Contents

List of Figures

List of Tables

ABBREVIATIONS

DBMS	Database Management System
SQL	Structure Query Language
PHP	Hypertext Preprocessor
HTML	Hypertext Markup Language
CSS	Cascading Style Sheet
ER Diagram	Entity Relationship Diagram

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

At the University of VTU, both freshmen and continuing students can apply to a large number of scholarships. They need to apply for scholarships explicitly illustrating their qualifications and eligibility for the scholarships they apply. There are various criteria to satisfy when a student applies for a scholarship. The majority of criteria are based on students' merits and other academic performances. Their financial information is another main consideration for how much scholarship they can receive. Some scholarships may also require that applicants have to take specific courses or select a particular major. The Scholarship Foundation Office plays the role as a sponsor in maintaining scholarships information, processing applications.

1.2 DBMS (DATABASE MANAGEMENT SYSTEM)

Database is a collection of related data and data is a collection of facts and figures that can be processed to produce information. Mostly data represents recordable facts. Data aids in producing information, which is based on facts. For example, if we have data about marks.

A database management system (DBMS) is a software package designed to define, manipulate, retrieve and manage data in a database. A DBMS generally manipulates the data itself, the data format, field names, record structure and file structure. It also defines rules to validate and manipulate this data.

A DBMS relieves users of framing programs for data maintenance. Fourth- generation query languages, such as SQL, are used along with the DBMS package to interact with a database.

- MySQL
- SQL Server
- Oracle
- dBase
- FoxPro

1.3 PHP (HYPERTEXT PREPROCESSOR)

PHP is the most popular and widely used server-side scripting language for web development. It is used to make the Dynamic pages in websites. Rasmus Lerdorf was the creator of PHP in 1995. PHP codes are embedding in HTML source codes for making the page dynamic. PHP can deal with most of the requirements in web development like Database, File handling, String operations, Arrays, Graphics, File Uploads, Data processing etc. PHP can be used in any operating system with a web server Supports PHP. Apache web server is one of the popular web servers dealing with PHP + MySQL. Moreover, PHP is absolutely free to use.

1.4 PROBLEM STATEMENT

Students across the universities should have the tools and means to apply for scholarship in a brief period of time without having them to go through a long process. Currently, paper applications are overwhelming many of the employees by having them to do more work manually. This hinders productivity and also requires more time to verify and sanction the scholarship to the students. We propose that, using an online portal for students to apply for scholarship by entering/uploading all the required information, which then will be verified by their respective colleges, later the scholarship department can sanction the scholarship for the applications verified by the colleges.

1.5 OBJECTIVES

The main objectives of this project is to:

1. Reduce paper works
2. Reduce delay
3. Offer an easy method for students to apply scholarship

CHAPTER 2

REQUIREMENT SPECIFICATION

2.1 HARDWARE REQUIREMENTS

The hardware required for the development of this project is:

Processor	:Intel Core i3
Processor speed	:1.7 GHz
RAM	:2 GB
System Type	:64-Bit Operating System

2.2 SOFTWARE REQUIREMENTS

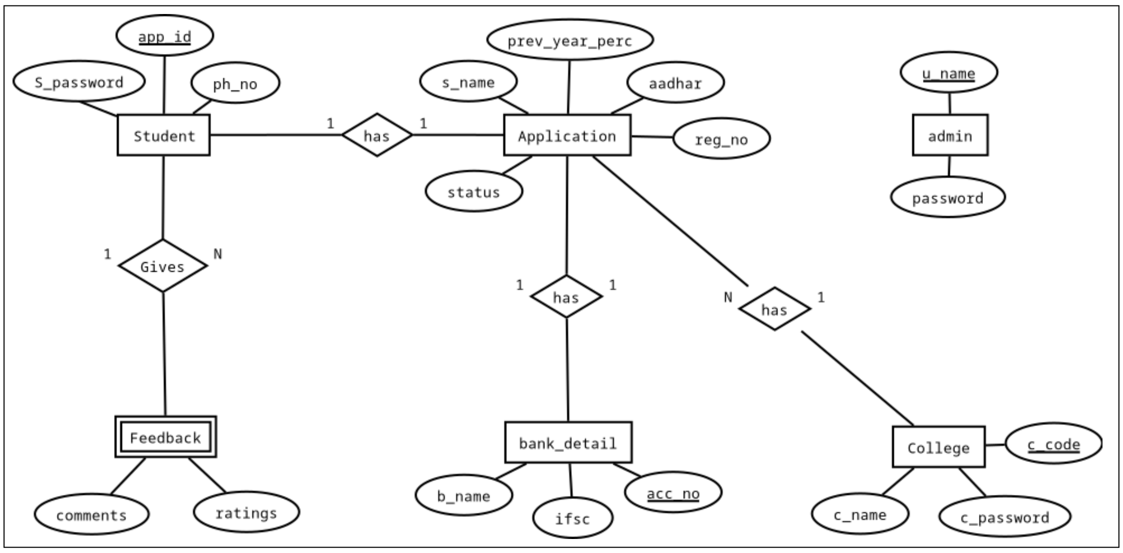
The software required for the development of this project is:

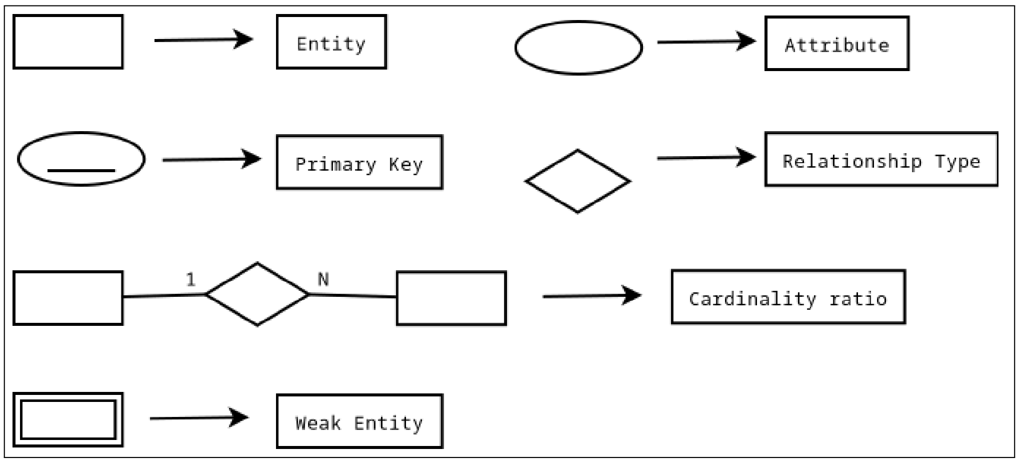
Sofware	:Xampp
OS	:Platform independent
Front End	:HTML and CSS
Programming Language	:PHP and SQL
Database	:MariaDB

CHAPTER 3

DESIGNS

3.1 ENTITY-RELATIONSHIP DIAGRAM





3.2 DESCRIPTION

The ER Model figure shows conceptual view of the database. It works around real-world entities and the associations among them. At view level, the ER model is considered a good option for designing databases. So, let's see each entity.

ADMIN TABLE

This entity stores the login credentials of the admin.

STUDENT TABLE

This entity stores the information about the student login credentials along with phone number. Attributes are app_id, ph_no, S_password.

APPLICATION TABLE

This entity stores the information about the student. The attributes are app_id, c_code, s_name, aadhar, reg_no, prev_year_perc, status.

COLLEGE TABLE

This entity stores information about the college. The attributes are c_code, c_name and c_password.

BANK_DETAIL TABLE

This entity stores the bank details of the student. Attributes are ifsc, acc_no, b_name, app_id.

FEEDBACK TABLE

This entity stores the details of student's feedback. Attibutes are app_id, ratings, comments.

3.3 SEVEN STEPS FOR ER TO SCHEMA CONVERSION

Step 1: Mapping of Regular Entity Types.

For each regular (strong) entity type E in the ER schema, create a relation R that includes all the simple attributes of E. Include only the simple component attributes of a composite attribute. Choose one of the key attributes of E as the primary key for R. If the chosen key of E is a composite, then the set of simple attributes that form it will together form the primary key of R. If multiple keys were identified for E during the conceptual design, the information describing the attributes that form each additional key is kept in order to specify secondary (unique) keys of relation R. Knowledge about keys is also kept for indexing purposes and other types of analyses.

Step 2: Mapping of Weak Entity Types.

For each weak entity type W in the ER schema with owner entity type E, create a relation R and include all simple attributes (or simple components of composite attributes) of was attributes of R. In addition, include as foreign key attributes of R, the primary key attribute(s) of the relation(s) that correspond to the owner entity type(s); this takes care of mapping the identifying relationship type of W. The primary key of R is the combination of the primary key(s) of the owner(s) and the partial key of the weak entity type W, if any. If there is a weak entity type E2 whose owner is also a weak entity type E1, then E1 should be mapped before E2 to determine its primary key first.

Step 3: Mapping of Binary 1:1 Relationship Types.

For each binary 1:1 relationship type R in the ER schema, identify the relations S and T that correspond to the entity types participating in R. There are three possible approaches:

1. The foreign key approach.
2. The merged relationship approach, and

The first approach is the most useful and should be followed unless special conditions exist, as we discuss below.

1. **Foreign key approach:**
   Choose one of the relations—S, say—and include as a foreign key in S the primary

key of T. It is better to choose an entity type with total participation in R in the role of S. Include all the simple attributes (or simple components of composite attributes) of the 1:1 relationship type R as attributes of S.

2. **Merged relation approach:**
   An alternative mapping of a 1:1 relationship type is to merge the two entity types and the relationship into a single relation. This is possible when both participations are total, as this would indicate that the two tables will have the exact same number of tuples at all times.

3. **Cross-reference or relationship relation approach:**
   The third option is to set up a third relation R for the purpose of cross-referencing the primary keys of the two relations S and T representing the entity types. As we will see, this approach is required for binary M: N relationships. The relation R is called a relationship relation (or sometimes a lookup table), because each tuple in R represents a relationship instance that relates one tuple from S with one tuple from T. The relation R will include the primary key attributes of S and T as foreign keys to S and T. The primary key of R will be one of the two foreign keys, and the other foreign key will be a unique key of R. The drawback is having an extra relation, and requiring an extra join operation when combining related tuples from the tables.

Step 4: Mapping of Binary 1: N Relationship Types.

For each regular binary 1: N relationship type R, identify the relation S that represents the participating entity type at the N-side of the relationship type. Include as foreign key in S the primary key of the relation T that represents the other entity type participating in R; we do this because each entity instance on the N-side is related to at most one entity instance on the 1-side of the relationship type. Include any simple attributes (or simple components of composite attributes) of the 1: N relationship type as attributes of S.

Step 5: Mapping of Binary M: N Relationship Types.

For each binary M: N relationship type R, create a new relation S to represent R. Include as foreign key attributes in S the primary keys of the relations that represent the participating entity types; their combination will form the primary key of S. Also include any simple attributes of the M: N relationship type (or simple components of composite attributes) as attributes of S. Notice that we cannot represent an M: N relationship type by a single foreign key attribute in one of the participating relations (as we did for 1:1 or 1: N relationship types) because of the M: N cardinality ratio; we must create a separate relationship relation S.

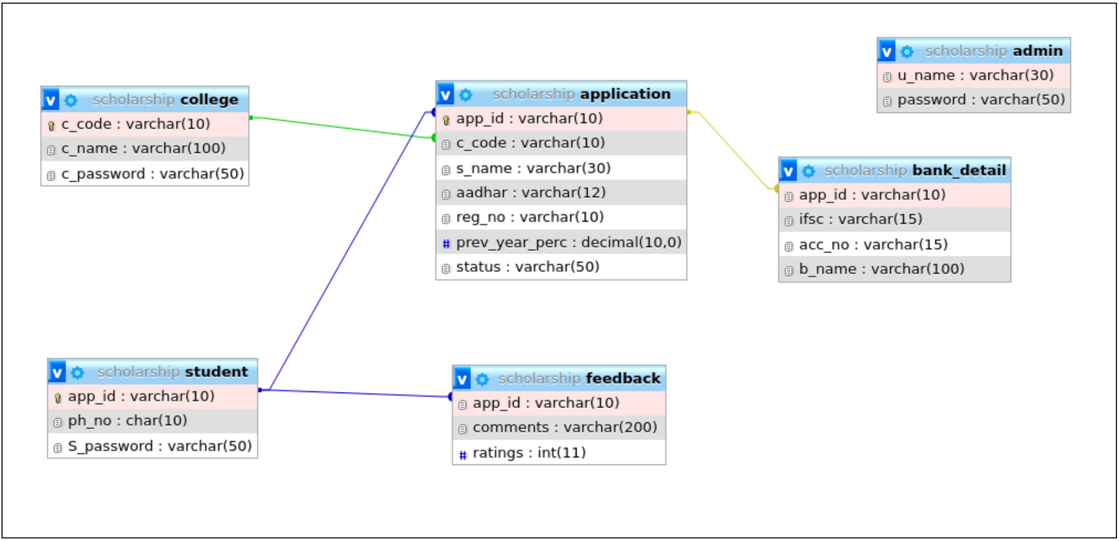
Step 6: Mapping of Multivalued Attributes.

For each multivalued attribute A, create a new relation R. This relation R will include an attribute corresponding to A, plus the primary key attribute K—as a foreign key in R—of the relation that represents the entity type or relationship type that has A as a multivalued attribute. The primary key of R is the combination of A and K. If the multivalued attribute is composite, we include its simple components.

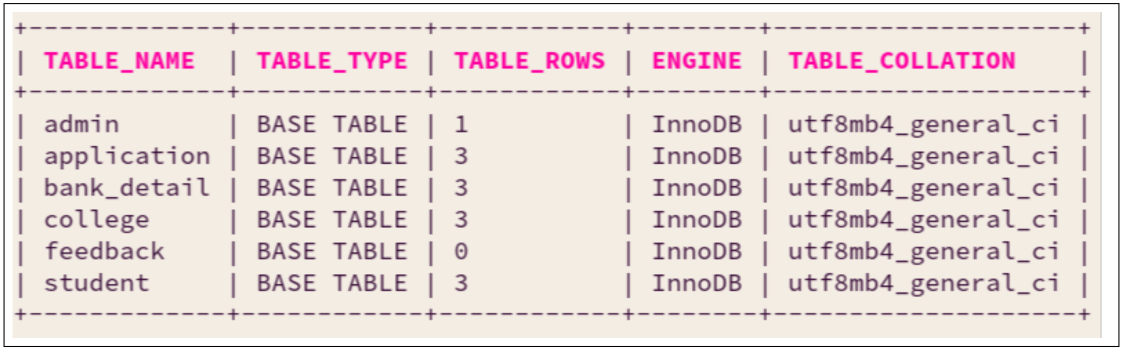
Step 7: Mapping of N-array Relationship Types.

For each n-array relationship type R, where n > 2, create a new relation S to represent R. Include as foreign key attributes in S the primary keys of the relations that represent the participating entity types. Also include any simple attributes of the n-array relationship type (or simple components of composite attributes) as attributes of S. The primary key of S is usually a combination of all the foreign keys that reference the relations representing the participating entity types. However, if the cardinality constraints on any of the entity types E participating in R is 1, then the primary key of S should not include the foreign key attribute that references the relation E ‘corresponding to E.

3.4 SCHEMA DIAGRAM

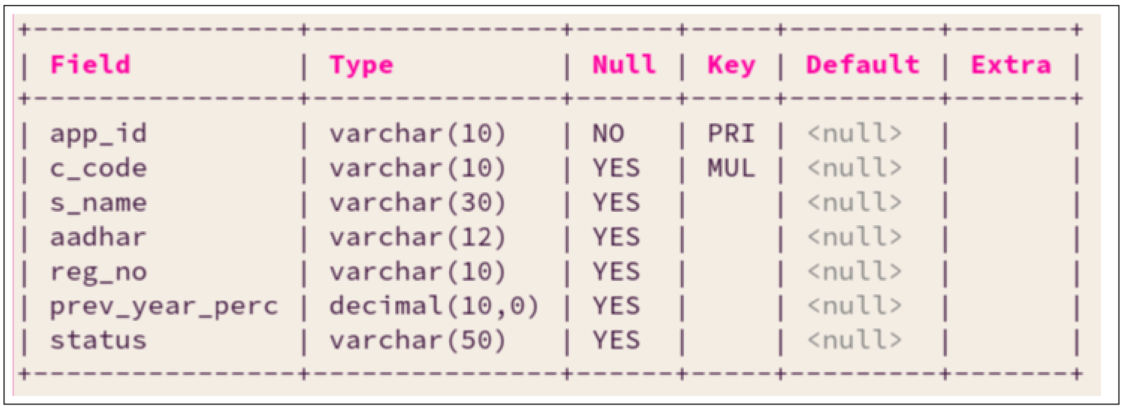


3.5 DATABASE DESCRIPTION



```
+-------------+------------+------------+--------+--------------------+
| TABLE_NAME  | TABLE_TYPE | TABLE_ROWS | ENGINE | TABLE_COLLATION    |
+-------------+------------+------------+--------+--------------------+
| admin       | BASE TABLE | 1          | InnoDB | utf8mb4_general_ci |
| application | BASE TABLE | 3          | InnoDB | utf8mb4_general_ci |
| bank_detail | BASE TABLE | 3          | InnoDB | utf8mb4_general_ci |
| college     | BASE TABLE | 3          | InnoDB | utf8mb4_general_ci |
| feedback    | BASE TABLE | 0          | InnoDB | utf8mb4_general_ci |
| student     | BASE TABLE | 3          | InnoDB | utf8mb4_general_ci |
+-------------+------------+------------+--------+--------------------+
```

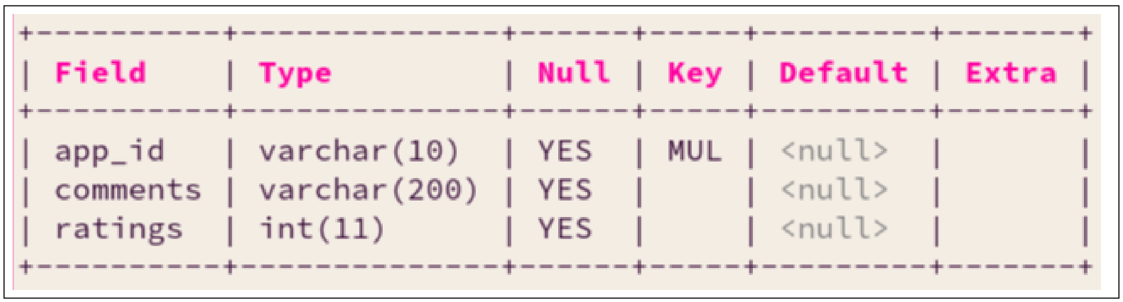
Table 3.5.1 Description of project database Table 3.5.1 shows description of all tables in the database scholarship



```
+----------------+----------------+------+-----+---------+-------+
| Field          | Type           | Null | Key | Default | Extra |
+----------------+----------------+------+-----+---------+-------+
| app_id         | varchar(10)    | NO   | PRI | <null>  |       |
| c_code         | varchar(10)    | YES  | MUL | <null>  |       |
| s_name         | varchar(30)    | YES  |     | <null>  |       |
| aadhar         | varchar(12)    | YES  |     | <null>  |       |
| reg_no         | varchar(10)    | YES  |     | <null>  |       |
| prev_year_perc | decimal(10,0)  | YES  |     | <null>  |       |
| status         | varchar(50)    | YES  |     | <null>  |       |
+----------------+----------------+------+-----+---------+-------+
```

Table 3.5.2: Application Table

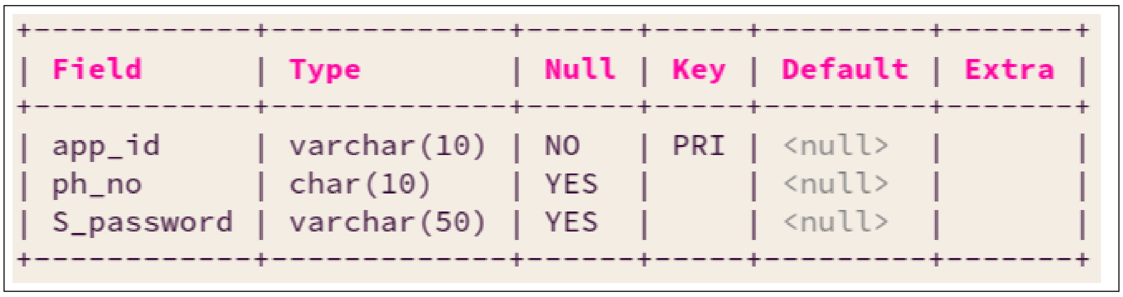
Table 3.5.2 shows structure and details of application table



```
+----------+--------------+------+-----+---------+-------+
| Field    | Type         | Null | Key | Default | Extra |
+----------+--------------+------+-----+---------+-------+
| app_id   | varchar(10)  | YES  | MUL | <null>  |       |
| comments | varchar(200) | YES  |     | <null>  |       |
| ratings  | int(11)      | YES  |     | <null>  |       |
+----------+--------------+------+-----+---------+-------+
```

Table 3.5.3: Feedback Table

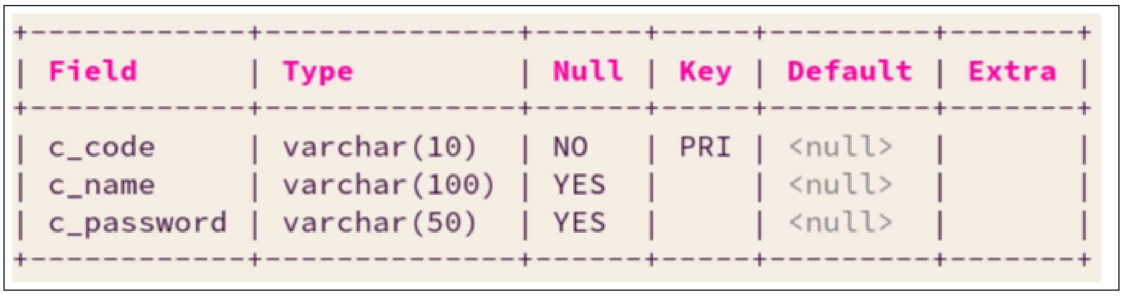
Table 3.5.3 shows structure and details of the user feedback table



```
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| app_id     | varchar(10) | NO   | PRI | <null>  |       |
| ph_no      | char(10)    | YES  |     | <null>  |       |
| S_password | varchar(50) | YES  |     | <null>  |       |
+------------+-------------+------+-----+---------+-------+
```

Table 3.5.4: Student Table

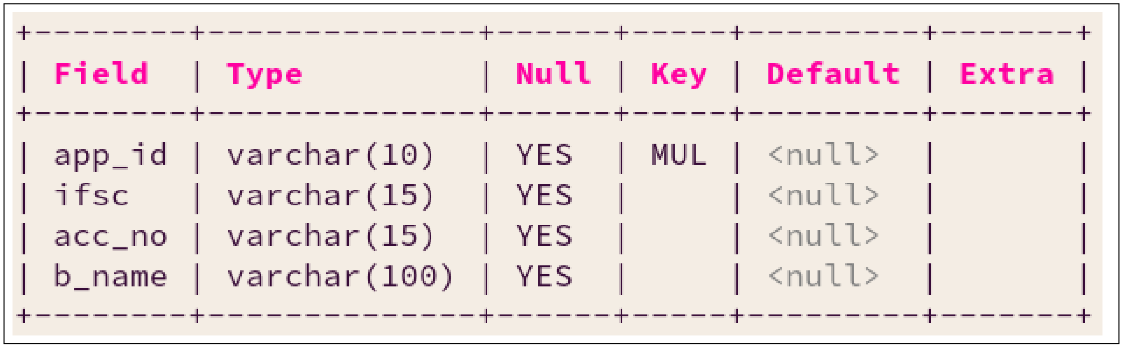
Table 3.5.4 shows structure and details of the student login information table



```
+------------+--------------+------+-----+---------+-------+
| Field      | Type         | Null | Key | Default | Extra |
+------------+--------------+------+-----+---------+-------+
| c_code     | varchar(10)  | NO   | PRI | <null>  |       |
| c_name     | varchar(100) | YES  |     | <null>  |       |
| c_password | varchar(50)  | YES  |     | <null>  |       |
+------------+--------------+------+-----+---------+-------+
```

Table 3.5.5: College Table

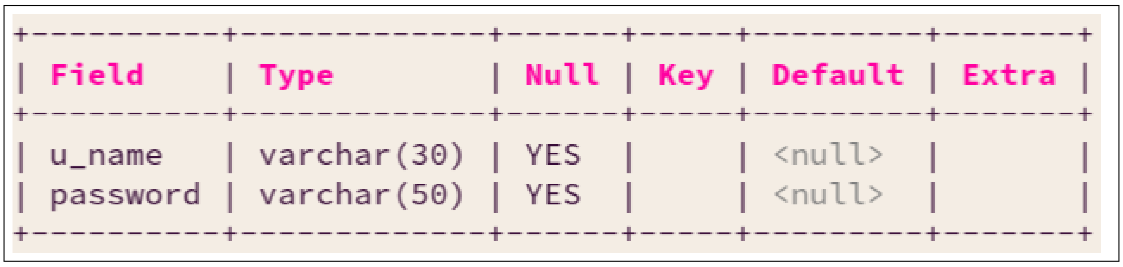
Table 3.5.5 shows structure and details of the College table



```
+--------+--------------+------+-----+---------+-------+
| Field  | Type         | Null | Key | Default | Extra |
+--------+--------------+------+-----+---------+-------+
| app_id | varchar(10)  | YES  | MUL | <null>  |       |
| ifsc   | varchar(15)  | YES  |     | <null>  |       |
| acc_no | varchar(15)  | YES  |     | <null>  |       |
| b_name | varchar(100) | YES  |     | <null>  |       |
+--------+--------------+------+-----+---------+-------+
```

Table 3.5.6: bank_detail Table

Table 3.5.6 shows structure and details of the bank_detail table



```
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| u_name   | varchar(30) | YES  |     | <null>  |       |
| password | varchar(50) | YES  |     | <null>  |       |
+----------+-------------+------+-----+---------+-------+
```

Table 3.5.7: Admin Table

Table 3.5.7 shows structure and details of the admin table

CHAPTER 4

IMPLEMENTATION CODE

4.1 IMPLEMENTATION CODE

4.1.1 CONNECTION CODE FOR FRONT END TO BACK END

```
<?php
// connect to the database
$conn = mysqli_connect('localhost',
                       'root', '',
                       'scholarship');

// check connection
if(!$conn){
        echo 'Connection error: '. mysqli_connect_error();
}
?>
```

4.1.2 CODE TO DELETE COLLEGE

```
<?php
if(!(array_filter($errors))){
    // escape sql chars
    $c_code = mysqli_real_escape_string($conn,
                                        $_POST['c_code']);
    // query to delete
    $sql = "DELETE FROM college WHERE c_code='$c_code'";

    // check whether query executed successfully
    if(mysqli_query($conn, $sql)){
        // Do nothing for now
    } else{
        $errors['query'] = 'Unable to delete: ' .
                                mysqli_error($conn);
    }
}
?>
```

4.1.3 CODE TO INSERT NEW COLLEGE

```
<?php
$sql = "INSERT INTO college VALUES ('$c_code',
                                    '$c_name',
                                    '$c_password')";


// check whether the query executed without any error
if(mysqli_query($conn, $sql)){
// Do nothing
} else{
  $errors['query'] = 'Unable to insert: ' . mysqli_error($conn);
}
?>
```

4.1.4 CODE FOR VERIFYING APPLICATION

```
<?php
$sql_ver = "UPDATE application\
            SET status='verified'\
            WHERE app_id='$id_to_update'";


if(mysqli_query($conn, $sql_ver)){
  header('Location: index.php');
} else {
  echo 'query error: '. mysqli_error($conn);
}
?>
```

4.1.5 CODE FOR DECLINING APPLICATION

```php
<?php
$sql_dec = "UPDATE application\
            SET status='declined'\
            WHERE app_id='$id_to_update'";

if(mysqli_query($conn, $sql_dec)){
  header('Location: index.php');
} else {
  echo 'query error: '. mysqli_error($conn);
}
?>
```

4.1.6 CODE FOR INSERTING DATA INTO STUDENT, BANK_DETAIL AND APPLICATION TABLE

```
<?php
$sql = "INSERT INTO student\
        VALUES ('$app_id',\
                '$phno',\
                '$password')";
if(mysqli_query($conn, $sql)){
} else {
  echo 'query error: '. mysqli_error($conn);}
$sql = "INSERT INTO application\
        VALUES('$app_id','$c_code',\
               '$s_name', '$aadhar',\
               '$reg_no', '$prev_year_perc',\
               'Application Submitted')";
if(mysqli_query($conn, $sql)){
} else {
  echo 'query error: '. mysqli_error($conn);}
$sql = "INSERT INTO bank_detail
        VALUES('$app_id','$ifsc',\
               '$acc_no', '$b_name')";
if(mysqli_query($conn, $sql)){
  session_start();
  $_SESSION['app_id'] = $app_id;
  header('Location: studetail.php');
} else {
  echo 'query error: ' . mysqli_error($conn);}?>
```

CHAPTER 5

SNAPSHOTS

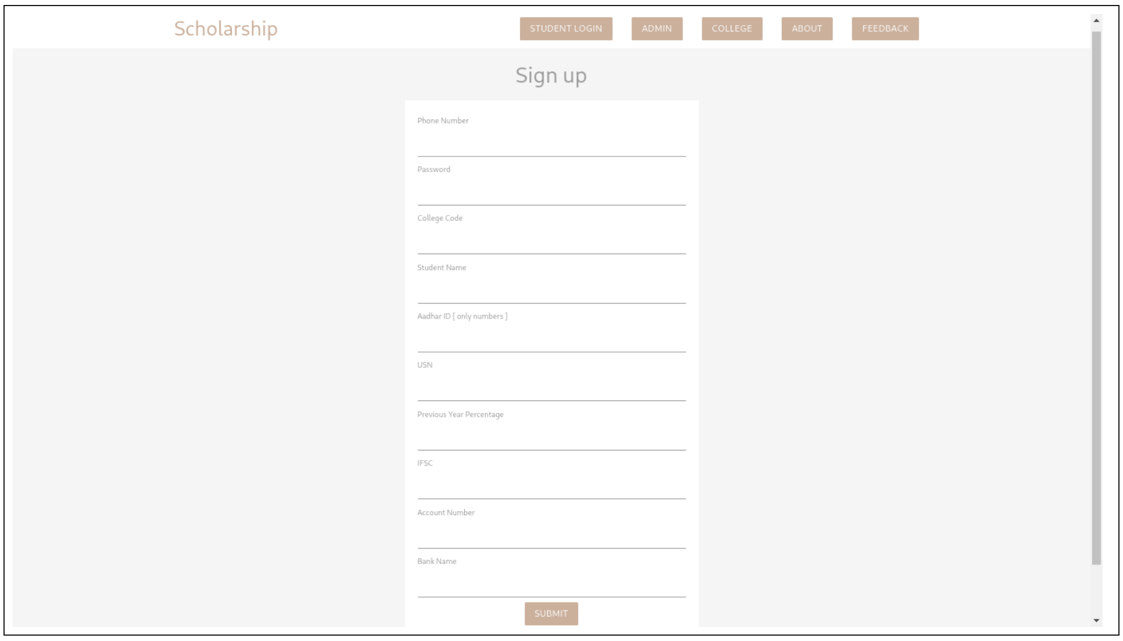


Figure 5.1: **Student Registration Page**

Figure 5.1 shows student can register to his new account to apply for scholarship.

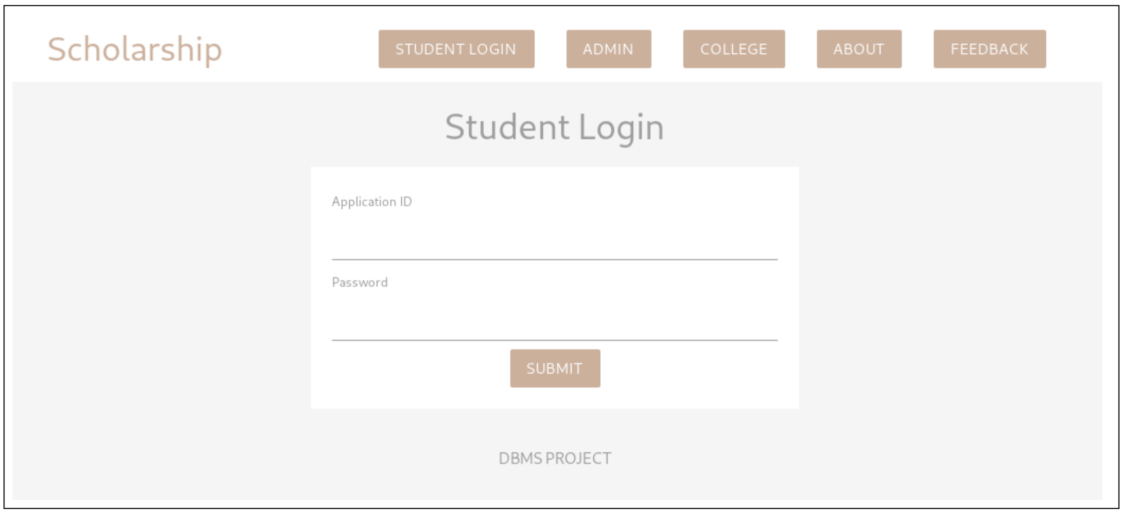
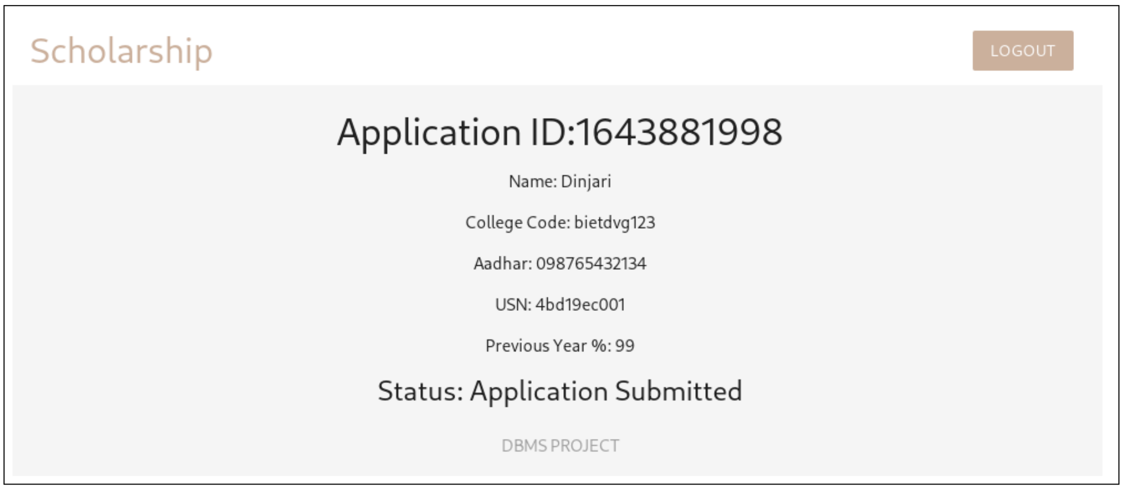


Figure 5.2: **Student Login Page**

Figure 5.2 shows student can Login with his credentials to view his application status.



Application ID:1643881998

Name: Dinjari

College Code: bietdvg123

Aadhar: 098765432134

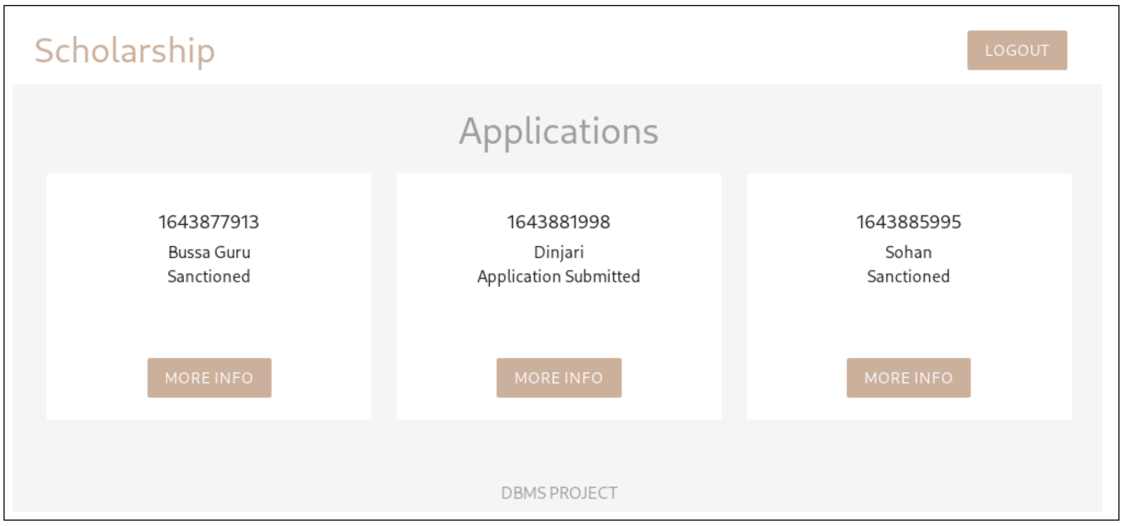
USN: 4bd19ec001

Previous Year %: 99

Status: Application Submitted

Figure 5.3: **Student Profile**

Figure 5.3 shows student can view status of his/her application



1643877913
Bussa Guru
Sanctioned

1643881998
Dinjari
Application Submitted

1643885995
Sohan
Sanctioned

Figure 5.4: **College view**

Figure 5.4 Shows College can view all the applications belonging to that college.

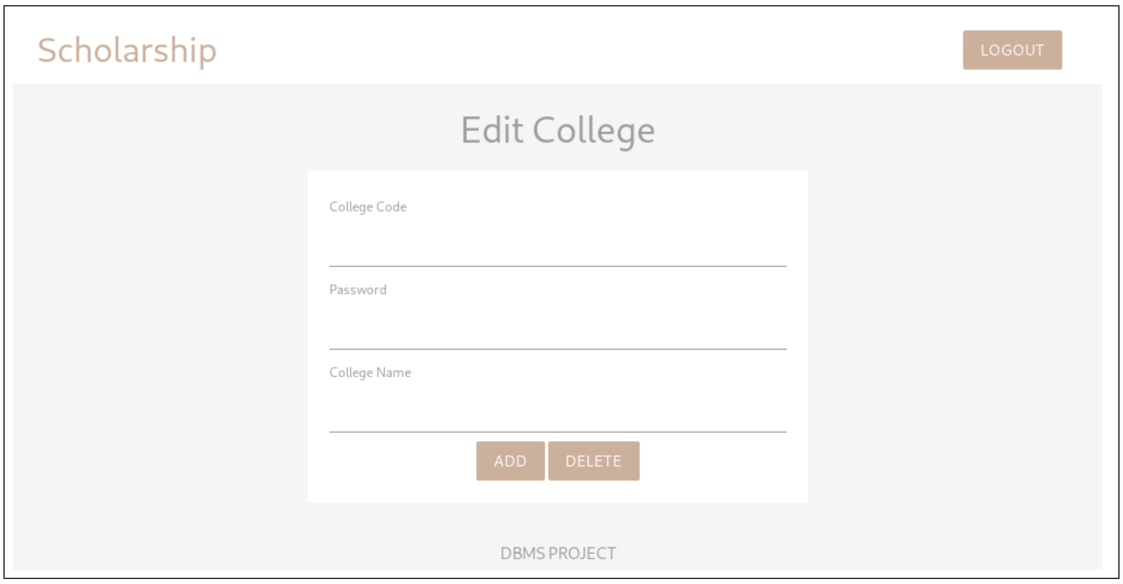
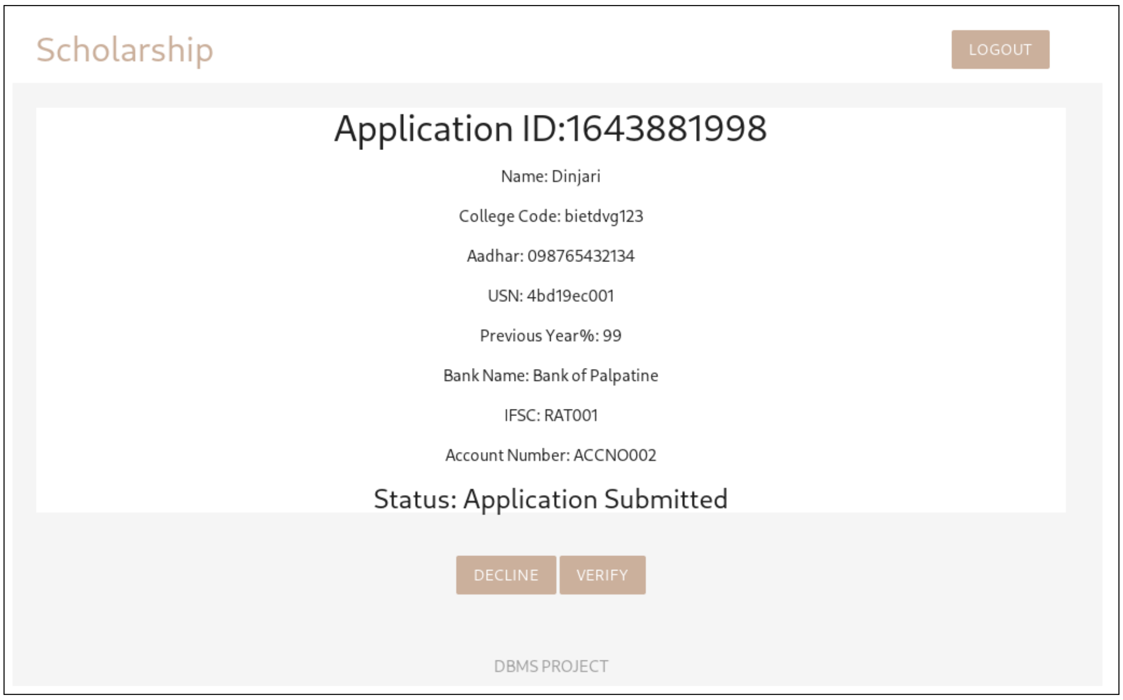


Figure 5.5: **Edit College**

Figure 5.5 shows Admin can add/delete colleges.



Scholarship

LOGOUT

Application ID:1643881998

Name: Dinjari

College Code: bietdvg123

Aadhar: 098765432134

USN: 4bd19ec001

Previous Year%: 99

Bank Name: Bank of Palpatine

IFSC: RAT001

Account Number: ACCNO002

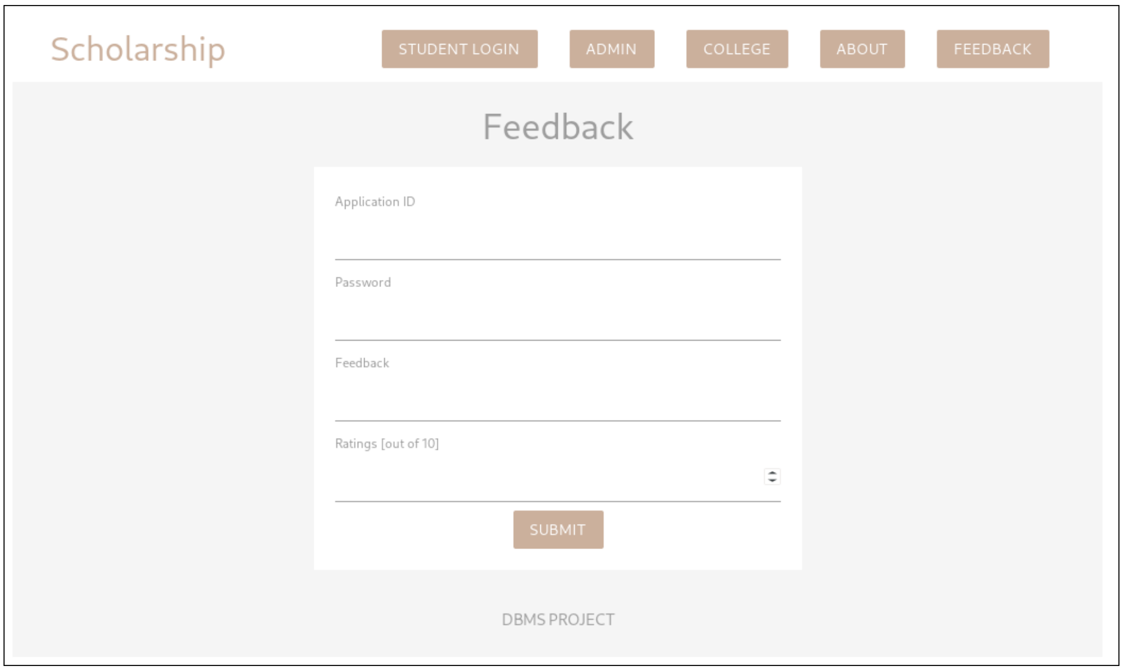
Status: Application Submitted

DECLINE VERIFY

DBMS PROJECT

Figure 5.6: **Verify/Decline Application**

Figure 5.6 shows College can either verify/decline the applications.



Scholarship

STUDENT LOGIN ADMIN COLLEGE ABOUT FEEDBACK

Feedback

Application ID

Password

Feedback

Ratings [out of 10]

SUBMIT

DBMS PROJECT

Figure 5.7: **Feedback Form**

Figure 5.7 shows student can submit feedbacks.

CONCLUSION

The scholarship portal handles the scholarship applications of various students belonging to different colleges. This project intends to reduce the heavy paperwork, which is otherwise involved, in the manual system by computerizing all the transactions of scholarship department like issuing of forms, submission of forms, and sanction from scholarship department and distribution to the students resulting in a faster processing of the scholarship applications.

REFERENCES

[1] Fundamentals of database systems, Ramez Elmasri and S B Navathe, 7^{th} Edition, 2017, Pearson.

[2] Database management systems, Ramakrishnan, and Gehrke, 3^{rd} Edition, 2014, McGraw Hill.

[3] Programming PHP, 4^{th} Edition, Kevin Tatroe and Peter MacIntyre, O'Reilly Media.

[4] Learning PHP, MySQL & JavaScript With jQuery, CSS & HTML5, 5^{th} Edition, Robin Nixon, O'Reilly Media.