**A Web-Based System to Keep**

**Track Of University-Related Information**

*A Project Report Submitted*

*to*

**MANIPAL ACADEMY OF HIGHER EDUCATION**

*For Partial Fulfillment of the Requirement for the*

*Award of the Degree*

*Of*

**Bachelor of Technology**

*in*

**Computer Science and Engineering**

*by*

**Bryan Jo Antony, Mavesh Damodaran**

**21095011, 21095021**

**28th APRIL 2023**

**SYNOPSIS & ABSTRACT**

**Our objective is to develop a web-based system to track student registration and grade data for university courses. When a student enrolls in the institution, they can examine their individual personal information, such as their registration number and contact information, on the Student Details page as well as their course information, timetables and grades on a separate Academics page.**

**The website will also have several other features, such as:**

**❖ Separate Teacher and Student Logins: This will increase security by granting**

**teachers and students access to the website at different levels. For instance,**

**professors will be able to update grades but students can solely view and verify**

**them.**

**❖ Student Feedback: Students can submit comments on courses, teachers, and the**

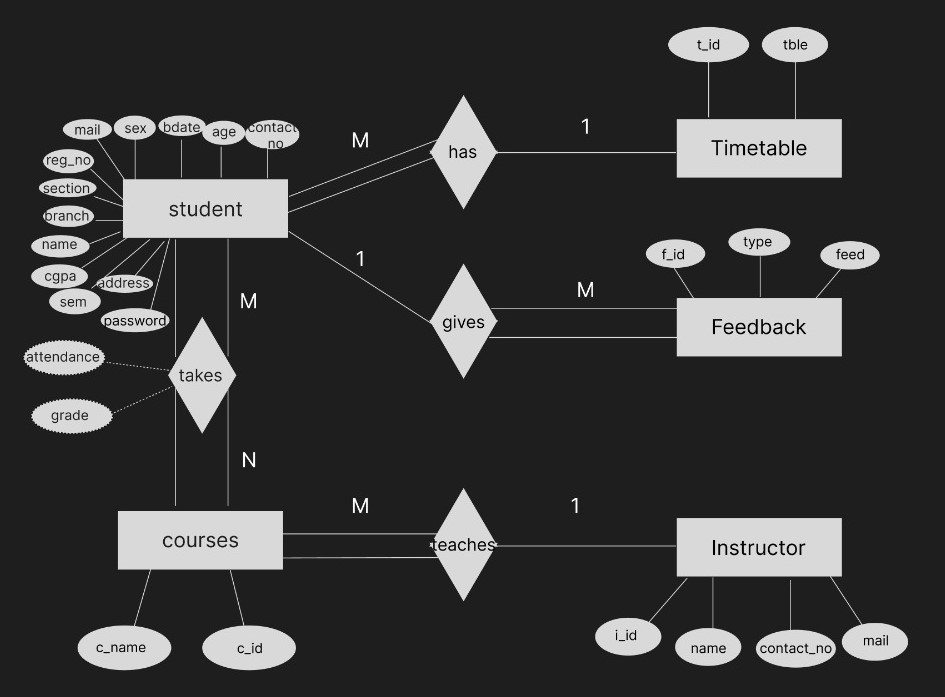
**student database website itself thanks to this feature.**

**❖ Timetables: Displays pre-filled timetables from the database in accordance to the**

**students respective branch and section.**

**❖ Robust Database: Usage of a well-structured database to store records in multiple tables.**

**Design Of Database**



**Schema:-**

Student(reg\_no, t\_id, name, age, sex, bdate, sem, contact\_no, mail, cgpa, section, branch, address, passwd)

Takes(reg\_no, c\_id, grade, attendace)

Courses(c\_id, c\_name, i\_id)

Instructor(i\_id, name, contact\_no, mail)

Feedback (f\_id, type, feedb, reg\_no)

Timetable(t\_id, tble)

**DDL Commands**

**create table student(**

**reg\_no number(10),**

**t\_id varchar(10) not null,**

**name varchar(20) not null,**

**age number(3) not null check(age between 1 and 99),**

**sex varchar(10) not null check(sex in ('M','F')),**

**bdate date not null,**

**sem number(1) not null,**

**contact\_no number(10) not null,**

**mail varchar(30) not null,**

**cgpa number(4,2) not null,**

**section varchar(1) not null,**

**branch varchar(20) not null,**

**address varchar(40) not null,**

**passwd varchar(30) not null,**

**primary key(reg\_no)**

**);**

**create table takes(**

**reg\_no number(10),**

**c\_id varchar(10),**

**grade varchar(1) not null,**

**attendance number(5,2) not null,**

**primary key (reg\_no,c\_id)**

**);**

**create table courses(**

**c\_id varchar(10),**

**c\_name varchar(20) not null,**

**i\_id varchar(5) not null,**

**primary key(c\_id)**

**);**

**create table instructor(**

**i\_id varchar(5),**

**name varchar(20) not null,**

**contact\_no number(10) not null,**

**mail varchar(30) not null,**

**primary key(i\_id)**

**);**

**create table feedback(**

**f\_id number(10),**

**type varchar(15) not null check(type in('Instructor','Course','Portal')),**

**feedb varchar(50) not null,**

**reg\_no number(10) not null,**

**primary key(f\_id)**

**);**

**create table timetable(**

**t\_id varchar(10),**

**tble varchar(200) not null,**

**primary key(t\_id)**

**);**

**alter table student add constraint foreign key(t\_id) references timetable(t\_id);**

**alter table takes add constraint foreign key(c\_id) references courses(c\_id);**

**alter table takes add constraint foreign key(reg\_no) references student(reg\_no);**

**alter table feedback add constraint foreign key(reg\_no) references student(reg\_no);**

**alter table courses add constraint foreign key(i\_id) references instructor(i\_id);**

**alter table login add constraint foreign key(mail) references student(mail);**

**Queries & PL/Sql blocks**

**1. List all Students in the University.**

Select reg\_no,name from student;

**2. List all Instructors who teach the course 'Database Systems'.**

select i\_id,name from instructor as i,courses as c where i.i\_id = c.i\_id and c\_name='Database Systems';

**3. Find the number of students in each course.**

select c\_id,count(distinct reg\_no) from takes group by c\_id;

**4. List all students who have given feedback.**

select distinct(reg\_no),name from student as s,feedback as f where s.reg\_no=f.reg\_no;

**5. Create a view of all instructors whose name starts with 'a' and retrieve their contact information.**

create view instr as select \* from instructor where name like 'a%';

select name,contact\_no,mail from instr;

**6. List all female students who take the course 'MeX-001'.**

with temp as(select \* from student natural join courses where sex='F')

select reg\_no,name from temp as tm,takes as ta where tm.c\_id=ta.c\_id and c\_id='MeX-001';

**7. List all students and the courses in which their attendance is below 75%.**

select reg\_no,name,c\_id from student natural join takes where attendance<75;

**8. List the students and the length of their name who are from the Comp. Sci Branch and are female.**

with temp as(select name from student where branch='Comp. Sci.' and sex='F')

select name,length(name) as lenght from student;

**9. List all Course names in Uppercase.**

select c\_id,upper(c\_name) from courses;

**10. List all students who have given feedback and take the course 'CSE-001'.**

with temp as(select name,reg\_no from student natural join takes natural join courses where c\_id='CSE-001')

select reg\_no,name from temp as t,feedback as f where t.reg\_no=f.reg\_no;

**11. Write a PL/SQL block to display the address of a given student**

SET SERVEROUTPUT ON

DECLARE

REGNO student.reg\_no%TYPE;

A student.reg\_no.address%TYPE;

BEGIN

REGNO := '&regno';

SELECT address

INTO A

FROM student

WHERE reg\_no = REGNO;

DBMS\_OUTPUT.PUT\_LINE('THE GPA is: ' || A);

END;

/

**12. Create a trigger to check whether a new student's name is valid**

CREATE OR REPLACE TRIGGER instrInsert

BEFORE INSERT ON instructor

BEGIN

IF LENGTH(TRIM(TRANSLATE(:NEW.name, 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ', ' '))) > 0 THEN

RAISE\_APPLICATION\_ERROR(-20100,'Name must contain only alphabets');

END IF;

END;

/

Insert into student values(210905005,t002,'Jane',18,'F',14-02-2004,3,9000000005,'jane@gmail.com',5.23,'B','Humanities','Yellow Avenue 2');

**13. Create a trigger to add a students details to a new warning table if their attendance is below 75%.**

create table warning(

reg\_no varchar(10),

attendance number(5,2),

PRIMARY KEY (reg\_no));

create or replace trigger warnAttendance

after update of attendance on takes

begin

insert into warning values(:old.reg\_no, :old.c\_id, :new.attendance);

end;

/

**14. Write a function to Retrieve the passwords of a student of given registration number.**

CREATE OR REPLACE function password(reg VARCHAR)

RETURN AS

pass student.passwd%TYPE;

DECLARE

CURSOR C1 IS SELECT reg\_no,passwd FROM student;

BEGIN

for i in C1

if i.reg\_no=reg THEN

pass=i.passwd;

return pass;

END;

/

Set SERVEROUTPUT on

DECLARE

reg student.reg\_no%type;

BEGIN

reg=’&RegistrationNumber’

DBMS\_OUTPUT.PUT\_LINE('Password of ' || reg || ' is ' || password(reg));

END;

/

**15. Write a procedure using a PL/SQL block to display all students with attendance below 75 in each course.**

Set serveroutput on

CREATE OR REPLACE PROCEDURE disp(cid VARCHAR) IS

CURSOR C1 IS SELECT reg\_no,c\_id,attendance FROM takes WHERE c\_id = cid;

BEGIN

DBMS\_OUTPUT.PUT\_LINE(‘Reg\_no c\_id attendance% ');

FOR I IN C1

LOOP

IF (I.attendance<75) THEN

DBMS\_OUTPUT.PUT\_LINE(I.reg\_no || ' ' || I.c\_id||’ ’||I.attendance);

END IF;

END LOOP;

END;

/

DECLARE

CURSOR C IS SELECT distinct(c\_id) from Takes;

BEGIN

For I in C

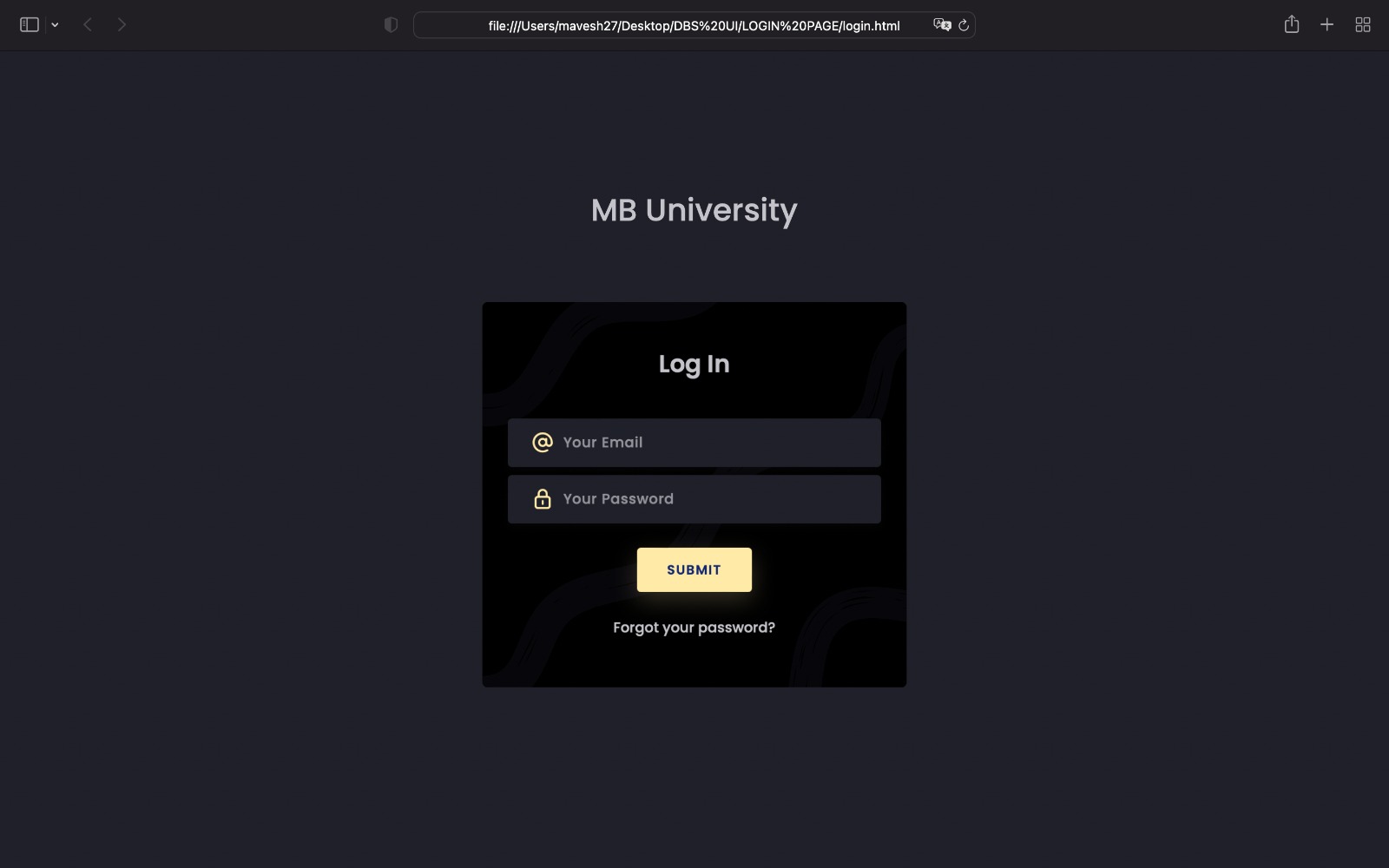
disp(I.c\_id);

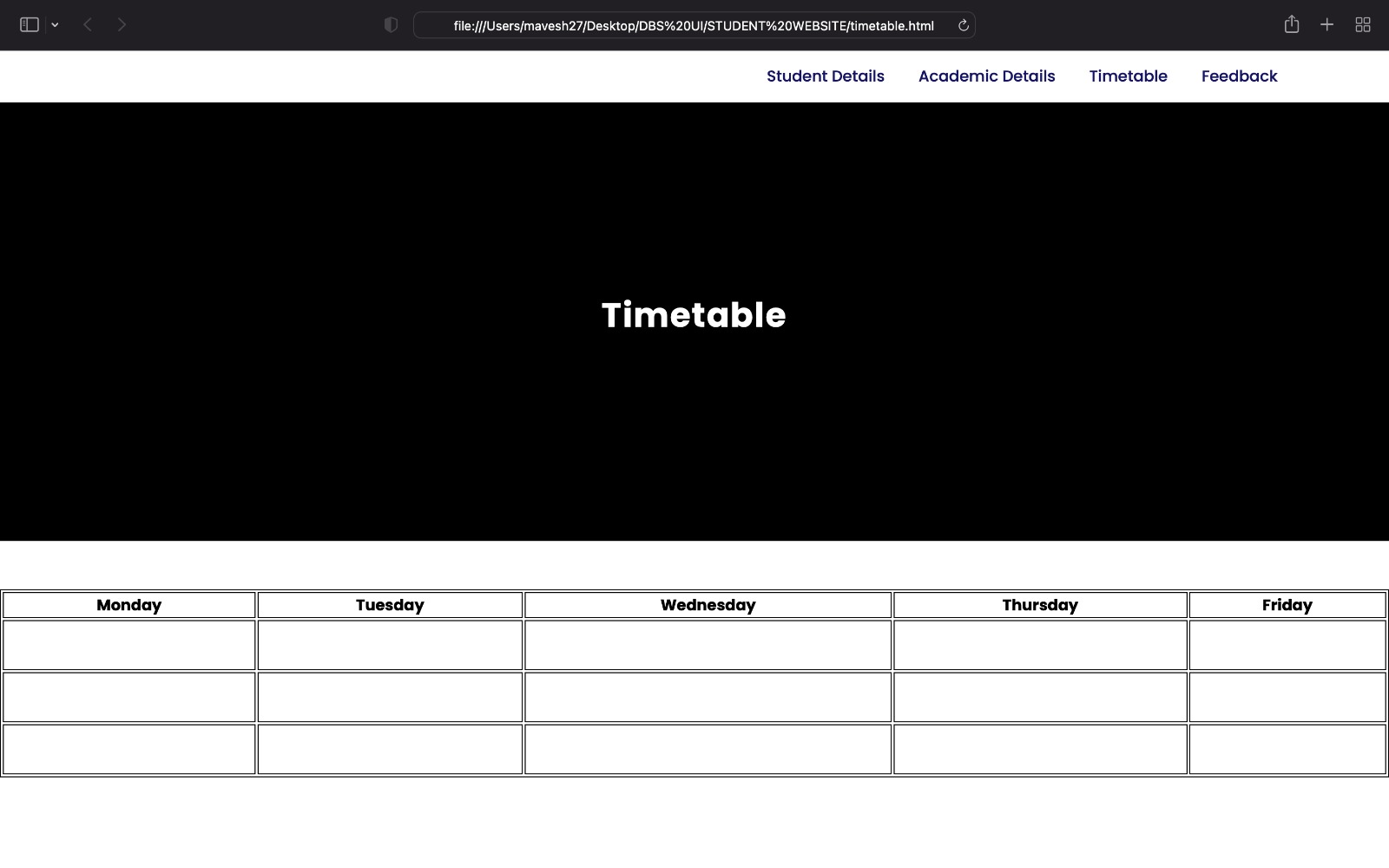
END LOOP;

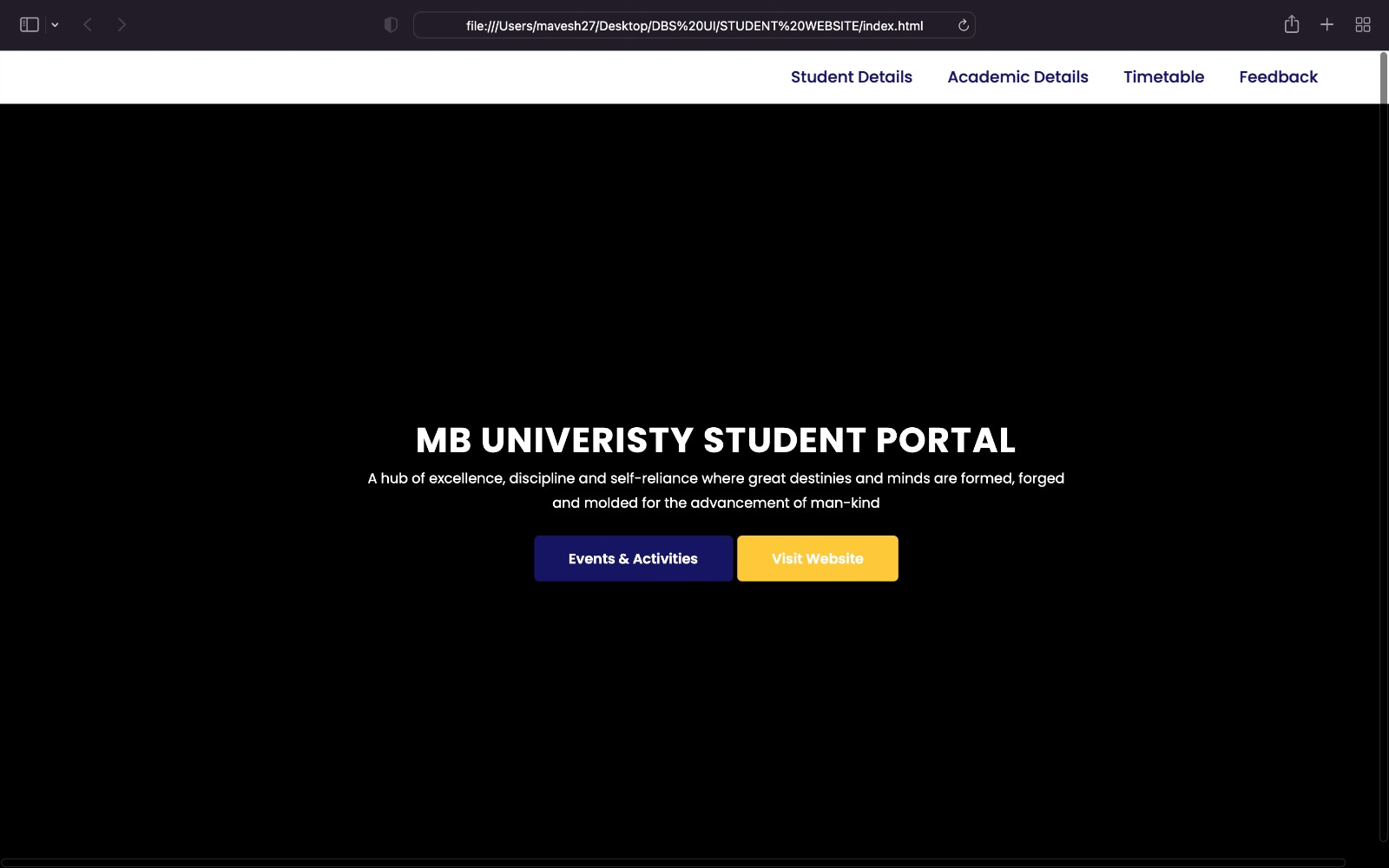
END;

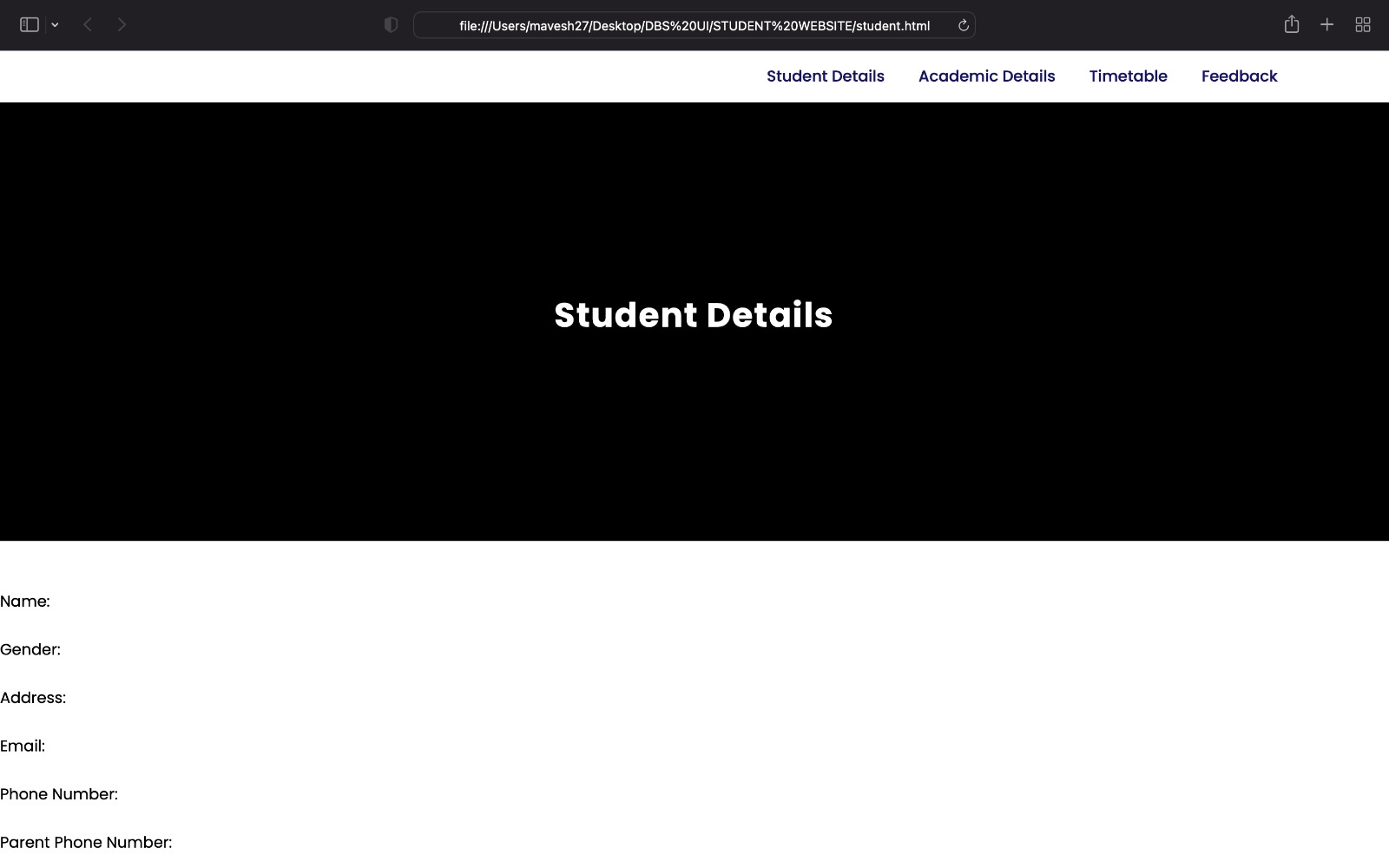
/

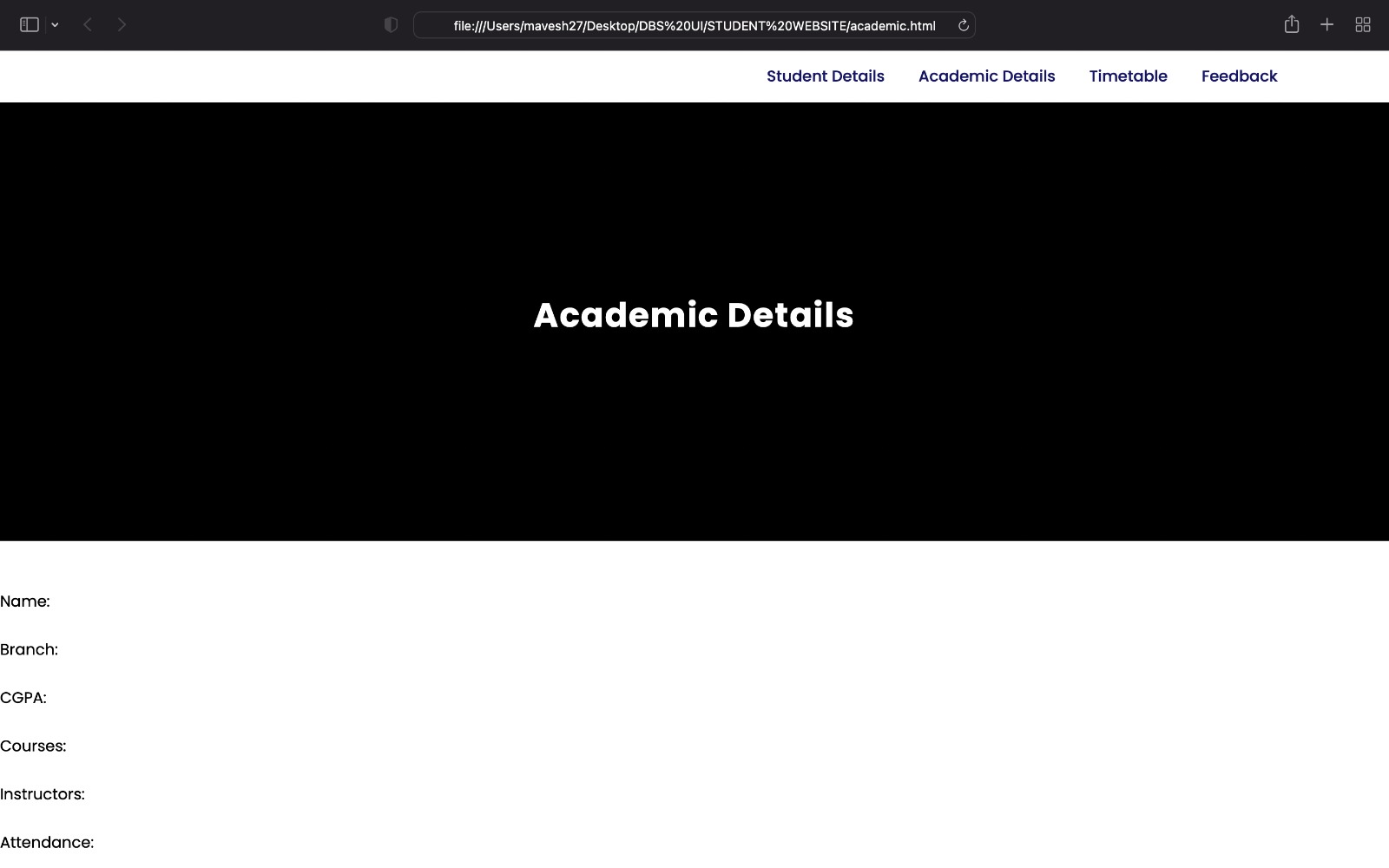
**UI Design**

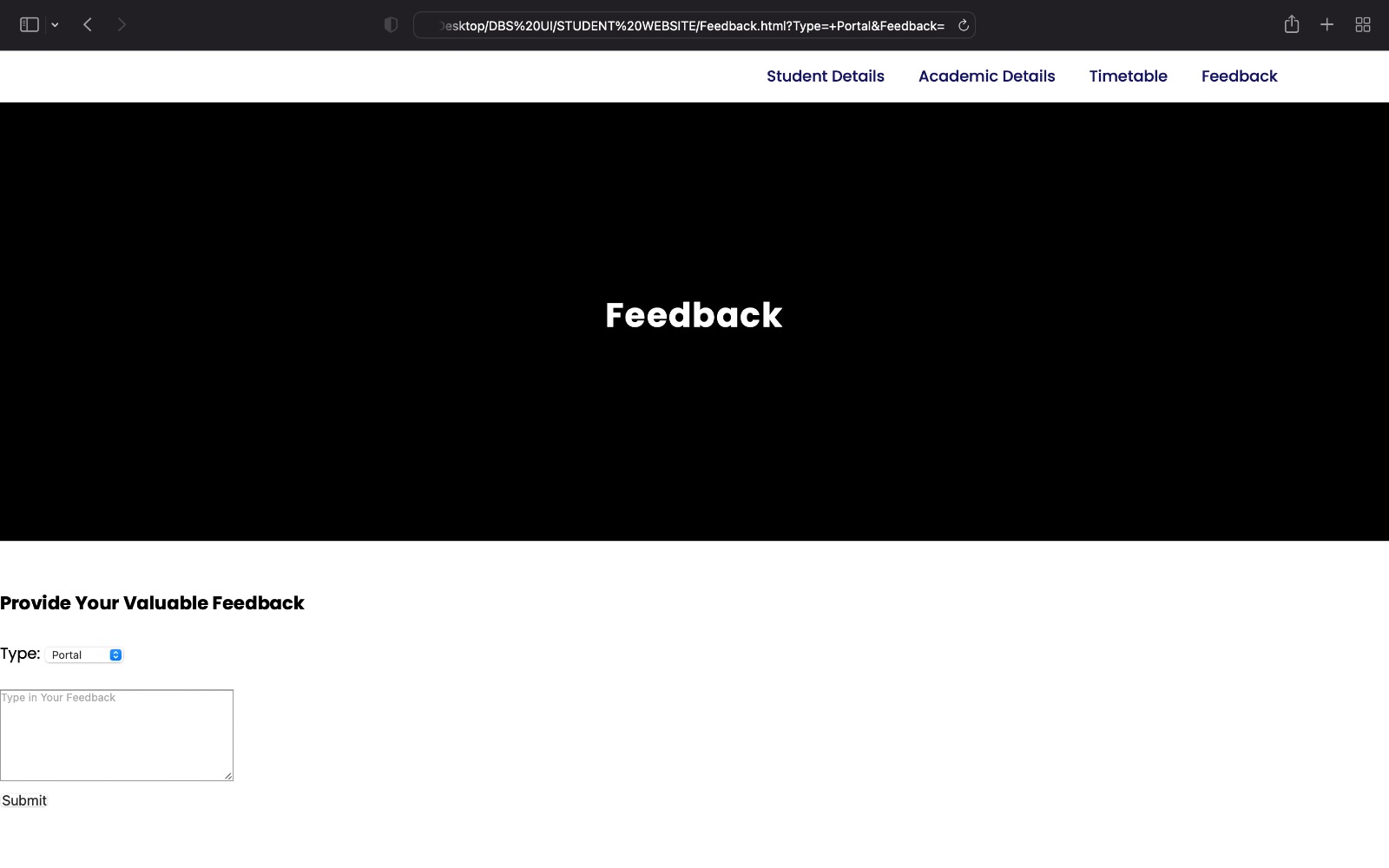
****











**References**

**[1] A. Silberchatz, H. F. Korth, and S. Sudarshan, Database System Concepts, 6th Edition, 2011.**