Project #4 (Application System) CMSC389N Due Thu Jun 25 11:59 pm Summer 2015

Objectives

- To practice sessions, MySQL processing through php scripts, and object-oriented programming in PHP.

Overview

For this project you will write an application that allow students to apply to a set of classes. Students will go online and fill out an application where they will provide name, e-mail, current school year and additional information related to the class they would like to apply for. The application will also provide options (e.g., displaying current applicants, submitting an SQL query against the database) for the administrator of the application system.

You need to implement your code in a folder named **ApplicationSystem.** We should be able to execute your application through the file **main.html** (which resides in the **ApplicationSystem** folder). You can define as many php script files you understand you need.

Grading

- (15 pts) Submit application functionality.
- (15 pts) Review application functionality.
- (15 pts) Update application functionality.
- (5 pts) Logging to administrative interface.
- (15 pts) Administrative interface functionality.
- (15 pts) Project implemented using object-oriented design (you must define and use php classes).
- (20 pts) Other requirements

Demo

You will do a demo of your project with a TA or instructor in lecture. For the demo:

- Load the application table with the data available at Table Data.
- Wait for the grader in order to download your submission from the submit server.
- You can do your demo on Friday if you are ready. Just notify your instructor at the beginning of Friday's lecture.

Requirements

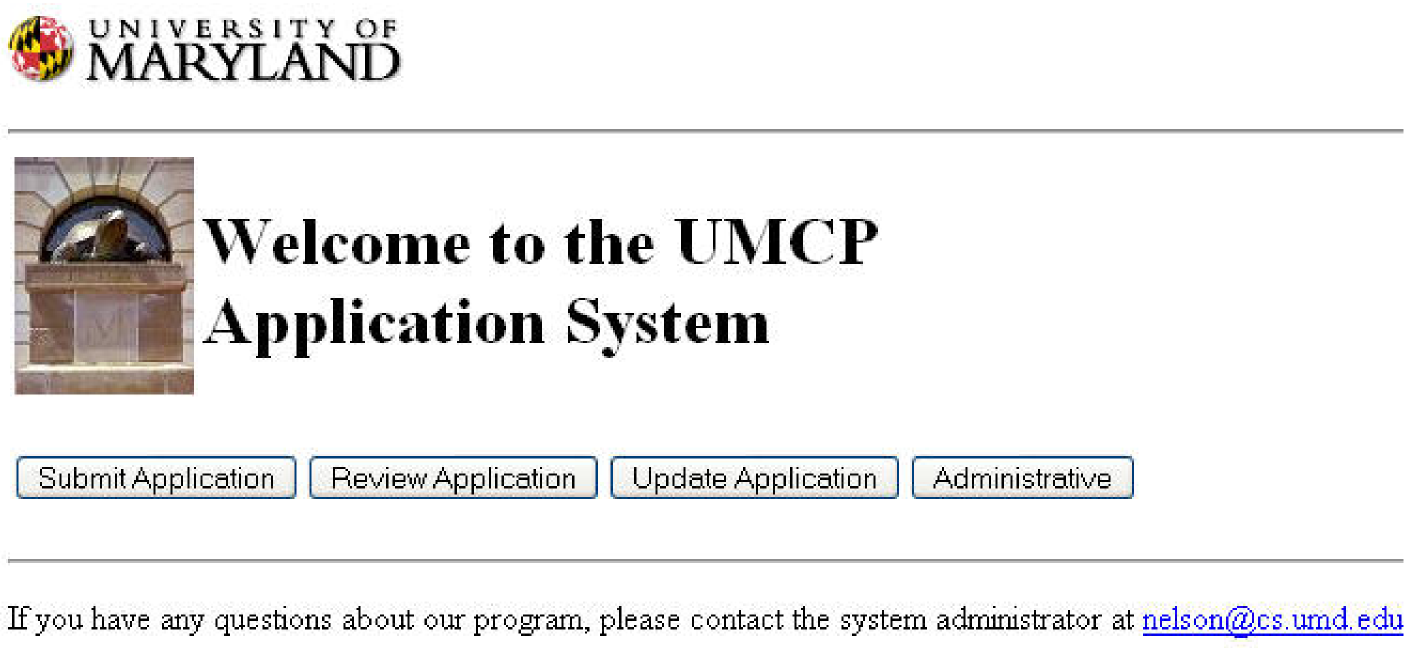
- You must submit forms using the post method.
- Your program must define a table to keep information about applicants. The table has the following specifications:
    - Database Name - applicationdb
    - Table Name - applicants
    - Fields
        - **name** - string with a maximum of 50 characters
        - **email** - string with a maximum of 100 characters. The e-mail represents the primary key and cannot be null.
        - **gpa** - float
        - **year** - integer
        - **gender** - enumeration type that can assume the values 'M' or 'F'
        - **password** - string (you decide the maximum length).
    - username - dbuser
    - password - goodbyeWorld
    - For simplicity all database operations (including administrative) will use the dbuser account.
    - You will create the database by using the MySQL console (not through a php script).
- The main tasks your application must implement are:
    - **submit application** - An applicant's name, e-mail, gpa, year (value between 10 and 12), gender, and password will be read through a form. See the Sample section for the form and web page contents to use.
    - **review application** - Allows an applicant to see his/her application information. A form will request the applicant's email and password and your application will display the information available on the database. See the Sample section for the form and web page contents to use.
    - **update application** - Allows an applicant to update the information present in the database. A form will request the applicant's email and password and your application will display a form with the current information on the database. The user will be able to update the information through that form. See the Sample section for the form and web page contents to use.
    - **administrative** - Allows a system administrator to display information about applicants. The administrator can select which fields (e.g., name, gpa, etc.) to display, which field to sort the data by, and an optional filter condition (e.g., gpa > 3.0). The administrative task is password protected using authentication via the header function; use "main" as user name and "terps" as password. Your user name and password information must be encrypted using php's **crypt** function. See the Sample section for the form and web page contents to use.
- You can assume every applicant has a unique e-mail address.
- At least one of your scripts must be a self-referencing one.
- Every page displayed (except main menu) must have a button that allows you to return to the main menu.
- Make sure you remove any leading or trailing spaces from text fields.
- In the update and submit application tasks your application must verify that the provided password matches the password verification. If that is not the case, your application must display a web page with the message "Password and password verification values do not match."
- In the review and update application tasks your application must generate the message "No

entry exists in the database for the specified email and password" if an invalid email/password combination is provided.
- You must store the password value as an encrypted value in the database.
- In the main menu use your e-mail address and **NOT nelson@cs.umd.edu** as the administrator's email address. Use href="mailto:your address" to define the e-mail address.
- For the administrative task you must use drop-down select boxes to select the fields to display (multiple fields can be selected) and the sorting field (only one field will be selected).
- The filter condition of the administrative task corresponds to the predicate to use with the where clause of an SQL query.
- Use "order by" to display fields according to the sort field.
- Links to the images on the main menu are umdLogo.gif, testudo.jpg.
- You should define at least two functions in your application (in any file).
- The formatting associated with the forms and web pages should resemble the sample output format (but it does not need to be exact). For example, ignore the yellow background color you see for some of the fields. Feel free to decorate the interface :)
- Your application must have the same set of page transitions we have in the sample output.
- Please do not add any extra functionality to the application.
- You do not need to provide comments.
- You cannot use form hidden fields.
- You do not need to validate your html forms.
- As with all programming assignments:
  - You must use meaningful variable names.
  - You must use good indentation.

Sample

The following is a sample run of the application. Remember that this is not the only scenario your application is expected to handle.

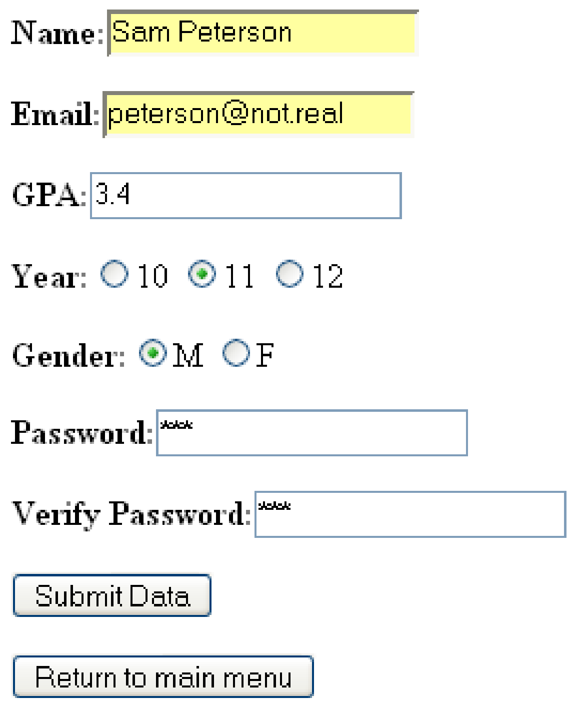
Application started (Main Menu)



Welcome to the UMCP Application System

If you have any questions about our program, please contact the system administrator at nelson@cs.umd.edu

After selecting "Submit Application" and providing some information



Name: Sam Peterson

Email: peterson@not.real

GPA: 3.4

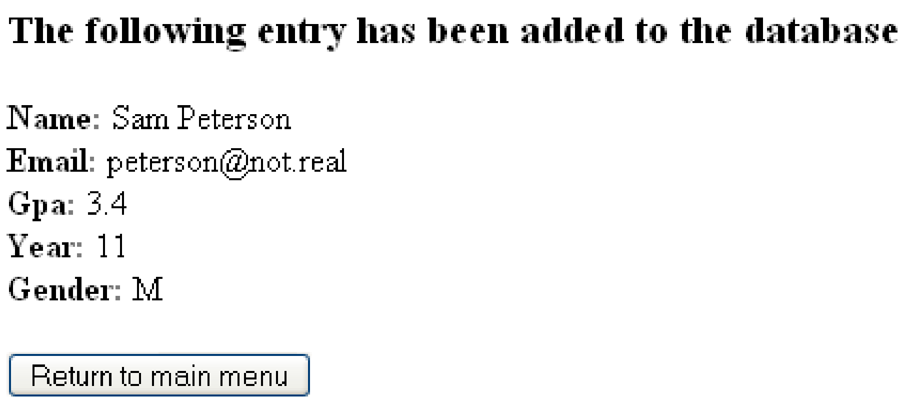
Year: 10 [✓ 11] 12

Gender: [✓ M] F

Password: ***

Verify Password: ***

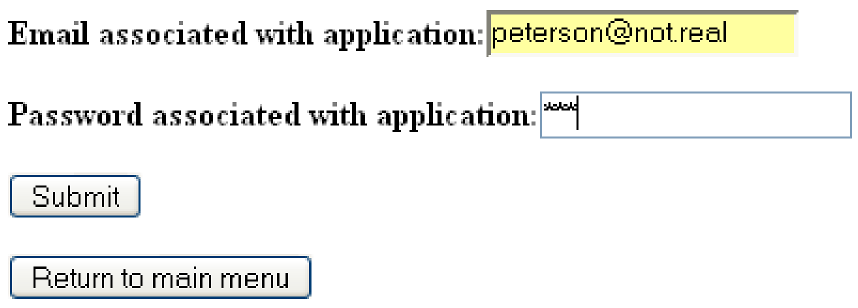
After selecting "Submit Data"



The following entry has been added to the database

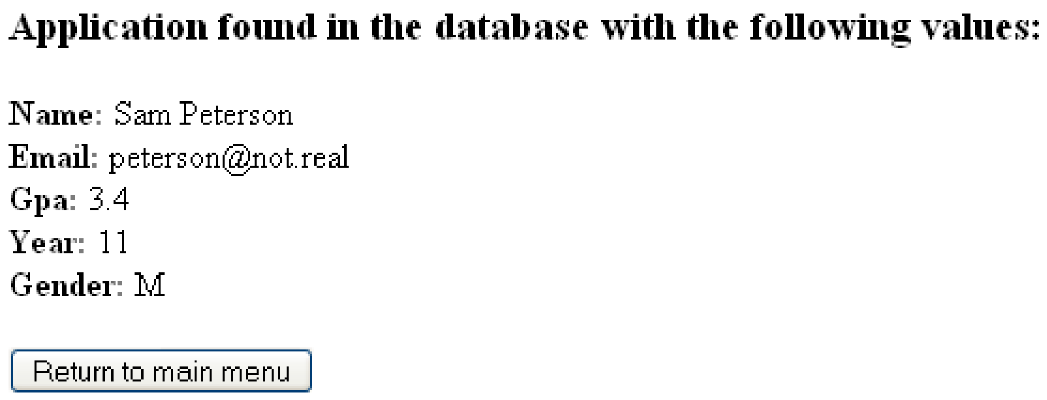
Name: Sam Peterson
Email: peterson@not.real
Gpa: 3.4
Year: 11
Gender: M

After returning to the main menu, selecting "Review Application", and entering data


Email associated with application: peterson@not.real

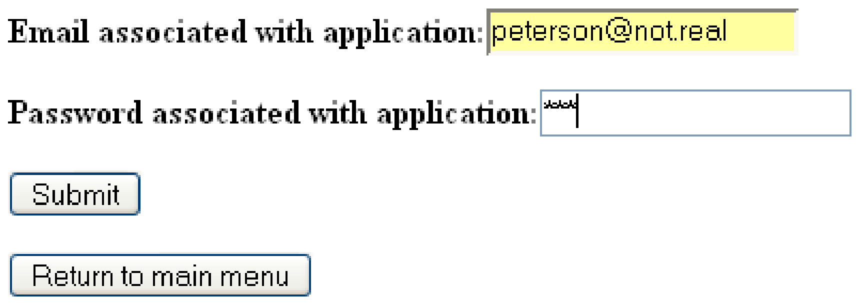
Password associated with application: ***

After selecting "Submit"


Application found in the database with the following values:

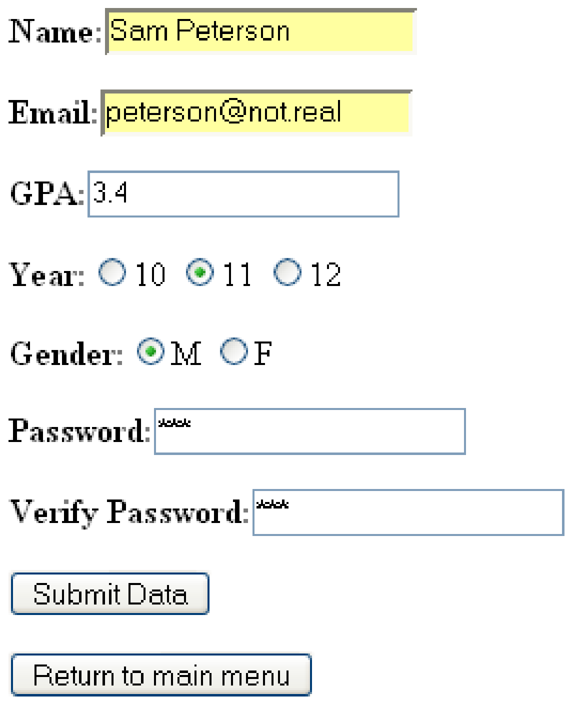
Name: Sam Peterson
Email: peterson@not.real
Gpa: 3.4
Year: 11
Gender: M

After returning to the main menu, selecting "Update Application", and entering data


Email associated with application: peterson@not.real

Password associated with application: ***

After selecting "Submit"



Name: Sam Peterson

Email: peterson@not.real

GPA: 3.4

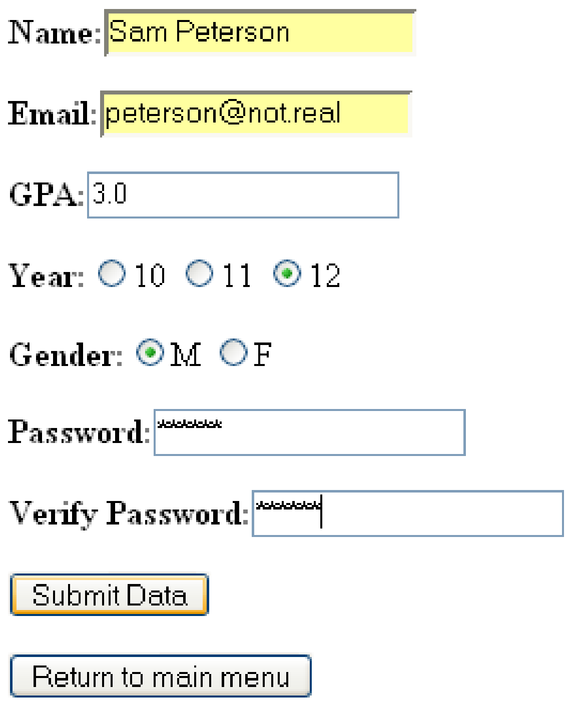
Year: 10 [✓ 11] 12

Gender: [✓ M] F

Password: ***

Verify Password: ***

After editing some of the data



Name: Sam Peterson

Email: peterson@not.real

GPA: 3.0

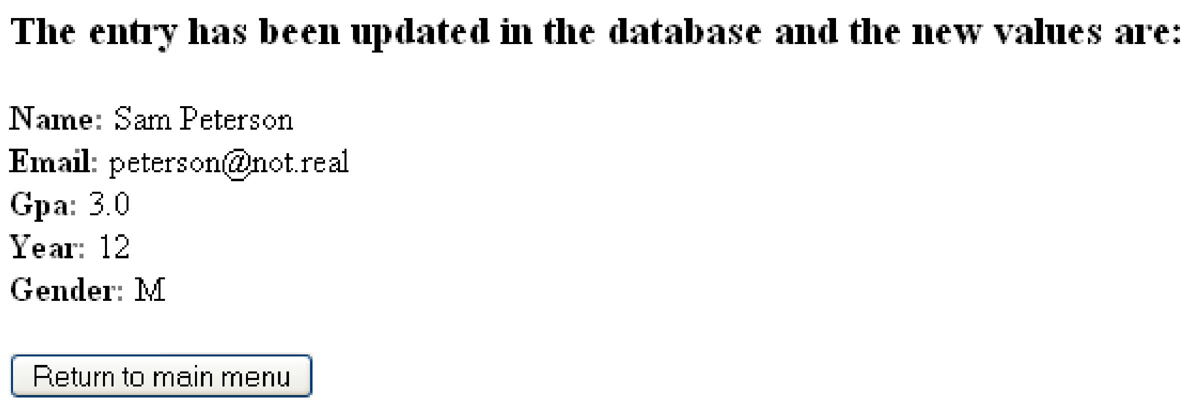
Year: 10 11 [✓ 12]

Gender: [✓ M] F

Password: ******

Verify Password: ******

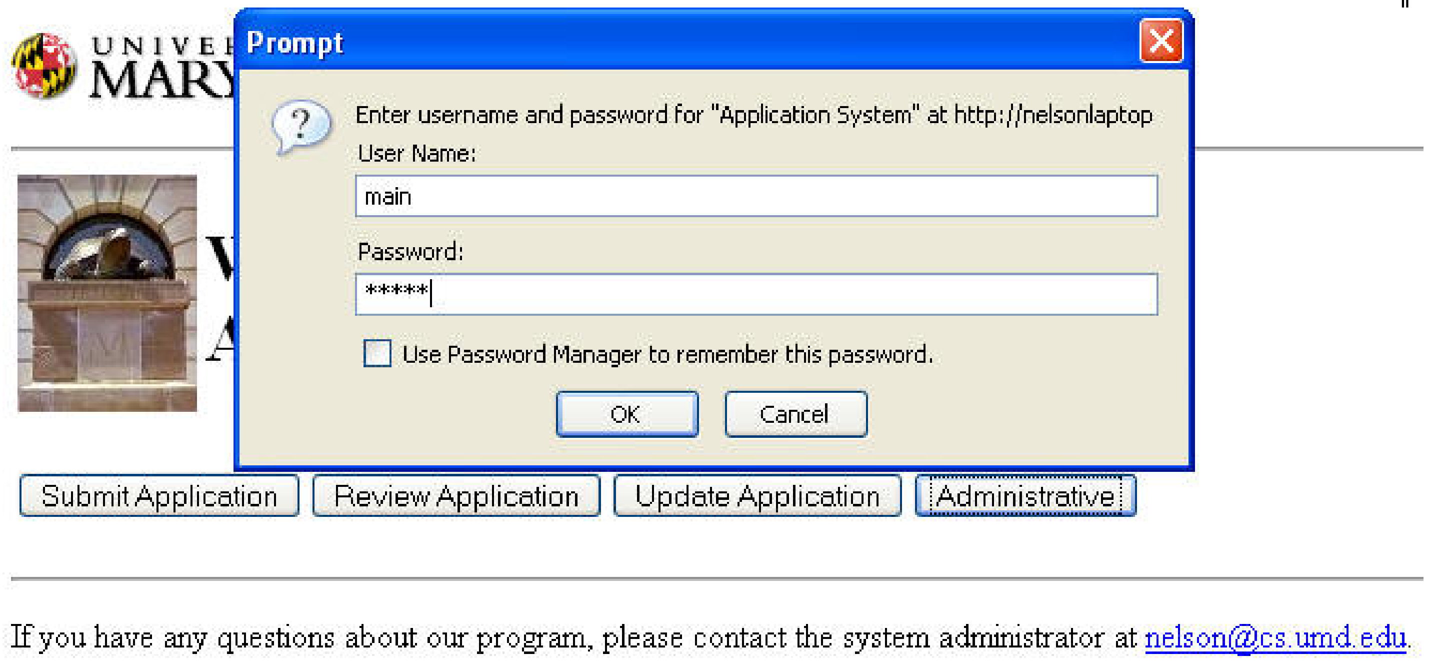
After selecting "Submit Data"



The entry has been updated in the database and the new values are:

Name: Sam Peterson
Email: peterson@not.real
Gpa: 3.0
Year: 12
Gender: M

After returning to the main menu, selecting "Administrative", and entering user name and password



Prompt

Enter username and password for "Application System" at http://nelsonlaptop

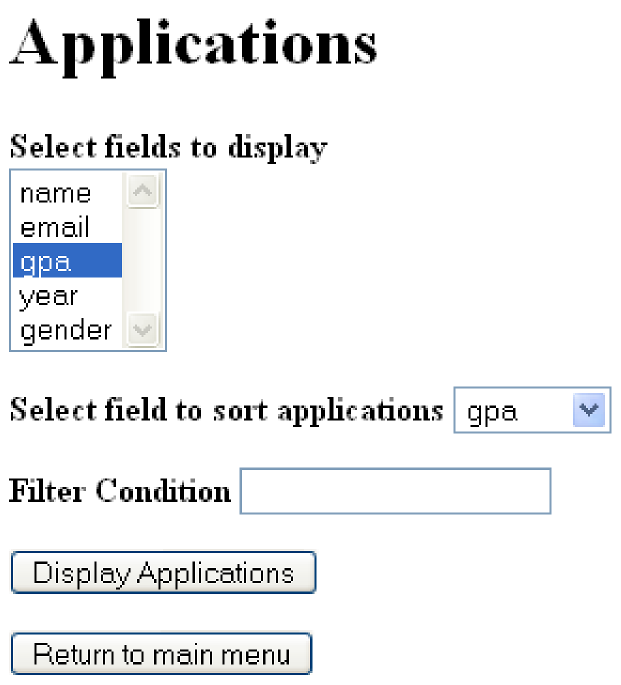
User Name: main

Password: *****

☐ Use Password Manager to remember this password.

If you have any questions about our program, please contact the system administrator at nelson@cs.umd.edu.

After selecting OK



Applications

Select fields to display

name
email
[✓ gpa]
year
gender

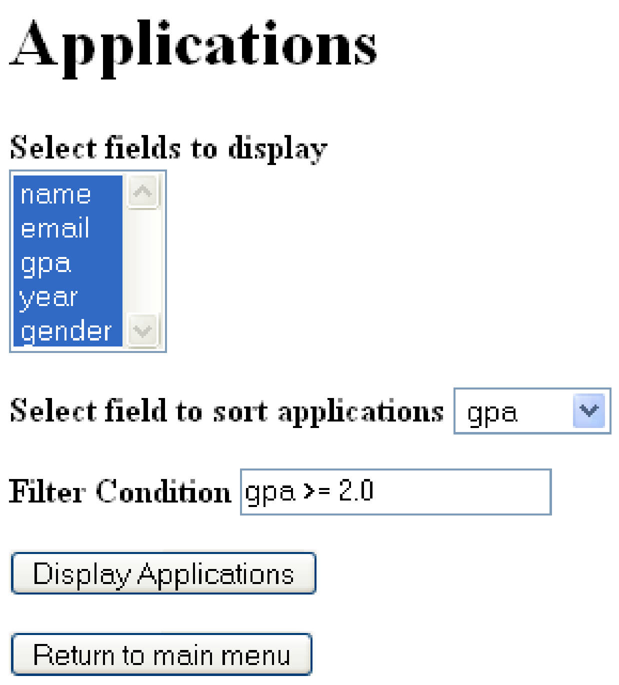
Select field to sort applications gpa

Filter Condition

Display Applications

Return to main menu

After selecting all fields to display, selecting gpa for sorting and entering gpa>=2.0



Applications

Select fields to display

[✓ name]
[✓ email]
[✓ gpa]
[✓ year]
[✓ gender]

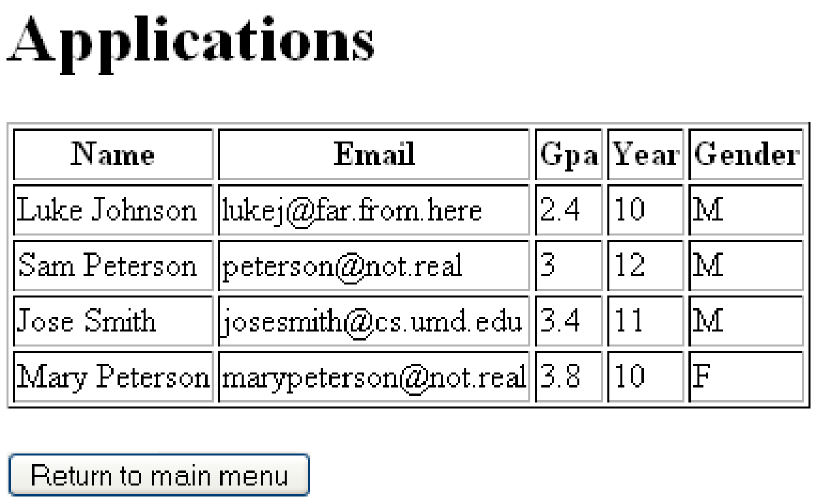
Select field to sort applications gpa

Filter Condition gpa >= 2.0

Display Applications

Return to main menu

After selecting "Display Applications"



Applications

Name	Email	Gpa	Year	Gender
Luke Johnson	lukej@far.from.here	2.4	10	M
Sam Peterson	peterson@not.real	3	12	M
Jose Smith	josesmith@cs.umd.edu	3.4	11	M
Mary Peterson	marypeterson@not.real	3.8	10	F

Submission

To submit your project, follow these steps:

1. Create a zip file that includes all your files.
2. Upload the zip file you created in the previous step using the submit server available at: https://submit.cs.umd.edu/summer2015. Make sure you select the submit server entry (Project #4) that corresponds to this project.
3. After submitting your project, make sure you download from the submit server the submitted file and verify that what you have submitted is correct.

Academic Integrity

Please make sure you read the academic integrity section of the syllabus so you understand what is permissible in our programming projects. We want to remind you that we check your project against other students' projects and any case of academic dishonesty will be referred to the University's Office of Student Conduct.