)   | NO   | PRI | NULL    |       |
| subj_name | varchar(100) | NO   |     | NULL    |       |
| semester  | int          | NO   |     | NULL    |       |
| credits   | int          | NO   |     | NULL    |       |
| branch_id | varchar(100) | NO   | MUL | NULL    |       |
+-----------+--------------+------+-----+---------+-------+
5 rows in set (0.39 sec)
```

Fig 3.4.1c : Desc Table Subject

Fig 3.4.1c shows a MySQL command line snapshot of table Subject description with subj_code as Primary Key and branch_id as Foreign Key from table Department and subj_name, semester and credits as attributes.

Student Table

Django snippet –

```
class Student(models.Model):
     usn = models.CharField(max_length=10,primary_key=True)
     name = models.CharField(max_length=100)
     email = models.EmailField(max_length=100)
     phone = models.DecimalField(max_digits=10,decimal_places=0)
     password = models.CharField(max_length=100)
     branch = models.ForeignKey(Department,on_delete=models.CASCADE)
     semsec=models.ForeignKey(Class_Section,on_delete=models.CASCADE)
```

The above snipped is analogous to the following SQL DDL(Data Definition Language):

SQL command –

CREATE TABLE student{

usn varchar(10) primary key,

name varchar(100).

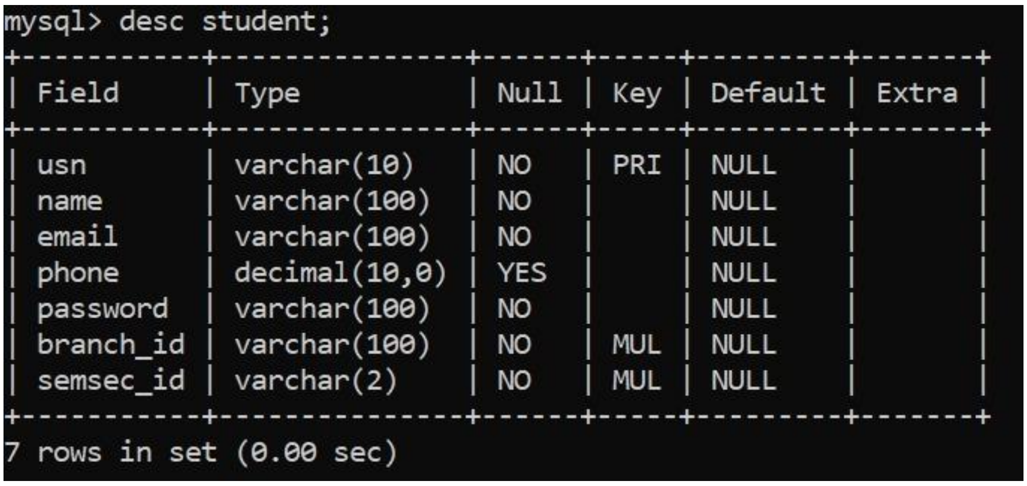
email varchar(100),

phone NUMBER(10),

password varchar(100),

branch references department(d_id) ON DELELTE CASCADE,

semsec references class_section(semsec) ON DELETE CASCADE };



```
mysql> desc student;
+-----------+---------------+------+-----+---------+-------+
| Field     | Type          | Null | Key | Default | Extra |
+-----------+---------------+------+-----+---------+-------+
| usn       | varchar(10)   | NO   | PRI | NULL    |       |
| name      | varchar(100)  | NO   |     | NULL    |       |
| email     | varchar(100)  | NO   |     | NULL    |       |
| phone     | decimal(10,0) | YES  |     | NULL    |       |
| password  | varchar(100)  | NO   |     | NULL    |       |
| branch_id | varchar(100)  | NO   | MUL | NULL    |       |
| semsec_id | varchar(2)    | NO   | MUL | NULL    |       |
+-----------+---------------+------+-----+---------+-------+
7 rows in set (0.00 sec)
```

Fig 3.4.1d : Desc Table Student

Fig 3.4.1d shows a MySQL command line snapshot of table Student description with usn as Primary Key and branch_id, semsec_id as Foreign Key from table Department and Class_Section and name, email, phone and password as attributes.

Teacher Table

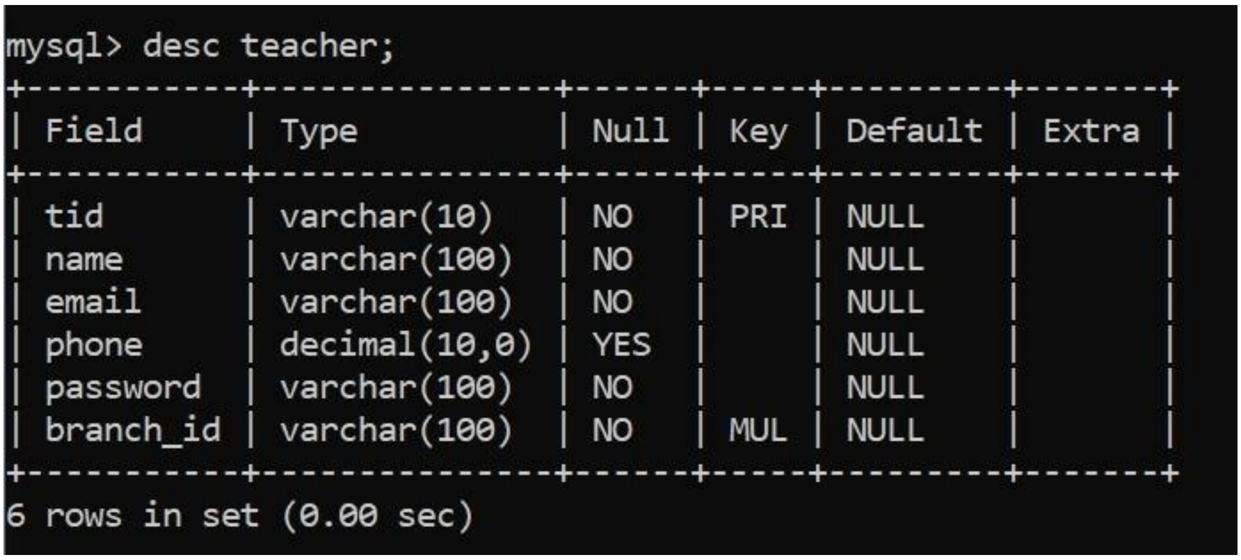
Django snippet –

```
class Teacher(models.Model):
    tid = models.CharField(max_length=10,primary_key=True)
    name = models.CharField(max_length=100)
    email = models.EmailField(max_length=100)
    phone = models.DecimalField(max_digits=10,decimal_places=0)
    password = models.CharField(max_length=100)
    branch = models.ForeignKey(Department,on_delete=models.CASCADE)
```

The above snipped is analogous to the following SQL DDL(Data Definition Language):

SQL command –

```
CREATE TABLE teacher{
tid varchar(10) primary key,
name varchar(100).
email varchar(100),
phone NUMBER(10),
password varchar(100),
branch references department(d_id) ON DELELTE CASCADE};
```



```
mysql> desc teacher;
+-----------+---------------+------+-----+---------+-------+
| Field     | Type          | Null | Key | Default | Extra |
+-----------+---------------+------+-----+---------+-------+
| tid       | varchar(10)   | NO   | PRI | NULL    |       |
| name      | varchar(100)  | NO   |     | NULL    |       |
| email     | varchar(100)  | NO   |     | NULL    |       |
| phone     | decimal(10,0) | YES  |     | NULL    |       |
| password  | varchar(100)  | NO   |     | NULL    |       |
| branch_id | varchar(100)  | NO   | MUL | NULL    |       |
+-----------+---------------+------+-----+---------+-------+
6 rows in set (0.00 sec)
```

Fig 3.4.1e : Desc Table Teacher

Fig 3.4.1e shows a MySQL command line snapshot of table Teacher description with tid as Primary Key and branch_id as Foreign Key from table Department and name, email, phone and password as attributes.

Project Table

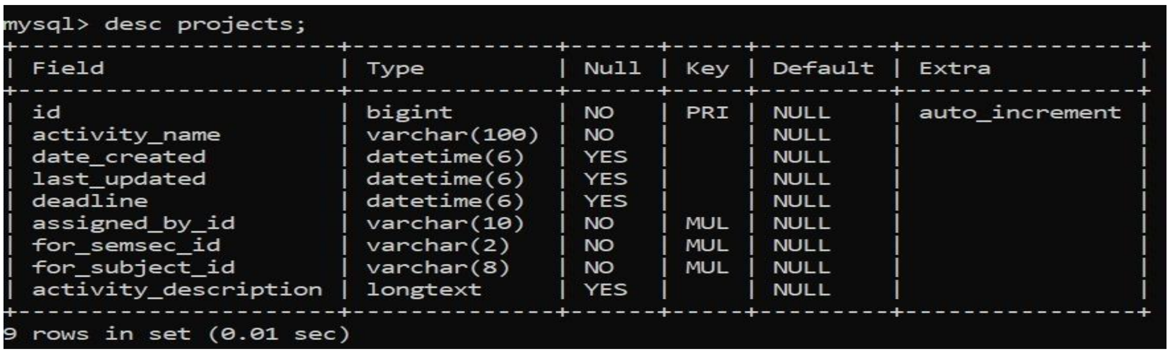
Django snippet –

```
class Project(models.Model):
    activity_name = models.CharField(max_length=100)
    activity_description = models.TextField()
    date_created = models.DateTimeField(auto_now_add=True)
    last_updated = models.DateTimeField(auto_now=True)
    deadline = models.DateTimeField()
    for_semsec = models.ForeignKey (Class_Section, on_delete =
    models.CASCADE)
    for_subject= models.ForeignKey(Subject,on_delete=models.CASCADE)
    assigned_by=models.ForeignKey(Teacher,on_delete=models.CASCADE)
```

The above snipped is analogous to the following SQL DDL(Data Definition Language):

SQL command –

```
CREATE TABLE project{
p_id INTEGER primary key,
activity_name varchar(100),
activity_description varchar(200),
date_created DATE,
last_updated DATE,
deadline DATE,
for_Semsec references class_section(semsec) ON DELETE CASCADE,
for_subject references subject(subj_code) ON DELETE CASCADE,
assigned_by references teacher(tid) ON DELETE CASCADE};
```



```
mysql> desc projects;
+----------------------+--------------+------+-----+---------+----------------+
| Field                | Type         | Null | Key | Default | Extra          |
+----------------------+--------------+------+-----+---------+----------------+
| id                   | bigint       | NO   | PRI | NULL    | auto_increment |
| activity_name        | varchar(100) | NO   |     | NULL    |                |
| date_created         | datetime(6)  | YES  |     | NULL    |                |
| last_updated         | datetime(6)  | YES  |     | NULL    |                |
| deadline             | datetime(6)  | YES  |     | NULL    |                |
| assigned_by_id       | varchar(10)  | NO   | MUL | NULL    |                |
| for_semsec_id        | varchar(2)   | NO   | MUL | NULL    |                |
| for_subject_id       | varchar(8)   | NO   | MUL | NULL    |                |
| activity_description | longtext     | YES  |     | NULL    |                |
+----------------------+--------------+------+-----+---------+----------------+
9 rows in set (0.01 sec)
```

Fig 3.4.1f : Figure showing description for Project table

Fig 3.4.1f shows a MySQL command line snapshot of table Projects description with id as Primary Key and assigned_by_id, for_semsec_id, for_subject_id as Foreign Key from table

Teacher, Class_Section and Subjects and activity_name, date_created, last_updated, deadline and activity_description as attributes.

SubmitProject Table

Django snippet –

```
class SubmitProject(models.Model):
    project_title = models.CharField(max_length=100)
    source_url = models.URLField(max_length=400)
    grade = models.IntegerField()
    submitted_by=models.ForeignKey(Student,on_delete=models.CASCADE)
    submisstion_date = models.DateTimeField(auto_now_add=True)
  for_project_id=models.ForeignKey(Project,on_delete=models.CASCADE)
```

The above snipped is analogous to the following SQL DDL(Data Definition Language):

SQL command –

CREATE TABLE submitproject{

sid INTEGER primary key,

project_title varchar(100),

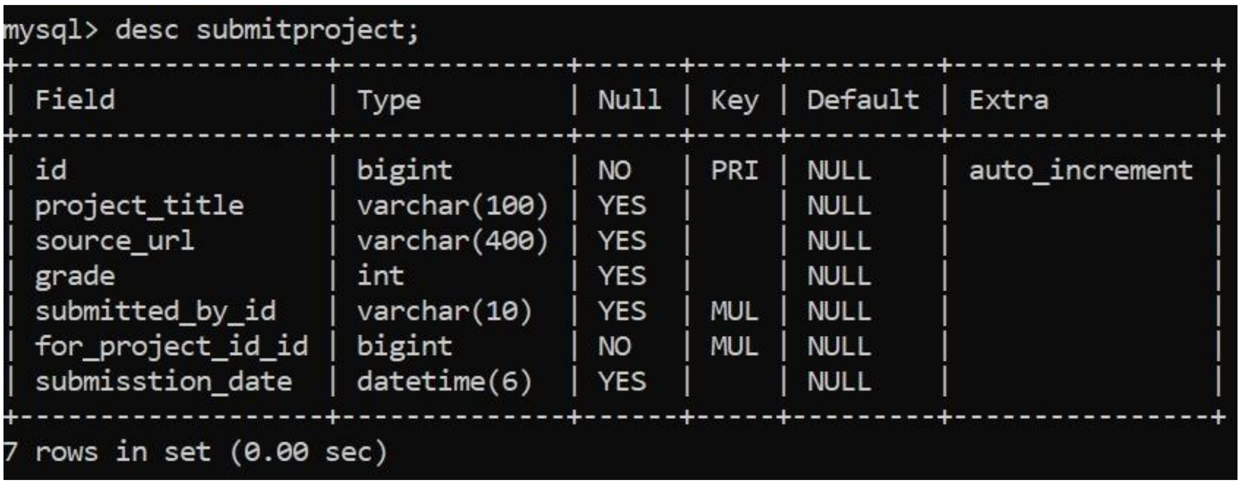
source_url varchar(400),

grade INTEGER,

submission_date DATE,

submitted_by references student(usn) ON DELETE CASCADE,

for_project_id references project(pid) };



```
mysql> desc submitproject;
+-------------------+--------------+------+-----+---------+----------------+
| Field             | Type         | Null | Key | Default | Extra          |
+-------------------+--------------+------+-----+---------+----------------+
| id                | bigint       | NO   | PRI | NULL    | auto_increment |
| project_title     | varchar(100) | YES  |     | NULL    |                |
| source_url        | varchar(400) | YES  |     | NULL    |                |
| grade             | int          | YES  |     | NULL    |                |
| submitted_by_id   | varchar(10)  | YES  | MUL | NULL    |                |
| for_project_id_id | bigint       | NO   | MUL | NULL    |                |
| submisstion_date  | datetime(6)  | YES  |     | NULL    |                |
+-------------------+--------------+------+-----+---------+----------------+
7 rows in set (0.00 sec)
```

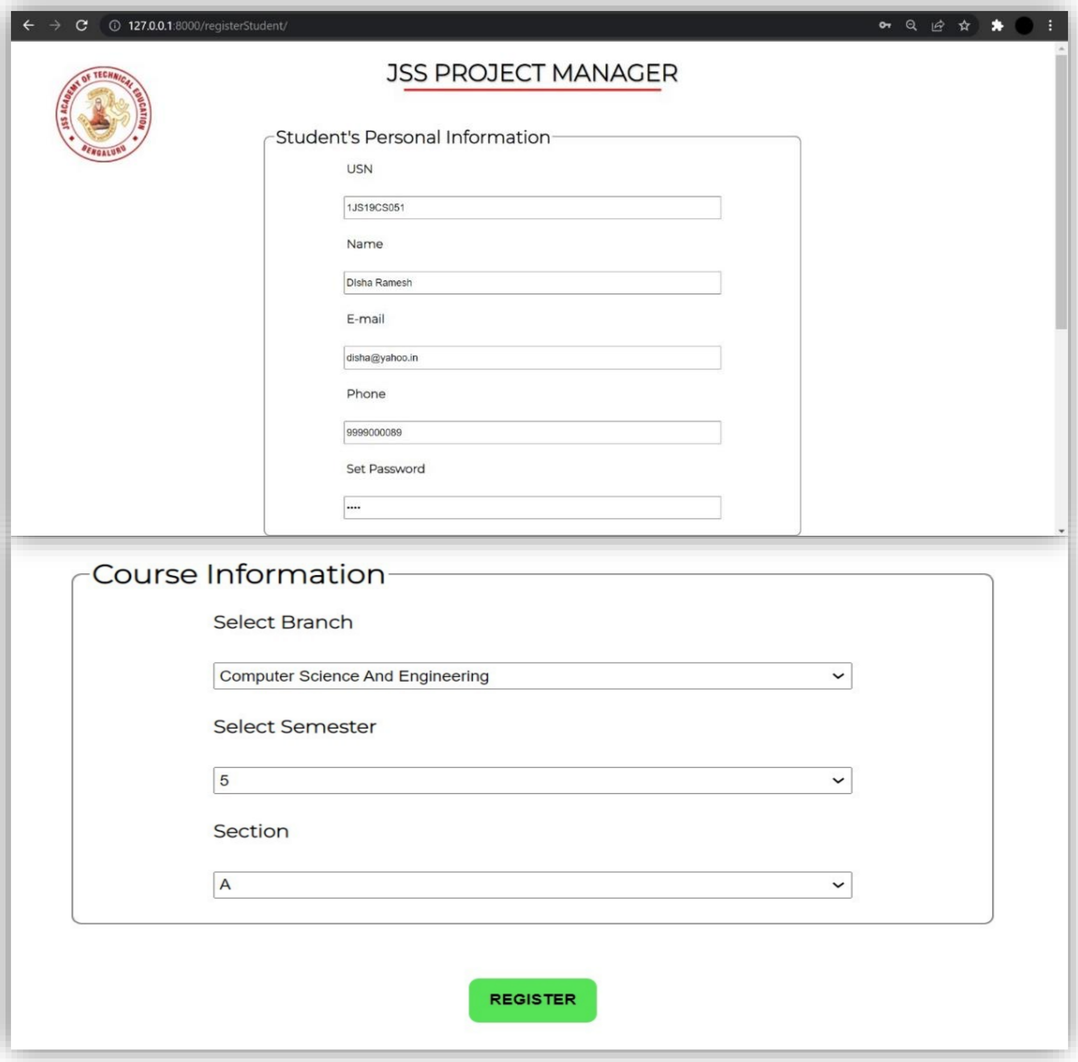
Fig 3.4.1g : Desc Table SubmitProject

Fig 3.4.1g shows a MySQL command line snapshot of table SubmitProject description with id as Primary Key and submitted_by_id, for_project_id as Foreign Key from table Student and Project and project_title, source_url, grade and submission_date as attributes.

3.4.2 Insertion

In Student Project Management System (SPMS), we have provided HTML pages that renders out required forms according to the place/table in the database that the information is to be stored. Data is collected in the frontend and passed on to the database on successful validation of entered data which is performed in the backend logic.

Example(1): New user registration (Registering User is a Student)



127.0.0.1:8000/registerStudent/

JSS PROJECT MANAGER

Student's Personal Information

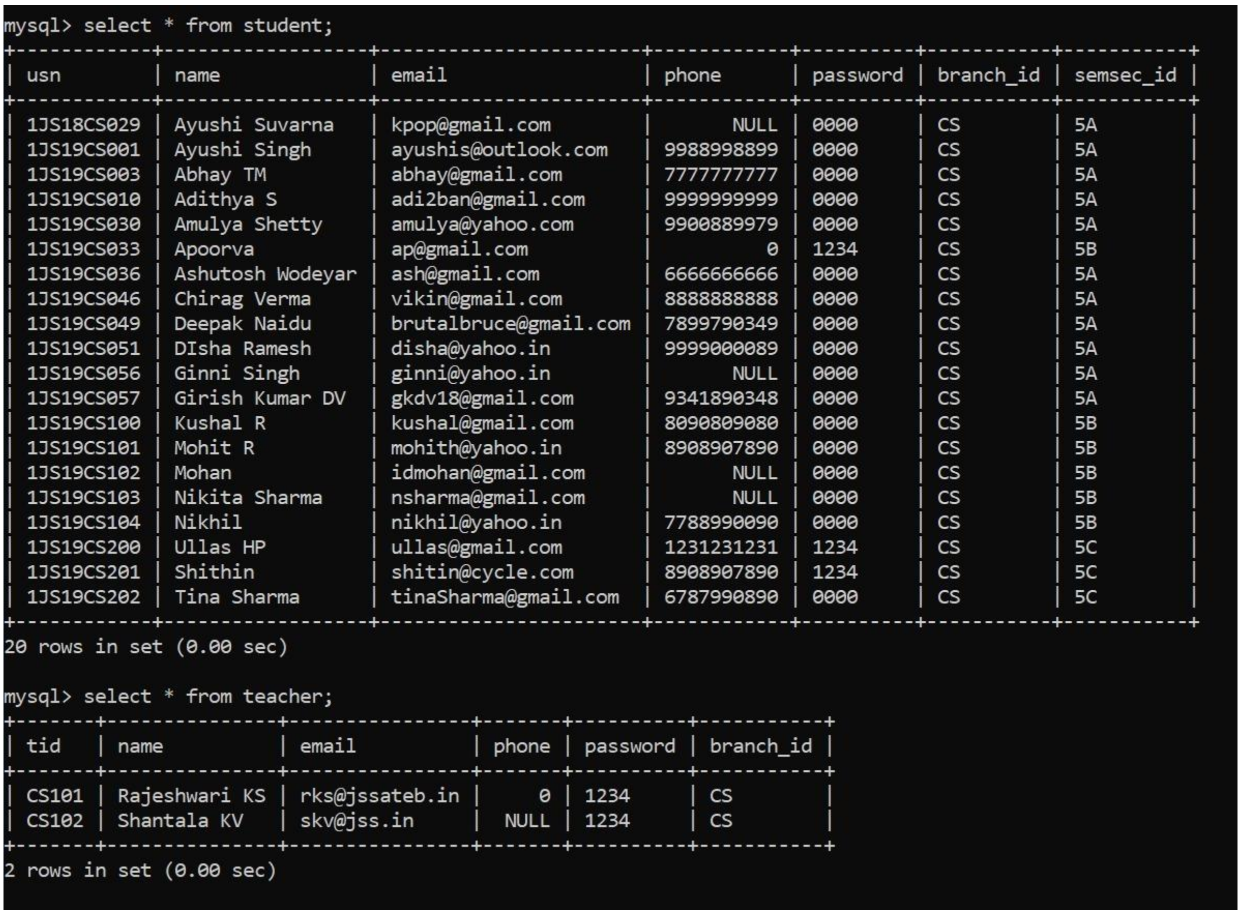
USN	1JS19CS051
Name	Disha Ramesh
E-mail	disha@yahoo.in
Phone	9999000089
Set Password	····

Course Information

Select Branch	Computer Science And Engineering
Select Semester	5
Section	A

Fig 3.4.2a : Frontend Data Insertion (registration)

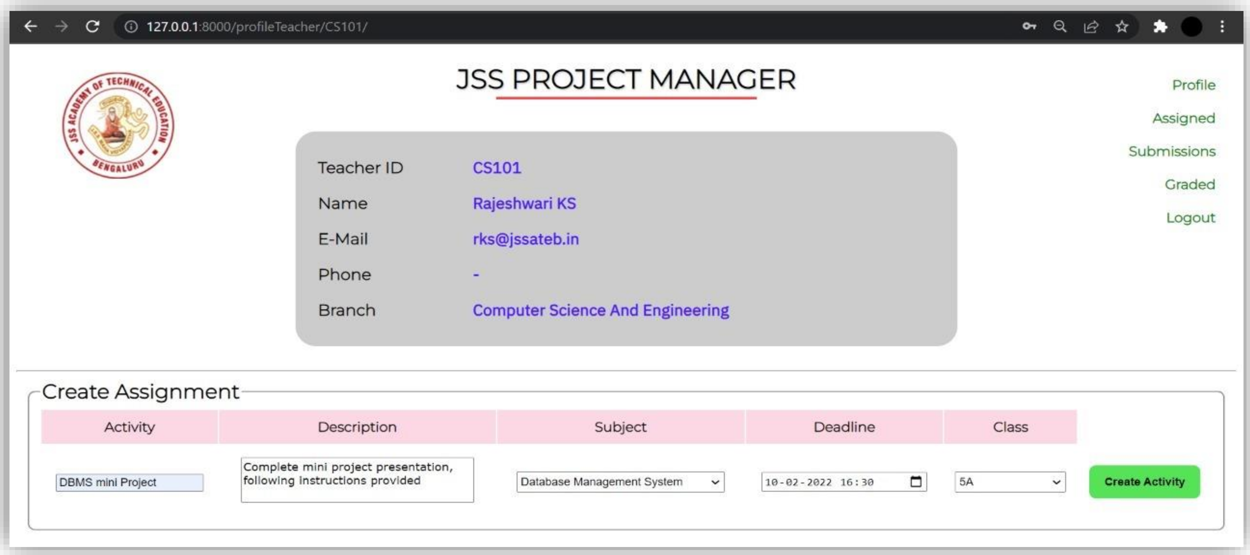
Fig 3.4.2a shows data insertion of a Student with HTML forms, this entered data is stored in the database in Table Student.



```
mysql> select * from student;
+------------+------------------+-----------------------+------------+----------+-----------+-----------+
| usn        | name             | email                 | phone      | password | branch_id | semsec_id |
+------------+------------------+-----------------------+------------+----------+-----------+-----------+
| 1JS18CS029 | Ayushi Suvarna   | kpop@gmail.com        |       NULL | 0000     | CS        | 5A        |
| 1JS19CS001 | Ayushi Singh     | ayushis@outlook.com   | 9988998899 | 0000     | CS        | 5A        |
| 1JS19CS003 | Abhay TM         | abhay@gmail.com       | 7777777777 | 0000     | CS        | 5A        |
| 1JS19CS010 | Adithya S        | adi2ban@gmail.com     | 9999999999 | 0000     | CS        | 5A        |
| 1JS19CS030 | Amulya Shetty    | amulya@yahoo.com      | 9900889979 | 0000     | CS        | 5A        |
| 1JS19CS033 | Apoorva          | ap@gmail.com          |          0 | 1234     | CS        | 5B        |
| 1JS19CS036 | Ashutosh Wodeyar | ash@gmail.com         | 6666666666 | 0000     | CS        | 5A        |
| 1JS19CS046 | Chirag Verma     | vikin@gmail.com       | 8888888888 | 0000     | CS        | 5A        |
| 1JS19CS049 | Deepak Naidu     | brutalbruce@gmail.com | 7899790349 | 0000     | CS        | 5A        |
| 1JS19CS051 | DIsha Ramesh     | disha@yahoo.in        | 9999000089 | 0000     | CS        | 5A        |
| 1JS19CS056 | Ginni Singh      | ginni@yahoo.in        |       NULL | 0000     | CS        | 5A        |
| 1JS19CS057 | Girish Kumar DV  | gkdv18@gmail.com      | 9341890348 | 0000     | CS        | 5A        |
| 1JS19CS100 | Kushal R         | kushal@gmail.com      | 8090809080 | 0000     | CS        | 5B        |
| 1JS19CS101 | Mohit R          | mohith@yahoo.in       | 8908907890 | 0000     | CS        | 5B        |
| 1JS19CS102 | Mohan            | idmohan@gmail.com     |       NULL | 0000     | CS        | 5B        |
| 1JS19CS103 | Nikita Sharma    | nsharma@gmail.com     |       NULL | 0000     | CS        | 5B        |
| 1JS19CS104 | Nikhil           | nikhil@yahoo.in       | 7788990090 | 0000     | CS        | 5B        |
| 1JS19CS200 | Ullas HP         | ullas@gmail.com       | 1231231231 | 1234     | CS        | 5C        |
| 1JS19CS201 | Shithin          | shitin@cycle.com      | 8908907890 | 1234     | CS        | 5C        |
| 1JS19CS202 | Tina Sharma      | tinaSharma@gmail.com  | 6787990890 | 0000     | CS        | 5C        |
+------------+------------------+-----------------------+------------+----------+-----------+-----------+
20 rows in set (0.00 sec)

mysql> select * from teacher;
+-------+---------------+-----------------+-------+----------+-----------+
| tid   | name          | email           | phone | password | branch_id |
+-------+---------------+-----------------+-------+----------+-----------+
| CS101 | Rajeshwari KS | rks@jssateb.in  |     0 | 1234     | CS        |
| CS102 | Shantala KV   | skv@jss.in      |  NULL | 1234     | CS        |
+-------+---------------+-----------------+-------+----------+-----------+
2 rows in set (0.00 sec)
```

Fig 3.4.2b : SQL – Student and teacher data

Fig 3.4.2b shows MySQL command line snapshot of all details of all registered users i.e. Teachers and Students.



127.0.0.1:8000/profileTeacher/CS101/

JSS PROJECT MANAGER

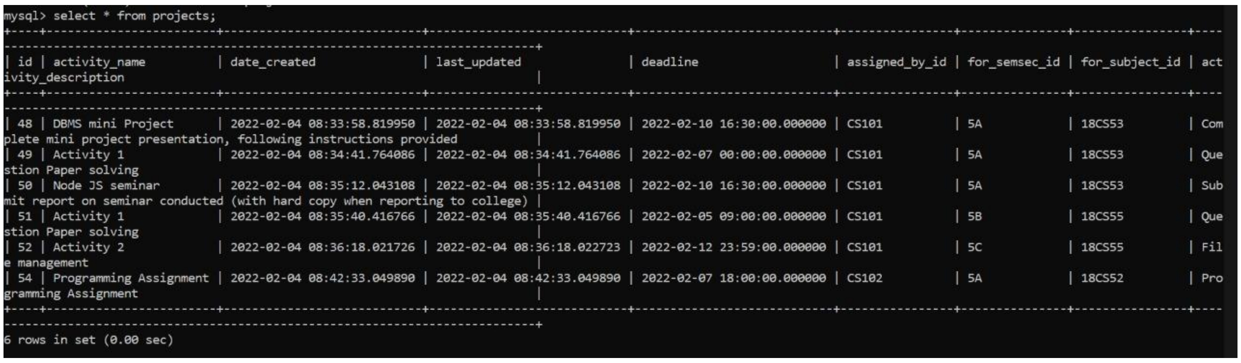
Teacher ID	CS101
Name	Rajeshwari KS
E-Mail	rks@jssateb.in
Phone	-
Branch	Computer Science And Engineering

Create Assignment

Activity	Description	Subject	Deadline	Class	
DBMS mini Project	Complete mini project presentation, following instructions provided	Database Management System	10-02-2022 16:30	5A	Create Activity

Fig 3.4.2c : Teacher View – add activity

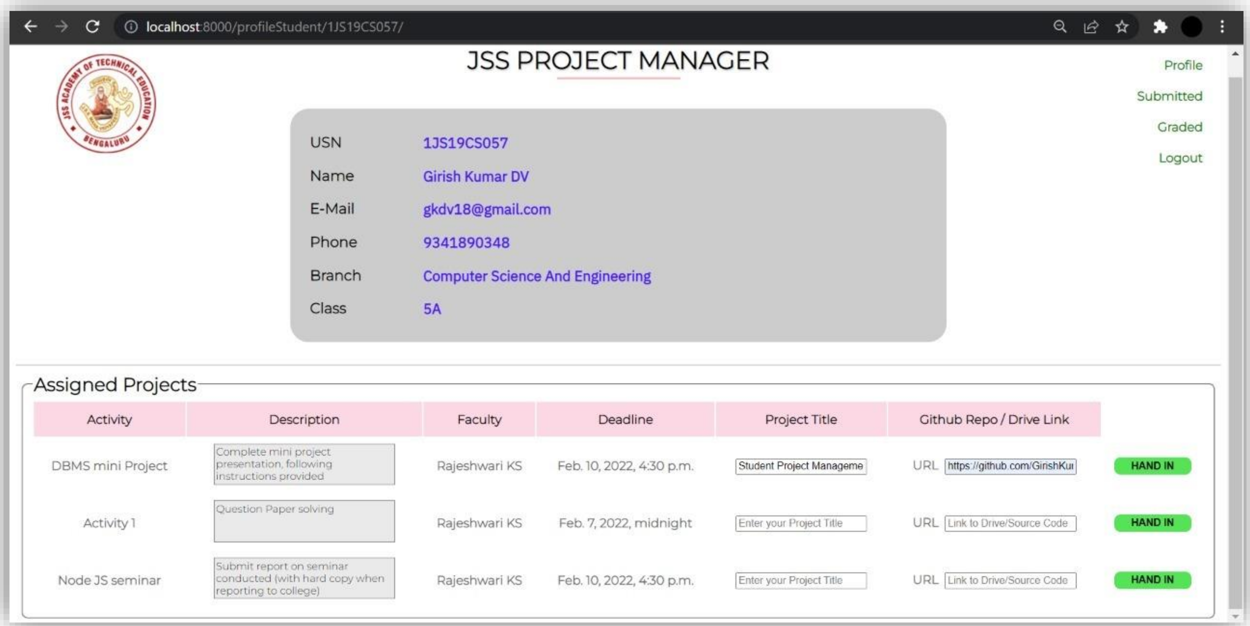
Fig 3.4.2c shows Teacher View of the application where the Teacher can add/assign and activity to a class and set deadline.



```
mysql> select * from projects;
+----+----------------------+----------------------------+----------------------------+----------------------------+----------------+---------------+----------------+----
------------------------------------------------------------------+
| id | activity_name        | date_created               | last_updated               | deadline                   | assigned_by_id | for_semsec_id | for_subject_id | act
ivity_description                                                 |
+----+----------------------+----------------------------+----------------------------+----------------------------+----------------+---------------+----------------+----
------------------------------------------------------------------+
| 48 | DBMS mini Project    | 2022-02-04 08:33:58.819950 | 2022-02-04 08:33:58.819950 | 2022-02-10 16:30:00.000000 | CS101          | 5A            | 18CS53         | Com
plete mini project presentation, following instructions provided |
| 49 | Activity 1           | 2022-02-04 08:34:41.764086 | 2022-02-04 08:34:41.764086 | 2022-02-07 00:00:00.000000 | CS101          | 5A            | 18CS53         | Que
stion Paper solving                                               |
| 50 | Node JS seminar      | 2022-02-04 08:35:12.043108 | 2022-02-04 08:35:12.043108 | 2022-02-10 16:30:00.000000 | CS101          | 5A            | 18CS53         | Sub
mit report on seminar conducted (with hard copy when reporting to college) |
| 51 | Activity 1           | 2022-02-04 08:35:40.416766 | 2022-02-04 08:35:40.416766 | 2022-02-05 09:00:00.000000 | CS101          | 5B            | 18CS55         | Que
stion Paper solving                                               |
| 52 | Activity 2           | 2022-02-04 08:36:18.021726 | 2022-02-04 08:36:18.022723 | 2022-02-12 23:59:00.000000 | CS101          | 5C            | 18CS55         | Fil
e management                                                      |
| 54 | Programming Assignment | 2022-02-04 08:42:33.049890 | 2022-02-04 08:42:33.049890 | 2022-02-07 18:00:00.000000 | CS102          | 5A            | 18CS52         | Pro
gramming Assignment                                               |
+----+----------------------+----------------------------+----------------------------+----------------------------+----------------+---------------+----------------+----
------------------------------------------------------------------+
6 rows in set (0.00 sec)
```

Fig 3.4.2d : Project Table data

Fig 3.4.2d shows all details of the assigned projects/activities by the teachers.



localhost:8000/profileStudent/1JS19CS057/

JSS PROJECT MANAGER

Profile
Submitted
Graded
Logout

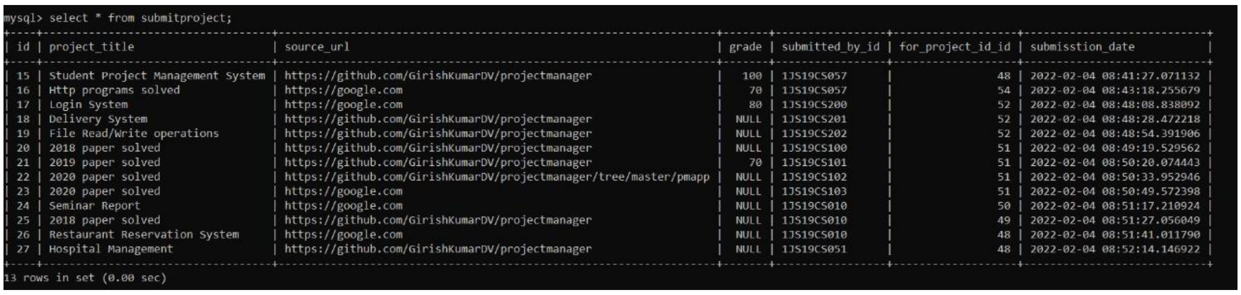
USN	1JS19CS057
Name	Girish Kumar DV
E-Mail	gkdv18@gmail.com
Phone	9341890348
Branch	Computer Science And Engineering
Class	5A

Assigned Projects

Activity	Description	Faculty	Deadline	Project Title	Github Repo / Drive Link	
DBMS mini Project	Complete mini project presentation, following instructions provided	Rajeshwari KS	Feb. 10, 2022, 4:30 p.m.	Student Project Manageme	URL https://github.com/GirishKur	HAND IN
Activity 1	Question Paper solving	Rajeshwari KS	Feb. 7, 2022, midnight	Enter your Project Title	URL Link to Drive/Source Code	HAND IN
Node JS seminar	Submit report on seminar conducted (with hard copy when reporting to college)	Rajeshwari KS	Feb. 10, 2022, 4:30 p.m.	Enter your Project Title	URL Link to Drive/Source Code	HAND IN

Fig 3.4.2e : Student View – Hand In Project

Fig 3.4.2e shows all the pending submissions of that student for those projects. Student can upload their work by specifying their project title and the source URL to their project.



```
mysql> select * from submitproject;
+----+-------------------------------+----------------------------------------------------------------+-------+-----------------+-------------------+----------------------------+
| id | project_title                 | source_url                                                     | grade | submitted_by_id | for_project_id_id | submisstion_date           |
+----+-------------------------------+----------------------------------------------------------------+-------+-----------------+-------------------+----------------------------+
| 15 | Student Project Management System | https://github.com/GirishKumarDV/projectmanager            |   100 | 1JS19CS057      |                48 | 2022-02-04 08:41:27.071132 |
| 16 | Http programs solved          | https://google.com                                             |    70 | 1JS19CS057      |                54 | 2022-02-04 08:43:18.255679 |
| 17 | Login System                  | https://google.com                                             |    80 | 1JS19CS200      |                52 | 2022-02-04 08:48:08.838092 |
| 18 | Delivery System               | https://github.com/GirishKumarDV/projectmanager                | NULL  | 1JS19CS201      |                52 | 2022-02-04 08:48:28.472218 |
| 19 | File Read/Write operations    | https://github.com/GirishKumarDV/projectmanager                | NULL  | 1JS19CS202      |                52 | 2022-02-04 08:48:54.391906 |
| 20 | 2018 paper solved             | https://github.com/GirishKumarDV/projectmanager                | NULL  | 1JS19CS100      |                51 | 2022-02-04 08:49:19.529562 |
| 21 | 2019 paper solved             | https://github.com/GirishKumarDV/projectmanager                |    70 | 1JS19CS101      |                51 | 2022-02-04 08:50:20.074443 |
| 22 | 2020 paper solved             | https://github.com/GirishKumarDV/projectmanager/tree/master/pmapp | NULL | 1JS19CS102     |                51 | 2022-02-04 08:50:33.952946 |
| 23 | 2020 paper solved             | https://google.com                                             | NULL  | 1JS19CS103      |                51 | 2022-02-04 08:50:49.572398 |
| 24 | Seminar Report                | https://google.com                                             | NULL  | 1JS19CS010      |                50 | 2022-02-04 08:51:17.210924 |
| 25 | 2018 paper solved             | https://github.com/GirishKumarDV/projectmanager                | NULL  | 1JS19CS010      |                49 | 2022-02-04 08:51:27.056049 |
| 26 | Restaurant Reservation System | https://google.com                                             | NULL  | 1JS19CS010      |                48 | 2022-02-04 08:51:41.011790 |
| 27 | Hospital Management           | https://github.com/GirishKumarDV/projectmanager                | NULL  | 1JS19CS051      |                48 | 2022-02-04 08:52:14.146922 |
+----+-------------------------------+----------------------------------------------------------------+-------+-----------------+-------------------+----------------------------+
13 rows in set (0.00 sec)
```

Fig 3.4.2f : SQL – SubmitProject Data

Fig 3.4.2f shows all the data of all the submissions by the students to that particular project in the column for_project_id, grade column is initially set to NULL, this is changed by teacher.

Some pre-requisite data can be entered statically (hardcoded)

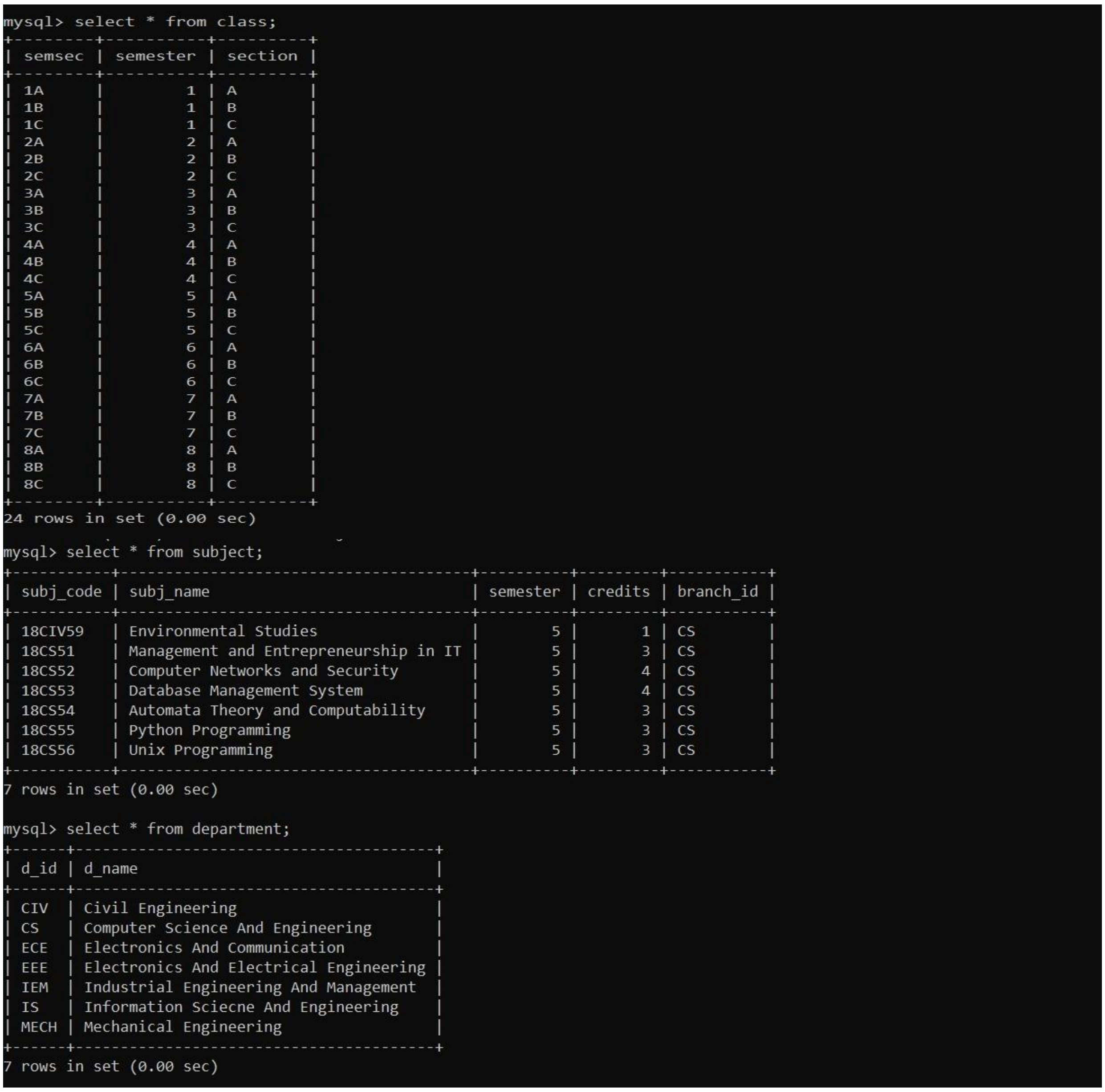
Ex: Data required in Subjecs table, Class_section table, and Department Table

SQL Queries examples:

INSERT INTO DEPARTMENT VALUES(“CS”,”Computer Science And Engineering”);

INSERT INTO SUBJECT VALUES(“18CS52”,”Computer Networks And Security”,5,4,”CS”);

INSERT INTO CLASS_SECTION VALUES(“5A”,5,”A”);



```
mysql> select * from class;
+--------+----------+---------+
| semsec | semester | section |
+--------+----------+---------+
| 1A     |        1 | A       |
| 1B     |        1 | B       |
| 1C     |        1 | C       |
| 2A     |        2 | A       |
| 2B     |        2 | B       |
| 2C     |        2 | C       |
| 3A     |        3 | A       |
| 3B     |        3 | B       |
| 3C     |        3 | C       |
| 4A     |        4 | A       |
| 4B     |        4 | B       |
| 4C     |        4 | C       |
| 5A     |        5 | A       |
| 5B     |        5 | B       |
| 5C     |        5 | C       |
| 6A     |        6 | A       |
| 6B     |        6 | B       |
| 6C     |        6 | C       |
| 7A     |        7 | A       |
| 7B     |        7 | B       |
| 7C     |        7 | C       |
| 8A     |        8 | A       |
| 8B     |        8 | B       |
| 8C     |        8 | C       |
+--------+----------+---------+
24 rows in set (0.00 sec)

mysql> select * from subject;
+----------+-------------------------------------+----------+---------+-----------+
| subj_code | subj_name                          | semester | credits | branch_id |
+----------+-------------------------------------+----------+---------+-----------+
| 18CIV59  | Environmental Studies               |        5 |       1 | CS        |
| 18CS51   | Management and Entrepreneurship in IT |      5 |       3 | CS        |
| 18CS52   | Computer Networks and Security      |        5 |       4 | CS        |
| 18CS53   | Database Management System          |        5 |       4 | CS        |
| 18CS54   | Automata Theory and Computability   |        5 |       3 | CS        |
| 18CS55   | Python Programming                  |        5 |       3 | CS        |
| 18CS56   | Unix Programming                    |        5 |       3 | CS        |
+----------+-------------------------------------+----------+---------+-----------+
7 rows in set (0.00 sec)

mysql> select * from department;
+------+-------------------------------------+
| d_id | d_name                              |
+------+-------------------------------------+
| CIV  | Civil Engineering                   |
| CS   | Computer Science And Engineering    |
| ECE  | Electronics And Communication       |
| EEE  | Electronics And Electrical Engineering |
| IEM  | Industrial Engineering And Management |
| IS   | Information Sciecne And Engineering |
| MECH | Mechanical Engineering              |
+------+-------------------------------------+
7 rows in set (0.00 sec)
```

Fig 3.4.2g : SQL-ClassSection, Department and Subject Data

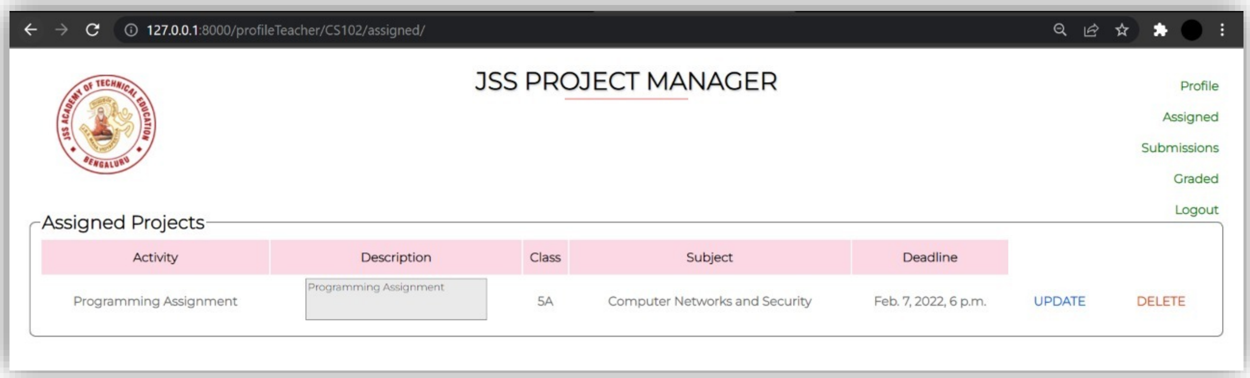
Fig 3.4.2g shows all the data entered from the command line for the tables, Class_Section, Department and Subject.

3.4.3 Deletion & Updating

Deletion and updating are crucial functions that needs to be provided to any DBMS application, in Student Project Management System (SPMS) we have given user widgets through which a user can either update or delete projects and functionalities for these widgets differ depending upon type of user i.e. Student or Teacher.

Update option for Teacher gives the following privileges:

- Teacher can update activity description
- Teacher can update (extend / prepone) deadline


127.0.0.1:8000/profileTeacher/CS102/assigned/

JSS PROJECT MANAGER

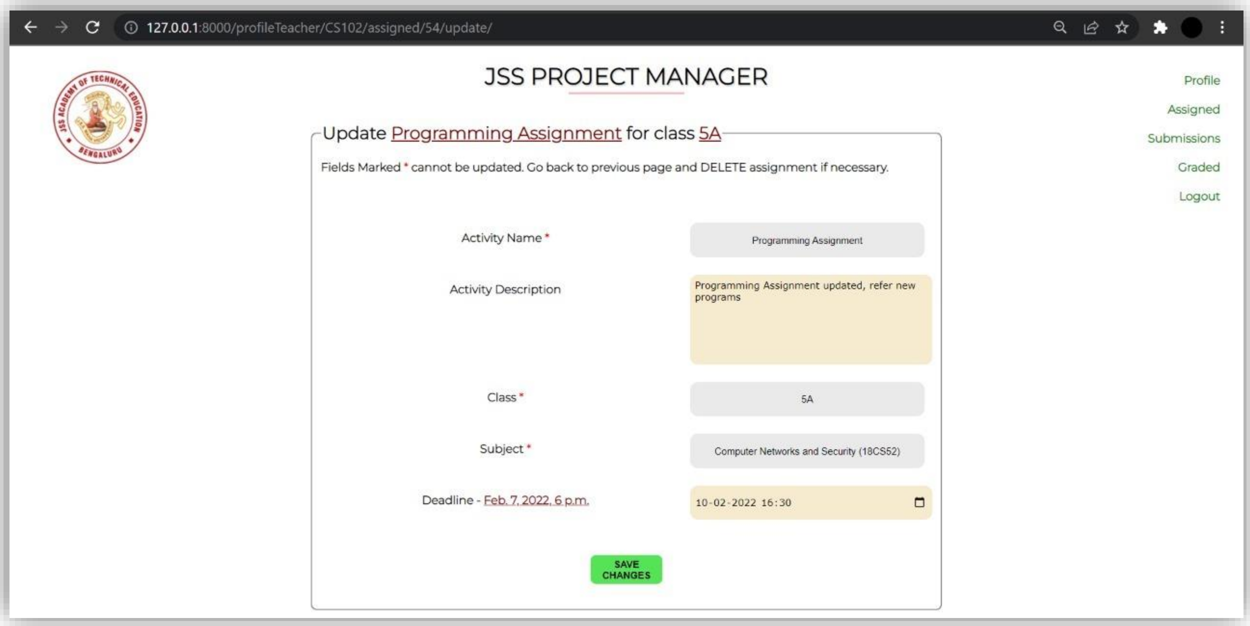
Profile
Assigned
Submissions
Graded
Logout

Assigned Projects

Activity	Description	Class	Subject	Deadline		
Programming Assignment	Programming Assignment	5A	Computer Networks and Security	Feb. 7, 2022, 6 p.m.	UPDATE	DELETE

Fig 3.4.3a : Teacher View – Before Update.

Fig 3.4.3a shows Teacher View – Assigned tab, before updating an assignment.


127.0.0.1:8000/profileTeacher/CS102/assigned/54/update/

JSS PROJECT MANAGER

Profile
Assigned
Submissions
Graded
Logout

Update Programming Assignment for class 5A

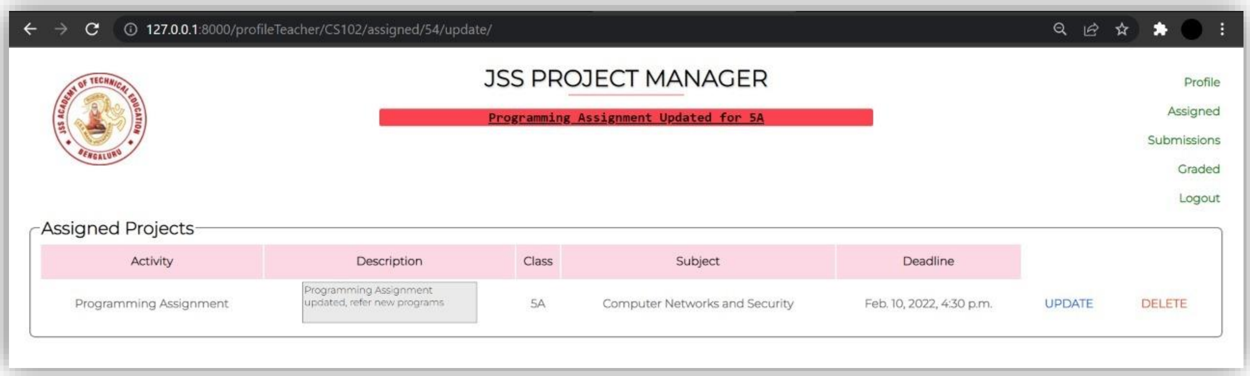
Fields Marked * cannot be updated. Go back to previous page and DELETE assignment if necessary.

Activity Name *	Programming Assignment
Activity Description	Programming Assignment updated, refer new programs
Class *	5A
Subject *	Computer Networks and Security (18CS52)
Deadline - Feb. 7, 2022, 6 p.m.	10-02-2022 16:30

SAVE CHANGES

Fig 3.4.3b: Teacher View – Update Page

Fig 3.4.3b shows the update page on clicking the update button and the updatable fields are shows in cream color while grey colored fields are non-updatable.



127.0.0.1:8000/profileTeacher/CS102/assigned/54/update/

JSS PROJECT MANAGER

Programming Assignment Updated for 5A

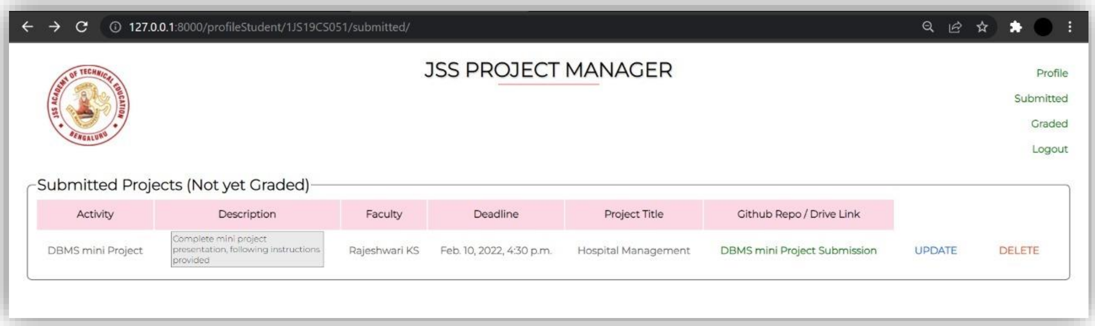
Profile
Assigned
Submissions
Graded
Logout

Assigned Projects

Activity	Description	Class	Subject	Deadline		
Programming Assignment	Programming Assignment updated, refer new programs	5A	Computer Networks and Security	Feb. 10, 2022, 4:30 p.m.	UPDATE	DELETE

Fig 3.4.3c: Teacher View – After Update

Fig 3.4.3c shows teacher assigned tab after updating, a pop up message is shown and updated details are reflected.



127.0.0.1:8000/profileStudent/1JS19CS051/submitted/

JSS PROJECT MANAGER

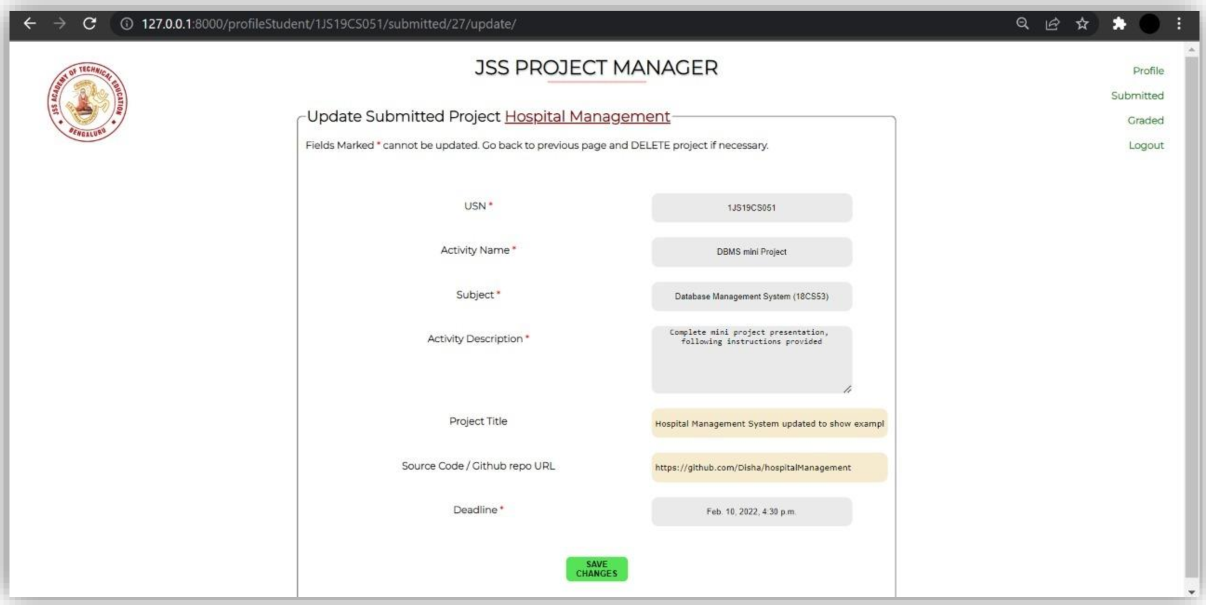
Profile
Submitted
Graded
Logout

Submitted Projects (Not yet Graded)

Activity	Description	Faculty	Deadline	Project Title	Github Repo / Drive Link		
DBMS mini Project	Complete mini project presentation, following instructions provided	Rajeshwari KS	Feb. 10, 2022, 4:30 p.m.	Hospital Management	DBMS mini Project Submission	UPDATE	DELETE

Fig 3.4.3d : Student View – Before Update

Fig 3.4.3d shows Student View – Submitted tab, before updating a submitted project.



127.0.0.1:8000/profileStudent/1JS19CS051/submitted/27/update/

JSS PROJECT MANAGER

Profile
Submitted
Graded
Logout

Update Submitted Project Hospital Management

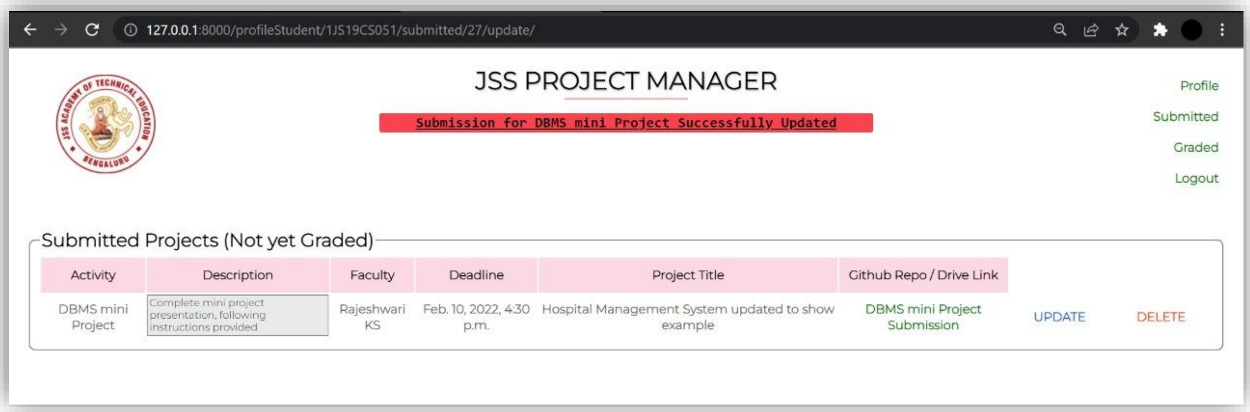
Fields Marked * cannot be updated. Go back to previous page and DELETE project if necessary.

USN *	1JS19CS051
Activity Name *	DBMS mini Project
Subject *	Database Management System (18CS53)
Activity Description *	Complete mini project presentation, following instructions provided
Project Title	Hospital Management System updated to show exampl
Source Code / Github repo URL	https://github.com/Disha/hospitalManagement
Deadline *	Feb. 10, 2022, 4.30 p.m.

SAVE CHANGES

Fig 3.4.3e : Student View – Update Page

Fig 3.4.3e shows the update page on clicking the update button and the updatable fields are shown in cream color and non-updatable fields are grey colored


127.0.0.1:8000/profileStudent/1JS19CS051/submitted/27/update/

JSS PROJECT MANAGER

Submission for DBMS mini Project Successfully Updated

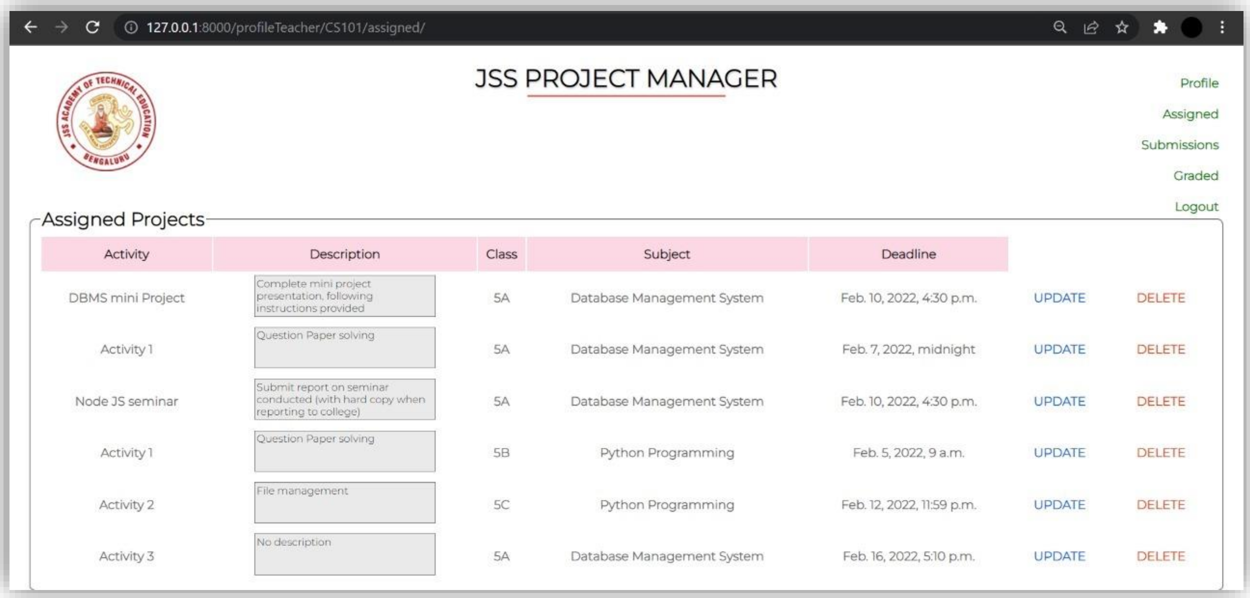
Submitted Projects (Not yet Graded)

Activity	Description	Faculty	Deadline	Project Title	Github Repo / Drive Link		
DBMS mini Project	Complete mini project presentation, following instructions provided	Rajeshwari KS	Feb. 10, 2022, 4:30 p.m.	Hospital Management System updated to show example	DBMS mini Project Submission	UPDATE	DELETE

Figure 3.4.3(f): After Update, alert message displayed and updated contents displayed

Fig 3.4.3f shows Student submitted tab after updating, a pop up message is shown and updated details are reflected.

Delete Option for teacher is given so that they can delete any assigned activity/assignment and assign any other activity or if all the students have submitted the present activity.


127.0.0.1:8000/profileTeacher/CS101/assigned/

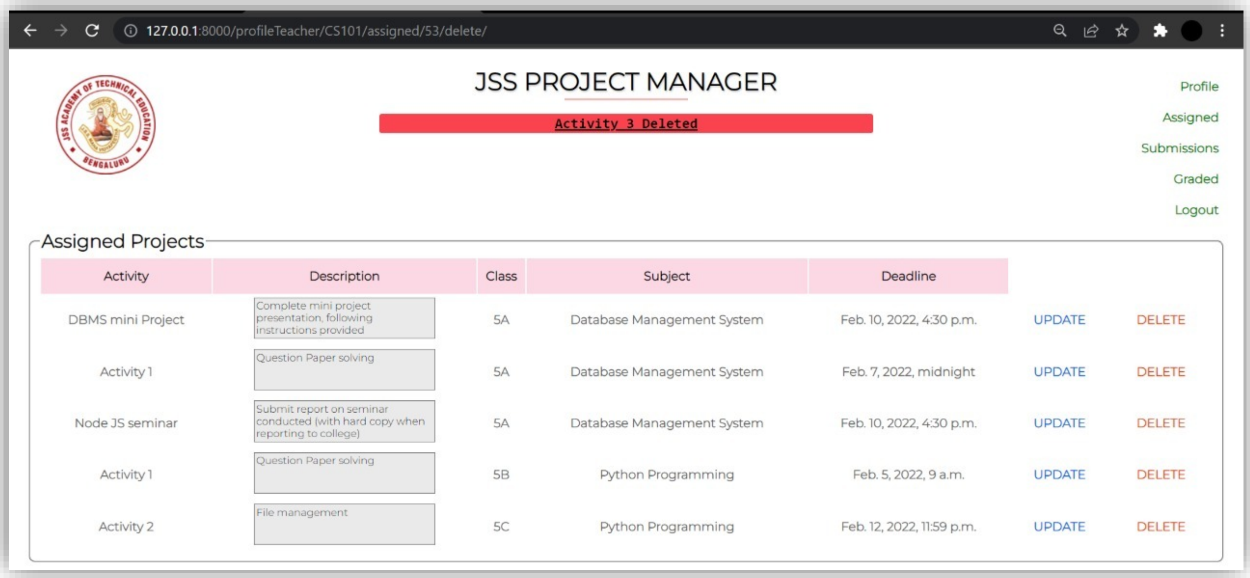
JSS PROJECT MANAGER

Assigned Projects

Activity	Description	Class	Subject	Deadline		
DBMS mini Project	Complete mini project presentation, following instructions provided	5A	Database Management System	Feb. 10, 2022, 4:30 p.m.	UPDATE	DELETE
Activity 1	Question Paper solving	5A	Database Management System	Feb. 7, 2022, midnight	UPDATE	DELETE
Node JS seminar	Submit report on seminar conducted (with hard copy when reporting to college)	5A	Database Management System	Feb. 10, 2022, 4:30 p.m.	UPDATE	DELETE
Activity 1	Question Paper solving	5B	Python Programming	Feb. 5, 2022, 9 a.m.	UPDATE	DELETE
Activity 2	File management	5C	Python Programming	Feb. 12, 2022, 11:59 p.m.	UPDATE	DELETE
Activity 3	No description	5A	Database Management System	Feb. 16, 2022, 5:10 p.m.	UPDATE	DELETE

Fig 3.4.3g: Teacher View – Before Deletion

Fig 3.4.3g shows Teacher View Assigned Tab where teacher can delete any assigned activity on clicking the delete button



127.0.0.1:8000/profileTeacher/CS101/assigned/53/delete/

JSS PROJECT MANAGER

Activity 3 Deleted

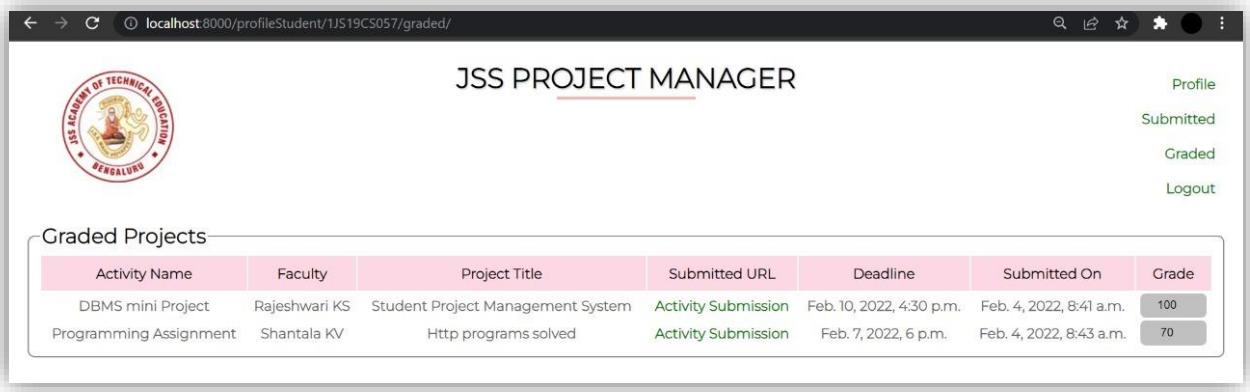
Assigned Projects

Activity	Description	Class	Subject	Deadline		
DBMS mini Project	Complete mini project presentation, following instructions provided	5A	Database Management System	Feb. 10, 2022, 4:30 p.m.	UPDATE	DELETE
Activity 1	Question Paper solving	5A	Database Management System	Feb. 7, 2022, midnight	UPDATE	DELETE
Node JS seminar	Submit report on seminar conducted (with hard copy when reporting to college)	5A	Database Management System	Feb. 10, 2022, 4:30 p.m.	UPDATE	DELETE
Activity 1	Question Paper solving	5B	Python Programming	Feb. 5, 2022, 9 a.m.	UPDATE	DELETE
Activity 2	File management	5C	Python Programming	Feb. 12, 2022, 11:59 p.m.	UPDATE	DELETE

Fig 3.4.3h: Teacher View – Submission page after Delete

Fig 3.4.3h shows Teacher View Assigned Page after teacher has deleted an assignment , pop up message is displayed and changes are reflected.

Delete Option for student is only given before the Teacher grades the project, once the project has been graded they cannot update/delete the project. They can only view the work and the marks awarded.



localhost:8000/profileStudent/1JS19CS057/graded/

JSS PROJECT MANAGER

Graded Projects

Activity Name	Faculty	Project Title	Submitted URL	Deadline	Submitted On	Grade
DBMS mini Project	Rajeshwari KS	Student Project Management System	Activity Submission	Feb. 10, 2022, 4:30 p.m.	Feb. 4, 2022, 8:41 a.m.	100
Programming Assignment	Shantala KV	Http programs solved	Activity Submission	Feb. 7, 2022, 6 p.m.	Feb. 4, 2022, 8:43 a.m.	70

Fig 3.4.3i: Student View – Graded page

Fig 3.4.3i shows Student View graded page where the student can view the marks awarded for each of their work and the details of their work (URL, deadline, title).

With this, Student Project Management System presents all the CRUD (Creation, Retrieval, Updating, Deletion) operations that can be performed on our DBMS application. Also, by providing isolated views, we make sure that there is no duplication of entries and hence prove to be efficient.

3.5 Functions

Django file structure is as shown:

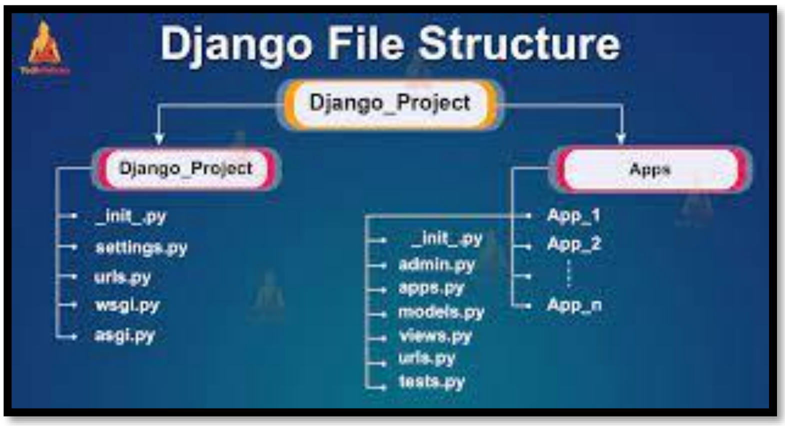
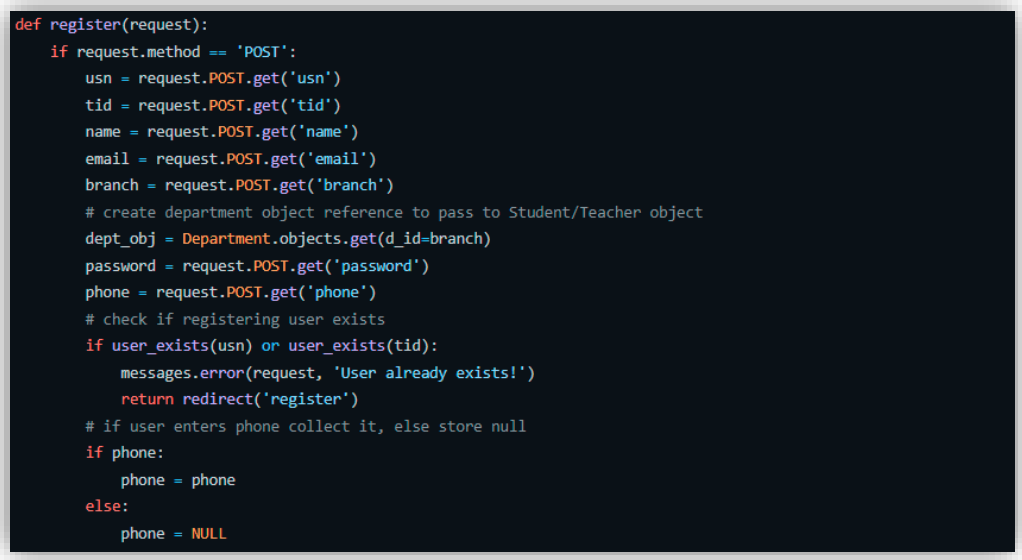


Fig 3.5 : Django File Structure

Fig 3.5 shows Django File Structure. In Django the logic like Stored procedures or the Triggers and all other logical implementations are written in the *'Apps'* `views.py` file to perform the functions and required tasks as assumed in the Requirement Analysis while describing the problem statement before starting to build the app. The `urls.py` file is used to define the URL route mappings which occur in the website. A few functions written in `views.py` file are presented below:

Functions for login and registration

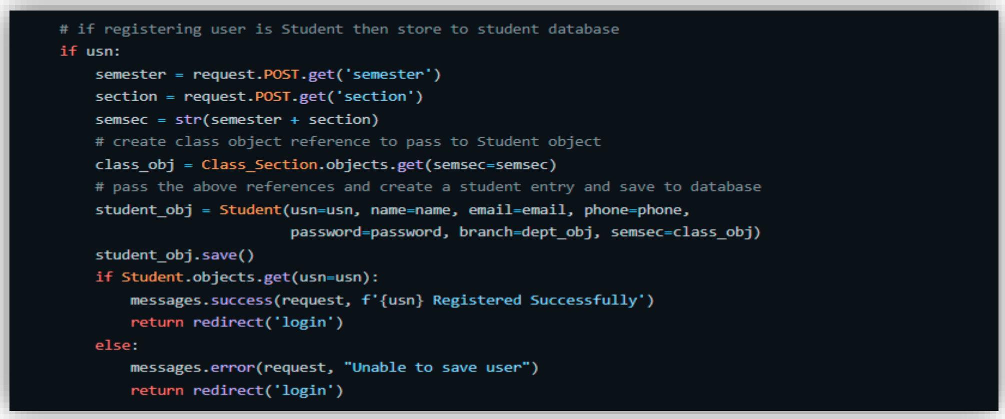
3.5.1 Registration



```
def register(request):
    if request.method == 'POST':
        usn = request.POST.get('usn')
        tid = request.POST.get('tid')
        name = request.POST.get('name')
        email = request.POST.get('email')
        branch = request.POST.get('branch')
        # create department object reference to pass to Student/Teacher object
        dept_obj = Department.objects.get(d_id=branch)
        password = request.POST.get('password')
        phone = request.POST.get('phone')
        # check if registering user exists
        if user_exists(usn) or user_exists(tid):
            messages.error(request, 'User already exists!')
            return redirect('register')
        # if user enters phone collect it, else store null
        if phone:
            phone = phone
        else:
            phone = NULL
```

Fig 3.5.1 : Views.py - register

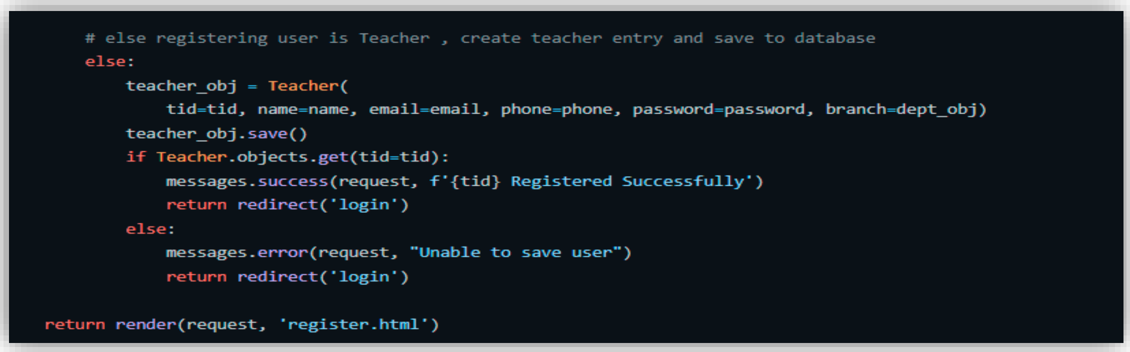
Fig 3.5.1 shows the Views.py file with the logic to register a new user writted in python.



```python
    # if registering user is Student then store to student database
    if usn:
        semester = request.POST.get('semester')
        section = request.POST.get('section')
        semsec = str(semester + section)
        # create class object reference to pass to Student object
        class_obj = Class_Section.objects.get(semsec=semsec)
        # pass the above references and create a student entry and save to database
        student_obj = Student(usn=usn, name=name, email=email, phone=phone,
                              password=password, branch=dept_obj, semsec=class_obj)
        student_obj.save()
        if Student.objects.get(usn=usn):
            messages.success(request, f'{usn} Registered Successfully')
            return redirect('login')
        else:
            messages.error(request, "Unable to save user")
            return redirect('login')
```

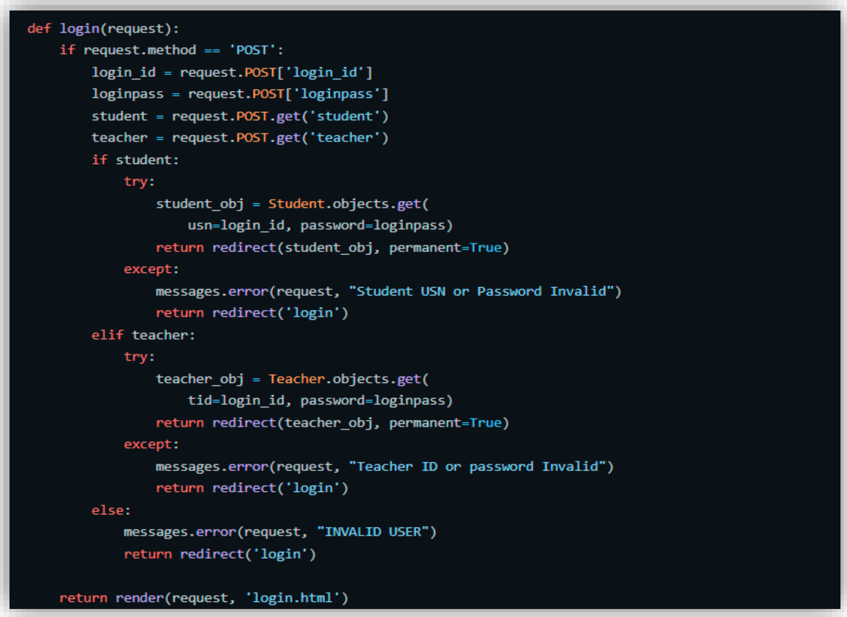
Fig 3.5.1a : Views.py – registerStudent

Fig 3.5.1a is part of Fig 3.5.1, this contains logic to register a Student.



```python
    # else registering user is Teacher , create teacher entry and save to database
    else:
        teacher_obj = Teacher(
            tid=tid, name=name, email=email, phone=phone, password=password, branch=dept_obj)
        teacher_obj.save()
        if Teacher.objects.get(tid=tid):
            messages.success(request, f'{tid} Registered Successfully')
            return redirect('login')
        else:
            messages.error(request, "Unable to save user")
            return redirect('login')

return render(request, 'register.html')
```

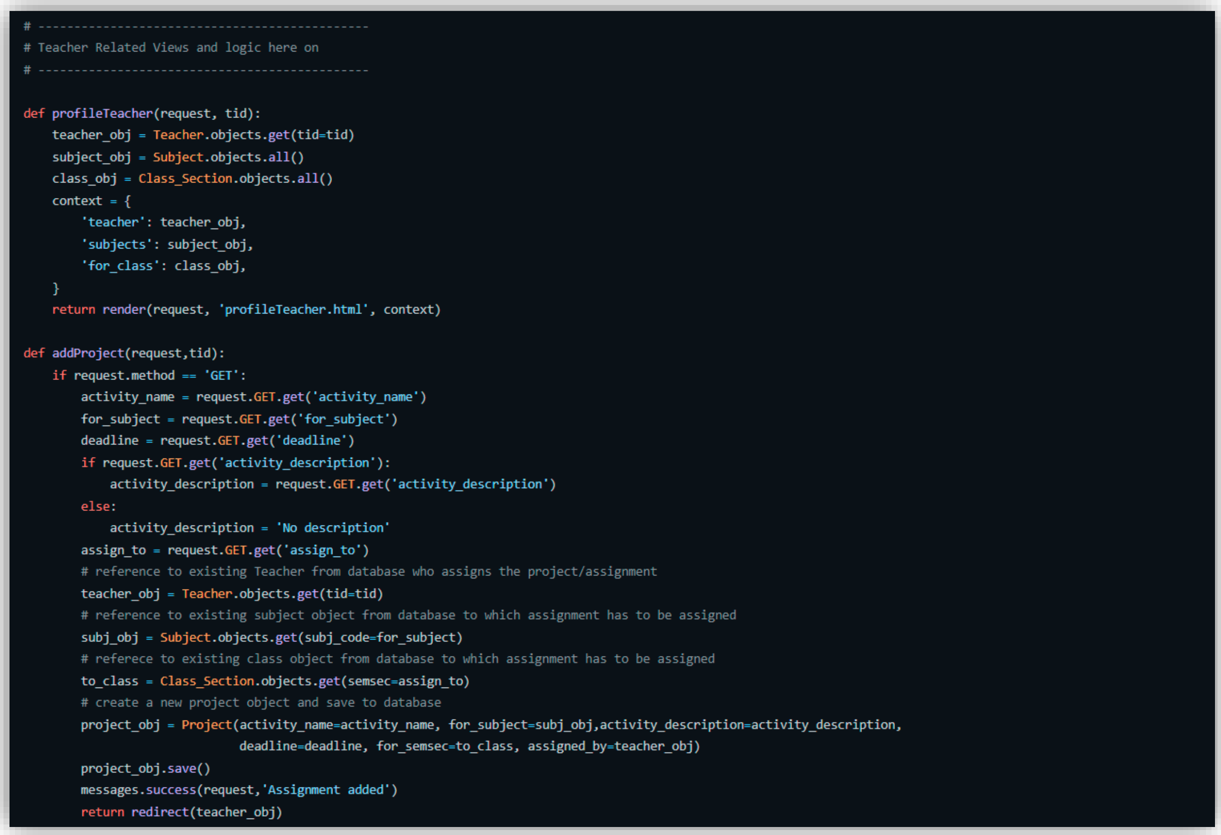
Fig 3.5.1b : Views.py – register Teacher
Fig 3.5.1b is also a part of Fig 3.5.1 and it contains the logic to register a Teacher



```python
def login(request):
    if request.method == 'POST':
        login_id = request.POST['login_id']
        loginpass = request.POST['loginpass']
        student = request.POST.get('student')
        teacher = request.POST.get('teacher')
        if student:
            try:
                student_obj = Student.objects.get(
                    usn=login_id, password=loginpass)
                return redirect(student_obj, permanent=True)
            except:
                messages.error(request, "Student USN or Password Invalid")
                return redirect('login')
        elif teacher:
            try:
                teacher_obj = Teacher.objects.get(
                    tid=login_id, password=loginpass)
                return redirect(teacher_obj, permanent=True)
            except:
                messages.error(request, "Teacher ID or password Invalid")
                return redirect('login')
        else:
            messages.error(request, "INVALID USER")
            return redirect('login')

    return render(request, 'login.html')
```

Fig 3.5.1c : Views.py – login
Fig 3.5.1c contains the logic written to authorize access to a user logging into the site. On successful authentication, access is granted else, pop-up error message is displayed.

3.5.2 Teacher Profile and add activity

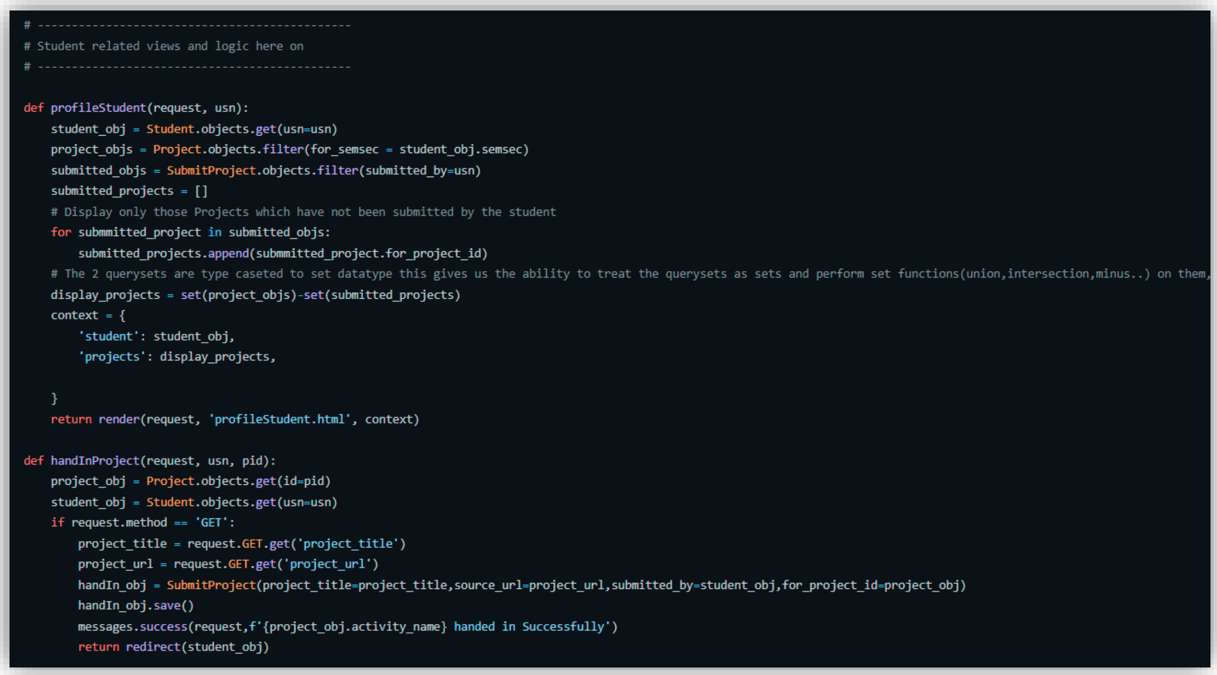


```python
# --------------------------------------------
# Teacher Related Views and logic here on
# --------------------------------------------

def profileTeacher(request, tid):
    teacher_obj = Teacher.objects.get(tid=tid)
    subject_obj = Subject.objects.all()
    class_obj = Class_Section.objects.all()
    context = {
        'teacher': teacher_obj,
        'subjects': subject_obj,
        'for_class': class_obj,
    }
    return render(request, 'profileTeacher.html', context)

def addProject(request,tid):
    if request.method == 'GET':
        activity_name = request.GET.get('activity_name')
        for_subject = request.GET.get('for_subject')
        deadline = request.GET.get('deadline')
        if request.GET.get('activity_description'):
            activity_description = request.GET.get('activity_description')
        else:
            activity_description = 'No description'
        assign_to = request.GET.get('assign_to')
        # reference to existing Teacher from database who assigns the project/assignment
        teacher_obj = Teacher.objects.get(tid=tid)
        # reference to existing subject object from database to which assignment has to be assigned
        subj_obj = Subject.objects.get(subj_code=for_subject)
        # referece to existing class object from database to which assignment has to be assigned
        to_class = Class_Section.objects.get(semsec=assign_to)
        # create a new project object and save to database
        project_obj = Project(activity_name=activity_name, for_subject=subj_obj,activity_description=activity_description,
                              deadline=deadline, for_semsec=to_class, assigned_by=teacher_obj)
        project_obj.save()
        messages.success(request,'Assignment added')
        return redirect(teacher_obj)
```

Fig 3.5.2: View.py – Teacher profile Add Project

Fig 3.5.2 shows logic for to display Teacher profile and logic to add a new assignment to database.

3.5.3 Student Profile and hand In project

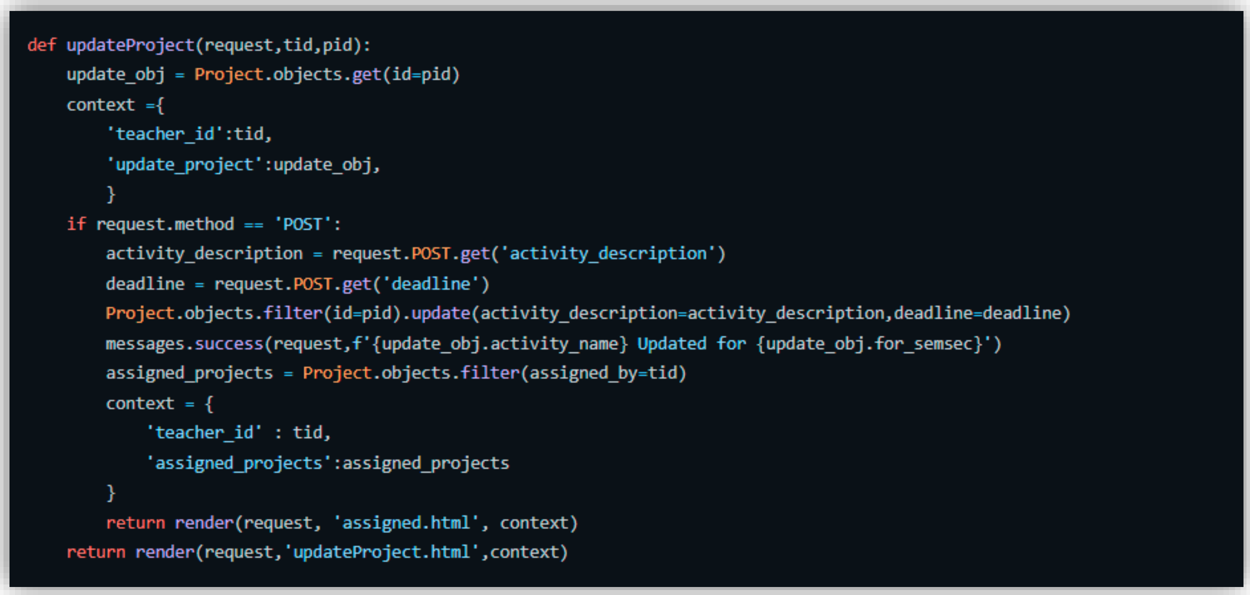


```python
# --------------------------------------------
# Student related views and logic here on
# --------------------------------------------

def profileStudent(request, usn):
    student_obj = Student.objects.get(usn=usn)
    project_objs = Project.objects.filter(for_semsec = student_obj.semsec)
    submitted_objs = SubmitProject.objects.filter(submitted_by=usn)
    submitted_projects = []
    # Display only those Projects which have not been submitted by the student
    for submmitted_project in submitted_objs:
        submitted_projects.append(submmitted_project.for_project_id)
    # The 2 querysets are type caseted to set datatype this gives us the ability to treat the querysets as sets and perform set functions(union,intersection,minus..) on them,
    display_projects = set(project_objs)-set(submitted_projects)
    context = {
        'student': student_obj,
        'projects': display_projects,

    }
    return render(request, 'profileStudent.html', context)

def handInProject(request, usn, pid):
    project_obj = Project.objects.get(id=pid)
    student_obj = Student.objects.get(usn=usn)
    if request.method == 'GET':
        project_title = request.GET.get('project_title')
        project_url = request.GET.get('project_url')
        handIn_obj = SubmitProject(project_title=project_title,source_url=project_url,submitted_by=student_obj,for_project_id=project_obj)
        handIn_obj.save()
        messages.success(request,f'{project_obj.activity_name} handed in Successfully')
        return redirect(student_obj)
```

Fig 3.5.3: Views.py Student Profile Hand In Project

Fig 3.5.3 shows logic to display Student profile and accept a project handed in by student and store to database.

3.5.4 Updating details of Assigned Project (Teacher View - description / deadline)

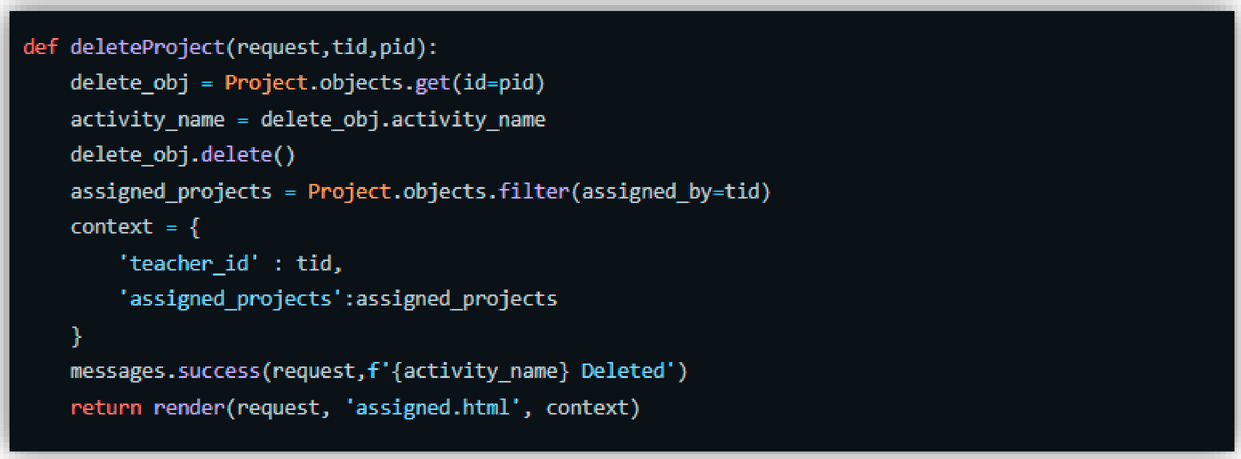


```python
def updateProject(request,tid,pid):
    update_obj = Project.objects.get(id=pid)
    context ={
        'teacher_id':tid,
        'update_project':update_obj,
        }
    if request.method == 'POST':
        activity_description = request.POST.get('activity_description')
        deadline = request.POST.get('deadline')
        Project.objects.filter(id=pid).update(activity_description=activity_description,deadline=deadline)
        messages.success(request,f'{update_obj.activity_name} Updated for {update_obj.for_semsec}')
        assigned_projects = Project.objects.filter(assigned_by=tid)
        context = {
            'teacher_id' : tid,
            'assigned_projects':assigned_projects
        }
        return render(request, 'assigned.html', context)
    return render(request,'updateProject.html',context)
```

Fig 3.5.4: Views.py – Update Project (Teacher)

Fig 3.5.4 shows logic written to update record of an assignment in database when teacher clicks on update button.

3.5.5 Deleting an assigned project/activity (Teacher View)



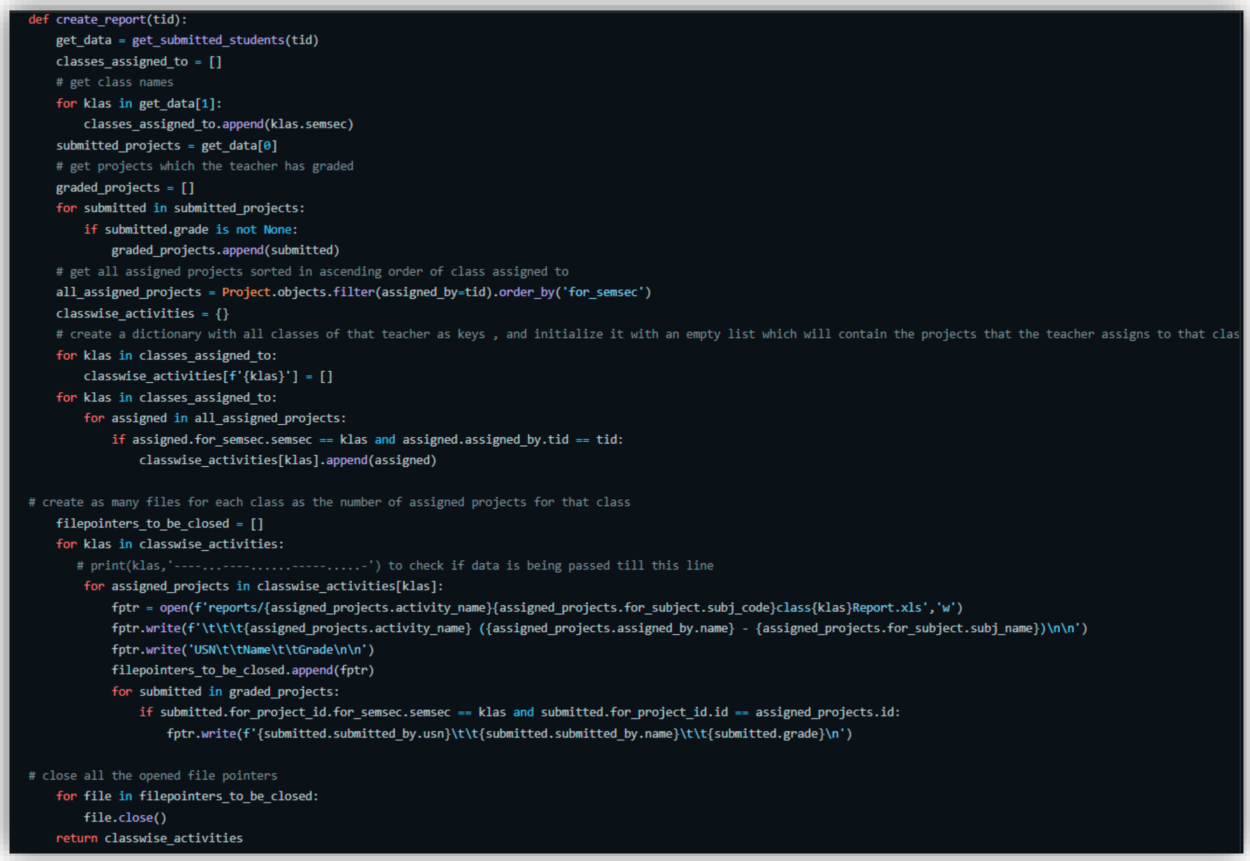
```python
def deleteProject(request,tid,pid):
    delete_obj = Project.objects.get(id=pid)
    activity_name = delete_obj.activity_name
    delete_obj.delete()
    assigned_projects = Project.objects.filter(assigned_by=tid)
    context = {
        'teacher_id' : tid,
        'assigned_projects':assigned_projects
    }
    messages.success(request,f'{activity_name} Deleted')
    return render(request, 'assigned.html', context)
```

Fig 3.5.5: Views.py – Delete Project (Teacher)

Fig 3.5.5 shows logic written to delete record of an assignment in database when teacher clicks on delete button.

3.5.6 Generating report for each activity section-wise

Spreadsheet files (Excel / .xls) are generated for each class for each assignment with Activity Details, Assigned Teacher Name , Student details such as USN, Name and the marks awarded. Logic is written in such a way that Files that are created are the same files that are updated if any data is updated in the site, hence avoiding the problem of creating numerous files for every request, in turn reducing server load.



```
def create_report(tid):
    get_data = get_submitted_students(tid)
    classes_assigned_to = []
    # get class names
    for klas in get_data[1]:
        classes_assigned_to.append(klas.semsec)
    submitted_projects = get_data[0]
    # get projects which the teacher has graded
    graded_projects = []
    for submitted in submitted_projects:
        if submitted.grade is not None:
            graded_projects.append(submitted)
    # get all assigned projects sorted in ascending order of class assigned to
    all_assigned_projects = Project.objects.filter(assigned_by=tid).order_by('for_semsec')
    classwise_activities = {}
    # create a dictionary with all classes of that teacher as keys , and initialize it with an empty list which will contain the projects that the teacher assigns to that clas
    for klas in classes_assigned_to:
        classwise_activities[f'{klas}'] = []
    for klas in classes_assigned_to:
        for assigned in all_assigned_projects:
            if assigned.for_semsec.semsec == klas and assigned.assigned_by.tid == tid:
                classwise_activities[klas].append(assigned)

# create as many files for each class as the number of assigned projects for that class
    filepointers_to_be_closed = []
    for klas in classwise_activities:
       # print(klas,'----...----......-----.....-') to check if data is being passed till this line
        for assigned_projects in classwise_activities[klas]:
            fptr = open(f'reports/{assigned_projects.activity_name}{assigned_projects.for_subject.subj_code}class{klas}Report.xls','w')
            fptr.write(f'\t\t\t{assigned_projects.activity_name} ({assigned_projects.assigned_by.name} - {assigned_projects.for_subject.subj_name})\n\n')
            fptr.write('USN\t\tName\t\tGrade\n\n')
            filepointers_to_be_closed.append(fptr)
            for submitted in graded_projects:
                if submitted.for_project_id.for_semsec.semsec == klas and submitted.for_project_id.id == assigned_projects.id:
                    fptr.write(f'{submitted.submitted_by.usn}\t\t{submitted.submitted_by.name}\t\t{submitted.grade}\n')

# close all the opened file pointers
    for file in filepointers_to_be_closed:
        file.close()
    return classwise_activities
```

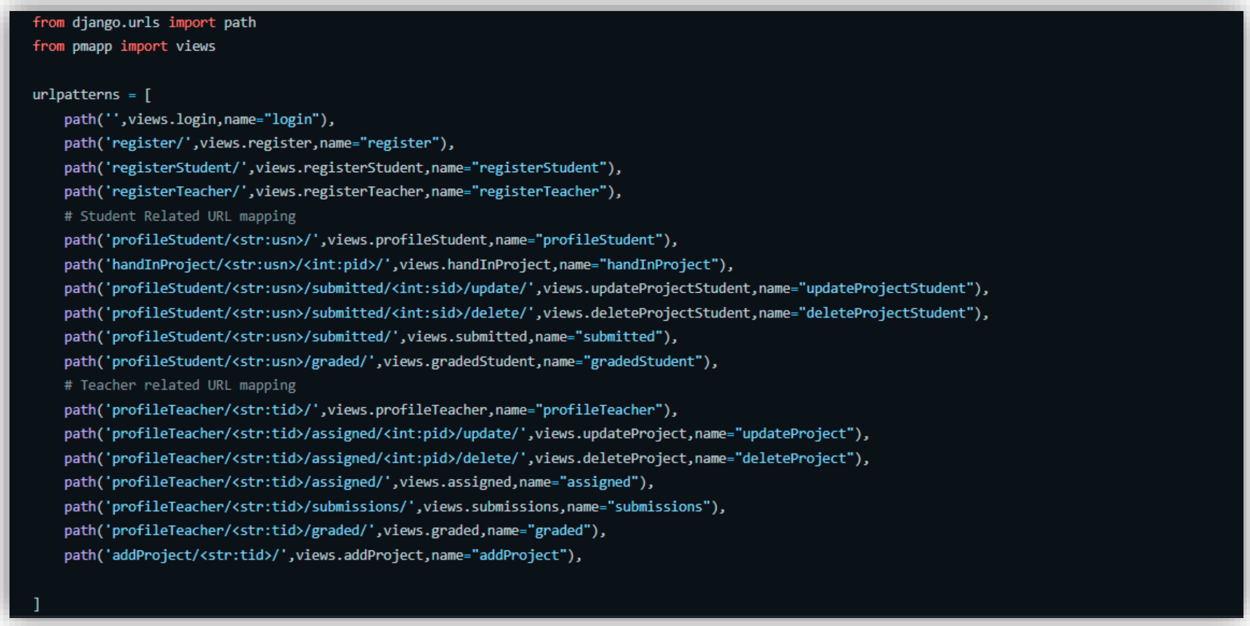
Fig 3.5.6: Methods.py – Generate report (Excel / .xls file)

Fig 3.5.6 shows logic written to generate report for each activity assigned to a class. The report will contain Student details such as USN, name and marks obtained.

These are a few function based views which implement the required logic for the application. The entire source code is available in our Github repository – *projectmanager* .
Source code link is provided at the end of the report.

3.6 URL Mappings

As this is a web application, we need URL mappings to handle all the different html templates rendering using regex and dynamic URL mappings and this functionality is implemented using the urls.py file.



```python
from django.urls import path
from pmapp import views

urlpatterns = [
    path('',views.login,name="login"),
    path('register/',views.register,name="register"),
    path('registerStudent/',views.registerStudent,name="registerStudent"),
    path('registerTeacher/',views.registerTeacher,name="registerTeacher"),
    # Student Related URL mapping
    path('profileStudent/<str:usn>/',views.profileStudent,name="profileStudent"),
    path('handInProject/<str:usn>/<int:pid>/',views.handInProject,name="handInProject"),
    path('profileStudent/<str:usn>/submitted/<int:sid>/update/',views.updateProjectStudent,name="updateProjectStudent"),
    path('profileStudent/<str:usn>/submitted/<int:sid>/delete/',views.deleteProjectStudent,name="deleteProjectStudent"),
    path('profileStudent/<str:usn>/submitted/',views.submitted,name="submitted"),
    path('profileStudent/<str:usn>/graded/',views.gradedStudent,name="gradedStudent"),
    # Teacher related URL mapping
    path('profileTeacher/<str:tid>/',views.profileTeacher,name="profileTeacher"),
    path('profileTeacher/<str:tid>/assigned/<int:pid>/update/',views.updateProject,name="updateProject"),
    path('profileTeacher/<str:tid>/assigned/<int:pid>/delete/',views.deleteProject,name="deleteProject"),
    path('profileTeacher/<str:tid>/assigned/',views.assigned,name="assigned"),
    path('profileTeacher/<str:tid>/submissions/',views.submissions,name="submissions"),
    path('profileTeacher/<str:tid>/graded/',views.graded,name="graded"),
    path('addProject/<str:tid>/',views.addProject,name="addProject"),

]
```

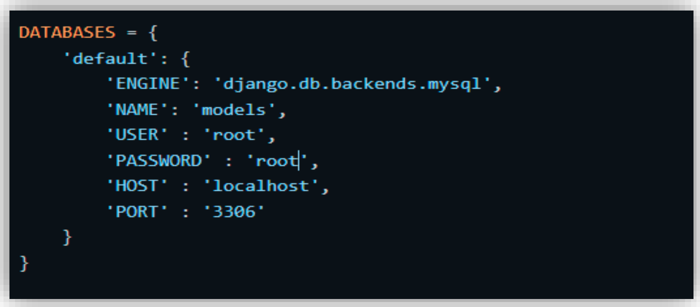
Fig 3.6: Urls.py file.

Fig 3.6 shows the urls.py file, this is where all the url mapping is done with the views.py file.

3.7 Database connection to Django

The database is connected in the settings.py file and the models that are created in the models.py file are migrated to the database by executing the following commands.

- `python manage.py makemigrations`
- `Python manage.py migrate`



```python
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'models',
        'USER' : 'root',
        'PASSWORD' : 'root',
        'HOST' : 'localhost',
        'PORT' : '3306'
    }
}
```

Fig 3.7: Django database connectivity

Fig 3.7 shows the settings needed to be included in settings.py file to connect to mysql database

Chapter 4

Results and Discussions

This project is a Django web app. It runs on Client side browser. Screenshots of most of the working are shown above. We have put a in few more screenshots here to show the working of our application.

Main Webapp Screen Shots

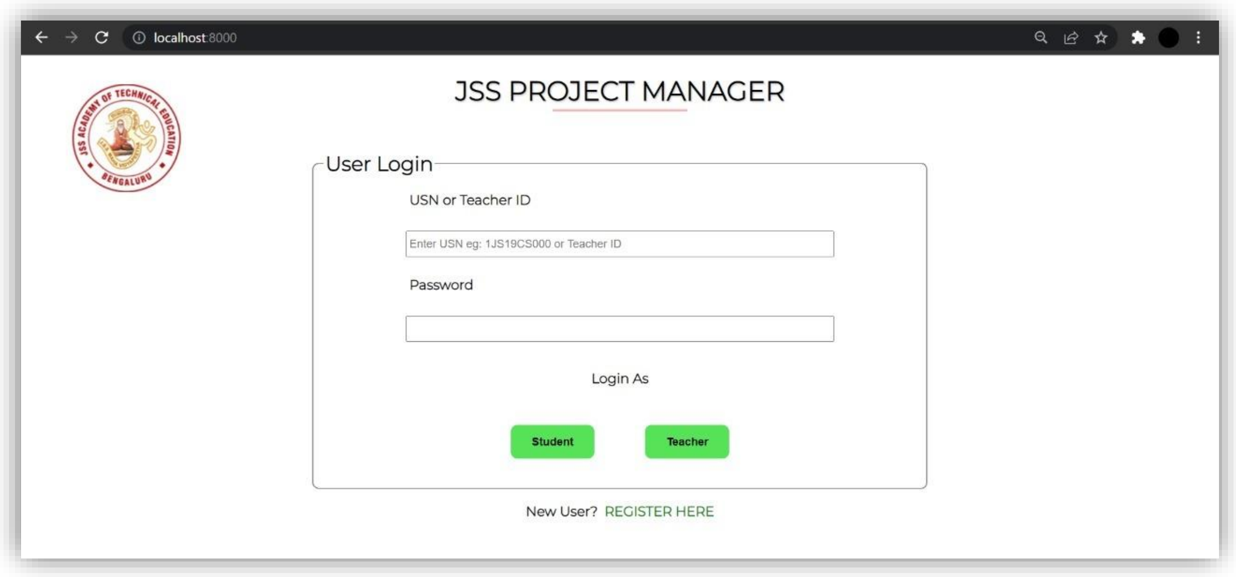
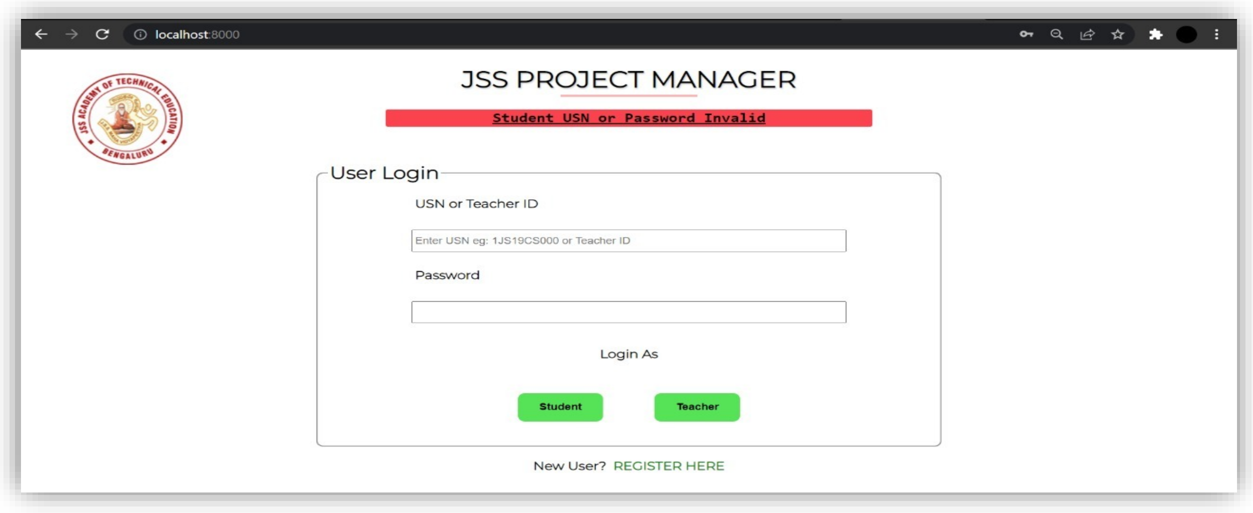


Fig 4.1 : Login Page

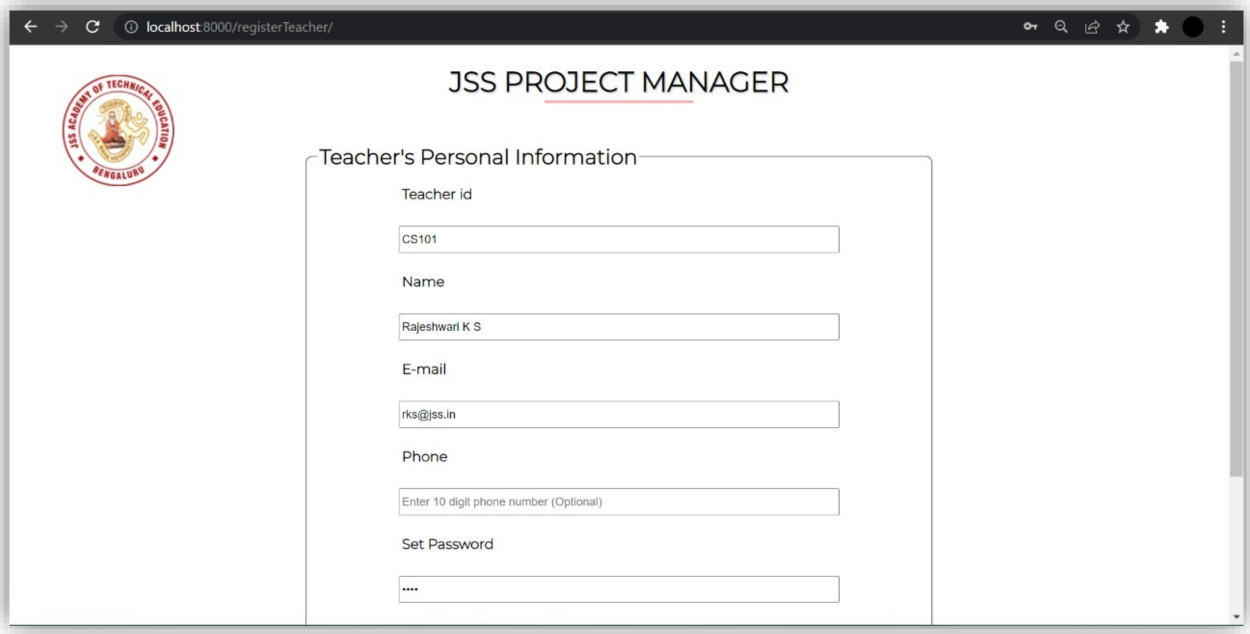
Fig 4.1 shows the screenshot of the login page of our application.



Student USN or Password Invalid

Fig 4.2 : Authenticate Login

Fig 4.2 screenshot of webapp where unauthorized user is denied access to the portal.


localhost:8000/registerTeacher/

JSS PROJECT MANAGER

Teacher's Personal Information

Teacher id: CS101

Name: Rajeshwari K S

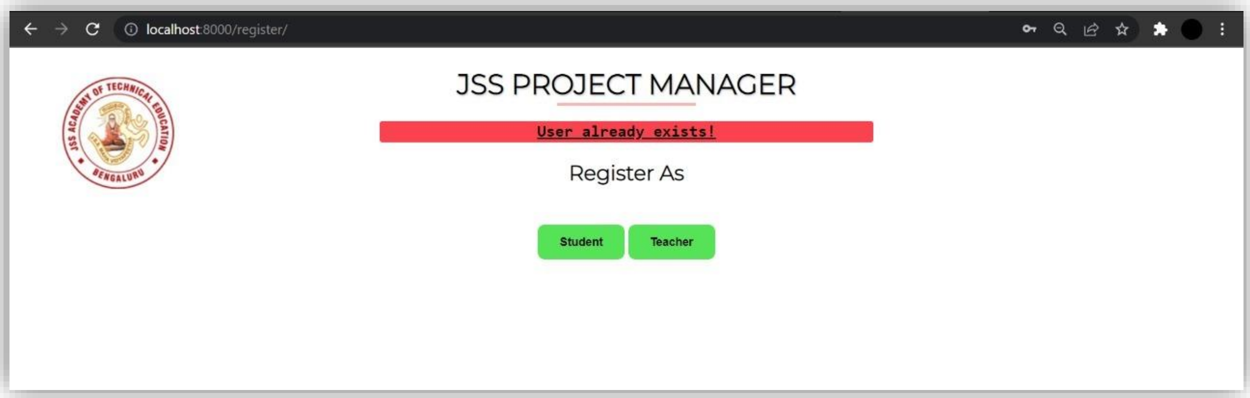
E-mail: rks@jss.in

Phone: Enter 10 digit phone number (Optional)

Set Password: ····

Fig 4.3 : User Registration

Fig 4.3 shows screenshot of webapp where user(teacher) can register by entering details.


localhost:8000/register/

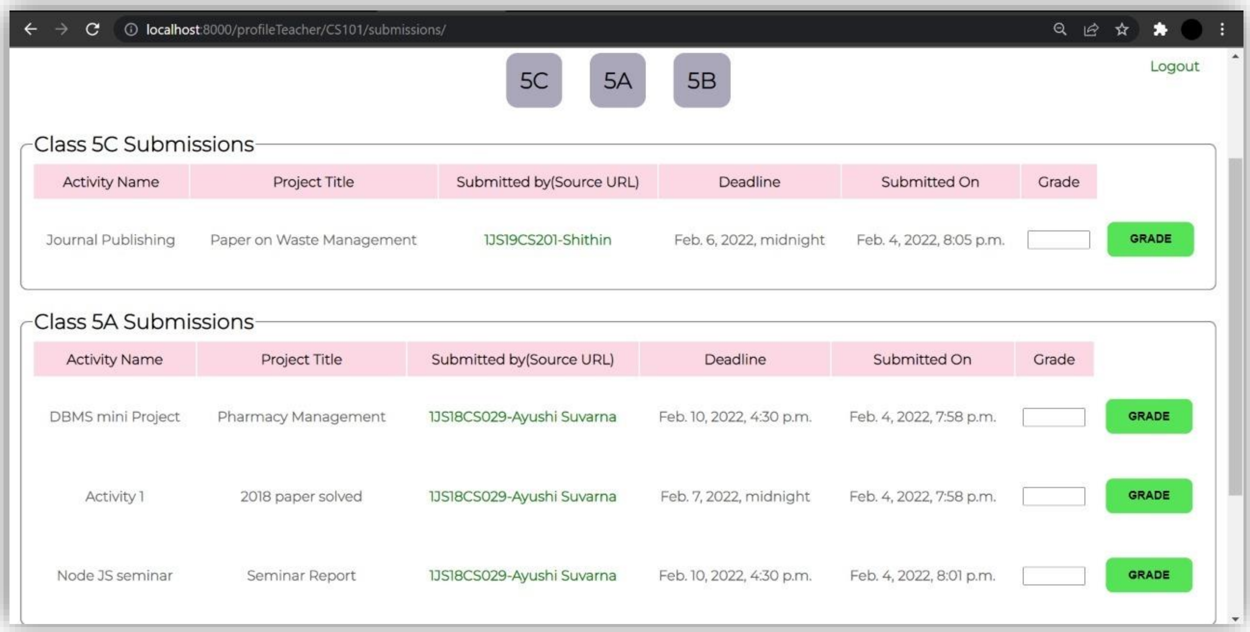
JSS PROJECT MANAGER

User already exists!

Register As

Fig 4.4 : Alert Message – User validation

Fig 4.4 shows screenshot of webapp where alert message is shown if user is registering with an already existing id(usn or tid). Hence duplication of data is avoided.


localhost:8000/profileTeacher/CS101/submissions/

Class 5C Submissions

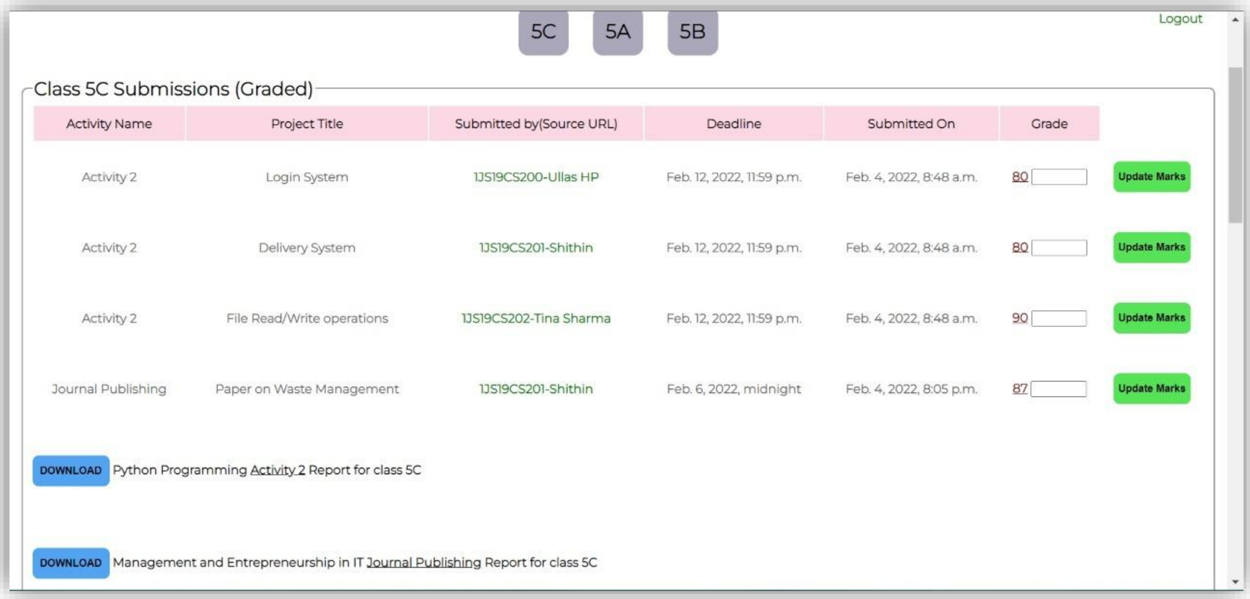
Activity Name	Project Title	Submitted by(Source URL)	Deadline	Submitted On	Grade	
Journal Publishing	Paper on Waste Management	1JS19CS201-Shithin	Feb. 6, 2022, midnight	Feb. 4, 2022, 8:05 p.m.		GRADE

Class 5A Submissions

Activity Name	Project Title	Submitted by(Source URL)	Deadline	Submitted On	Grade	
DBMS mini Project	Pharmacy Management	1JS18CS029-Ayushi Suvarna	Feb. 10, 2022, 4:30 p.m.	Feb. 4, 2022, 7:58 p.m.		GRADE
Activity 1	2018 paper solved	1JS18CS029-Ayushi Suvarna	Feb. 7, 2022, midnight	Feb. 4, 2022, 7:58 p.m.		GRADE
Node JS seminar	Seminar Report	1JS18CS029-Ayushi Suvarna	Feb. 10, 2022, 4:30 p.m.	Feb. 4, 2022, 8:01 p.m.		GRADE

Fig 4.5 : Submissions Page (Teacher View)

Fig 4.5 shows teacher view – submissions page. It Allows teachers to view submitted projects classwise, the Submitted By Column contain clickable elements which redirect to the students work. Teacher can grade students work here.



Class 5C Submissions (Graded)

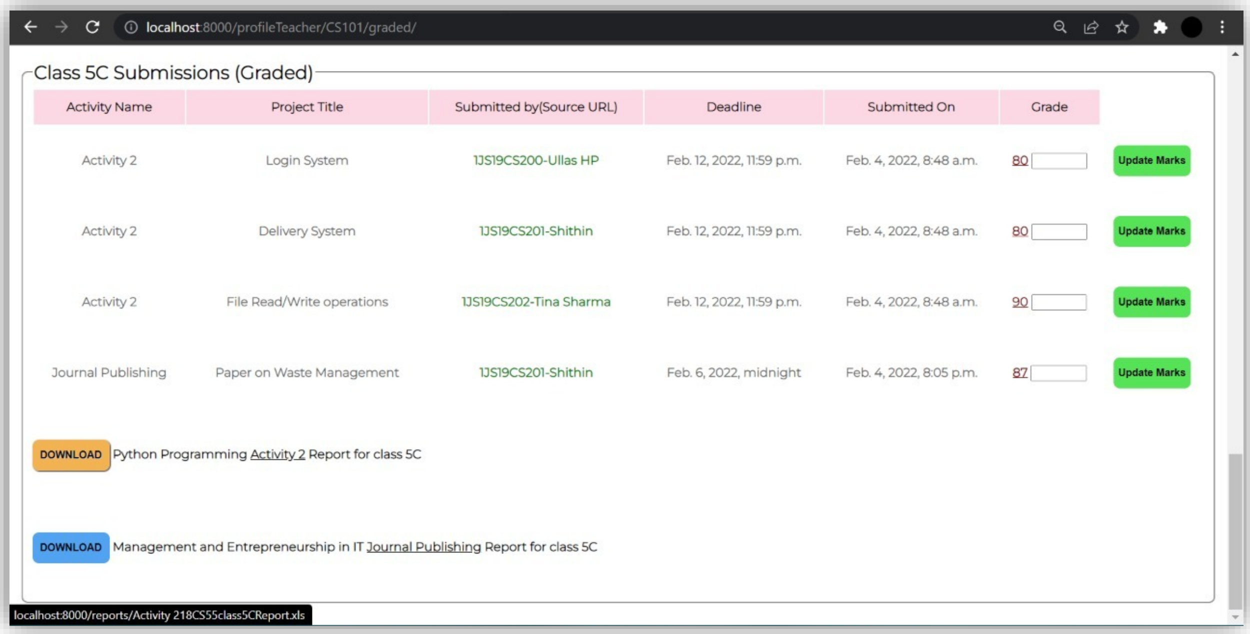
Activity Name	Project Title	Submitted by(Source URL)	Deadline	Submitted On	Grade	
Activity 2	Login System	1JS19CS200-Ullas HP	Feb. 12, 2022, 11:59 p.m.	Feb. 4, 2022, 8:48 a.m.	80	Update Marks
Activity 2	Delivery System	1JS19CS201-Shithin	Feb. 12, 2022, 11:59 p.m.	Feb. 4, 2022, 8:48 a.m.	80	Update Marks
Activity 2	File Read/Write operations	1JS19CS202-Tina Sharma	Feb. 12, 2022, 11:59 p.m.	Feb. 4, 2022, 8:48 a.m.	90	Update Marks
Journal Publishing	Paper on Waste Management	1JS19CS201-Shithin	Feb. 6, 2022, midnight	Feb. 4, 2022, 8:05 p.m.	87	Update Marks

DOWNLOAD Python Programming Activity 2 Report for class 5C

DOWNLOAD Management and Entrepreneurship in IT Journal Publishing Report for class 5C

Fig 4.6 : Graded page (Teachers View)

Fig 4.6 shows graded page it Teachers view. Here teachers are given the functionality to update the awarded marks as well and download reports of that particular class for that particular assignment



localhost:8000/profileTeacher/CS101/graded/

Class 5C Submissions (Graded)

Activity Name	Project Title	Submitted by(Source URL)	Deadline	Submitted On	Grade	
Activity 2	Login System	1JS19CS200-Ullas HP	Feb. 12, 2022, 11:59 p.m.	Feb. 4, 2022, 8:48 a.m.	80	Update Marks
Activity 2	Delivery System	1JS19CS201-Shithin	Feb. 12, 2022, 11:59 p.m.	Feb. 4, 2022, 8:48 a.m.	80	Update Marks
Activity 2	File Read/Write operations	1JS19CS202-Tina Sharma	Feb. 12, 2022, 11:59 p.m.	Feb. 4, 2022, 8:48 a.m.	90	Update Marks
Journal Publishing	Paper on Waste Management	1JS19CS201-Shithin	Feb. 6, 2022, midnight	Feb. 4, 2022, 8:05 p.m.	87	Update Marks

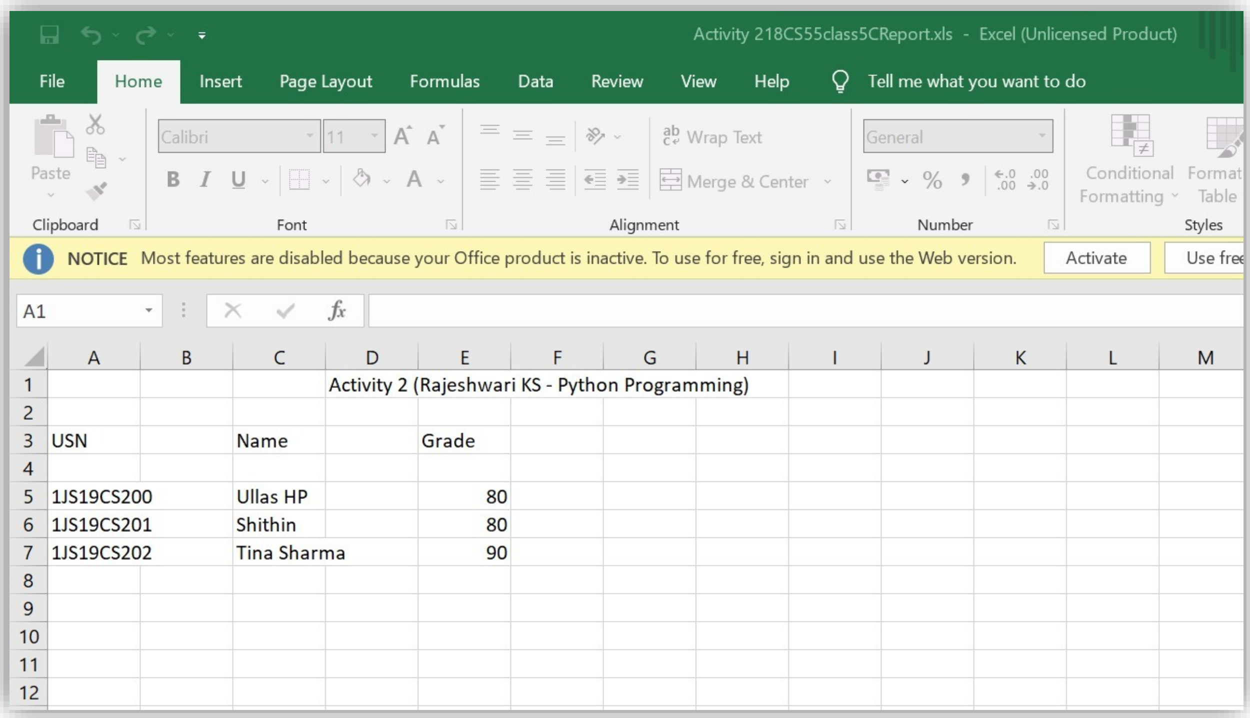
DOWNLOAD Python Programming Activity 2 Report for class 5C

DOWNLOAD Management and Entrepreneurship in IT Journal Publishing Report for class 5C

localhost:8000/reports/Activity 218CS55class5CReport.xls

Fig 4.7 : Download Report (Teacher View)

Fig 4.7 shows the screenshot of Teacher View where, on clicking the download button, the generated report is downloaded.



Activity 218CS55class5CReport.xls - Excel (Unlicensed Product)

NOTICE Most features are disabled because your Office product is inactive. To use for free, sign in and use the Web version.

	A	B	C	D	E
1				Activity 2 (Rajeshwari KS - Python Programming)	
2					
3	USN		Name		Grade
4					
5	1JS19CS200		Ullas HP		80
6	1JS19CS201		Shithin		80
7	1JS19CS202		Tina Sharma		90

Fig 4.8 : Excel file containing report of Student.

Fig 4.8 shows screenshot of Excel file which contains the details of the student

Conclusion and Future Enhancements

Conclusion

Student Project Management System has been a great learning experience as we have experimented and played around with various aspects of development such as front-end, back-end, database manipulation, server side processing and rendering. This is an experiment that we have conducted on ourselves to learn a new technology(Django) and implement it on our problem statement. SPMS provides a range of facilities to users and hence partly solves the purpose it is built for by providing

- Ease of use
- User friendly GUI (graphical user interface)
- Constraints and Implied logic are performed behind the screen rapidly to give user the best experience
- User security and data integrity
- Goes a step further by generating reports dynamically hence helping teachers

Future Enhancements

The future scope of Student Project Management System is quite diverse and scalable.

The application currently focuses on educational institutes like colleges, schools and universities which is in itself a vast acquirable market for this domain of the project.

However, we can always take this into other organizations too, because every organization follow a chain of order or hierarchy and where there is this hierarchal model, it is essential to implement monitoring and tracking of work and progress. This project can be diversified in this aspect as well hence improving the scope of the project.

At its current state, that is now, it can be scaled up to provide more security features and recovery methods and it would be ready to be deployed on a campus.

References

[1]. Fundamentals of Database Systems, Ramez Elmasri and Shamkant B. Navathe, 7th Edition, 2017, Pearson.

[2]. Database management systems , Ramkrishnan and Gehrke , 3rd Edition ,2014 , McGraw Hill

Website Reference Links

Django Official Documentation - https://docs.djangoproject.com/en/4.0/

Codemy Youtube Playlist - https://www.youtube.com/watch?v=HHx3tTQWUx0&list=PLCC34OHNcOtqW9BJmgQPPzUpJ8hl49AGy

Corey Schafer Youtube Playlist – https://www.youtube.com/watch?v=UmljXZIypDc&list=PL-osiE80TeTtoQCKZ03TU5fNfx2UY6U4p

Numerous posts from – https://stackoverflow.com/

Source Code (GitHub repository link) :

https://github.com/GirishKumarDV/projectmanager