

Final Project Report - Platypus Spring 2015

Title: Platypus

Who: Paige Alleman, Caleb Hsu, Amos Leo Kim, Thomas Wagner, Mitchell Lewis

Methodologies: Agile, Pair Programming

Project Tracker: Trello <https://trello.com/b/RJYTYbXx/platypus>

Project Plan:

The screenshot shows a Trello board titled "Platypus". The board is organized into several columns:

- Current Tasks:** Contains cards for "Finish Final Report" and "Project Final Presentation in Google Docs".
- Done:** Contains cards for "CSS dynamic resizing to accomodate various screen sizes", "Print error messages when user fails login or registration", "Fix live site errors", "Fix autoresizing class and group lists", and "Make login page layout more intuitive (out Sign Up button in a ...)".
- Key:** Contains categories: Structural, Urgent, Needs To Be Done...Eventually, and Secondary.
- Upcoming Deadlines:** Contains cards for "P9 - Final Report" (due Apr 28), "P10 - Peer Evaluations" (due Apr 28), and "P8 - Presentation" (due May 1).
- Requirements:** Contains user stories: [B-01] As accounts address the using the, [B-02] As app to be so that st, [U-01] As edit my pr for new cl, and [U-02] As search for Add a card...

VCS: GitHub https://github.com/leokim89/CSCI3308_Project1

Member Contributions:

Paige Alleman: Pair programming with Caleb for CSS and HTML, peer code review, templates (the ones I made without CSS were never used), Auto documentation

```
div id="sidebar">
  <div style="text-align: center; margin-top: 84px">
    
  </div>
</div>

div id="header">
  <div class="btn-group">
    <button type="button" class="navFont" onClick="window.location.href='/matchApp/addcourses/'">Add</button>
    <button type="button" class="navFont" onClick="window.location.href='/matchApp/logout/'">Logout</button>
  </div>
</div>

div id="classInfoBlock">
  <div id="title">Welcome!</div>
  <p class="title">I'm currently enrolled in:</p>
  <ul style="list-style-type: none; padding-left: 0;">
    <li>({ for course in course_list )>
      <p id="descourseList">{course}</p>
    </li>
  </ul>
  <button class="matchbutton" onClick="window.location.href='/matchApp/classpage'">View More</button>
</div>
</div>
```



Platypus

Welcome to Platypus!!

Username:

Password:

Submit

Sign up

What is Platypus?

Platypus is an online application designed to help students share and collaborate by aiding in project group selection! Our goal is to have Platypus be the new social media platform of academia.

The screenshot shows the Platypus 1.0 website. The layout includes:

- Header:** Features a Platypus logo and a search bar.
- Navigation Bar:** Includes links for "Main Page", "Namespaces", "Classes", and "Files".
- Content Area:** Displays "Platypus Documentation".
- Sidebar:** Shows the "Platypus" namespace under "Namespaces".
- Footer:** Includes a copyright notice: "Generated on Wed Apr 22 2015 18:13:06 by /DOCUMENTATION.godot".

Amos Leo Kim:

Matching algorithms and back-end code, much of it dealing with parsing string literals into data that could be saved to the database.

The screenshot displays a multi-monitor setup for Python development. The left monitor shows a code editor with multiple tabs for files like `returnCourseList.py`, `removeCourse.py`, `queryAndMatchSections.py`, `addCourse.py`, `queryAndMatchCourses.py`, `returnMatchesBySection.py`, and `returnMatchesBySection.py`. The right monitor shows another set of code editors for files such as `queryAndMatchCourses.py`, `returnMatchesBySection.py`, and `returnMatchesBySection.py`. A bottom dock contains icons for various tools and applications, including a browser, file manager, and system status indicators.

```
returnCourseList.py
28 def returnCourseList(student_id):
29     """ Queries the database for information about a specific student.
30     Arguments: student_id - student's id
31     Returns the course list for that student"""
32     queried_student = Student.objects.get(user__username=student_id)
33     course_array = str(queried_student.course_list).split(',')
34     print(course_array)
35
36     return_array = []
37     for course in course_array:
38         query = Course.objects.get(class_id=course)
39         if query:
40             queried_course = query
41             dept_id = str(queried_course.course_title.dept_id)
42             course_number = str(queried_course.course_title.course_number)
43             print(course_number)
44             course_title = str(queried_course.course_title)
45             return_array.append(dept_id+course_number + " "+course_title)
46
47     return return_array

removeCourse.py
33 def removeCourse(student_id, section_id):
34     """ Remove a course from a students course list.
35     Arguments: student_id - student's id
36     Returns nothing
37     """
38     queried_student = Student.objects.get(user__username=student_id)
39     course_list = str(queried_student.course_list)
40
41     course_array = course_list.split(',')
42     if section_id in course_array:
43         course_array.remove(section_id)
44
45     else:
46         print("Course is not in list of registered sections")
47
48     course_list = ""
49     for course in course_array:
50         if course_list == "":
51             course_list += str(course)
52         else:
53             course_list += ","+str(course)
54
55     student.course_list = course_list
56     student.save()

queryAndMatchSections.py
23
24 def queryAndMatchSections():
25     """ Function that takes section numbers as keys and arrays
26     of student objects as values that saves section numbers as keys
27     and student objects as values that are enrolled in that section
28     Returns a dictionary of the sections.
29
30     """
31     #Create a hashtable that will save sections numbers as keys
32     sectionsDict = {}
33
34     #Query all rows in Section table
35     #For each section, create an entry with the section number
36     for section in Section.objects.all():
37         sectionString = str(section).strip()
38         sectionsDict[sectionString] = []
39
40     #Query all students in Student table
41     #Concatenate the SID and course_list attributes of each
42     #Save course-separated entries of the concatenated string
43     #For every student that is enrolled in, add the SID
44     for student in Student.objects.all():
45         sid = str(student).strip()
46         courseString = str(student.course_list).strip()
47         concatenated = sid + ":" + courseString
48         for index in range(1, len(row)):
49             sectionsDict[row[index]].append(row[0])
50
51     return sectionsDict

addCourse.py
33 def addCourse(student_id, section_id):
34     """Function to add a course to be available on Platypus.
35     Arguments: student_id - student's id
36     Returns nothing
37     Saves course to database.
38     """
39     queried_student = Student.objects.get(user__username=student_id)
40     course_list = str(queried_student.course_list)
41
42     course_array = course_list.split(',')
43     if section_id in course_array:
44         print("Section already in list of enrolled courses"
45     else:
46         course_array.append(section_id)
47
48     course_list = ""
49     for course in course_array:
50         if course_list == "":
51             course_list += str(course)
52         else:
53             course_list += ","+str(course)
54
55     student.course_list = course_list
56     student.save()

queryAndMatchCourses.py
25 def queryAndMatchCourses():
26     """ Function that saves course numbers as keys and arrays
27     of student objects as values that are taking the same course
28     Returns all sections in the database.
29     """
30     #Query all sections from course_list string into an array.
31     #Save each section number from course_list string into an array.
32     #Extract department ID and first four chars of section id.
33     #Create the course dictionary.
34
35     #Create a hashtable that will save course numbers as keys and array
36     courseDict = {}
37
38     #Query all sections in the database;
39     #Extract department ID and first four chars of section id;
40     #Save each section number in the array;
41     #Concatenate the two values coercing as strings and slicing first four characters of section id as key
42     for section in Section.objects.all():
43         deptID = str(section.course_title.dept_id).strip()
44         sectionString = str(section).strip()[0:4]
45
46         courseDict[deptID+sectionString] = []
47
48     #Query all students in the database;
49     #Save every section number from course_list string into an array;
50     #For each section number in the array:
51     #Query database for department ID for section with that number;
52     #Concatenate the two values coercing as strings and slicing first four characters of section id as key;
53     #Append student to courseDict[deptID+sectionString], and append student to courseDict[deptID]
54     for student in Student.objects.all():
55         courseString = str(student.course_list).strip()
56         courseArray = courseString.split(',')
57         for course in courseArray:
58             classID = Section.objects.get(class_id=course)
59             deptID = classID.course_title.dept_id
60             courseId = str(deptID)+str(classID)[0:4]
61
62             courseDict[courseId].append(str(student))
63
64     return courseDict

returnMatchesBySection.py
31 def returnMatchesBySection(student_id):
32     """ Find a list of students that are taking the same course
33     Arguments: student_id - student's id
34     Returns dictionary of matching students. """
35     courseList = returnCourseList(student_id)
36
37     course_dict = queryAndMatchCourses()
38
39     return_dict = {}
40
41     for course in courseList:
42         if student_id in course_dict[course]:
43             course_dict[course].remove(student_id)
44             return_dict[course] = course_dict[course]
45
46     #print return_dict
47
48     #Key: course; value: student
49     return return_dict

returnMatchesBySection.py
31 def returnMatchesBySection(student_id):
32     """ Find a list of students that are taking the same course
33     Arguments: student_id - student's id
34     Returns dictionary of matching students. """
35     courseList = returnCourseList(student_id)
36
37     course_dict = queryAndMatchCourses()
38
39     return_dict = {}
40
41     for course in courseList:
42         if student_id in course_dict[course]:
43             course_dict[course].remove(student_id)
44             return_dict[course] = course_dict[course]
45
46     #print return_dict
47
48     #Key: course; value: student
49     return return_dict

returnStudentData.py
27 def returnStudentData(student_id):
28     """Find information associated with a student user.
29     Arguments: student_id - student's id
30     Returns the attributes of the student user. """
31     queried_student = Student.objects.get(user__username=student_id)
32     username = queried_student.user.username
33     name = queried_student.user.first_name + " " + str(queried_student.user.last_name)
34     email = queried_student.user.email
35
36     print("User ID: " + str(student_id))
37     print("Name: " + name)
38     print("Email: " + email + "\n")
39
40     return queried_student

INSERT MODE, Line 53, Column 1
Tab Size: 4
Py

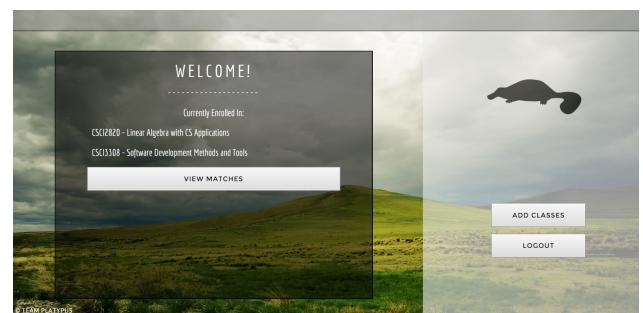
INSERT MODE, Line 53, Column 1
Tab Size: 4
Py

INSERT MODE, Line 48, Column 1
Tab Size: 4
Py

INSERT MODE, Line 22, Column 27
Tab Size: 4
Python
```

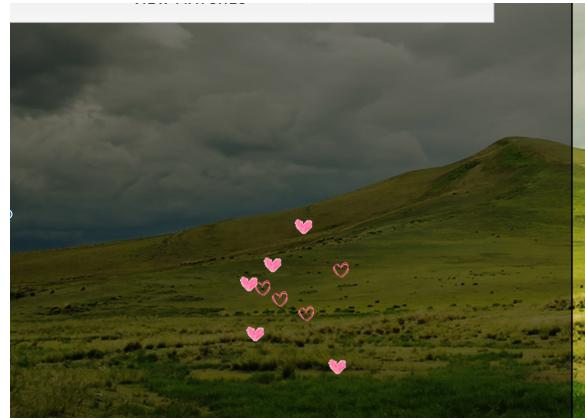
Caleb Hsu:

Most of the HTML/CSS for the site, page planning, navigation, login, and registration, peer code review



Thomas Wagner: Hearts following cursor, Some CSS (made site neat),

```
51     width: 80%;  
52     margin-left: 10%;  
53     height: 50px;  
54     color: white;  
55     margin-top: 15px;  
56 }  
57  
58 .classbutton {  
59     width: 100%;  
60     height: 30px;  
61     margin-left: 30%;  
62     color: white;  
63     margin-top: 2%;  
64 }  
65  
66 .sectionbutton {  
67     width: 300%;  
68     height: 30px;  
69     margin-left: 220%;  
70     margin-top: 10%;  
71 }  
72  
73 #back {  
74     width: 100px;  
75     bottom: 500px;  
76     right: 200px;  
77     position: fixed;  
78 }  
79  
80 #classnav {  
81     width: 0%;  
82     top: 38%;  
83     left: 23.3%;  
84     position: fixed;  
85     z-index: 1;
```



Mitchell Lewis: Worked with the database and ensured that it paired with the rest of the code. Managed login authentication functionality.

```
    self.assertEqual(response.status_code, 200, "Status code was not 200")
    self.assertContains(response.content, "Hello, World!", "Content did not contain 'Hello, World!'")

def test_register(self):
    response = self.client.post('/register', {
        'username': 'testuser',
        'password': 'testpassword',
        'email': 'testuser@example.com'
    })
    self.assertEqual(response.status_code, 200, "Status code was not 200")
    self.assertContains(response.content, "User registered successfully", "Content did not contain 'User registered successfully'")

def test_login(self):
    self.client.post('/register', {
        'username': 'testuser',
        'password': 'testpassword',
        'email': 'testuser@example.com'
    })
    response = self.client.post('/login', {
        'username': 'testuser',
        'password': 'testpassword'
    })
    self.assertContains(response.content, "User logged in successfully", "Content did not contain 'User logged in successfully'")
```

A screenshot of a software interface showing search results for course and section matches. The left panel displays 'COURSE MATCHES' with two entries: 'Section 330001' and 'Section 330034'. The right panel displays 'SECTION MATCHES' with two entries: 'Section 330001' and 'Section 330034'. Both panels show details like 'Matches:', 'Page', and 'PDF'. The background features a scenic landscape with rolling green hills under a cloudy sky.

Deployment: Python Anywhere <http://leokim89.pythonanywhere.com/matchApp/>

Differences: Over the course of the semester, we realized that many of the things we wanted to include in our web application would not be possible with our experience or time available. Some adjustments to our aspirations included

- limiting the available classes to just CS classes at CU Boulder
- decreasing social media aspects like a personalized profile page and built-in chat
- scrapping helper tools like an auto-scheduler for meetings

which we realized would be too difficult and time consuming to implement. However, our intended functionality remains the same: users are sorted according to class and section, then matched into groups. Also, we initially elected to pursue an Agile method of development with some elements of Iterfall, but in reality, over time our development method became simply Agile because once the project got going, some of the more structured aspects of Iterfall became unnecessary. Early on, we planned for weekly meetings where we would discuss what we did, what needed to be done, what we would work on as individuals, and any roadblocks we were facing. This was extremely helpful throughout development and really cemented the importance of the Agile methodology.