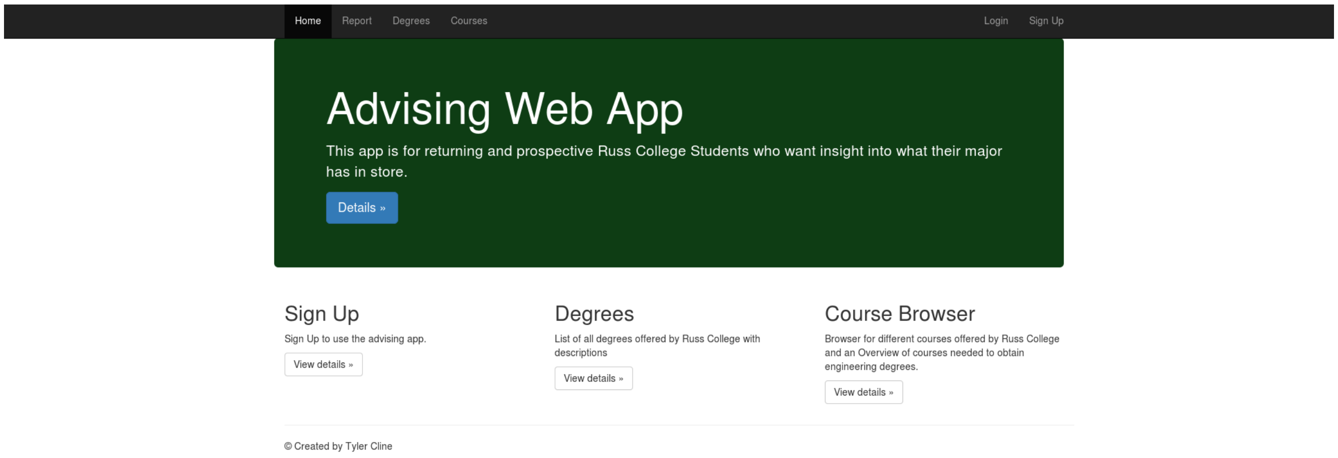
CS 4900
Final Document
Tyler Cline
Ohio University 2016

New Wireframes

home.html



Advising Web App

This app is for returning and prospective Russ College Students who want insight into what their major has in store.

Sign Up

Sign Up to use the advising app.

Degrees

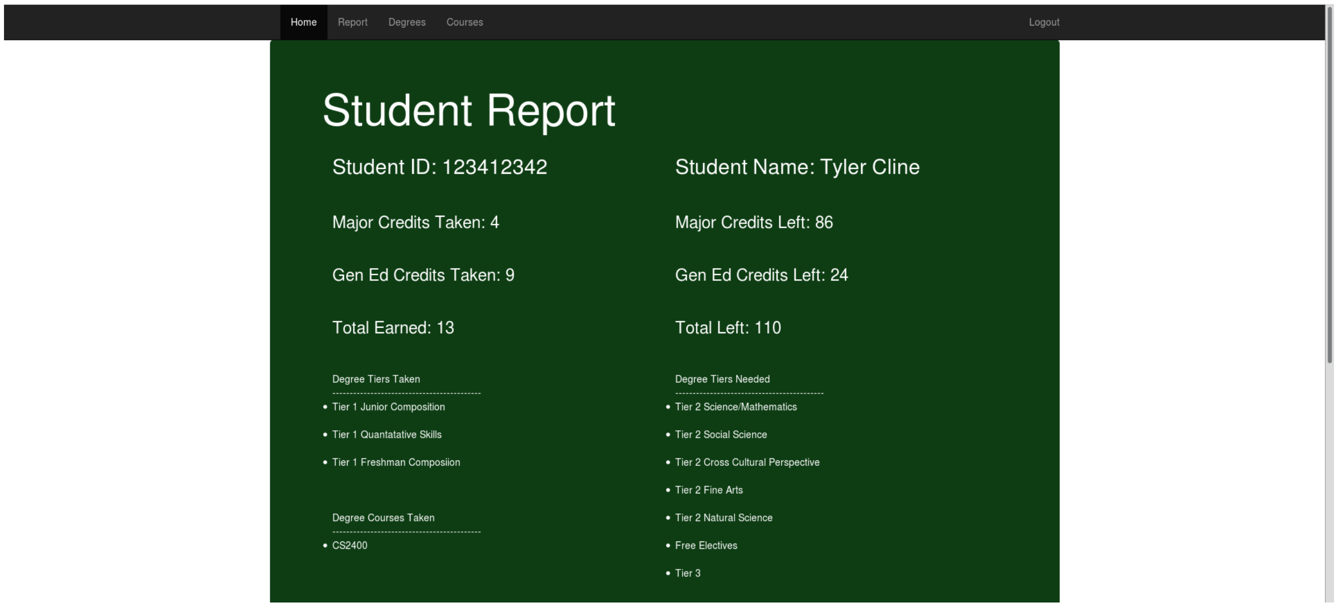
List of all degrees offered by Russ College with descriptions

Course Browser

Browser for different courses offered by Russ College and an Overview of courses needed to obtain engineering degrees.

© Created by Tyler Cline

report.html



Student Report

Student ID: 123412342	Student Name: Tyler Cline
Major Credits Taken: 4	Major Credits Left: 86
Gen Ed Credits Taken: 9	Gen Ed Credits Left: 24
Total Earned: 13	Total Left: 110

Degree Tiers Taken

- Tier 1 Junior Composition
- Tier 1 Quantatative Skills
- Tier 1 Freshman Composiion

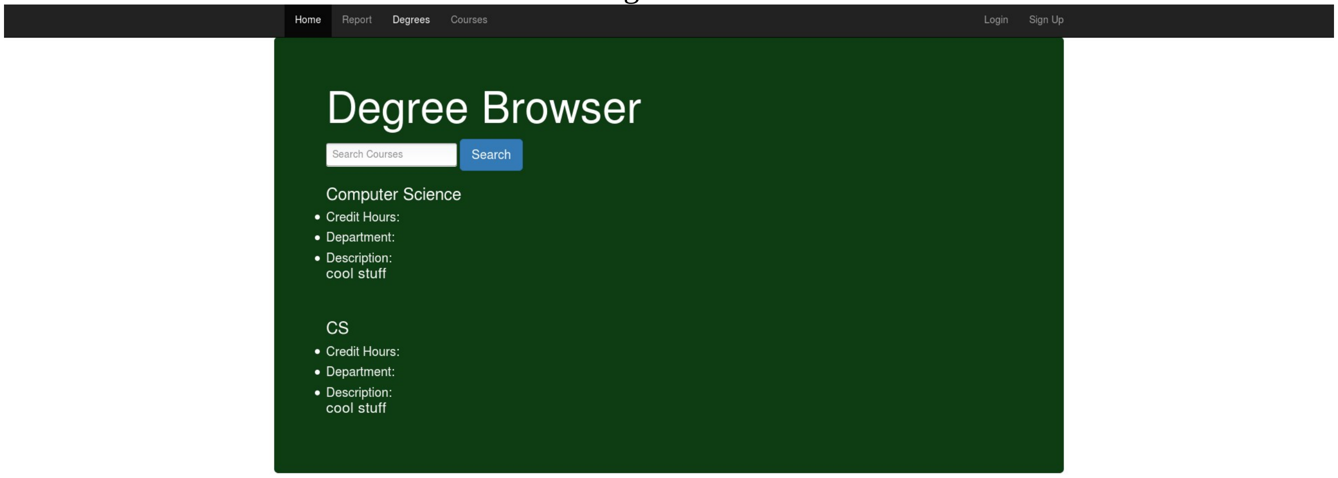
Degree Courses Taken

- CS2400

Degree Tiers Needed

- Tier 2 Science/Mathematics
- Tier 2 Social Science
- Tier 2 Cross Cultural Perspective
- Tier 2 Fine Arts
- Tier 2 Natural Science
- Free Electives
- Tier 3

degree.html

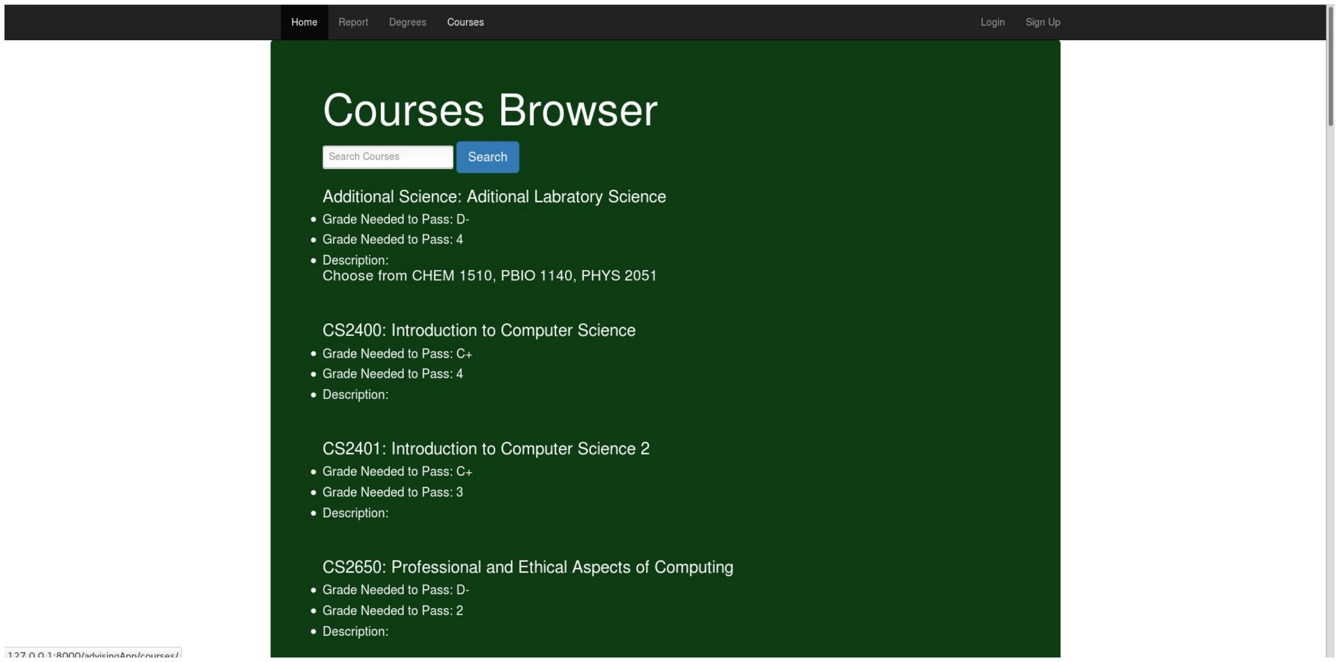


Degree Browser

Computer Science
- Credit Hours:
- Department:
- Description:
cool stuff

CS
- Credit Hours:
- Department:
- Description:
cool stuff

course.html



Courses Browser

Additional Science: Aditional Labratory Science

- Grade Needed to Pass: D-
- Grade Needed to Pass: 4
- Description:
  Choose from CHEM 1510, PBIO 1140, PHYS 2051

CS2400: Introduction to Computer Science

- Grade Needed to Pass: C+
- Grade Needed to Pass: 4
- Description:

CS2401: Introduction to Computer Science 2

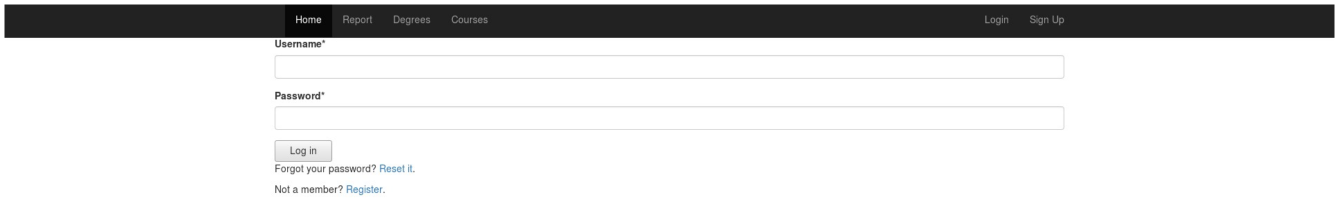
- Grade Needed to Pass: C+
- Grade Needed to Pass: 3
- Description:

CS2650: Professional and Ethical Aspects of Computing

- Grade Needed to Pass: D-
- Grade Needed to Pass: 2
- Description:

127.0.0.1:8000/advisingApp/courses/

login.html



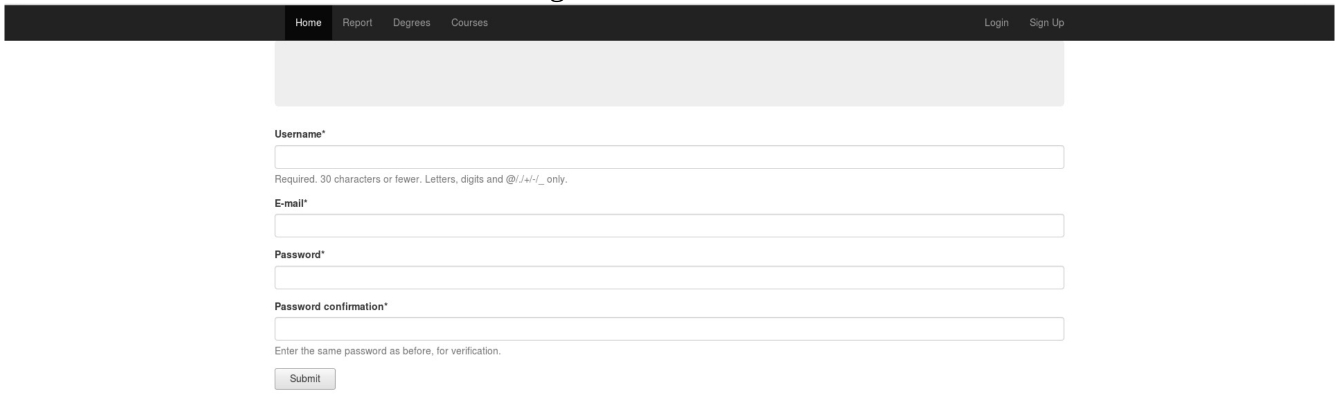
Username*

Password*

Forgot your password? Reset it.

Not a member? Register.

registration_form.html



Username*

Required. 30 characters or fewer. Letters, digits and @/./+/-/_ only.

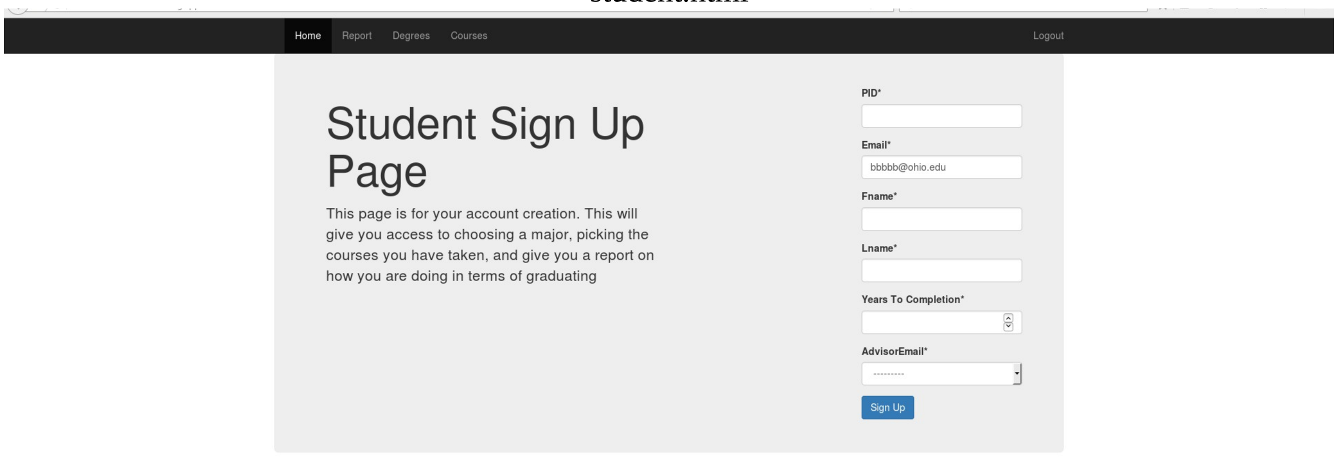
E-mail*

Password*

Password confirmation*

Enter the same password as before, for verification.

student.html


Home Report Degrees Courses Logout

Student Sign Up Page

This page is for your account creation. This will give you access to choosing a major, picking the courses you have taken, and give you a report on how you are doing in terms of graduating

PID*

Email*
bbbbb@ohio.edu

Fname*

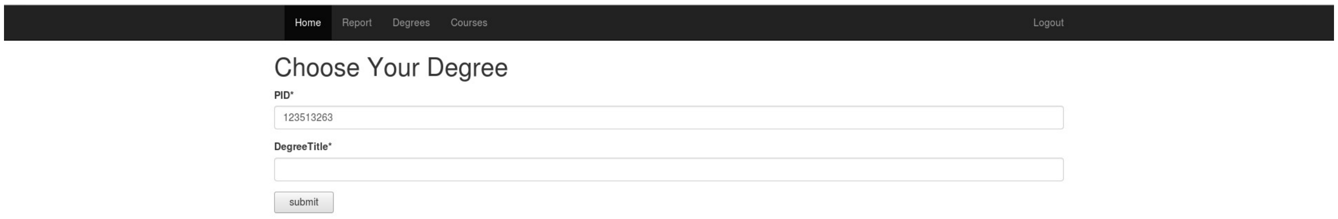
Lname*

Years To Completion*

AdvisorEmail*

Sign Up

chooses.html


Home Report Degrees Courses Logout

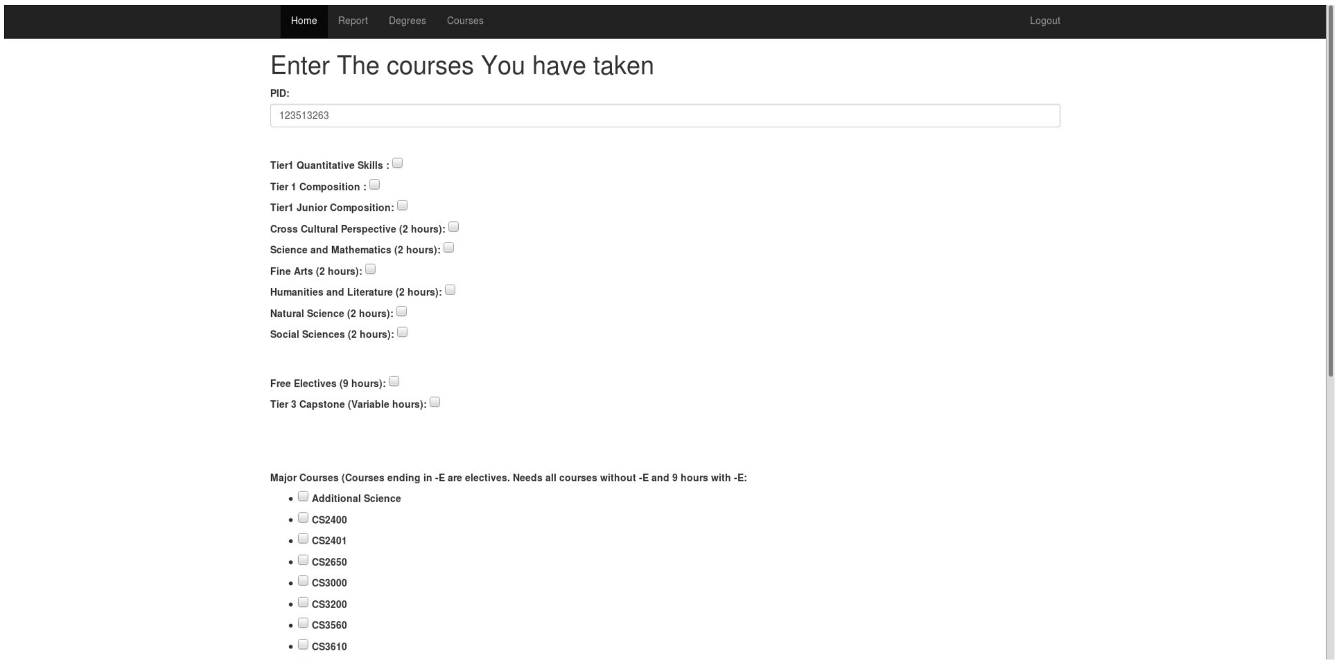
Choose Your Degree

PID*
123513263

DegreeTitle*

submit

takes.html


Home Report Degrees Courses Logout

Enter The courses You have taken

PID:
123513263

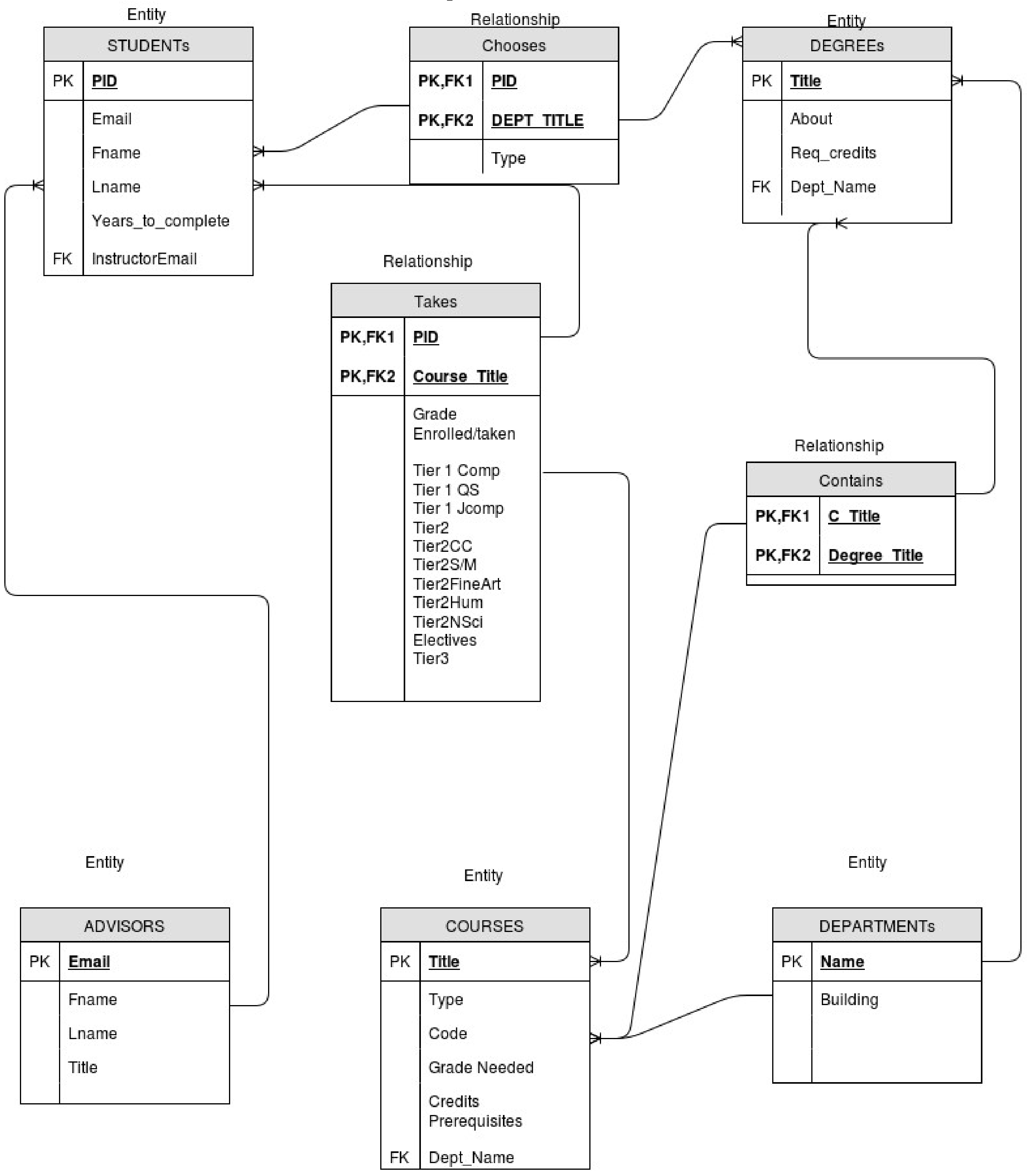
Tier1 Quantitative Skills : ☐
Tier 1 Composition : ☐
Tier1 Junior Composition: ☐
Cross Cultural Perspective (2 hours): ☐
Science and Mathematics (2 hours): ☐
Fine Arts (2 hours): ☐
Humanities and Literature (2 hours): ☐
Natural Science (2 hours): ☐
Social Sciences (2 hours): ☐

Free Electives (9 hours): ☐
Tier 3 Capstone (Variable hours): ☐

Major Courses (Courses ending in -E are electives. Needs all courses without -E and 9 hours with -E:

- ☐ Additional Science
- ☐ CS2400
- ☐ CS2401
- ☐ CS2650
- ☐ CS3000
- ☐ CS3200
- ☐ CS3560
- ☐ CS3610

Updated Schema



Extra Technologies Used

1. django registration redux: This is a method of rapid deployment of registration for the app.
2. django crispy forms: This is a method to make forms look more visually pleasing
3. Bootstrap 3: Where I got the tamplates for the project
4. Sublime Text 3: Text editor
5.

Relevant Python Files

Each file is heavily commented. This is a general explanation of why each file exists in the project. Among the comments are explanations of the code used and why you would prefer using or have to use that bit of code.

advising directory: Additions that I made to default django project files. Each file is heavily commented. This is a general explanation of why each file exists in the project. The more in depth explanation is within the comments in the file.

- **settings.py:** Changes include updating EMAIL_HOST and associated variables. This will need to be updated for whomever takes over the project to a different working email and host. INSTALLED_APPS was altered to hold the app created (advisingApp), registration, and a third party framework for forms called crispy_forms. For more information see comments in the code. Of course, the default database was changed to mysql with my root password. This will need to be updated for your credentials for your mysql verson. STATIC_URL, STATIC_ROOT, and STATICFILES_DIRS were altered to find static files in the project. This doesn't need changed unless you move the static files.

- **urls.py:** The url pattern for the advising app and django registration redux was added. There is a separate urls.py in the advisingApp project directory. This url pattern just matches that url and the urls within the urls.py in the advisingApp project are used.

AdvisingApp Directory: All app project files

- **models.py:** This is where the database schema was implemented. Each class corresponds to a table within the database schema and each attribute of the class (also called field) has a field type. This part creates the database for you.

- **admin.py:** This is where I customized the admin page for easy administration. The models are imported in from the top and a class is made for each model. The class is used for allowing for the admin page to display the data in the models, filter the data, data input, and more.

- **apps.py**: This is generated by default when a new app is started from the command line.

- **forms.py:** This is where the input forms for the templates(html files) are coded. This part could be done through raw html, but django has a framework for easily creating forms for your templates. Forms for the degree browser, course browser, takes and take_update page, student sign up page, and the degree choice page.

-g.py: This file is used to store global variables. The only global within the project holds the current users email address to be used within forms.py to be used in a database query to find the users PID.

- urls.py: This is what the urls.py in the advising project uses to find the urls for the web pages. The student report, student sign up, degree browser, course browser, takes, degree choice, base (home page), and the takes update page urls are defined here and given a name to be used in redirecting and creating links in templates. Since they are named, the programmer need only specify the name of the url in creating links and such.

- views.py: This file is used to render the templates and forms in forms.py. This is where you will put a lot of the project logic and queries, like the calculations made in the report generation. Once all calculations and queries are made, you save the results in a context, which is a dictionary containing multiple values. These values can be used/rendered on the template through this context.

StaticFiles Directory: This is where all of the javascript, css, and image files are stored. The project uses this directory through settings in settings.py. I spoke briefly about this when speaking about the settings.py file. You need not worry about changing this until you deploy the code to a server.

Templates Directory: Holds all html files for the project.
- **base.html**: base html document. This is what the rest of the template files inherit from to maintain continuity in how the project looks. This file is broken into 3 blocks. A style block for inline css not included in staticfiles, a jumbotron block for the big green jumbotron that can be seen in the wireframes, and a content block that can be seen in the home page. The content block is for content under the jumbotron.

- chooses.html: This is used to display the chooses form. It utilizes crispy forms.

- courses.html: This page displays the courses in the database. This also has html to add searching capabilites, so someone can browse courses. The courses are displayed by looping through the context and grabbing the title and such.

- **degree.html:** Logically equivalent to coures.html. Uses the same code except searches and renders degrees.

- **head_css.html:** This holds the links to the bootstrap stylesheets. To use this in other templates, just put {% include head_css.html %}.

- **home.html:** Home page for the project.

- **javascript.html:** Stores the links for bootstrap's javascript files. To use this in other templates, just put {% include javascript.html %}.

- **navbar.html:** Stores all code to create the navbar seen on all web pages.

- **report.html:** Code for the generation of a student report. This file checks wheather the student is in the database or not (if he/she is a user). If they are, it utilizes the logic generated in the view, follows some if statements to see if boolean values are true or not, then iterates through the manytomany field and grabs the coures title of courses already taken. Then, another column is generated telling the student what he/she still needs to take following the same method.

- student.html: used to render the student sign up form from forms.py. Utilizes django crispy forms.

- **takes.html:** Used to render the takes form from forms.py. Unlike the other templates, this form is rendered manually by explicitly stating each variable in the takes model. A very thourough explanation of what I did can be found here: https://docs.djangoproject.com/en/1.10/topics/forms/

- **takes_update.html:** This is essentially the same file as takes.html except the database is queried for the current user and data is prefilled based on what the user has already taken.

Naming Conventions for Data

In order for the apps elements to work correctly, data needs to be entered correctly, primarily just the course and degree data. The app works by taking the current users choice of degree from the chooses table by filtering the data by PID. Then, using that degree's name (computer science for example), the app filters the course table by that name, so, in this case, all courses with the attribute called degree with the name computer science will be collected. This has created a naming convention for course names like the following:

<course abbreviation><number>-<degree abbreviation>

for non major courses and the following for major courses:

<major abbreviation><number>

For major electives, use the following:

<major abbreviation><number>-E

Example 1: Math course requirement for a computer science degree

MATH3200-CS

Example 2: Computer Science course requirement for computer science degree:

CS4040

Example 3: Computer Science Elective requirement for computer science degree:

CS4600-E

This convention is used because the course title is a primary key and cannot be duplicated. There is also another field that has a naming convention. The tier field:

<major abbreviation>-Major
<major abbreviation>-Elective

examples:

CS-Major
CS-Elective

Known Issues:

On the takes web page, the courses are not categorized. They are also in one long list. The courses should display the name, title, and be categorized by type.

Future Work:

1. Add capabilities to email the user once he/she has created there account with a registration profile.
2. Add login form to navbar instead of redirecting user to a login page.
3. Create some sort of method to categorize the course names in the takes form.
4. Sync the app with OU's database so students need not create a separate profile and can get courses taken from there.
5. Break the courses taken/courses need to be taken columns into separate columns, so they don't the user doesn't have to scroll down the page.
6. Add more polish to the GUI and make it more user friendly.

Deployment
(This assumes you have an instance of mysql installed on your computer and django 1.9)

1. Download source files from: https://github.com/TylerCline/Advising-App.git
or
Clone the repository:
```
git clone https://github.com/TylerCline/Advising-App.git
```

2. Create a new directory for the project and Unpack the source files (if downloaded):
```
mkdir djangoProjects && cd djangoProjects
cp /home/<user>/downloads/Advising-App-master.zip .
unzip Advising-App-master.zip
```

3. Create a database in mysql called 'advising':
```
mysql -uroot -p
CREATE DATABASE advising;
```

4. Navigate to settings.py and look for an entry that looks like:
```
DATABASES = {
 'default': {
     'ENGINE': 'django.db.backends.mysql',
     'NAME': 'advising',
     'USER': 'root',
     'PASSWORD': '1Bermuda',
     'HOST': 'localhost',
     'PORT': '3306',
  }
}
```
The highlighted areas need to be changed to accommodate your mysql installation

5. Install django crispy forms (assumes you have pip installed):
```
pip install --upgrade django-crispy-forms
```
The CRISPY_TEMPLATE_PACK variable is already set in the settings file.

6. Install django registration redux:
```
pip install django-registration-redux
```

7. Create the database. Navigate to the directory with a file called manage.py and run:
```
python manage.py makemigrations
python manage.py migrate
```

8. Create a superuser for the website:
```
python manage.py createsuperuser
```

9. Run the server:
```
python manage.py runserver
```

10. Navigate the website:
urls:

1. 127.0.0.1:8000/admin – admin login page
2. 127.0.0.1:8000/advisingApp/base – home page
3. 127.0.0.1:8000/advisingApp/report – student report page
4. 127.0.0.1:8000/advisingApp/courses – course browser
5. 127.0.0.1:8000/advisingApp/degrees – degree browser
6. 127.0.0.1:8000/advisingApp/students – student information input
7. 127.0.0.1:8000/advisingApp/Chooses – student degree choice page
8. 127.0.0.1:8000/advisingApp/Takes – student takes form page