**College Semester Scheduling System**

**Team Members:**

Stephen Heeps, Jeffrey Paige, Justin Smith

**Problem Statement:**

1. Assist students and academic advisors in classes, majors, minors, concentrations, and organize this data into one central hub and create this as a SaaS for universities.

2. Current and future students enrolled in universities/colleges

**Comparisons**

1. Why is it better than existing technologies

● Many universities lack software to allow students to efficiently view their curriculum, this would provide an easy to access solution that would be able to branch multiple universities to allow students from all over to obtain the same information in one central place.

2. Why would a business buy this solution

● A university that provides open and easy to use methods for students to view and evaluate the classes they’ll need to take during their program would have a competitive advantage over many other universities. The Student is the customer, and ultimately you’re trying to sell them a product. If it is easier for the student to decide what they need, they’ll be more willing to participate.

3. What good things will come from your solution; what bad things won’t happen as a result of your solution.

● Good things will consist of a new piece of software making students and academic advisors lives easier. Bad things will consist of not having an easy system to work with and students and academic advisors being frustrated with class schedules and possible missing classes.

**Project Description:**

1. The solution will consist of three main parts, a web front end to act as an interface between the user and the system, web services that act as an intermediary between the user application and the database, and database that will be used to store and retrieve data. The user will be able to view the current, past, and projected course catalog for a given university, and sort their curriculum by major, minor, class, and professor. The user will also be able to create mockup schedules, and provide publicly visible feedback on professors and classes.

2. Our solution takes an approach that is more geared towards the student being able to get detailed information about the courses required by their program. It will also give students the opportunity to evaluate their experience in classes in a manner that is publicly available to their peers. While some universities may prefer to keep information like that under wraps, it is important in maintaining a healthy level of customer satisfaction between universities and their students. A university that is more able to match the needs of their customers, may perform better in the long run.

3. We will implement our solution via web page with web design and HTML standards. The web page will interact with web services, written in ASP.net+C#, that will retrieve information from a database that will operate using either Microsoft SQL Server or MySQL.

**Technologies you will be working with:**

o Web page

o ASP.net + C#

o MySQL(?)

o IIS

o Mobile devices (later in development when web page is finished and polished)

SQL Code

SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0;

SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0;

SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='TRADITIONAL,ALLOW\_INVALID\_DATES';

CREATE SCHEMA IF NOT EXISTS `mydb` DEFAULT CHARACTER SET utf8 COLLATE utf8\_general\_ci ;

USE `mydb` ;

-- -----------------------------------------------------

-- Table `mydb`.`Major`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`Major` (

`majorID` INT NOT NULL AUTO\_INCREMENT ,

`majorYears` INT NULL ,

`majorDesc` VARCHAR(300) NULL ,

PRIMARY KEY (`majorID`) ,

UNIQUE INDEX `idMajor\_UNIQUE` (`majorID` ASC) )

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`Student`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`Student` (

`idStudent` INT NOT NULL AUTO\_INCREMENT ,

`Major\_idMajor` INT NULL ,

`studentName` VARCHAR(45) NULL ,

`studentGender` VARCHAR(1) NULL ,

`studentAddr` VARCHAR(60) NULL ,

`studentEmail` VARCHAR(30) NULL ,

`studentPhone` VARCHAR(21) NULL ,

PRIMARY KEY (`idStudent`) ,

INDEX `fk\_Student\_Major\_idx` (`Major\_idMajor` ASC) ,

UNIQUE INDEX `idStudent\_UNIQUE` (`idStudent` ASC) ,

CONSTRAINT `fk\_Student\_Major`

FOREIGN KEY (`Major\_idMajor` )

REFERENCES `mydb`.`Major` (`majorID` )

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`Requirement`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`Requirement` (

`requirementID` INT NOT NULL AUTO\_INCREMENT ,

`Major\_idMajor` INT NOT NULL ,

`RequirementDesc` VARCHAR(50) NULL ,

PRIMARY KEY (`requirementID`) ,

INDEX `fk\_Requirement\_Major1\_idx` (`Major\_idMajor` ASC) ,

UNIQUE INDEX `idRequirement\_UNIQUE` (`requirementID` ASC) ,

CONSTRAINT `fk\_Requirement\_Major1`

FOREIGN KEY (`Major\_idMajor` )

REFERENCES `mydb`.`Major` (`majorID` )

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`Course`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`Course` (

`courseID` INT NOT NULL AUTO\_INCREMENT ,

`courseDesc` VARCHAR(300) NULL ,

PRIMARY KEY (`courseID`) ,

UNIQUE INDEX `idCourse\_UNIQUE` (`courseID` ASC) )

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`RequiredCourse`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`RequiredCourse` (

`Requirement\_idRequirement` INT NOT NULL ,

`Course\_idCourse` INT NOT NULL ,

PRIMARY KEY (`Requirement\_idRequirement`, `Course\_idCourse`) ,

INDEX `fk\_RequiredCourse\_Course1\_idx` (`Course\_idCourse` ASC) ,

CONSTRAINT `fk\_RequiredCourse\_Requirement1`

FOREIGN KEY (`Requirement\_idRequirement` )

REFERENCES `mydb`.`Requirement` (`requirementID` )

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_RequiredCourse\_Course1`

FOREIGN KEY (`Course\_idCourse` )

REFERENCES `mydb`.`Course` (`courseID` )

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`Semester`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`Semester` (

`semesterID` INT NOT NULL AUTO\_INCREMENT ,

`Student\_idStudent` INT NOT NULL ,

`semesterYear` YEAR NULL ,

`semesterNote` VARCHAR(100) NULL ,

PRIMARY KEY (`semesterID`, `Student\_idStudent`) ,

INDEX `fk\_Semester\_Student1\_idx` (`Student\_idStudent` ASC) ,

UNIQUE INDEX `idSemester\_UNIQUE` (`semesterID` ASC) ,

CONSTRAINT `fk\_Semester\_Student1`

FOREIGN KEY (`Student\_idStudent` )

REFERENCES `mydb`.`Student` (`idStudent` )

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`Professor`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`Professor` (

`professorID` INT NOT NULL ,

`professorName` VARCHAR(45) NULL ,

`professorGender` VARCHAR(1) NULL ,

`professorAddr` VARCHAR(60) NULL ,

`professorEmail` VARCHAR(30) NULL ,

`professorPhone` VARCHAR(21) NULL ,

`professorMajor` VARCHAR(20) NULL ,

PRIMARY KEY (`professorID`) )

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`Section`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`Section` (

`sectionID` INT NOT NULL ,

`Course\_idCourse` INT NOT NULL ,

`Professor\_idProfessor` INT NULL ,

`sectionSemester` VARCHAR(15) NULL ,

`sectionBooks` VARCHAR(45) NULL ,

PRIMARY KEY (`sectionID`) ,

INDEX `fk\_Section\_Professor1\_idx` (`Professor\_idProfessor` ASC) ,

INDEX `fk\_Section\_Course1\_idx` (`Course\_idCourse` ASC) ,

CONSTRAINT `fk\_Section\_Professor1`

FOREIGN KEY (`Professor\_idProfessor` )

REFERENCES `mydb`.`Professor` (`professorID` )

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_Section\_Course1`

FOREIGN KEY (`Course\_idCourse` )

REFERENCES `mydb`.`Course` (`courseID` )

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`InCourse`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`InCourse` (

`Student\_idStudent` INT NOT NULL ,

`Section\_idSection` INT NOT NULL ,

PRIMARY KEY (`Student\_idStudent`, `Section\_idSection`) ,

INDEX `fk\_InCourse\_Section1\_idx` (`Section\_idSection` ASC) ,

CONSTRAINT `fk\_InCourse\_Student1`

FOREIGN KEY (`Student\_idStudent` )

REFERENCES `mydb`.`Student` (`idStudent` )

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_InCourse\_Section1`

FOREIGN KEY (`Section\_idSection` )

REFERENCES `mydb`.`Section` (`sectionID` )

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`SectionEval`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`SectionEval` (

`idSectionEval` INT NOT NULL ,

`Section\_idSection` INT NOT NULL ,

PRIMARY KEY (`idSectionEval`) ,

INDEX `fk\_SectionEval\_Section1\_idx` (`Section\_idSection` ASC) ,

CONSTRAINT `fk\_SectionEval\_Section1`

FOREIGN KEY (`Section\_idSection` )

REFERENCES `mydb`.`Section` (`sectionID` )

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`Planned`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`Planned` (

`Semester\_idSemester` INT NOT NULL ,

`Section\_idSection` INT NOT NULL ,

`Semester\_Student\_idStudent` INT NOT NULL ,

PRIMARY KEY (`Semester\_idSemester`, `Section\_idSection`) ,

INDEX `fk\_Planned\_Semester1\_idx` (`Semester\_idSemester` ASC, `Semester\_Student\_idStudent` ASC) ,

INDEX `fk\_Planned\_Section1\_idx` (`Section\_idSection` ASC) ,

CONSTRAINT `fk\_Planned\_Semester1`

FOREIGN KEY (`Semester\_idSemester` , `Semester\_Student\_idStudent` )

REFERENCES `mydb`.`Semester` (`semesterID` , `Student\_idStudent` )

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_Planned\_Section1`

FOREIGN KEY (`Section\_idSection` )

REFERENCES `mydb`.`Section` (`sectionID` )

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

USE `mydb` ;

SET SQL\_MODE=@OLD\_SQL\_MODE;

SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS;

SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS;

HTML

<!DOCTYPE html>

<html>

<head>

<link rel="stylesheet" type="text/css" href="style.css">

</head>

<title>

University Catalogue Explorer

</title>

<body>

<ul>

<li><a href="index.html">Home</a></li>

<li><a href="ppt.html">Our presentation</a></li>

<li><a href="submitcatalogue.html">Submit Your Catalogue</a></li>

<li><a href="catalogue.html">View our current catalogues</a></li>

<li><a href="contactus.html">Contact Us</a></li>

<li><a href="projectproposal.html">Our project proposal</a></li>

<li><a href="aboutus.html">About Us</a></li>

</ul>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<link rel="stylesheet" type="text/css" href="style.css">

</head>

<title>

About Us

</title>

<body>

<ul>

<li><a href="index.html">Home</a></li>

<li><a href="submitcatalogue.html">Submit Your Catalogue</a></li>

<li><a href="catalogue.html">View our current catalogues</a></li>

<li><a href="contactus.html">Contact Us</a></li>

<li><a href="projectproposal.html">Our project proposal</a></li>

<li><a href="aboutus.html">About Us</a></li>

</ul>

<h1>

Stephen Heeps: Web designer / Programmer <br/>

Justin Smith: DBA<br/>

Jeffrey Paige: Systems Analyst<br/>

</h1>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<link rel="stylesheet" type="text/css" href="style.css">

</head>

<title>

Project Proposal

</title>

<body id = "top">

<ul>

<li><a href="index.html">Home</a></li>

<li><a href="submitcatalogue.html">Submit Your Catalogue</a></li>

<li><a href="catalogue.html">View our current catalogues</a></li>

<li><a href="contactus.html">Contact Us</a></li>

<li><a href="projectproposal.html">Our project proposal</a></li>

<li><a href="aboutus.html">About Us</a></li>

</ul>

<center>

Navigation for page

</center>

<ul class = "pp">

<li><a href="#intro"> Intro </a></li>

<li><a href="#team members"> Team Members</a></li>

<li><a href="#ps">Problem Statement</a></li>

<li><a href="#pd">Project Description</a></li>

</ul>

<br/><br/>

<center>

<p>

Project Title: University Catalogue Explorer

</p>

</center>

<br/>

<p id = "team members">

Team Members:

Stephen Heeps (Web designer)<br/>

Jeffrey Paige: Systems Analyst<br/>

Justin Smith: DBA<br/>

</p>

<p class="intro" id="intro">

We feel as though this project will both benefit our target audience as well as our experience and customer relationship. While it gives them an extremely useful tool to project schedules and possible graduation dates, it also helps us with experience in the industry.

</p>

<br/><br/>

<p class = "ps" id = "ps">

Problem Statement:

Assist students and academic advisers in classes, majors, minors, concentrations, and organize this data into one central hub and create this as a SaaS for universities.

Target Audience: Current and future students enrolled in universities/colleges

Why is this an important problem

Why is it better than existing technologies

Many universities lack software to allow students to efficiently view their curriculum, this would provide an easy to access solution that would be able to branch multiple universities to allow students from all over to obtain the same information in one central place.

Why would a business buy this solution

A university that provides open and easy to use methods for students to view and evaluate the classes they&#8217;ll need to take during their program would have a competitive advantage over many other universities. The Student is the customer, and ultimately you&#8217;re trying to sell them a product. If it is easier for the student to decide what they need, they&#8217;ll be more willing to participate.

What good things will come from your solution; what bad things won&#8217;t happen as a result of your solution.

Good things will consist of a new piece of software making students and academic advisers lives easier. Bad things will consist of not having an easy system to work with and students and academic advisers being frustrated with class schedules and possible missing classes.

<br/><br/>

</p>

<p class = "pd" id = "pd">

Project Description:

The solution will consist of three main parts, a web front end to act as an interface between the user and the system, web services that act as an intermediary between the user application and the database, and database that will be used to store and retrieve data. The user will be able to view the current, past, and projected course catalogue for a given university, and sort their curriculum by major, minor, class, and professor. The user will also be able to create mock-up schedules, and provide publicly visible feedback on professors and classes.

Our solution takes an approach that is more geared towards the student being able to get detailed information about the courses required by their program. It will also give students the opportunity to evaluate their experience in classes in a manner that is publicly available to their peers. While some universities may prefer to keep information like that under wraps, it is important in maintaining a healthy level of customer satisfaction between universities and their students. A university that is more able to match the needs of their customers, may perform better in the long run.

We will implement our solution via web page with web design and HTML standards. The web page will interact with web services, written in ASP.net + C#, that will retrieve information from a database that will operate using either Microsoft SQL Server or MySQL.

Technologies you want to work with:

At the beginning of development, yes we will be limited to technology. As development furthers and iterations are met, more technology can be utilized I.E. mobile applications, software, etc.

What are you technologies of choice?

Web page

ASP.net + C#

SQL Server or MySQL(?)

IIS

Mobile devices (later in development when web page is finished and polished)

<br/><br/>

</p>

<center>Client (if you have one):

N/A: This is currently an academic endeavour. </center>

<br/><br/>

<center><a href="#top">Top of page</a></center>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<link rel="stylesheet" type="text/css" href="style.css">

</head>

<title>

Catalogue

</title>

<body>

<ul>

<li><a href="index.html">Home</a></li>

<li><a href="ppt.html">Our presentation</a></li>

<li><a href="submitcatalogue.html">Submit Your Catalogue</a></li>

<li><a href="catalogue.html">View our current catalogues</a></li>

<li><a href="contactus.html">Contact Us</a></li>

<li><a href="projectproposal.html">Our project proposal</a></li>

<li><a href="aboutus.html">About Us</a></li>

</ul>

<center>

<p>

Site under construction. Please come back later!

</p>

</center>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<link rel="stylesheet" type="text/css" href="style.css">

</head>

<title>

Contact Us

</title>

<body>

<ul>

<li><a href="index.html">Home</a></li>

<li><a href="ppt.html">Our presentation</a></li>

<li><a href="submitcatalogue.html">Submit Your Catalogue</a></li>

<li><a href="catalogue.html">View our current catalogues</a></li>

<li><a href="contactus.html">Contact Us</a></li>

<li><a href="projectproposal.html">Our project proposal</a></li>

<li><a href="aboutus.html">About Us</a></li>

</ul>

<br />

<br />

<center> If you have any questions or comments, please e-mail us <a href="mailto:stephen.heeps@snhu.edu"> here!</a>

<br/>

<br/>

Comments: <br/>

<form name = "text-area" action="commentsubmit.html" action="mailto:stephen.heeps@snhu.edu">

<textarea rows = "10" cols = "50" input type = "text" name = textarea">

</textarea><br/>

<input type = "submit" value = "Submit">

</form>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<link rel="stylesheet" type="text/css" href="style.css">

</head>

<title>

Submit Your Catalogue

</title>

<body>

<ul>

<li><a href="index.html">Home</a></li>

<li><a href="ppt.html">Our presentation</a></li>

<li><a href="submitcatalogue.html">Submit Your Catalogue</a></li>

<li><a href="catalogue.html">View our current catalogues</a></li>

<li><a href="contactus.html">Contact Us</a></li>

<li><a href="projectproposal.html">Our project proposal</a></li>

<li><a href="aboutus.html">About Us</a></li>

</ul>

<h1>What are you trying to submit?</h1>

<a href="submitug.html">Undergraduate</a><br/>

<a href="submitg.html">Graduate</a><br/>

<a href="submitc.html">COCE</a><br/>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<link rel="stylesheet" type="text/css" href="style.css">

</head>

<title>

Comment Submitted

</title>

<body>

<ul>

<li><a href="index.html">Home</a></li>

<li><a href="ppt.html">Our presentation</a></li>

<li><a href="submitcatalogue.html">Submit Your Catalogue</a></li>

<li><a href="catalogue.html">View our current catalogues</a></li>

<li><a href="contactus.html">Contact Us</a></li>

<li><a href="projectproposal.html">Our project proposal</a></li>

<li><a href="aboutus.html">About Us</a></li>

</ul>

<center> Comment submitted! Thank you for your input! :-) </center>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<link rel="stylesheet" type="text/css" href="style.css">

</head>

<title>

University Catalogue Explorer

</title>

<body>

<ul>

<li><a href="index.html">Home</a></li>

<li><a href="ppt.html">Our presentation</a></li>

<li><a href="submitcatalogue.html">Submit Your Catalogue</a></li>

<li><a href="catalogue.html">View our current catalogues</a></li>

<li><a href="contactus.html">Contact Us</a></li>

<li><a href="projectproposal.html">Our project proposal</a></li>

<li><a href="aboutus.html">About Us</a></li>

</ul>

<center>

<p>

Site under construction. Please come back later!

</p>

</center>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<link rel="stylesheet" type="text/css" href="style.css">

</head>

<title>

University Catalogue Explorer

</title>

<body>

<ul>

<li><a href="index.html">Home</a></li>

<li><a href="ppt.html">Our presentation</a></li>

<li><a href="submitcatalogue.html">Submit Your Catalogue</a></li>

<li><a href="catalogue.html">View our current catalogues</a></li>

<li><a href="contactus.html">Contact Us</a></li>

<li><a href="projectproposal.html">Our project proposal</a></li>

<li><a href="aboutus.html">About Us</a></li>

</ul>

<center>

<p>

Site under construction. Please come back later!

</p>

</center>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<link rel="stylesheet" type="text/css" href="style.css">

</head>

<title>

University Catalogue Explorer

</title>

<body>

<ul>

<li><a href="index.html">Home</a></li>

<li><a href="ppt.html">Our presentation</a></li>

<li><a href="submitcatalogue.html">Submit Your Catalogue</a></li>

<li><a href="catalogue.html">View our current catalogues</a></li>

<li><a href="contactus.html">Contact Us</a></li>

<li><a href="projectproposal.html">Our project proposal</a></li>

<li><a href="aboutus.html">About Us</a></li>

</ul>

<center>

<iframe src="https://docs.google.com/presentation/d/1EGsmi9QJaKO7Qd9nmcovmxF4K8FadMGICNH7vr1bReY/present#slide=id.p" style="width:1200px; height:600px;" frameborder="0"></iframe>

</center>

</body>

</html>

CSS

a:link {color:#FF0000;} /\*unvisited\*/

a:visited {color:#FFFFFF;} /\*visited\*/

a:hover {color:#3399FF;} /\*mouseover\*/

a:active {color:#1919D1;} /\*selected\*/

body

{

background-image:url("images/background.jpg");

background-repeat: no-repeat;

background-size: cover;

color:#FFFFFF;

font-family:"Century Gothic";

/\* cursor:wait; \*/

}

ul

{

width: 570px;

padding:15px;

margin: 0px auto 0px auto;

border-top: 2px solid #FFFFFF;

border-bottom: 1px solid #FFFFFF;

text-align: center;

}

ul.pp

{

border-top:#000000;

}

li

{

display: inline;

margin: 0px 3px;

}

p

{

width: 500px;

border: 2px solid #0088dd;

padding: 10px;

font-size: 20px;

text-align: center;

}

p.intro

{

margin-left:600px;

}

p.ps

{

margin-left: 750px;

width:1000px;

}

p.pd

{

width:1150px;

}

PHP

<?php>

// Establish the connection

$con = mysqli\_connect ("50.163.101.38:3306","root","w6IEAPQY4KaMl13R5csA","mydb"); //host = host name or IP address.

//Check connection

if (mysqli\_connect\_errno())

{

echo "Failed to connect to MySQL: " . mysqli\_connect\_error();

}

$result = mysqli\_query($con,"SELECT \* FROM ");

echo "<table border='1'>

<tr>

<th>University</th>

<th>Types of Degrees</th>

<th>Types of Majors</th>

<th>Types of Minors</th>

<th>Types of Concentrations</th>

</tr>"

while($row = mysqli\_fetch\_array($result))

{

echo "<tr>";

echo "<td>" . $row[''] . "</td>"; //Here is where we'll get data for University. Create links

echo "<td>" . $row[''] . "</td>"; //Here is where we'll get data for types of degress. Create links

echo "<td>" . $row[''] . "</td>"; //Here is where we'll get data for majors. Create links

echo "<td>" . $row[''] . "</td>"; //Here is where we'll get data for minors. Create links

echo "<td>" . $row[''] . "</td>"; //Here is where we'll get data for concentrations. Create links

echo "</tr>";

}

echo "</table>";

mysqli\_close($con);

?>

**C# Login Web Service**

This is a prototype web service to use for storing and checking passwords against a database. It salts and hashes passwords before storing them, and checks the password against the hash.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.Serialization;

using System.ServiceModel;

using System.ServiceModel.Web;

using System.Text;

using System.Data;

using System.Security.Cryptography;

using MySql.Data.MySqlClient;

namespace LoginService

{

// NOTE: You can use the "Rename" command on the "Refactor" menu to change the class name "Service1" in code, svc and config file together.

// NOTE: In order to launch WCF Test Client for testing this service, please select Service1.svc or Service1.svc.cs at the Solution Explorer and start debugging.

public class Service1 : IService1

{

public bool registerAccount(string email, string password)

{

MySqlConnection connection = new MySqlConnection();

//Create connection string, using test info. Realistically you would read this from a config file or something

connection.ConnectionString = "Server=127.0.0.1;" + "Database=mydb;" + "UID=root;" + "Password=w6IEAPQY4KaMl13R5csA;";

//Hash and salt password

PasswordHash hashedPassword = new PasswordHash(password);

byte[] hashedBytes = hashedPassword.ToArray();

string hashedPasswordString = Convert.ToBase64String(hashedBytes);

//Open our connection

connection.Open();

//Create a MySQLCommand in order to execute a query

MySqlCommand command = connection.CreateCommand();

//Add our parameters

command.Parameters.AddWithValue("email", email);

command.Parameters.AddWithValue("password", password);

//Set the query to be executed

command.CommandText = "INSERT INTO student (studentEmail, studentPassword) VALUES ('" + email + "','" + hashedPasswordString + "');";

//execute the query

command.ExecuteNonQuery();

//close the query

connection.Close();

//send true back to the client to say it successfully executed

return true;

}

public bool verifyUser(string email, string password)

{

//Create a MySQL Connection

MySqlConnection connection = new MySqlConnection();

//Create connection string, using test info. Realistically you would read this from a config file or something

connection.ConnectionString = "Server=127.0.0.1;" + "Database=mydb;" + "UID=root;" + "Password=test;";

//open the connection

connection.Open();

//Create a MySQLCommand in order to execute a query

MySqlCommand command = connection.CreateCommand();

//Set the query to be executed

command.CommandText = "SELECT studentEmail, studentPassword FROM student WHERE studentEmail = '" + email + "';";

//Read the results

MySqlDataReader result = command.ExecuteReader();

//If there are no results send back false

//this occurs if there are no entries with the email

if(result.Read() == false)

{

connection.Close();

return false;

}

//read the fields from the result

string retrievedEmail = (string)result.GetValue(0);

string retrievedPassword = (string)result.GetValue(1);

//Since the hash is stored as a base64 string, we need to convert back to a byte array

byte[] hashedBytes = Convert.FromBase64String(retrievedPassword);

PasswordHash hashedPassword = new PasswordHash(hashedBytes);

//Check to see if the hashed password matches the input

if(hashedPassword.Verify(password))

{

connection.Close();

return true;

}

else

{

connection.Close();

return false;

}

}

}

//Using code found here (http://csharptest.net/470/another-example-of-how-to-store-a-salted-password-hash/) to generate and verify salted and hashed passwords

public sealed class PasswordHash

{

const int SaltSize = 16, HashSize = 20, HashIter = 10000;

readonly byte[] \_salt, \_hash;

public PasswordHash(string password)

{

new RNGCryptoServiceProvider().GetBytes(\_salt = new byte[SaltSize]);

\_hash = new Rfc2898DeriveBytes(password, \_salt, HashIter).GetBytes(HashSize);

}

public PasswordHash(byte[] hashBytes)

{

Array.Copy(hashBytes, 0, \_salt = new byte[SaltSize], 0, SaltSize);

Array.Copy(hashBytes, SaltSize, \_hash = new byte[HashSize], 0, HashSize);

}

public PasswordHash(byte[] salt, byte[] hash)

{

Array.Copy(salt, 0, \_salt = new byte[SaltSize], 0, SaltSize);

Array.Copy(hash, 0, \_hash = new byte[HashSize], 0, HashSize);

}

public byte[] ToArray()

{

byte[] hashBytes = new byte[SaltSize + HashSize];

Array.Copy(\_salt, 0, hashBytes, 0, SaltSize);

Array.Copy(\_hash, 0, hashBytes, SaltSize, HashSize);

return hashBytes;

}

public byte[] Salt { get { return (byte[])\_salt.Clone(); } }

public byte[] Hash { get { return (byte[])\_hash.Clone(); } }

public bool Verify(string password)

{

byte[] test = new Rfc2898DeriveBytes(password, \_salt, HashIter).GetBytes(HashSize);

for (int i = 0; i < HashSize; i++)

if (test[i] != \_hash[i])

return false;

return true;

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.Serialization;

using System.ServiceModel;

using System.ServiceModel.Web;

using System.Text;

namespace LoginService

{

// NOTE: You can use the "Rename" command on the "Refactor" menu to change the interface name "IService1" in both code and config file together.

[ServiceContract]

public interface IService1

{

[OperationContract]

bool registerAccount(string email, string password);

[OperationContract]

bool verifyUser(string email, string password);

}

}

SQL Code

SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0;

SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0;

SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='TRADITIONAL,ALLOW\_INVALID\_DATES';

CREATE SCHEMA IF NOT EXISTS `mydb` DEFAULT CHARACTER SET utf8 COLLATE utf8\_general\_ci ;

USE `mydb` ;

-- -----------------------------------------------------

-- Table `mydb`.`Major`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`Major` (

`majorID` INT NOT NULL AUTO\_INCREMENT ,

`majorYears` INT NULL ,

`majorDesc` VARCHAR(300) NULL ,

PRIMARY KEY (`majorID`) ,

UNIQUE INDEX `idMajor\_UNIQUE` (`majorID` ASC) )

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`Student`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`Student` (

`idStudent` INT NOT NULL AUTO\_INCREMENT ,

`Major\_idMajor` INT NULL ,

`studentName` VARCHAR(45) NULL ,

`studentGender` VARCHAR(1) NULL ,

`studentAddr` VARCHAR(60) NULL ,

`studentEmail` VARCHAR(30) NULL ,

`studentPhone` VARCHAR(21) NULL ,

PRIMARY KEY (`idStudent`) ,

INDEX `fk\_Student\_Major\_idx` (`Major\_idMajor` ASC) ,

UNIQUE INDEX `idStudent\_UNIQUE` (`idStudent` ASC) ,

CONSTRAINT `fk\_Student\_Major`

FOREIGN KEY (`Major\_idMajor` )

REFERENCES `mydb`.`Major` (`majorID` )

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`Requirement`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`Requirement` (

`requirementID` INT NOT NULL AUTO\_INCREMENT ,

`Major\_idMajor` INT NOT NULL ,

`RequirementDesc` VARCHAR(50) NULL ,

PRIMARY KEY (`requirementID`) ,

INDEX `fk\_Requirement\_Major1\_idx` (`Major\_idMajor` ASC) ,

UNIQUE INDEX `idRequirement\_UNIQUE` (`requirementID` ASC) ,

CONSTRAINT `fk\_Requirement\_Major1`

FOREIGN KEY (`Major\_idMajor` )

REFERENCES `mydb`.`Major` (`majorID` )

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`Course`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`Course` (

`courseID` INT NOT NULL AUTO\_INCREMENT ,

`courseDesc` VARCHAR(300) NULL ,

PRIMARY KEY (`courseID`) ,

UNIQUE INDEX `idCourse\_UNIQUE` (`courseID` ASC) )

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`RequiredCourse`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`RequiredCourse` (

`Requirement\_idRequirement` INT NOT NULL ,

`Course\_idCourse` INT NOT NULL ,

PRIMARY KEY (`Requirement\_idRequirement`, `Course\_idCourse`) ,

INDEX `fk\_RequiredCourse\_Course1\_idx` (`Course\_idCourse` ASC) ,

CONSTRAINT `fk\_RequiredCourse\_Requirement1`

FOREIGN KEY (`Requirement\_idRequirement` )

REFERENCES `mydb`.`Requirement` (`requirementID` )

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_RequiredCourse\_Course1`

FOREIGN KEY (`Course\_idCourse` )

REFERENCES `mydb`.`Course` (`courseID` )

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`Semester`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`Semester` (

`semesterID` INT NOT NULL AUTO\_INCREMENT ,

`Student\_idStudent` INT NOT NULL ,

`semesterYear` YEAR NULL ,

`semesterNote` VARCHAR(100) NULL ,

PRIMARY KEY (`semesterID`, `Student\_idStudent`) ,

INDEX `fk\_Semester\_Student1\_idx` (`Student\_idStudent` ASC) ,

UNIQUE INDEX `idSemester\_UNIQUE` (`semesterID` ASC) ,

CONSTRAINT `fk\_Semester\_Student1`

FOREIGN KEY (`Student\_idStudent` )

REFERENCES `mydb`.`Student` (`idStudent` )

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`Professor`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`Professor` (

`professorID` INT NOT NULL ,

`professorName` VARCHAR(45) NULL ,

`professorGender` VARCHAR(1) NULL ,

`professorAddr` VARCHAR(60) NULL ,

`professorEmail` VARCHAR(30) NULL ,

`professorPhone` VARCHAR(21) NULL ,

`professorMajor` VARCHAR(20) NULL ,

PRIMARY KEY (`professorID`) )

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`Section`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`Section` (

`sectionID` INT NOT NULL ,

`Course\_idCourse` INT NOT NULL ,

`Professor\_idProfessor` INT NULL ,

`sectionSemester` VARCHAR(15) NULL ,

`sectionBooks` VARCHAR(45) NULL ,

PRIMARY KEY (`sectionID`) ,

INDEX `fk\_Section\_Professor1\_idx` (`Professor\_idProfessor` ASC) ,

INDEX `fk\_Section\_Course1\_idx` (`Course\_idCourse` ASC) ,

CONSTRAINT `fk\_Section\_Professor1`

FOREIGN KEY (`Professor\_idProfessor` )

REFERENCES `mydb`.`Professor` (`professorID` )

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_Section\_Course1`

FOREIGN KEY (`Course\_idCourse` )

REFERENCES `mydb`.`Course` (`courseID` )

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`InCourse`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`InCourse` (

`Student\_idStudent` INT NOT NULL ,

`Section\_idSection` INT NOT NULL ,

PRIMARY KEY (`Student\_idStudent`, `Section\_idSection`) ,

INDEX `fk\_InCourse\_Section1\_idx` (`Section\_idSection` ASC) ,

CONSTRAINT `fk\_InCourse\_Student1`

FOREIGN KEY (`Student\_idStudent` )

REFERENCES `mydb`.`Student` (`idStudent` )

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_InCourse\_Section1`

FOREIGN KEY (`Section\_idSection` )

REFERENCES `mydb`.`Section` (`sectionID` )

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`SectionEval`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`SectionEval` (

`idSectionEval` INT NOT NULL ,

`Section\_idSection` INT NOT NULL ,

PRIMARY KEY (`idSectionEval`) ,

INDEX `fk\_SectionEval\_Section1\_idx` (`Section\_idSection` ASC) ,

CONSTRAINT `fk\_SectionEval\_Section1`

FOREIGN KEY (`Section\_idSection` )

REFERENCES `mydb`.`Section` (`sectionID` )

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`Planned`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`Planned` (

`Semester\_idSemester` INT NOT NULL ,

`Section\_idSection` INT NOT NULL ,

`Semester\_Student\_idStudent` INT NOT NULL ,

PRIMARY KEY (`Semester\_idSemester`, `Section\_idSection`) ,

INDEX `fk\_Planned\_Semester1\_idx` (`Semester\_idSemester` ASC, `Semester\_Student\_idStudent` ASC) ,

INDEX `fk\_Planned\_Section1\_idx` (`Section\_idSection` ASC) ,

CONSTRAINT `fk\_Planned\_Semester1`

FOREIGN KEY (`Semester\_idSemester` , `Semester\_Student\_idStudent` )

REFERENCES `mydb`.`Semester` (`semesterID` , `Student\_idStudent` )

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_Planned\_Section1`

FOREIGN KEY (`Section\_idSection` )

REFERENCES `mydb`.`Section` (`sectionID` )

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

USE `mydb` ;

SET SQL\_MODE=@OLD\_SQL\_MODE;

SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS;

SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS;

HTML

<!DOCTYPE html>

<html>

<head>

<link rel="stylesheet" type="text/css" href="style.css">

</head>

<title>

University Catalogue Explorer

</title>

<body>

<ul>

<li><a href="index.html">Home</a></li>

<li><a href="ppt.html">Our presentation</a></li>

<li><a href="submitcatalogue.html">Submit Your Catalogue</a></li>

<li><a href="catalogue.html">View our current catalogues</a></li>

<li><a href="contactus.html">Contact Us</a></li>

<li><a href="projectproposal.html">Our project proposal</a></li>

<li><a href="aboutus.html">About Us</a></li>

</ul>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<link rel="stylesheet" type="text/css" href="style.css">

</head>

<title>

About Us

</title>

<body>

<ul>

<li><a href="index.html">Home</a></li>

<li><a href="submitcatalogue.html">Submit Your Catalogue</a></li>

<li><a href="catalogue.html">View our current catalogues</a></li>

<li><a href="contactus.html">Contact Us</a></li>

<li><a href="projectproposal.html">Our project proposal</a></li>

<li><a href="aboutus.html">About Us</a></li>

</ul>

<h1>

Stephen Heeps: Web designer / Programmer <br/>

Justin Smith: DBA<br/>

Jeffrey Paige: Systems Analyst<br/>

</h1>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<link rel="stylesheet" type="text/css" href="style.css">

</head>

<title>

Project Proposal

</title>

<body id = "top">

<ul>

<li><a href="index.html">Home</a></li>

<li><a href="submitcatalogue.html">Submit Your Catalogue</a></li>

<li><a href="catalogue.html">View our current catalogues</a></li>

<li><a href="contactus.html">Contact Us</a></li>

<li><a href="projectproposal.html">Our project proposal</a></li>

<li><a href="aboutus.html">About Us</a></li>

</ul>

<center>

Navigation for page

</center>

<ul class = "pp">

<li><a href="#intro"> Intro </a></li>

<li><a href="#team members"> Team Members</a></li>

<li><a href="#ps">Problem Statement</a></li>

<li><a href="#pd">Project Description</a></li>

</ul>

<br/><br/>

<center>

<p>

Project Title: University Catalogue Explorer

</p>

</center>

<br/>

<p id = "team members">

Team Members:

Stephen Heeps (Web designer)<br/>

Jeffrey Paige: Systems Analyst<br/>

Justin Smith: DBA<br/>

</p>

<p class="intro" id="intro">

We feel as though this project will both benefit our target audience as well as our experience and customer relationship. While it gives them an extremely useful tool to project schedules and possible graduation dates, it also helps us with experience in the industry.

</p>

<br/><br/>

<p class = "ps" id = "ps">

Problem Statement:

Assist students and academic advisers in classes, majors, minors, concentrations, and organize this data into one central hub and create this as a SaaS for universities.

Target Audience: Current and future students enrolled in universities/colleges

Why is this an important problem

Why is it better than existing technologies

Many universities lack software to allow students to efficiently view their curriculum, this would provide an easy to access solution that would be able to branch multiple universities to allow students from all over to obtain the same information in one central place.

Why would a business buy this solution

A university that provides open and easy to use methods for students to view and evaluate the classes they&#8217;ll need to take during their program would have a competitive advantage over many other universities. The Student is the customer, and ultimately you&#8217;re trying to sell them a product. If it is easier for the student to decide what they need, they&#8217;ll be more willing to participate.

What good things will come from your solution; what bad things won&#8217;t happen as a result of your solution.

Good things will consist of a new piece of software making students and academic advisers lives easier. Bad things will consist of not having an easy system to work with and students and academic advisers being frustrated with class schedules and possible missing classes.

<br/><br/>

</p>

<p class = "pd" id = "pd">

Project Description:

The solution will consist of three main parts, a web front end to act as an interface between the user and the system, web services that act as an intermediary between the user application and the database, and database that will be used to store and retrieve data. The user will be able to view the current, past, and projected course catalogue for a given university, and sort their curriculum by major, minor, class, and professor. The user will also be able to create mock-up schedules, and provide publicly visible feedback on professors and classes.

Our solution takes an approach that is more geared towards the student being able to get detailed information about the courses required by their program. It will also give students the opportunity to evaluate their experience in classes in a manner that is publicly available to their peers. While some universities may prefer to keep information like that under wraps, it is important in maintaining a healthy level of customer satisfaction between universities and their students. A university that is more able to match the needs of their customers, may perform better in the long run.

We will implement our solution via web page with web design and HTML standards. The web page will interact with web services, written in ASP.net + C#, that will retrieve information from a database that will operate using either Microsoft SQL Server or MySQL.

Technologies you want to work with:

At the beginning of development, yes we will be limited to technology. As development furthers and iterations are met, more technology can be utilized I.E. mobile applications, software, etc.

What are you technologies of choice?

Web page

ASP.net + C#

SQL Server or MySQL(?)

IIS

Mobile devices (later in development when web page is finished and polished)

<br/><br/>

</p>

<center>Client (if you have one):

N/A: This is currently an academic endeavour. </center>

<br/><br/>

<center><a href="#top">Top of page</a></center>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<link rel="stylesheet" type="text/css" href="style.css">

</head>

<title>

Catalogue

</title>

<body>

<ul>

<li><a href="index.html">Home</a></li>

<li><a href="ppt.html">Our presentation</a></li>

<li><a href="submitcatalogue.html">Submit Your Catalogue</a></li>

<li><a href="catalogue.html">View our current catalogues</a></li>

<li><a href="contactus.html">Contact Us</a></li>

<li><a href="projectproposal.html">Our project proposal</a></li>

<li><a href="aboutus.html">About Us</a></li>

</ul>

<center>

<p>

Site under construction. Please come back later!

</p>

</center>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<link rel="stylesheet" type="text/css" href="style.css">

</head>

<title>

Contact Us

</title>

<body>

<ul>

<li><a href="index.html">Home</a></li>

<li><a href="ppt.html">Our presentation</a></li>

<li><a href="submitcatalogue.html">Submit Your Catalogue</a></li>

<li><a href="catalogue.html">View our current catalogues</a></li>

<li><a href="contactus.html">Contact Us</a></li>

<li><a href="projectproposal.html">Our project proposal</a></li>

<li><a href="aboutus.html">About Us</a></li>

</ul>

<br />

<br />

<center> If you have any questions or comments, please e-mail us <a href="mailto:stephen.heeps@snhu.edu"> here!</a>

<br/>

<br/>

Comments: <br/>

<form name = "text-area" action="commentsubmit.html" action="mailto:stephen.heeps@snhu.edu">

<textarea rows = "10" cols = "50" input type = "text" name = textarea">

</textarea><br/>

<input type = "submit" value = "Submit">

</form>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<link rel="stylesheet" type="text/css" href="style.css">

</head>

<title>

Submit Your Catalogue

</title>

<body>

<ul>

<li><a href="index.html">Home</a></li>

<li><a href="ppt.html">Our presentation</a></li>

<li><a href="submitcatalogue.html">Submit Your Catalogue</a></li>

<li><a href="catalogue.html">View our current catalogues</a></li>

<li><a href="contactus.html">Contact Us</a></li>

<li><a href="projectproposal.html">Our project proposal</a></li>

<li><a href="aboutus.html">About Us</a></li>

</ul>

<h1>What are you trying to submit?</h1>

<a href="submitug.html">Undergraduate</a><br/>

<a href="submitg.html">Graduate</a><br/>

<a href="submitc.html">COCE</a><br/>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<link rel="stylesheet" type="text/css" href="style.css">

</head>

<title>

Comment Submitted

</title>

<body>

<ul>

<li><a href="index.html">Home</a></li>

<li><a href="ppt.html">Our presentation</a></li>

<li><a href="submitcatalogue.html">Submit Your Catalogue</a></li>

<li><a href="catalogue.html">View our current catalogues</a></li>

<li><a href="contactus.html">Contact Us</a></li>

<li><a href="projectproposal.html">Our project proposal</a></li>

<li><a href="aboutus.html">About Us</a></li>

</ul>

<center> Comment submitted! Thank you for your input! :-) </center>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<link rel="stylesheet" type="text/css" href="style.css">

</head>

<title>

University Catalogue Explorer

</title>

<body>

<ul>

<li><a href="index.html">Home</a></li>

<li><a href="ppt.html">Our presentation</a></li>

<li><a href="submitcatalogue.html">Submit Your Catalogue</a></li>

<li><a href="catalogue.html">View our current catalogues</a></li>

<li><a href="contactus.html">Contact Us</a></li>

<li><a href="projectproposal.html">Our project proposal</a></li>

<li><a href="aboutus.html">About Us</a></li>

</ul>

<center>

<p>

Site under construction. Please come back later!

</p>

</center>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<link rel="stylesheet" type="text/css" href="style.css">

</head>

<title>

University Catalogue Explorer

</title>

<body>

<ul>

<li><a href="index.html">Home</a></li>

<li><a href="ppt.html">Our presentation</a></li>

<li><a href="submitcatalogue.html">Submit Your Catalogue</a></li>

<li><a href="catalogue.html">View our current catalogues</a></li>

<li><a href="contactus.html">Contact Us</a></li>

<li><a href="projectproposal.html">Our project proposal</a></li>

<li><a href="aboutus.html">About Us</a></li>

</ul>

<center>

<p>

Site under construction. Please come back later!

</p>

</center>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<link rel="stylesheet" type="text/css" href="style.css">

</head>

<title>

University Catalogue Explorer

</title>

<body>

<ul>

<li><a href="index.html">Home</a></li>

<li><a href="ppt.html">Our presentation</a></li>

<li><a href="submitcatalogue.html">Submit Your Catalogue</a></li>

<li><a href="catalogue.html">View our current catalogues</a></li>

<li><a href="contactus.html">Contact Us</a></li>

<li><a href="projectproposal.html">Our project proposal</a></li>

<li><a href="aboutus.html">About Us</a></li>

</ul>

<center>

<iframe src="https://docs.google.com/presentation/d/1EGsmi9QJaKO7Qd9nmcovmxF4K8FadMGICNH7vr1bReY/present#slide=id.p" style="width:1200px; height:600px;" frameborder="0"></iframe>

</center>

</body>

</html>

CSS

a:link {color:#FF0000;} /\*unvisited\*/

a:visited {color:#FFFFFF;} /\*visited\*/

a:hover {color:#3399FF;} /\*mouseover\*/

a:active {color:#1919D1;} /\*selected\*/

body

{

background-image:url("images/background.jpg");

background-repeat: no-repeat;

background-size: cover;

color:#FFFFFF;

font-family:"Century Gothic";

/\* cursor:wait; \*/

}

ul

{

width: 570px;

padding:15px;

margin: 0px auto 0px auto;

border-top: 2px solid #FFFFFF;

border-bottom: 1px solid #FFFFFF;

text-align: center;

}

ul.pp

{

border-top:#000000;

}

li

{

display: inline;

margin: 0px 3px;

}

p

{

width: 500px;

border: 2px solid #0088dd;

padding: 10px;

font-size: 20px;

text-align: center;

}

p.intro

{

margin-left:600px;

}

p.ps

{

margin-left: 750px;

width:1000px;

}

p.pd

{

width:1150px;

}

PHP

<?php>

// Establish the connection

$con = mysqli\_connect ("50.163.101.38:3306","root","w6IEAPQY4KaMl13R5csA","mydb"); //host = host name or IP address.

//Check connection

if (mysqli\_connect\_errno())

{

echo "Failed to connect to MySQL: " . mysqli\_connect\_error();

}

$result = mysqli\_query($con,"SELECT \* FROM ");

echo "<table border='1'>

<tr>

<th>University</th>

<th>Types of Degrees</th>

<th>Types of Majors</th>

<th>Types of Minors</th>

<th>Types of Concentrations</th>

</tr>"

while($row = mysqli\_fetch\_array($result))

{

echo "<tr>";

echo "<td>" . $row[''] . "</td>"; //Here is where we'll get data for University. Create links

echo "<td>" . $row[''] . "</td>"; //Here is where we'll get data for types of degress. Create links

echo "<td>" . $row[''] . "</td>"; //Here is where we'll get data for majors. Create links

echo "<td>" . $row[''] . "</td>"; //Here is where we'll get data for minors. Create links

echo "<td>" . $row[''] . "</td>"; //Here is where we'll get data for concentrations. Create links

echo "</tr>";

}

echo "</table>";

mysqli\_close($con);

?>

**C# Login Web Service**

This is a prototype web service to use for storing and checking passwords against a database. It salts and hashes passwords before storing them, and checks the password against the hash.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.Serialization;

using System.ServiceModel;

using System.ServiceModel.Web;

using System.Text;

using System.Data;

using System.Security.Cryptography;

using MySql.Data.MySqlClient;

namespace LoginService

{

// NOTE: You can use the "Rename" command on the "Refactor" menu to change the class name "Service1" in code, svc and config file together.

// NOTE: In order to launch WCF Test Client for testing this service, please select Service1.svc or Service1.svc.cs at the Solution Explorer and start debugging.

public class Service1 : IService1

{

public bool registerAccount(string email, string password)

{

MySqlConnection connection = new MySqlConnection();

//Create connection string, using test info. Realistically you would read this from a config file or something

connection.ConnectionString = "Server=127.0.0.1;" + "Database=mydb;" + "UID=root;" + "Password=w6IEAPQY4KaMl13R5csA;";

//Hash and salt password

PasswordHash hashedPassword = new PasswordHash(password);

byte[] hashedBytes = hashedPassword.ToArray();

string hashedPasswordString = Convert.ToBase64String(hashedBytes);

//Open our connection

connection.Open();

//Create a MySQLCommand in order to execute a query

MySqlCommand command = connection.CreateCommand();

//Add our parameters

command.Parameters.AddWithValue("email", email);

command.Parameters.AddWithValue("password", password);

//Set the query to be executed

command.CommandText = "INSERT INTO student (studentEmail, studentPassword) VALUES ('" + email + "','" + hashedPasswordString + "');";

//execute the query

command.ExecuteNonQuery();

//close the query

connection.Close();

//send true back to the client to say it successfully executed

return true;

}

public bool verifyUser(string email, string password)

{

//Create a MySQL Connection

MySqlConnection connection = new MySqlConnection();

//Create connection string, using test info. Realistically you would read this from a config file or something

connection.ConnectionString = "Server=127.0.0.1;" + "Database=mydb;" + "UID=root;" + "Password=test;";

//open the connection

connection.Open();

//Create a MySQLCommand in order to execute a query

MySqlCommand command = connection.CreateCommand();

//Set the query to be executed

command.CommandText = "SELECT studentEmail, studentPassword FROM student WHERE studentEmail = '" + email + "';";

//Read the results

MySqlDataReader result = command.ExecuteReader();

//If there are no results send back false

//this occurs if there are no entries with the email

if(result.Read() == false)

{

connection.Close();

return false;

}

//read the fields from the result

string retrievedEmail = (string)result.GetValue(0);

string retrievedPassword = (string)result.GetValue(1);

//Since the hash is stored as a base64 string, we need to convert back to a byte array

byte[] hashedBytes = Convert.FromBase64String(retrievedPassword);

PasswordHash hashedPassword = new PasswordHash(hashedBytes);

//Check to see if the hashed password matches the input

if(hashedPassword.Verify(password))

{

connection.Close();

return true;

}

else

{

connection.Close();

return false;

}

}

}

//Using code found here (http://csharptest.net/470/another-example-of-how-to-store-a-salted-password-hash/) to generate and verify salted and hashed passwords

public sealed class PasswordHash

{

const int SaltSize = 16, HashSize = 20, HashIter = 10000;

readonly byte[] \_salt, \_hash;

public PasswordHash(string password)

{

new RNGCryptoServiceProvider().GetBytes(\_salt = new byte[SaltSize]);

\_hash = new Rfc2898DeriveBytes(password, \_salt, HashIter).GetBytes(HashSize);

}

public PasswordHash(byte[] hashBytes)

{

Array.Copy(hashBytes, 0, \_salt = new byte[SaltSize], 0, SaltSize);

Array.Copy(hashBytes, SaltSize, \_hash = new byte[HashSize], 0, HashSize);

}

public PasswordHash(byte[] salt, byte[] hash)

{

Array.Copy(salt, 0, \_salt = new byte[SaltSize], 0, SaltSize);

Array.Copy(hash, 0, \_hash = new byte[HashSize], 0, HashSize);

}

public byte[] ToArray()

{

byte[] hashBytes = new byte[SaltSize + HashSize];

Array.Copy(\_salt, 0, hashBytes, 0, SaltSize);

Array.Copy(\_hash, 0, hashBytes, SaltSize, HashSize);

return hashBytes;

}

public byte[] Salt { get { return (byte[])\_salt.Clone(); } }

public byte[] Hash { get { return (byte[])\_hash.Clone(); } }

public bool Verify(string password)

{

byte[] test = new Rfc2898DeriveBytes(password, \_salt, HashIter).GetBytes(HashSize);

for (int i = 0; i < HashSize; i++)

if (test[i] != \_hash[i])

return false;

return true;

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.Serialization;

using System.ServiceModel;

using System.ServiceModel.Web;

using System.Text;

namespace LoginService

{

// NOTE: You can use the "Rename" command on the "Refactor" menu to change the interface name "IService1" in both code and config file together.

[ServiceContract]

public interface IService1

{

[OperationContract]

bool registerAccount(string email, string password);

[OperationContract]

bool verifyUser(string email, string password);

}

}