Diabetes Decision System

Team members: Dimitrije Prosevski, Matthew Jackson

Advisor: Arjan Durresi

Advisor's email: adurresi@iupui.edu

Introduction

In this project, we focused on the problem of connecting a machine learning back-end with a front-end website and application. This is an important area because it is needed with any project that has many components that must connect to each other. We found that there are many ways to do this but our focus was to use an SQL database. The project that we were trying to successfully connect involved a dietary and exercise system for a website and application. The back-end python would take the exercise and diet input from the front end after any specified update to the database and perform machine learning to provide a recommended direction that the user should take with their diet and exercise.

Problem

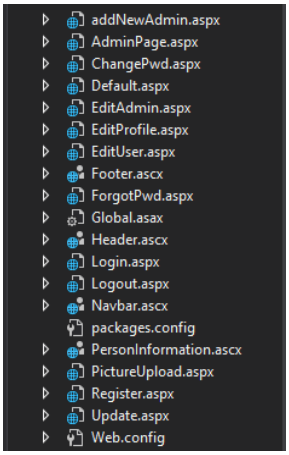
There were many specific things we needed to solve before we could have a working system. The first being our database and how its logical data model should look. A model that would fit our user information and input is what we needed. Other than a database, we needed a front-end connection that would be able to receive the input and information from our users and correctly store and manage it in the database. It would also need to generate a form of output that would be useful for the required input in the other sections of this project.

Website

Built in the ASP.NET framework, the website provides strong security, efficiency and user-friendliness. All the code is written using HTML and CSS for the structure and design of the website, along with C# in the back end to add all of the logic, functionality of the <asp> elements, and connection to the SQL server. For the design of the webpage we initially used a bootstrap template “Grayscale” and built on it from there.

Navigation and functionality:

There are 12 usable pages in total:



Registration: allows to register new users

Login: allows user to log in

Home Page: talks about general idea of the software, contact information, and other

Update: allows the user to update his/her nutritional and activity information

Picture Upload: allows the user to upload a picture for image recognition

Change Password: allows user to safely change password

Forgot Password: allows user to reset password using email address

Edit Profile: allows user to edit his profile information

Admin Page: allows links to Add Admin, Edit Admin and Edit User pages

Add Admin: allows super admin to add new admins

Edit Admin: allows super admin to edit other admins

Edit User: allows super admin and admins to edit regular users

The website was built using bootstrap. It has a very friendly and clear layout, with a navigation bar at the top linking all the pages accessible to the current user. Each form has an appropriate form validation with required field and regular expression validators (for emails and passwords). Each form submission, or change, has a confirmation form that allows the user to recheck/confirm the wanted change.

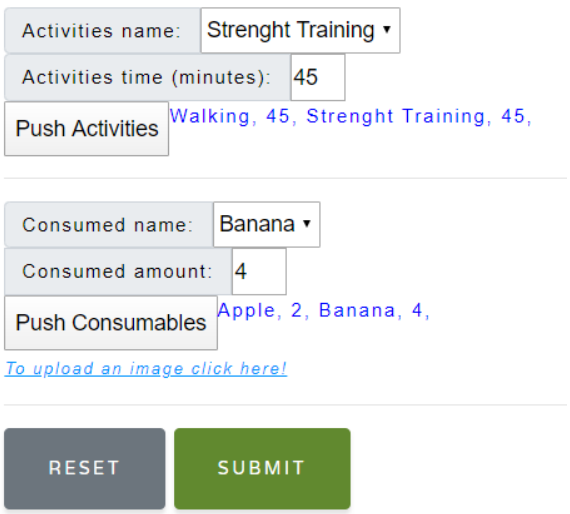
Example, edit profile form confirmation:

Prefix: Ms
First name: Nadine
Middle name: Hensley
Last name: Grant
Sufix: PhD
Alternative Email: weby12@mailinator.net
Phone: 11267054582
Address: 32 S Street
Date of Birth:
Gender: Female
Height (Ft): 5
Height (In): 6
Weight (lb): 152
Blood Type: A+
Allergies: Pollen
Terms accepted.

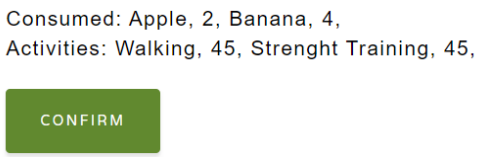
CONFIRM

Update Page:

The main functionality of the website is to update user nutrition and activity information through the “Update” page. Logged in users can choose the activity and consumable from the drop-down lists, put in the time and amount respectively and push that into vectors. Each time a user pushes something onto the vectors it concatenates the input.



Once the user submits the information, a confirmation form pops up



When confirmed, the database is updated

	email	inputAct	outputAct	inputCons	outputCons
1	test@gmail.com	Walking, 45, Strenght Training, 45,	NULL	Apple, 2, Banana, 4,	NULL

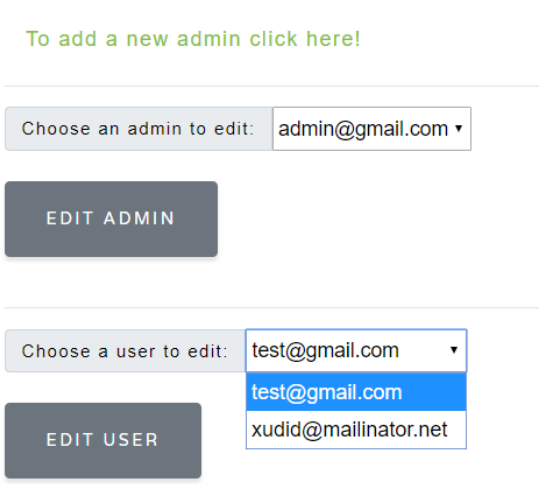
it was planned to have a python machine learning script that would trigger after user change of the activity and consumed updates, run the algorithm on it and prompt the results back. Another way to run the script would be through an image upload. The highlighted part has not been implemented yet due to time restrictions.

Accessibility hierarchy:

Access Type: super admin > admin > regular user

Regular users: can update their nutrition and activity
Admin: can do the same as regular user, and also can edit regular users
Super admin: can do the same as admin, also can edit and add new admins

Example, super admin's admin page options:



Security and Access:

The website is highly secured; SQL injection proof and access of the user is restricted to his/her access type.

All the pages that require login cannot be accessed without log in.

All the pages and functionality that require level admin or super admin access cannot be accessed by regular users.

All the pages that require super admin access cannot be accessed by admins or regular users.

Database

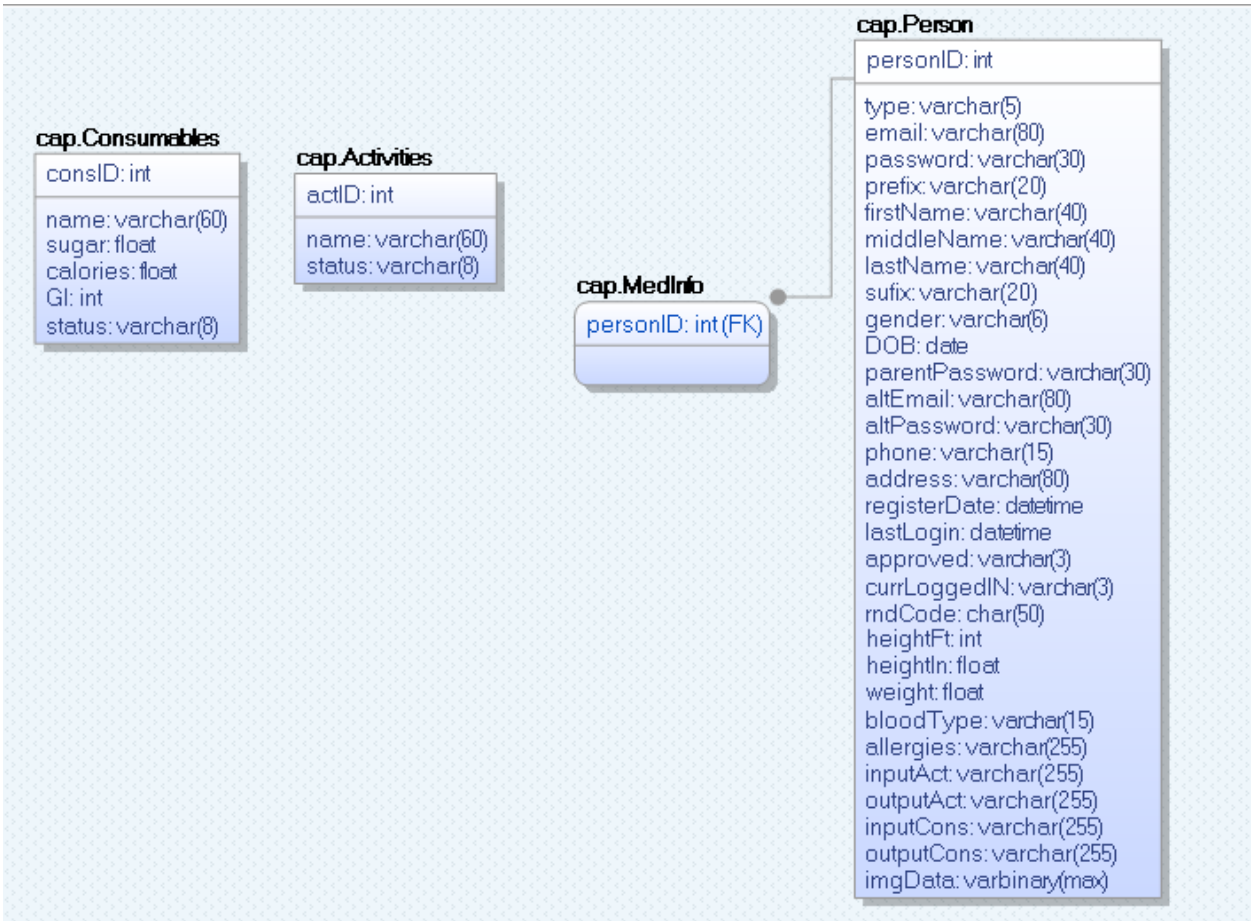
ER diagram:

cap.Person is used to store user's information

cap.Consumables used to store options of possible consumables and their attributes

cap.Activities used to store options of possible activities and their attributes

cap.MedInfo needs more work (personal medical information)



Explanation of columns:

Note: The reasons you see NULL in some columns is because the first 3 rows were manually input through the DB query for testing purposes

	personID	type	email	password	prefix	firstName	middleName	lastName	sufix
1	4	user	test@gmail.com	test123!!!		firstuser		lastuser	
2	5	admin	admin@gmail.com	test123!!!		firstadmin		lastadmin	
3	6	super	super@gmail.com	test123!!!		firstsuper		lastsuper	
4	7	user	xudid@mailinator.net	test123!!!	Ms	Nadine	Hensley	Grant	PhD

- Person ID: Each person upon registering gets an auto incremented unique ID
- Type: Each person upon registering is set to "user", if super admin adds a new admin, type is set to "admin". Super admins have to be added manually through the database query by the DBA.
- Email: is unique to each user, represents username
- Password: minimum length 10 and a mixture of letters, numbers and signs.
- Name consists of:
    - Prefix
    - First Name (required)
    - Middle Name
    - Last Name (required)
    - Suffix

gender	DOB	parentPassword	altEmail	altPassword	phone	address
	1900-01-01					
	1900-01-01					
	1900-01-01					
Female	1980-05-01	test123!!!	weby2@mailinator.net	test123!!!	11267054582	32 S Street

- Gender: input taken by radio buttons (male or female)
- DOB: date of birth of the user (yyyy-mm-dd)
- Parent's password: minimum length 10 and a mixture of letters, numbers and signs.
- Alternative Email: serves as an alternative (notification) email
- Phone number: user's phone number
- Address: user's address

registerDate	lastLogin	approved	currLoggedIN	rndCode
2020-03-15 22:21:19.000	2020-05-01 12:05:20.000	no	yes	AAGUBTMKJPUQTFPWHAXOJWPSCHEHEIHRPKOPGKILMWAHXEWRBQ
2020-03-15 22:21:19.000	2020-05-01 11:52:45.000	no	no	AAGUBTMKJPUQTFPWHAXOJWPSCHEHEIHRPKOPGKILMWAHXEWRBQ
2020-03-15 22:21:19.000	2020-05-01 11:54:16.000	NULL	yes	AAGUBTMKJPUQTFPWHAXOJWPSCHEHEIHRPKOPGKILMWAHXEWRBQ
2020-05-01 12:13:12.000	2020-05-01 12:19:14.000	no	no	EGXFVGABBRNQXEPFLOOWVMBFLMVFMFJBADFLJCMNAKMODUXJVE

- Register Date: date and time user registered
- Last login: last date and time user logged in
- Approved: (not implemented - does not affect any logic currently)
- Currently Logged in: shows if the user is currently logged in
- Random Code: random code generated for the confirmation email once the user registers (email server currently not implemented)

heightFt	heightIn	weight	bloodType	allergies	inputAct	outputAct	inputCons	outputCons	imgData
NULL	NULL	NULL	NULL	NULL	Sports, 45,	NULL	Banana, 3,	NULL	0xFFD8FFE000104A46494
NULL	NULL	NULL	NULL	NULL	Running, 45,	NULL	Apple, 2, Orange, 2,	NULL	NULL
NULL	NULL	NULL	NULL	NULL	Strenght Training, 35, Walking, 20,	NULL	Pear, 3, Orange, 3,	NULL	NULL
5	6	152	A+	Pollen	Sports, 45,	NULL	Apple, 2,	NULL	0xFFD8FFE1001845786968

- Personal information:
    - Height Feet: user's height in feet
    - Height In: user's height in inches
    - Weight: user's weight
    - Blood Type: user's blood type
    - Allergies: if any allergies

  (some of the stats here were implemented solely to show an example of type of inputs)
- Input/Output:
    - Activity:
        - Input activity
        - Output activity
    - Consumed:
        - Input consumed
        - Output consumed
- Image Data: uploaded consumed image converted to bytes

View created, IO helps visualize input and output of the users and email associated with the account:

	email	inputAct	outputAct	inputCons	outputCons
1	test@gmail.com	Walking, 45, Strenght Training, 45,	NULL	Apple, 2, Banana, 4,	NULL
2	admin@gmail.com	Running, 45,	NULL	Apple, 2, Orange, 2,	NULL
3	super@gmail.com	Strenght Training, 35, Walking, 20,	NULL	Pear, 3, Orange, 3,	NULL
4	xudid@mailinator.net	Sports, 45,	NULL	Apple, 2,	NULL

Example, SQL statement used for registration of a new user:

```
con.Open();
sql = "INSERT INTO  cap.Person ([type], [email], [password], [prefix], [firstName], [middleName], [lastName], [sufix], [gender], [DOB], [parentPassword], " +
   "[altEmail], [altPassword], [phone], [address], [registerDate], [approved], [currLoggedIN], [rndCode]," +
   " [heightFt], [heightIn], [weight], [bloodType], [allergies])" +
   "values('user', @email, @password, @prefix, @first, @middle, @last, @sufix, @gender, @dob, @parentPass, @altEmail, @altPassword," +
   "@phone, @address, @currentTime, 'no', 'no', @strCodeForm, @heightFt, @heightIn, @weight, @bloodType, @allergies);";
cmd = new SqlCommand(sql, con);

cmd.Parameters.Add(new SqlParameter("@email", Session["emailRegForm"]));
cmd.Parameters.Add(new SqlParameter("@password", Session["passRegForm"]));
cmd.Parameters.Add(new SqlParameter("@prefix", Session["prefixRegForm"]));
cmd.Parameters.Add(new SqlParameter("@first", Session["firstRegForm"]));
cmd.Parameters.Add(new SqlParameter("@middle", Session["middleRegForm"]));
cmd.Parameters.Add(new SqlParameter("@last", Session["lastRegForm"]));
cmd.Parameters.Add(new SqlParameter("@sufix", Session["sufixRegForm"]));
cmd.Parameters.Add(new SqlParameter("@gender", Session["genderRegForm"]));
cmd.Parameters.Add(new SqlParameter("@dob", Session["dobRegForm"]));
cmd.Parameters.Add(new SqlParameter("@parentPass", Session["parentPassRegForm"]));
cmd.Parameters.Add(new SqlParameter("@altEmail", Session["altEmailRegForm"]));
cmd.Parameters.Add(new SqlParameter("@altPassword", Session["altPassRegForm"]));
cmd.Parameters.Add(new SqlParameter("@phone", Session["phoneRegForm"]));
cmd.Parameters.Add(new SqlParameter("@address", Session["addressRegForm"]));
cmd.Parameters.Add(new SqlParameter("@currentTime", currentTime));
cmd.Parameters.Add(new SqlParameter("@strCodeForm", strCodeForm));
cmd.Parameters.Add(new SqlParameter("@heightFt", Session["heightFtRegForm"]));
cmd.Parameters.Add(new SqlParameter("@heightIn", Session["heightInRegForm"]));
cmd.Parameters.Add(new SqlParameter("@weight", Session["weightRegForm"]));
cmd.Parameters.Add(new SqlParameter("@bloodType", Session["bloodTypeRegForm"]));
cmd.Parameters.Add(new SqlParameter("@allergies", Session["allergiesRegForm"]));

cmd.ExecuteNonQuery();
con.Close();
```

Each variable is passed as a parameter to prevent SQL injection.

Application

A final addition to the project was our Xamarin Forms Android application. Which uses XML and C#. This application was an attempt to better understand the process involved in having a mobile connection to our database. Its main focus would be image uploading for the input needed within the OCR section of the project.

Here is a picture of the application running:

The design is simple due to the time restriction of our project. However, the functionality to take/upload images to the database is working. And more additions to the application, like login etc. could be added in the future. As for functionality of the application, the icon image is a placeholder for an image view. This image view is overwritten with a photo taken or uploaded and the current photo in the view is then sent to the database as a byte array.

The most important code of the application resides in the MainActivity.cs. It begins with accessing the correct permissions needed to use the camera and storage of the device and initializing the Xamarin platform. The former involves the AndroidManifest.xml and file_paths.xml files. After this it creates two button onclick functions, one for taking a photo and one for uploading a photo.

Upon receiving a photo, the application then changes the image into a byte array and then sets it as a bitmap to display in the view. This is done here:

```
byte[] imageArray = System.IO.File.ReadAllBytes(file.Path);
Android.Graphics.Bitmap bitmap = BitmapFactory.DecodeByteArray(imageArray, 0, imageArray.Length);
theImageView.SetImageBitmap(bitmap);
```

After setting the view the application then sends the byte array of the image to the database.

```
SqlConnection con = new SqlConnection(cs);
string sql;
sql = "";
SqlCommand cmd = new SqlCommand(sql, con);

try
{
    sql = "UPDATE cap.Person SET imgData = @imgData;";
    cmd = new SqlCommand(sql, con);
    cmd.Parameters.Add(new SqlParameter("@imgData", imageArray));
    con.Open();
    cmd.ExecuteNonQuery();
}
catch (Exception err)
{
    con.Close();
}
finally
{
    con.Close();
}
```

Division of work

Dimitrije - Website, Database management/design.

Matthew - Application, Database management/design

GitHub link of the repository with all the code:
https://github.iu.edu/dprosev/CapstoneProject

Conclusion

Coming to the end of this semester, we have gained an increased understanding of the requirements needed in connecting and using a database system. We have learned how to develop a database model that will fit the needs of an exercise/dieting project. Also, we have become more familiar with the technical skills involved in using the ASP.NET and Xamarin frameworks, as well as in using an SQL database. There are still improvements that can be made, and we look forward to adding these in the future.