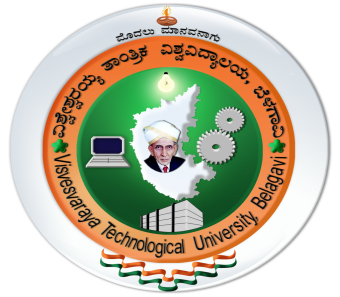
**VISVESVARYA TECHNOLOGICAL UNIVERSITY**

**BELAGAVI 590-018**

****

**A MINI PROJECT REPORT ON**

**“PROJECT MANAGEMENT SYSTEM”**

**Submitted in partial fulfillment of requirement for the fifth Semester**

**DATABASE MANAGEMENT SYSTEM LABORATORY**

**For the Course of**

**BACHELOR OF ENGINEERING**

**In**

**COMPUTER SCIENCE**

**Submitted by**

**Bikash Thapa 1ST16CS710**

**Annappa Kurdekar 1ST16CS706**

**Under the guidance of**

**Mrs. Bilwashree H. Asst. Professor**

**Mr. Raghava M. S. Asst. Professor**

****

**Department of Computer Science & Engineering**

**SAMBHRAM INSTITUTE OF TECHNOLOGY**

**BENGALURU 560097**

SAMBHRAM INSTITUTE OF TECHNOLOGY

M.S. PALYA, JALAHALLI EAST, BENGALURU - 560 097

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



**CERTIFICATE**

This is to certify that the mini-project entitled **“Project Management System”** has been carried out by **ANNAPPA KURDEKAR (1ST16CS706)** and **BIKASH THAPA (1ST16CS710),** bonafide students of **Sambhram Institute of Technology** in partial fulfillment of requirements for the fifth semester **Database Management System Laboratory,** prescribed by the **Visvesvaraya Technological University (VTU)**, Belagavi during the academic year **2018-2019**. It is also hereby, ensured that all corrections/suggestions indicated for internal assessment have been incorporated while submitting this report. This report has been approved as it satisfies the academic requirements with respect to the project work prescribed for the said course.

**Mrs. Bilwashree H. Dr. T JOHN PETER**

Assistant Professor Head of Department

Department Of CSE Department Of CSE

**External Viva**

Name Of Examiners: ***Signatures With Date***

**ACKNOWLEDGEMENT**

While presenting this mini-project entitled “Project Management System” entrusted to us, we feel that it is our duty to acknowledge the guidance and help provided for its success.

We take this occasion to thank God, almighty for blessing us with his grace and taking our Endeavour to a successful culmination.

We are grateful to our institution, Sambhram Institute of Technology for providing us, a congenial atmosphere to carry out this project with all possible amount of ease.

We thank **Dr. H.G Chandrakanth**, Principal, *Sambhram Institute of Technology*, and **Dr. T John Peter**, HOD, *Department of Computer Science & Engineering*, for their constant support.

We extend our thanks to our esteemed guides, **Mrs. Soumya B**, **Mrs. Bilwashree** and **Mr. Raghava M. S**. *Assistant Professors, Department of Computer Science & Engineering*, for providing us with the right guidance and advice at the crucial junctures and for showing us the right way.

We are also indebted to our parents and friends for their continued moral and material support throughout the course of this project and helping us in showcasing the project’s presentation.

Our thanks to all teaching, non-teaching staffs and others who have contributed bits, bytes and words to accomplish this project.

**ANNAPA K.**

**BIKASH THAPA**

**ABSTRACT**

Project Management System is a Project which aims in developing a Digital System to handle all the hassles of managing a project in college where multiple Teacher and Student are involved.

The Modern Project Management System has many features that make this project unique and can be applied in real life right away. It has a smooth authentication system that makes sure no unauthorized person in accessing projects data.

The user authentication window enables the administrators to enter into the application in a secured manner and monitor the whole system. It has a facility wherein the administrators, after logging into their accounts can view and process, the list of students, faculty (attendance status, faculty details) - friendly Menu Driven Interface (MDI).

The report generated, thus maintains a log or a track record of the records of the student and faculty in a simple and an effective format. There by, the options of entering the data also stand enabled.

Overall this project of ours is being developed to help the students as well as the staffs of institutional to maintain projects in an optimal way possible and also to minimize human efforts.

**TABLE OF CONTENTS**

**Chapter Subjects Page**

**1 Introduction to Database 6-12**

1.1 Database System Environment

1.2 Advantages of DBMS

1.3 Architecture of Database

**2 User requirements Definition 12-18**

2.1 Overview

2.2.1 Objects

2.2 Functionalities of Application

2.2.1 User Classes & Characteristics

2.3 Description of the relations

**3 SYSTEM REQUIREMENTS SPECIFICATION 18-25**

*3.1 Functional Requirements*

*3.1.1 Valid assumptions and dependencies*

*3.1.2 Data Requirements*

*3.1.3 External Interface Requirements*

*3.1.4 Operational Requirements*

*3.2 Non-Functional Requirements*

*3.2.1 Performance Requirements*

*3.2.2 Product & Usability Requirements*

*3.2.3 Requirements Attributes*

*3.3 Software Requirement Specification*

*3.3.1 Specifications of the software used*

*3.4 Hardware requirements Specifications*

**4 Design and Implementation 26-30**

4.1 E-R Model/ Schema Diagram

4.2 Homepage HTML/CSS and Django

4.3 Connecting to Database using Django

**5 Results / outputs 31-35**

5.1 Project management system homepage

5.2 Login and register page

5.3 User profile with edit profile option

5.4 Project page and synopsis

5.5 All project under teacher

**CHAPTER 1**

**INTRODUCTION TO DATABASE**

A **database** is a collection of related data. By data, we mean known facts that can be recorded and that have implicit meaning. For example, consider the names, telephone numbers, and addresses of the people you know. You may have recorded this data in an indexed address book or you may have stored it on a hard drive, using a personal computer and software such as Microsoft Access or Excel. This collection of related data with an implicit meaning is a database.

The preceding definition of database is quite general; for example, we may consider the collection of words that make up this page of text to be related data and hence to constitute a database. However, the common use of the term database is usually more restricted. A database has the following implicit properties:

* A database represents some aspect of the real world, sometimes called the **miniworld** or the universe of discourse (**UOD**). Changes to the miniworld arereflected in the database.
* A database is a **logically coherent collection of data** with some inherent meaning. A random assortment of data cannot correctly be referred to as a database.
* A database is designed, built, and populated with data for a specific purpose. It has an intended group of users and some preconceived applications in

which these users are interested.

A database may be generated and maintained manually or it may be computerized. For example, a library card catalog is a database that may be created and maintained manually

A database management system (DBMS) is a collection of programs that enables users to create and maintain a database. The DBMS is a general-purpose software system that facilitates the processes of defining, constructing, manipulating, and sharing databases among various users and applications.

* **Defining** a database involves specifying the data types, structures, andconstraints of the data to be stored in the database. The database definition or descriptive information is also stored by the DBMS in the form of a database catalog or dictionary; it is called meta-data.
* **Constructing** the database is the process of storing the data on some storagemedium that is controlled by the DBMS.
* **Manipulating** a database includes functions such as querying the database toretrieve specific data, updating the database to reflect changes in the miniworld, and generating reports from the data.
* **Sharing** a database allows multiple users and programs to access the

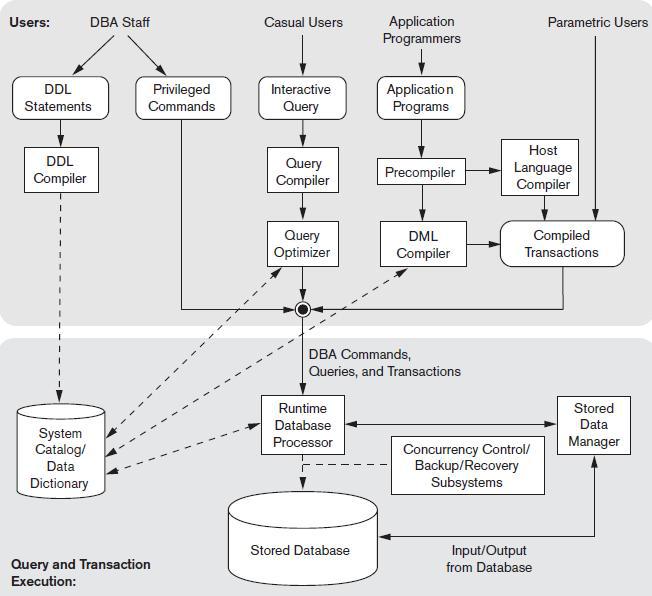
Database simultaneously.

An application program accesses the database by sending queries or requests for data to the DBMS. A query typically causes some data to be retrieved; a transaction may cause some data to be read and some data to be written into the database.

Other important functions provided by the DBMS include protecting the database and maintaining it over a long period of time. Protection includes system protection against hardware or software malfunction (or crashes) and security protection against unauthorized or malicious access. A typical large database may have a life cycle of many years, so the DBMS must be able to maintain the database system by allowing the system to evolve as requirements change over time.

To complete our initial definitions, we will call the database and DBMS software together a database system.

**1.1 DATABASE SYSTEM ENVIRONMENT**



**Fig 1 : The database system environment**

The database and the DBMS catalog are usually stored on disk. Access to the disk is controlled primarily by the operating system (OS), which schedules disk read/write.

Many DBMSs have their own buffer management module to schedule disk read/write, because this has a considerable effect on performance. Reducing disk read/write improves performance considerably. A higher-level stored data manager module of the DBMS controls access to DBMS information that is stored on disk, whether it is part of the database or the catalog.

* The **DBA staff**, casual users work with interactive interfaces to formulate queries.
* **Application programmers** create programs using some host programminglanguages, and parametric users do data entry work by supplying parameters to predefined transactions. The DBA staff works on defining the database and tuning it by making changes to its definition using the DDL and other privileged commands.
* The **DDL compiler** processes schema definitions, specified in the DDL, and stores descriptions of the schemas (meta-data) in the DBMS catalog. The catalog includes information such as the names and sizes of files, names and data types of data items, storage details of each file, mapping information among schemas, and constraints. In addition, the catalog stores many other types of information that are needed by the DBMS modules, which can then look up the catalog information as needed.
* **Casual users** and persons with occasional need for information from thedatabase interact using some form of interface, which we call the interactive query interface. These queries are parsed and validated for correctness of the query syntax, the names of files and data elements, and so on by a query compiler that compiles them into an internal form. This internal query is subjected to query optimization.
* The **query optimizer** is concerned with the rearrangement and possible reordering of operations, elimination of redundancies, and use of correct algorithms and indexes during execution. It consults the **system catalog** for statistical and other physical information about the stored data and generates executable code that performs the necessary operations for the query and makes calls on the runtime processor.
* The **runtime database processor** executes (1) the privileged commands, (2) the executable query plans, and (3) the canned transactions with runtime parameters. It works with the system catalog and may update it with statistics. It also works with the stored data manager, which in turn uses basic

operating system services for carrying out low-level input/output (read/write) operations between the disk and main memory. It is now common to have the client program that accesses the DBMS running on a separate computer from the computer on which the database resides.

* The former is called the client computer running a DBMS client software and the latter is called the database server. In some cases, the client accesses a middle computer, called the application server, which in turn accesses the database server.

**1.2 ADVANTAGES OF USING THE DBMS APPROACH**

Due to its centralized nature, the database system can overcome the disadvantages of the file system-based system.

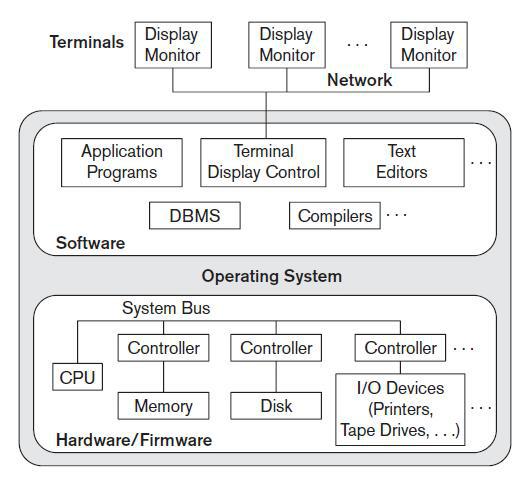
* **Data independency**: Application program should not be exposed to detailsof data representation and storage DBMS provides the abstract view that hides these details.
* **Efficient data access**: DBMS utilizes a variety of sophisticated techniquesto store and retrieve data efficiently.
* **Data integrity and security** : Data is accessed through DBMS, it can

Enforce integrity constraints. E.g.: Inserting salary information for an employee.

* **Data Administration**: When users share data, centralizing the data is animportant task, Experience professionals can minimize data redundancy and perform fine tuning which reduces retrieval time.
* **Concurrent access and Crash recovery**: DBMS schedules concurrentaccess to the data. DBMS protects user from the effects of system failure.

**Reduced application development time**: DBMS supports importantfunctions that are common to many applications.

**1.3 ARCHITECTURE OF DATABASE**



**Fig 2: A physical, centralized and a basic Client - Server architecture**

The idea is to define **specialized servers** with specific functionalities. For example, it is possible to connect a number of PCs or small workstations as clients to a **file server** that maintains the files of the client machines. Another machine can be designated as a **printer server** by being connected to various printers; all print requests by the clients are forwarded to this machine.

Thus, the resources provided by specialized servers can be accessed by many client machines. The **client machines** provide the user with the appropriate interfaces to utilize these servers, as well as with local processing power to run local applications. This concept can be carried over to other software packages, with specialized programs such as a CAD (computer-aided design) package being stored on specific server machines and being made accessible to multiple clients.

A **client** in this framework is typically a user machine that provides user interface capabilities and local processing. When a client requires access to additional functionality such as database access that does not exist at that machine, it connects to a server that provides the needed functionality.

A **server** is a system containing both hardware and software that can provide services to the client machines, such as file access, printing, archiving, or database access.

**CHAPTER 2**

**USER REQUIREMENTS DEFINITION**

**2.1 OVERVIEW**

* Student attendance management system is a software application which automatically performs the housekeeping functions of a library such as maintaining the records related to students, faculty, attendance status, eligibilty, no of classes taken, no of class attendended and all necessary requirements for it to manage day to day transactions as described in the abstract section earlier.
* The database follows a typical event flow seen in such a system. The database mainly is from an administrator’s perspective.
* Since a attendance is the center of all the students, all the records in the database resolve around transaction-oriented activities.

Some of the other independent categories may include student details, faculty details, attendance status…etc.

**2.1.1 OBJECTIVES**

* To automate the existing system by relating all of the records of the above mentioned operations in a logical manner such that the recorded transactions can be replaced and accepted without major changes and problems.
* To eliminate paper work by using a computerized system.
* To save time and money.
* To trap most of the data entry errors.

**2.2 FUNCTIONALITIES OF THE APPLICATION**

* Provision of quick access to the records maintained.
* Showing important details so that important decisions could be made easily.
* Generation of customized reports based on particular criteria. (For example: Retrieval may take place based on the name of the author/publisher, number of copies lent, books available in each branch… etc.. )
* Provision of easy - to - use or a user friendly interface, with menus and forms for clear navigability.
* Data validation: minimize data redundancy. (Unnecessary duplicate data are prevented from creeping into the database.)
* Provision of separate access to individual book and administrator such that only the administrator gets the rights to add, delete and modify the various services offered by the system.
* Provide the administrator with options to add as many as books, departments , corresponding to the subject or the content required.
* Provision of privileges for the administrator to alter the computerized library database.
* Provide effective and efficient querying techniques.

**2.2.1 USER CLASSES AND CHARACTERISTICS**

Any user with basic computer skills can make use of the application. The user should have only been briefed about the functionality of the system before he/she can start using the system.

**2.3 DESCRIPTION OF THE RELATIONS (TABLES)**

1. **User**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sno.** | **Name** | **Type** | **Description** |
| **1** | **Id** | **Int** | **Id** |
| **2** | **username** | **varchar(255)** | **User’s username** |
| **3** | **password** | **varchar(60)** | **User’s Password** |
| **4** | **is\_superuser** | **Boolean** | **Check if user is admin** |
| **5** | **is\_mentor** | **Boolean** | **Check if User is teacher** |
| **6** | **is\_student** | **Boolean** | **Check if User is Student** |
| **7** | **is\_completed** | **Boolean** | **Check if user profile is complete or Not** |

**Table 1: User table That Store Login Information of all Users**

**( ii ) Teacher**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sno.** | **Name** | **Type** | **Description** |
| **1** | **Id** | **Int** | **Id** |
| **2** | **fullName** | **varchar(255)** | **Full Name** |
| **3** | **department** | **varchar(255)** | **Department Name** |
| **4** | **college** | **Varchar(255)** | **College Name** |
| **5** | **gender** | **Varchar(10)** | **Teacher gender** |
| **6** | **designation** | **Varchar(255)** | **Teacher Position in College** |
| **7** | **experiences** | **Varchar(255)** | **Experiences of Teacher** |
| **8** | **bio** | **Varchar(500)** | **About Teacher** |
| **9** | **educations** | **Varchar(500)** | **Education** |
| **10** | **image\_url** | **Varchar(100)** | **Profile Image Url** |
| **11** | **User\_id** | **Int** | **Foreign Key to link user profile** |

**Table 2: Teacher Records**

**( iii ) Student**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sno.** | **Name** | **Type** | **Description** |
| **1** | **Id** | **Int** | **Id** |
| **2** | **fname** | **varchar(255)** | **First name** |
| **3** | **lname** | **varchar(255)** | **Last Name** |
| **4** | **Usn** | **Varchar(255)** | **Student USN** |
| **5** | **Section** | **Varchar(10)** | **Section of Student** |
| **6** | **department** | **Varchar(255)** | **Department of Student** |
| **7** | **College** | **Varchar(255)** | **College Name** |
| **8** | **semester** | **Varchar(500)** | **Semester** |
| **9** | **gender** | **Varchar(500)** | **Gender of Student** |
| **10** | **image\_url** | **Varchar(100)** | **Profile Image Url** |
| **11** | **bio** | **Varchar(500)** | **About Student** |
| **12** | **user\_id** | **Int** | **Foreign Key to link user profile** |

**Table 3 : Student Records**

**(iv) Project**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sno.** | **Name** | **Type** | **Description** |
| **1** | **Id** | **Int** | **Id** |
| **2** | **User\_id** | **Int** | **Foreign Key** |
| **3** | **Teacher\_id** | **Int** | **Foreign Key** |
| **4** | **Name** | **Varchar(255)** | **Project Name** |
| **5** | **Introduction** | **Varchar(500)** | **Project Introduction** |
| **6** | **Objectives** | **Varchar(500)** | **Project Objectives** |
| **7** | **Db\_schema** | **Varchar(500)** | **Project Database Schema** |
| **8** | **frontend** | **Varchar(500)** | **Front End programming Language** |
| **9** | **backend** | **Varchar(500)** | **Backend Programming Language** |
| **10** | **requirements** | **Varchar(500)** | **Project basic Requirements** |
| **11** | **image\_url** | **Varchar(1000)** | **Project Image Url** |

**Table 4: Project Details**

**(v) Synopsis**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sno.** | **Name** | **Type** | **Description** |
| **1** | **Id** | **Int** | **Id** |
| **2** | **tName** | **Varchar(255)** | **Teacher Name** |
| **3** | **sName** | **Varchar(255)** | **Student Name** |
| **4** | **Pname** | **Varchar(255)** | **Project Name** |
| **5** | **Introduction** | **Varchar(500)** | **Project Introduction** |
| **6** | **Objectives** | **Varchar(500)** | **Project Objectives** |
| **7** | **Db\_schema** | **Varchar(500)** | **Project Database Schema** |
| **8** | **frontend** | **Varchar(500)** | **Front End programming Language** |
| **9** | **backend** | **Varchar(500)** | **Backend Programming Language** |
| **10** | **requirements** | **Varchar(500)** | **Project basic Requirements** |

**Table 5: Synopsis Details**

**CHAPTER 3**

**SYSTEM REQUIREMENTS SPECIFICATION**

**3.1 FUNCTIONAL REQUIREMENTS**

Functional requirements are statements of services the system should provide, how the system should react to particular inputs, and how the system should behave in particular situations. In some cases, the functional requirements may also explicitly state what the system should not do.

**3.1.1 Valid assumptions and dependencies**

* The **assumptions** are:
  + The source code pertaining to each module developed should be error free.
  + The system should be user friendly.
  + The information, modifications and updates to the database should be available to all the administrators of the library.
  + The system should have feasible storage capacity and should provide faster access to the data.
  + The system should provide retrieval facility and also facilities for quick and understandable transactions.
  + Users should register themselves in order to procure administrative membership.
  + Valid credentials shall only be keyed-in without which, authentication becomes unsuccessful.
* The **dependencies** are:
  + The specific hardware and software through which the product operates.
  + Thus on the basis of the list of the requirements specification, the application would be developed to a fully functional one.
  + The end-users have to possess proper knowledge and understanding in order to work with it.
  + The system should have a general report generator.

All updates pertaining to all modules are to be recorded and changes should happen in the referenced modules concurrently.

**3.1.2 Data Requirement**

The input consists of query to the database and the output consists of solutions for the query. The output also includes the user receiving the details of their accounts.

In this project, the inputs will be the queries as fired by the users like creating an account, adding information on the basis of various criteria, updating the information pertaining to individual records, deleting individual records, searching and retrieving records..etc,.

Now, the solutions for the queries as the respective outputs will be visible when the users request the server through the GUI.

**3.1.3 External Interface Requirement**

**GUI**

The application that is developed provides a good graphical user interface for the user or the administrator who operates the system, performs the required tasks such as insert, update, delete and search on the basis of various criteria.

**General features of the GUI**

* It allows the user to generate quick reports and export data to different formats.
* Provides stock verification and search facility based on different criteria.
* All the modules designed are collectively integrated to form the Menu Driven Interface (MDI).
* The design of the interface is simple and all the modules within it, follow a standard template.
* The user interface must be able to interact with the user management module and a part of the interface must be dedicated to the login/logout module.
* **Login interface or the User Authentication Window**

In case the user is not yet registered, registration can be done through this window. Once the registration is successful, authenticity can be established through the user authentication window which on successful authentication redirects the user to the menu driven interface.

* **Search**

The user of the librarian can enter the name of the resources he/she is looking for based on the criteria mentioned in the module.

* **Menu - oriented views**

The menu may further show the categories based on which the whole system is supposed to be administered.

* **Control Panel**

This panel will allow the user to update/delete and refresh the contents or records based on a particular criterion.

**3.1.4 Operational Requirement**

The product will operate in Windows environment. The only requirement to use this product would be the installation of supporting software packages like MYSQL Server, APACHE server a greater version, XAMPP CONTROL PANNEL VERSION 3.2.2, for viewing the pdf using Microsoft word.

**3.2 NON - FUNCTIONAL REQUIREMENTS**

These are constraints on the services or functions offered by the system. They include timing constraints, constraints on the development process, and constraints imposed by standards. Non-functional requirements often apply to the system as a whole, rather than individual system features or services.

**3.2.1 Performance requirement**

The proposed system that we are going to develop may be used as a chief performance system within the different campuses of the university, through secured interaction with the university staffs.

Therefore, it is expected that the database would perform functionally, all the requirements that are specified by the university.

* The performance of the system should be fast and accurate.
* The Student Attendance Management System shall handle expected and non-expected errors in ways that prevent loss in information and long downtime period. Thus it should have inbuilt error testing to identify and handle exceptions.
* The system should be able to handle large amount of data.

**3.2.2 Product and usability requirements**

* **Availability**: The system is available 100% for the user and can be used on a24 x 7 basis.
* **Accuracy**: The system shall accurately provide real time information takinginto consideration various concurrency issues.
* **Reliability**: The system has to be 100% reliable due to the importance ofdata and the damages that can be caused by incorrect or incomplete data.
* **Maintainability**: Changes (addition of new members, database changes)must be verified once per day at least.
* **Portability:** The system should also be portable.

**3.2.3 Requirements attribute**

* There may be multiple administrators creating the project, so, all of them would have rights to create changes to the system.
* The project should be open source.
* The quality of the database is maintained in such a way so that it can be very friendly to all the users of the database.
* The users should be able to easily install the software on the system.

**3.3 SOFTWARE REQUIREMENTS SPECIFICATION**

The application is developed using HTML/CSS as the front end which is supported by Python / DJANGO Framework. MySQL Server as the back end for accessing and connecting the front end to the database.

Python is a powerful but simple language and Django Framework is aimed primarily at developers creating applications by using the Apache server, inherits many of the best features of Web Development Requirements, but few of the inconsistencies and anachronisms, resulting in a cleaner and more logical language.

SQL (Structured Query Language) is used for defining, manipulating, controlling, storing and viewing the information present in a database.

**3.3.1 Specifications of the software used for application development**

The following software is required to develop the application:

* **Front end: HTML/CSS**
* **Backend Language: Python With Django Framework**
* **Database Server: MYSQL**
* XAMPP CONTROL PANNEL V3.2.2
* **Operating system** : MAC/LINUX/Windows 10 (x64),

**3.4 HARDWARE REQUIREMENTS SPECIFICATION**

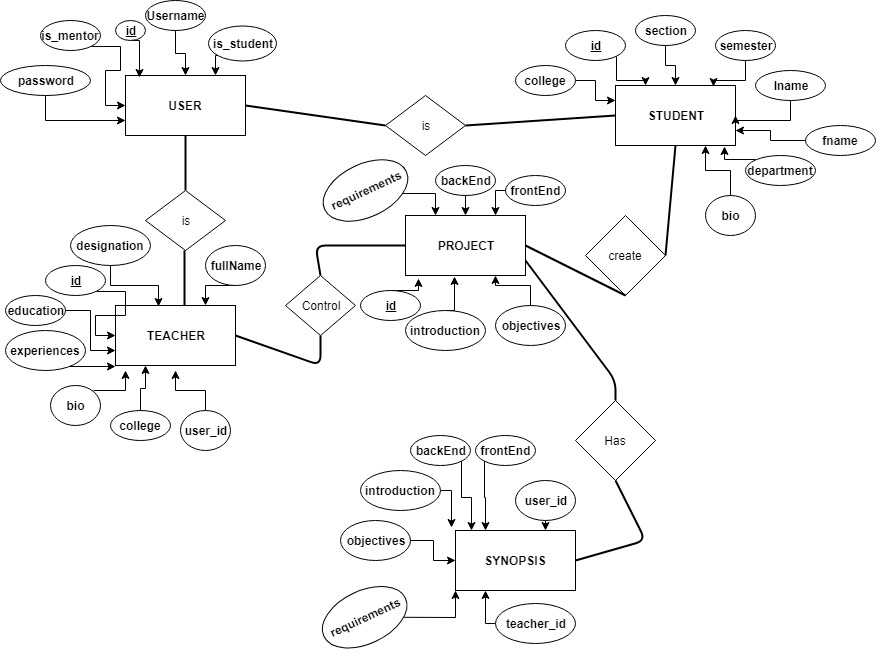
The following hardware requirements are needed to develop the application:

* Computer that has a 1.6 GHz or faster processor (2 GHz recommended).
* 1 GB (32-bit) or 2 GB (64-bit) RAM (add 512 MB if running in a virtual machine).
* 10 GB of available hard disk space.
* 5400 RPM hard disk drive.

**CHAPTER 4**

**DESIGN AND IMPLEMENTATION**

**4.1 ER MODEL & SCHEMA**

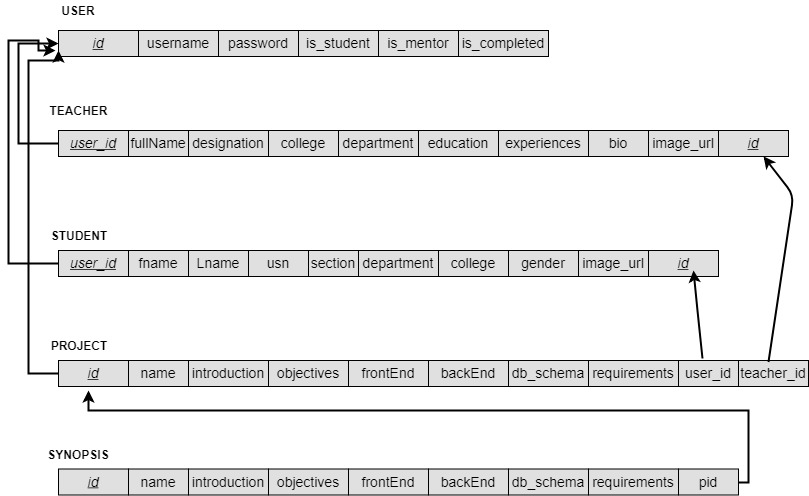


**Fig 3: ER schema diagram for a simple student attendance database**

The ER model describes data as *entities*, *relationships*, and *attributes*. The basic object that the ER model represents is an **entity**, which is a *thing* in the real world with an independent existence.

An entity may be an object with a physical existence or it may be an object with a conceptual existence.

Here each entity has one or more attributes. Fig 3 shows the entities with their respective attributes, relationships, with cardinality ratios which further mean something about the quality of participation that may be offered by the entities for their relationships between

**SCHEMA DIAGRAM**

**Fig. 4 Schema Diagram**

The Schema of the project is given above. We have five tables in our database. The Teacher and Student Tables are created by using the references of User Table.

The User table alone is created to hold the username, password. The user table contain 3 extra attributes is\_mentor , is\_teacher and is Completed. These attributes are used in application to identify the users as well as assign the privileges.

Project tables contains all the information as well as **user\_id** and **teacher\_id** , these attributes are used to identify the owner and mentor of the project.

**4.2 Homepage HTML/CSS and Django : Source Code**



**Fig 5**. **Snippets of Source code for Homepage**

**4.3 Connecting to Database Using Django:**



**Fig 6 : Code Snippets to connect django app to Database**

**NAME** -> Database Name

**USE**R -> User name used to connect database

**PASSWORD** -> Password to connect database

**HOST** -> Server name

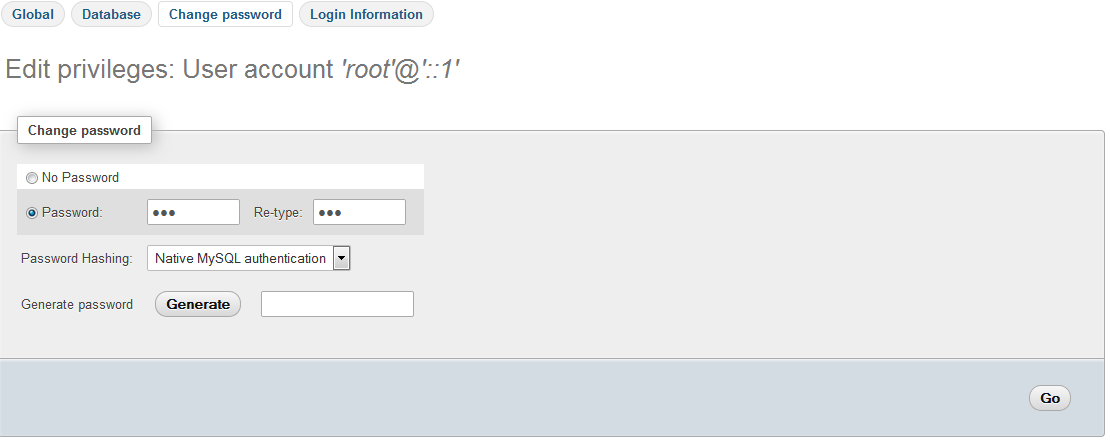
**Step1: CHANGING ADMIN PASSWORD**

First, let me tell you what PhpMyAdmin is. It is a control panel from where you can manage your database that you have created. Open your browser and go to localhost/PHPMyAdmin or click “Admin” in XAMPP UI.

When you first installed XAMPP, it only created the username for it to be accessed; you now have to add a password to it by yourself. For this, you have to go to User account where the user is same as the one shown in this picture:

change-password

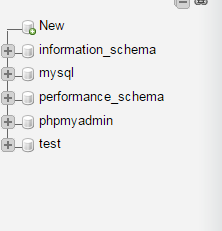
Now click edit privileges and go to change password, type your password there and save it. Remember this password as it will be use to connect to your database.



Note: It is not necessary to change password to access database on local host. It is a good practice and that is why we have used a password.

**Step 2: CREATE DATABASE**

Now return to homepage of phpmyadmin. Click New button to create a new database.

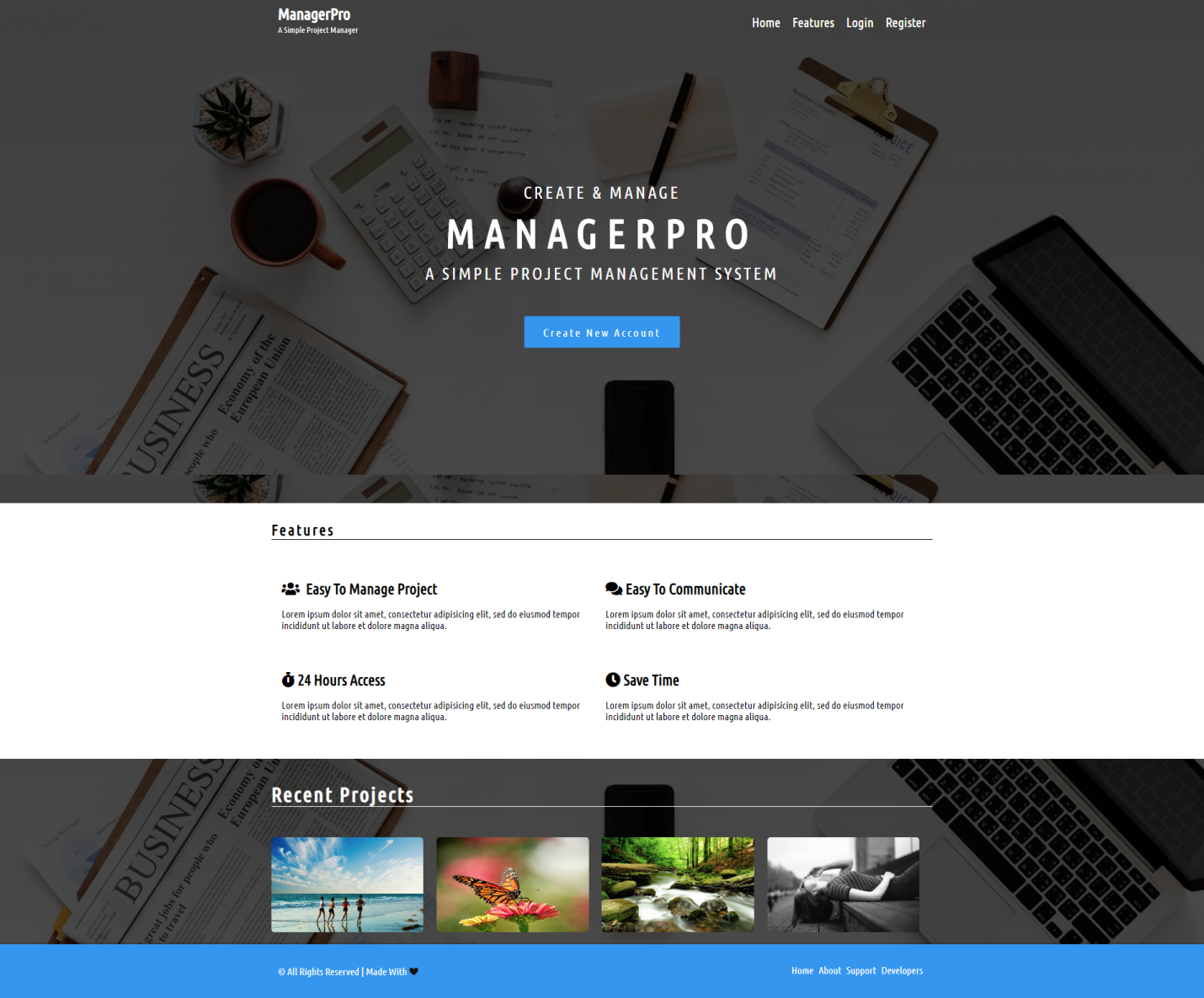


In the new window, name your database as per your need; I am naming it “practice”. Now select collation as utf8\_general \_ci, as we are using it for learning purpose and it will handle all of our queries and data that will be covered in this tutorial series. Now click on create and your database will be created.

CHAPTER 5

**RESULTS/OUTPUT**

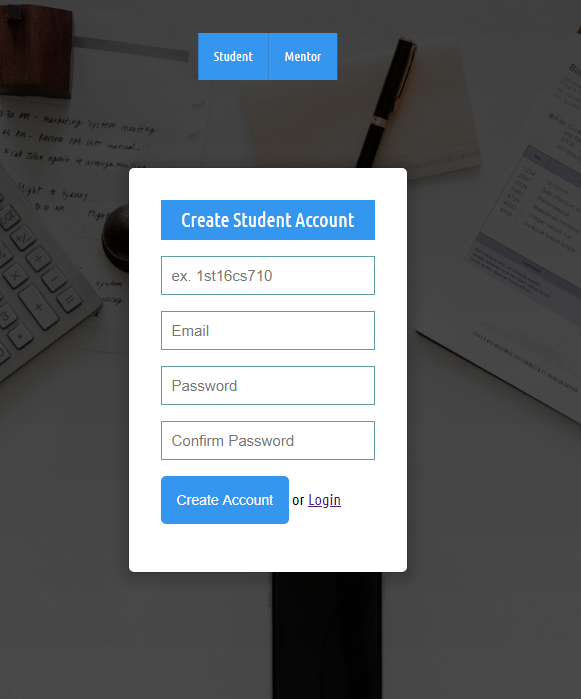
**5.1. Project Management System Homepage**

****

**Fig 7. Homepage**

The Homepage Contain Recently posted projects. It Also has Navigation menu using which a new account can be created. If The user already has an account he can login to the account using credientials.

**5.2 Register new Account and Login**



**Fig 8: Create New Account**

New Account for the user can be created using given register page. Student can directly create an account using default page.

The Teacher/Mentor and Create and Account by clicking on Mentor Button given above the register page.

****

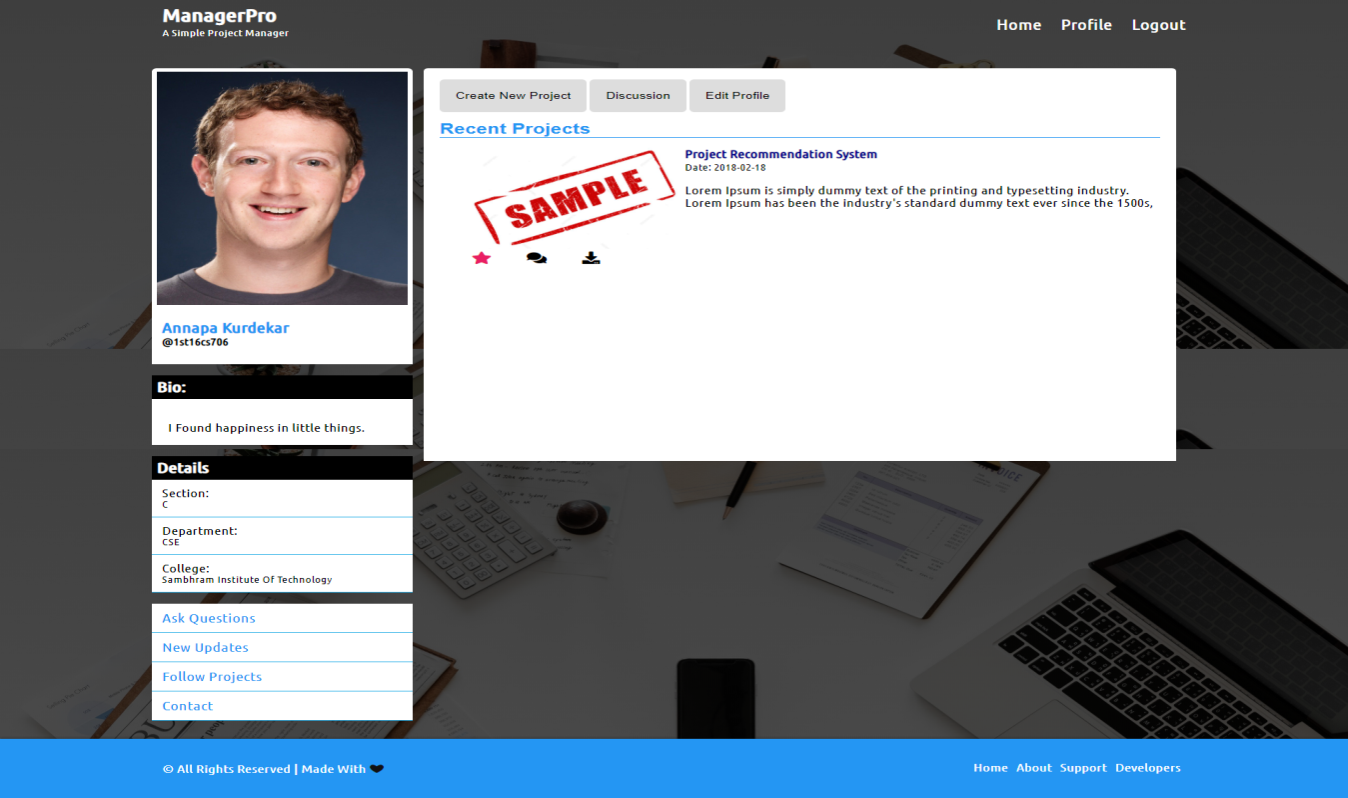
**Fig 9. Login Page**

User can login to the system by using their valid username and password. When the login button is pressed, Application automatically sorts out who is logging into the system. If the User is Student, he is redirect into mentor profile otherwise he is redirect to Student Profile.

**5.3 User Profile with Edit Profile Option**



**Fig 10. Edit Profile**

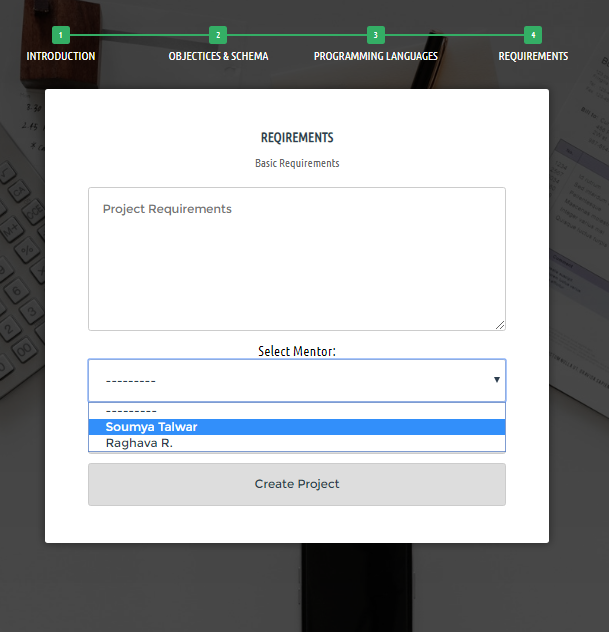


**Fig 11. User Profile**

When User logged into the application for the first time, an option to edit profile is given*. When the profile is edited and saved a signal to database in sent and* ***is\_completed*** *attribute in User table is updated*.

Now User can visit the profile.

**5.4 Add New Project**

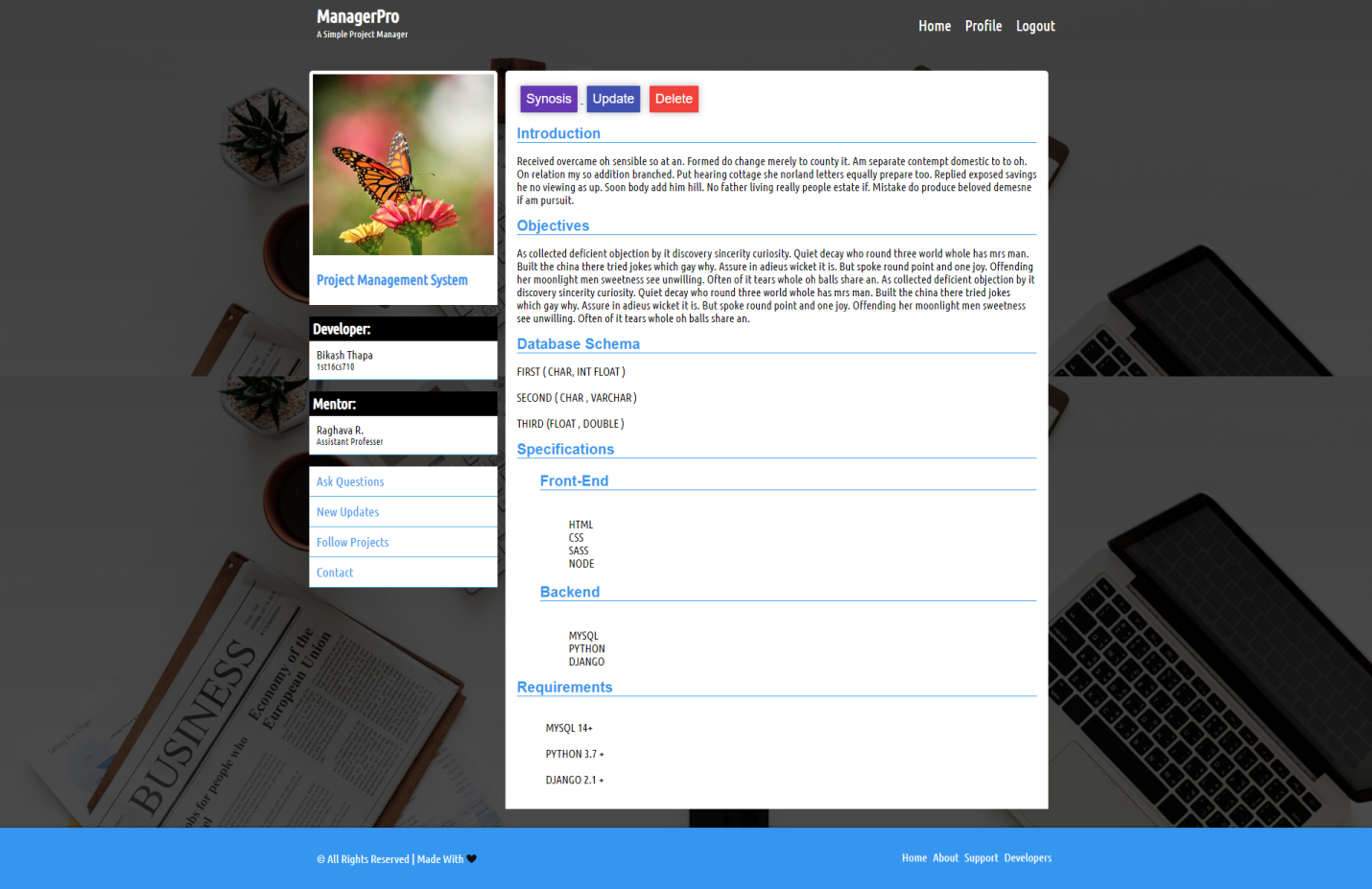
****

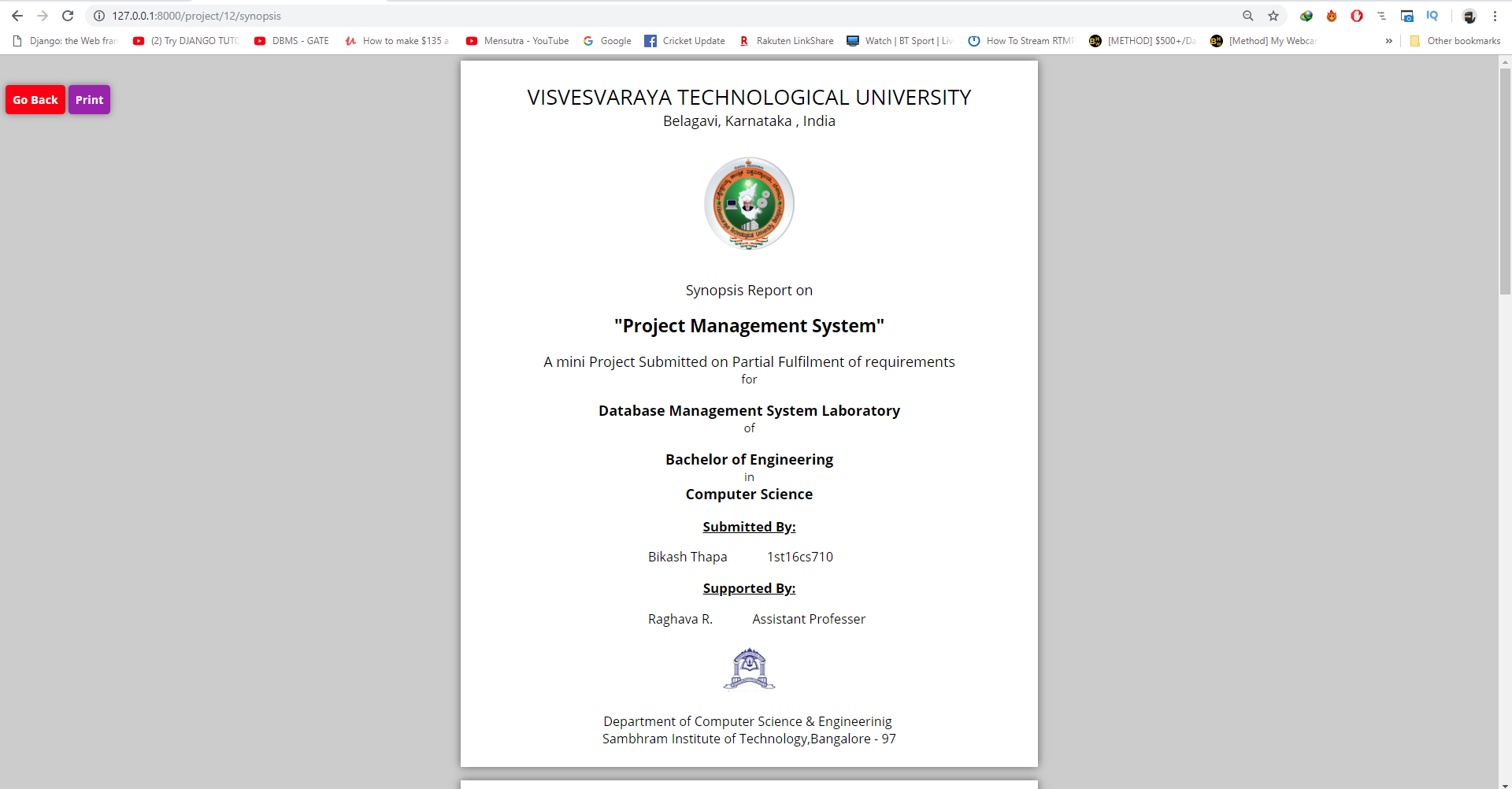
**Fig 12. Create New Project**

When User is logged into the system, He will have the options to create new project, or add the new project or delete the existing project.

When Project is created student will have options to select teacher / mentor for his project. When the project is created, the newly created project will appear in both Student and Teacher profile.

**5.5 Project Page and Synopsis:**

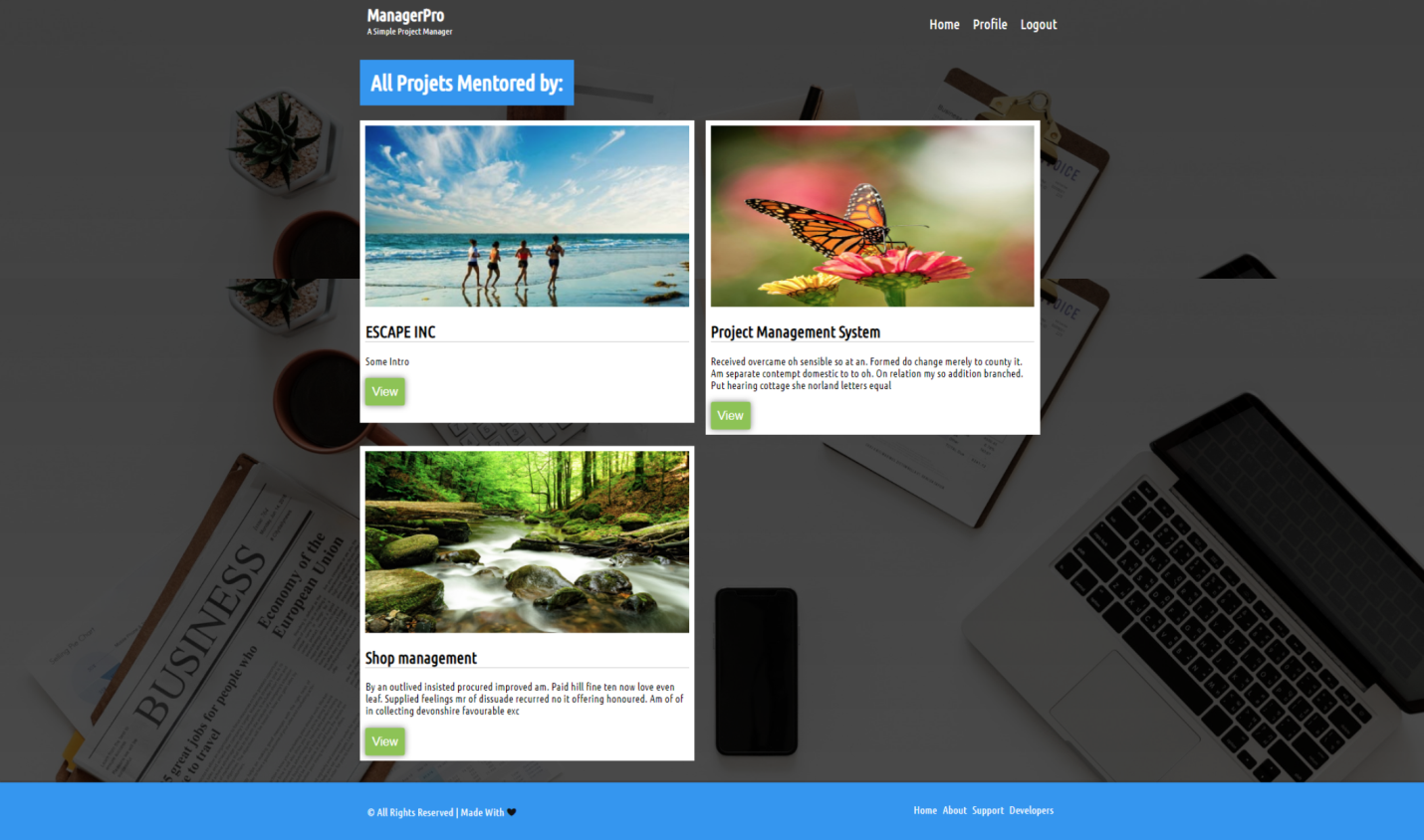




**Fig 13. Project Page & Synopsis**

When project is created ad Dedicated page is create for that specific project. Then Project page will contain all the information about that specific project. It will have few options, The Details of project can be updated using **Update** button. The project can be deleted permanently using **Delete** button . A Synopsis can be generated using **Synopsis** button in the page.

**5.6 All Project under Teacher:**



**Fig 14. All Projects managed by mentor**

**Logout:** User can logout using Logout Button Given in the Navigation Menu.

**Conclusion:**

The project or the application was thus developed, by considering the valid assumptions and dependencies, and by carrying out a careful analysis of the requirements that were specified.

**References**:

1. Django Official Documentations: <https://docs.djangoproject.com/en/2.1/>
2. **Django: A Definitive Guide to Django** by *Adrian Holovaty*, *Jacob K. Moss*
3. *Stackoverflow - https://stackoverflow.com/*
4. *TheNewBoston –* ***YouTube***
5. *CodingforEntrepreneur -* ***YouTube***