Project-3

This project is a web programming project which contains five pieces. The Front End, the Java Script, the Python and the Back End Database .

MahaLakshmi and Deepa worked on the entire project together. Using Zoom video chat’s screen share and remote control facility. We had the project running on Deepa’s computer.

To facilitate the workload sharing, Mahalakshmi designed the front end programs:

genhtml.py

genDisplayhtml.py

These two programs generated the interests.html, display\_interests.html files which become the front-end pieces of the project.

Deepa designed the back end and the middleware pieces. The files she programmed are:

Webforms.py (the main program that calls all the other functions)

GenerateJS.py

pyServer.py

GenSQL.py

The error checking part of the program is within the ValidateJSON.py

The report was created by Mahalakshmi. The backend explanations were added by Deepa

# Front End Outline:

* HTML, Python, JSON – To build the Front end we used Html, to generate the files- python.
* mySQL database. We create a new database before running the SQL query that is generated by the sql generator python file.
* Front end is created by running the html file. It is used to display the given form and to take the input from the user. Depending on whether the WebForm.py uses form1.json or form2.json the generator files create student.html or the interest.html and the corresponding python server, sql queries and the display html files.
* We used Notepad++ and pycharm to write the python programs, debug the JavaScript and the html files.

# Working:

To understand how all this works and what was needed to be generated by the python program we first created the html files using form1.json and form2.json. Using our knowledge from project1 we created the java script and python programs and the sql queries to create the database and generate the tables. We now had a working program that took data from a html form and updated the database. We verified that the database was updated by using the display\_html file to display the contents from the database.   
Once this was working we were now ready to write the python programs that generated these files. The main thing we had to keep in mind was that it was generic and could work with any json form giving the dynamic feel to the working of the programs.

**Front End: HTML and JavaScript:**

As the data input forms (form1.json, form2.json) had form elements of various types (interest/ student.html) which can accept the input from a user. We used 7 types of form elements which includes textbox, submit, reset, check box, radio buttons, select list, multiple select list and submit and reset has a data type associated with this.

The python program (WebForms.py) will take form1.json or form2.json as an input file and should generate the html file (student/interest.html) display the form and the input from the user. On clicking the submit button it stores information of a user in the database it is connected to. This is done by the python program (genhtml.py)

Below is the listing of all fields embedded within the html page that is created using the JSON form.

**Textbox**: The form attribute specifies the form the text area belongs to. The value of this attribute must be equal to the id attribute of a <form> element in the same document.

**Select** : It is used to select a particular option from the list given .

**Multi-select** : The multiple attribute is a boolean attribute.

When present, it specifies that multiple options can be selected at once.

**Radio button** : Radio buttons are normally presented in radio groups (a collection of radio buttons describing a set of related options).

Only one radio button in a group can be selected at the same time.

**Check Box** : The checkbox is shown as a square box that is ticked (checked) when activated.

Checkboxes are used to let a user select one or more options of a limited number of choices.

**Submit**: The submit button defines which submits all form values to a form-handler.

The form-handler is typically a server page with a script for processing the input data.

**Reset**: The reset button which resets all form values to its initial value.

The JavaScript file (student.js) uses the jquery file to connect to the server using ajax protocols. The Java Script program contains the ajax program code to connect to the back end server and transmit the data collected from the html fields in form of JSON objects. The call to the javascript program is embedded in the html files. This program is generated by the Python program (generateJS.py)

The JSON validator validates the JSON key-value pairs within the json form.

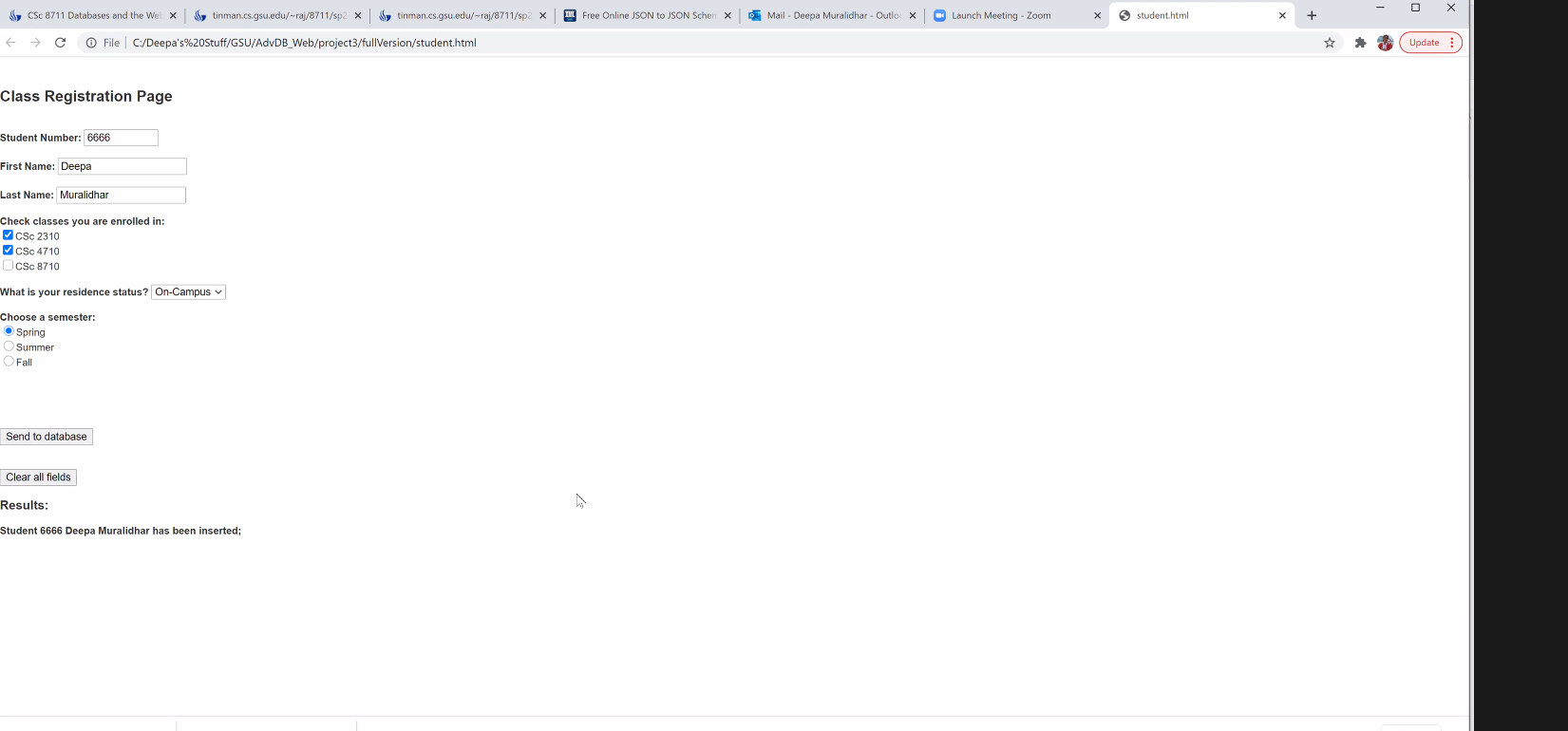
The Python Server (pyserver.py) creates the commands to get the flask server working and generates a listener. This listener looks for data from the front end (jquery programs) and the backend. (my SQL database). This program also generates queries to update the tables and in the front end waits from the javascript file in form of JSON queries. The python program generated (student/interest.py) is run on the command prompt to get the web server working.

The SQL generator (gensql.py) creates the SQL queries that are needed to create the tables and the columns in the database. The SQL script (student/interest.sql) runs on the database schema created in mySQL and creates the tables and the columns.

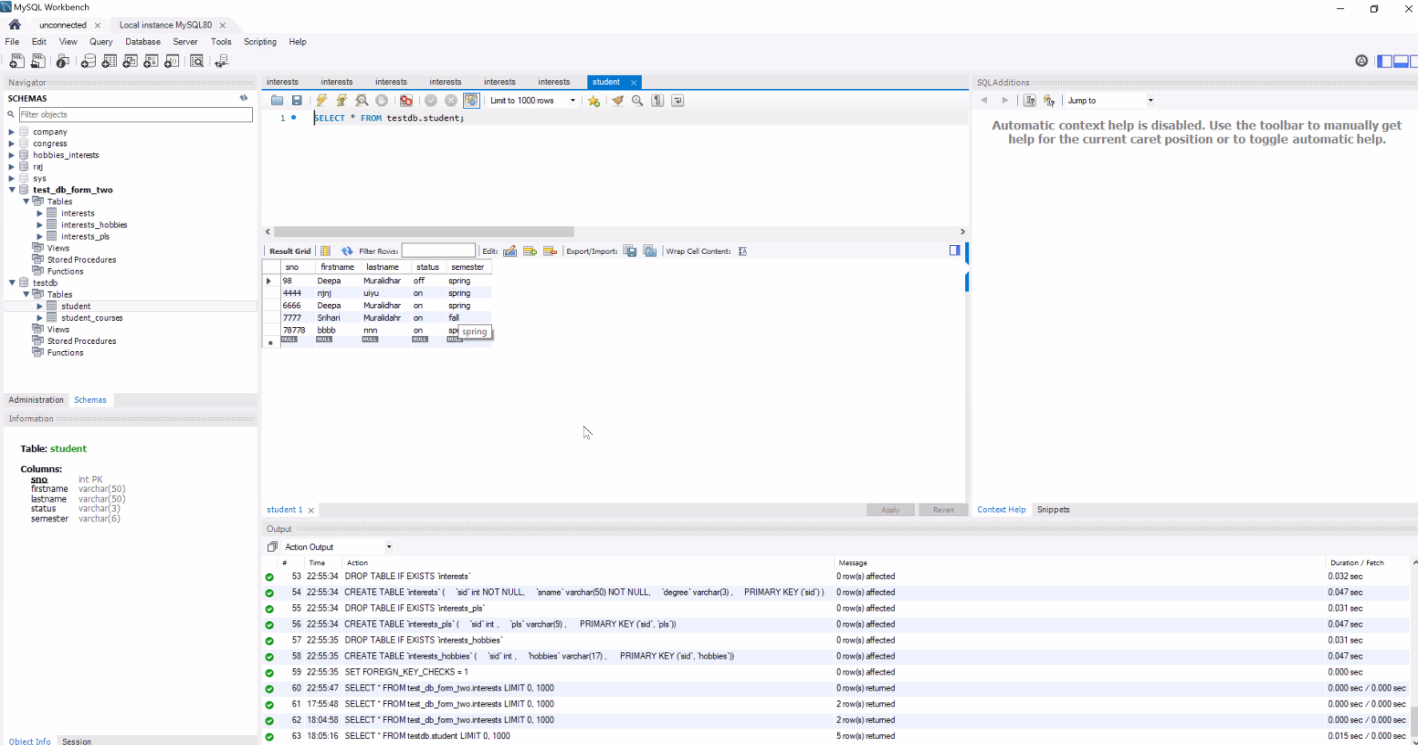
The python program (gendisplayhtml.py) creates the html file that displays the data from the database. (display\_student/display\_interest.html). By running the html file and clicking display data button the contents of the data base are shown in a tabular form as designed.

**Working Screen Shots:**

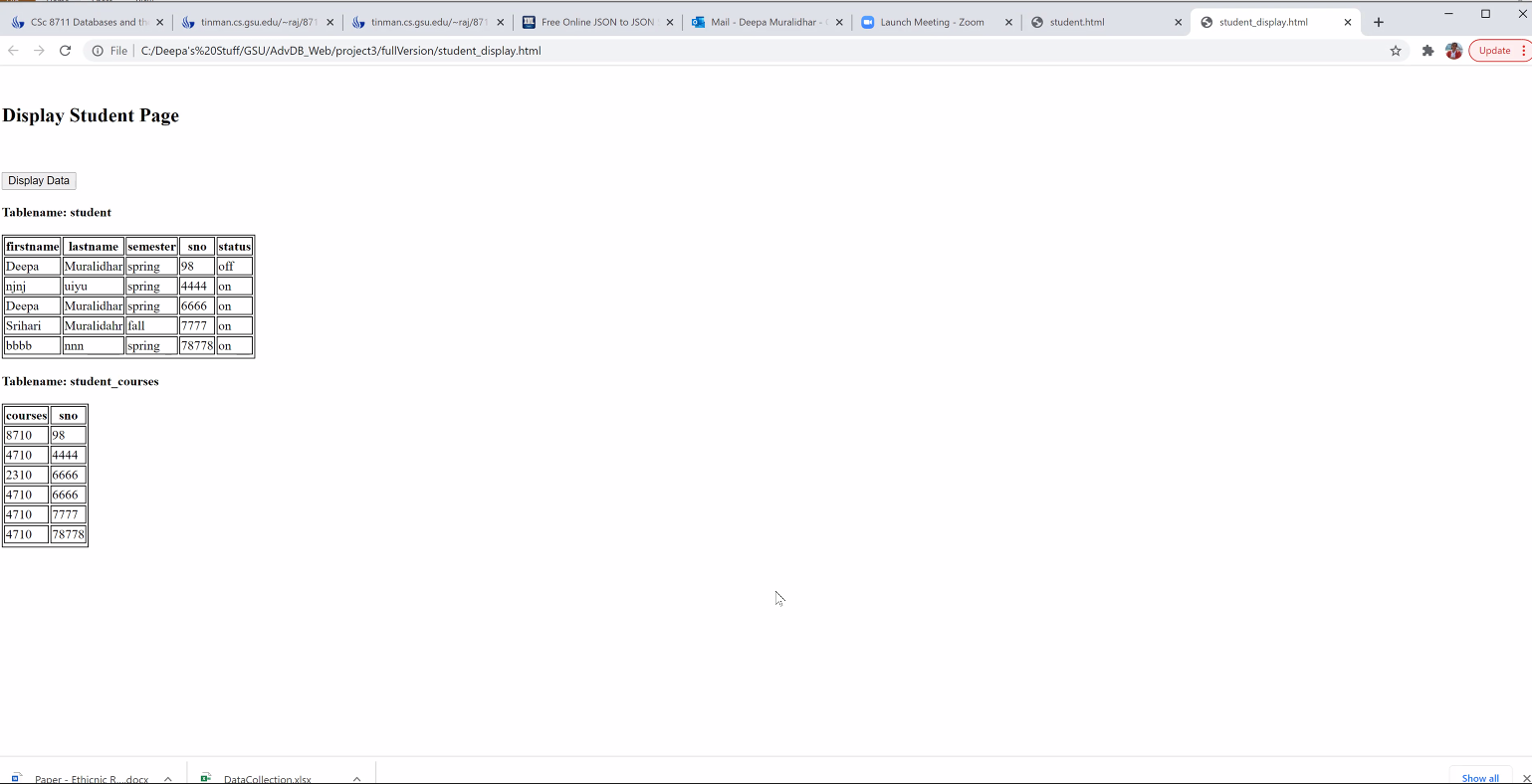
**Form 1:** Giving input data in the form1.json



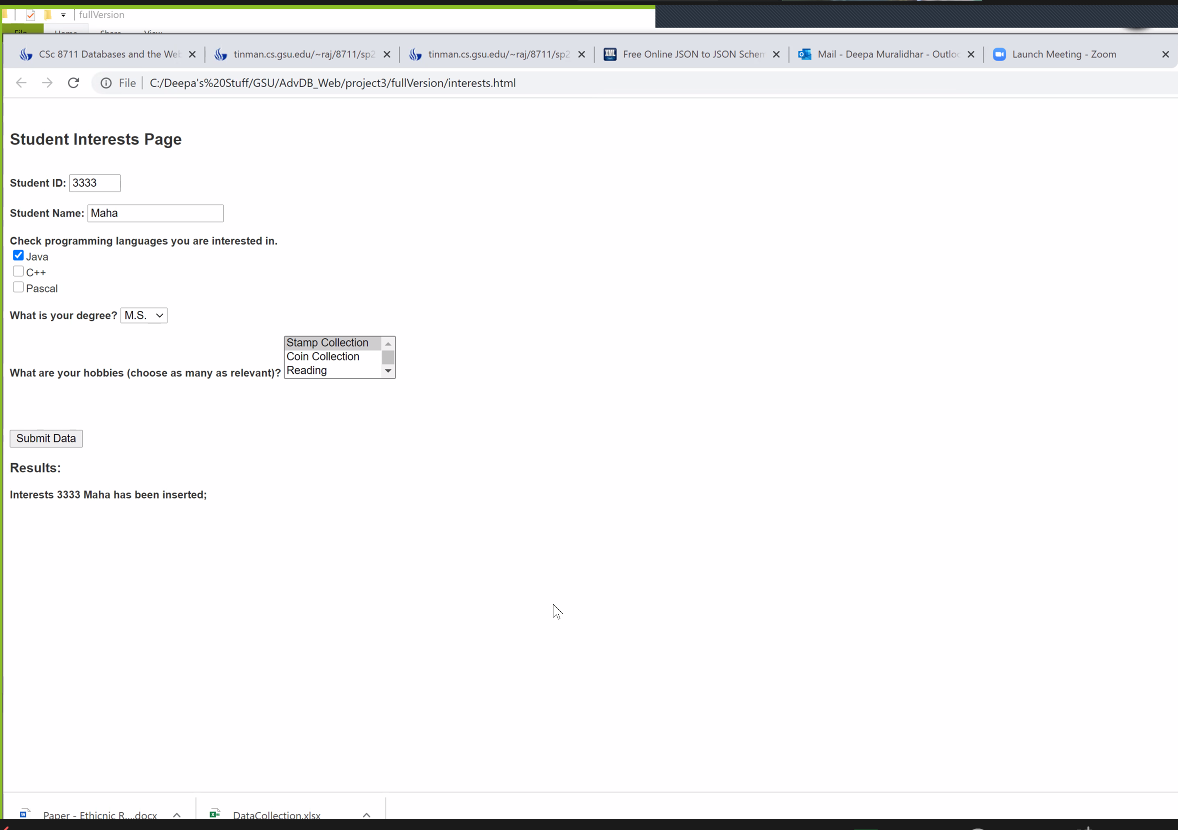
Data being loaded in the database



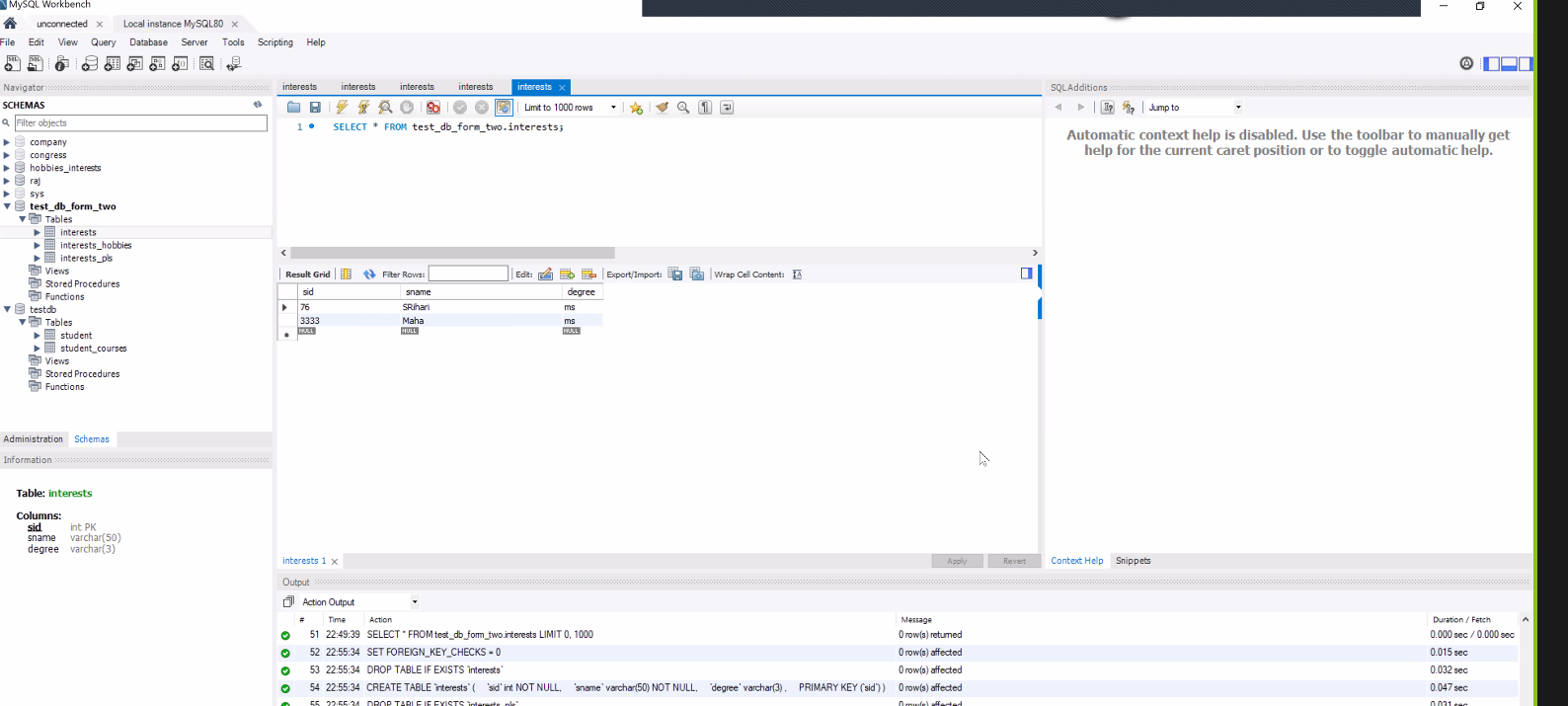
Displaying the data from the database on clicking the display



**Form 2 :** Giving form2.json as input



Data being loaded in the database



Data being displayed on the clicking the display button .

