SW Engineering CSC648/848 Spring 2020
Milestone 04

CsGetDegrees.com

Section 02

Team 04

Team Members:
1. Team lead: Anh Le - email: ale4@mail.sfsu.edu
2. Front end lead: Mohamad Farah
3. Front end developer and Github Master: YeeJian(Jeans) Tan
4. Back end lead: Aramis Knox
5. Back end developer: Timothy Wells
6. Back end developer: Russell Azucenas

History Table

Date	Details
4/28/2020	Submitted Milestone 4
TBD	Revised Milestone 4

1.Product Summary: URL http://34.94.123.246/

CsGetDegrees is a website designed to meet the expectations of an e-commerce website. CsGetDegrees focuses mainly on San Francisco State students. Every student here at San Francisco State will have full access to our e-commerce website. The functions students would have are as follows.

- Users on CsGetDegrees will be verified, so that they are approved to buy products.
- Users on CsGetDegrees are also verified to be approved to buy services such as tutoring services.
- Users who decide to sell products are verified before doing so. Users who are also selling services are approved before posting any type of service.
- System is set in a way that allows general users to become approved users via login.
- System is also set in a way that allows all users to browse through the site regardless of being logged in.
- CsGetDegrees also allows general users who may or may not be logged in to search through the site using the search bar.
- CsGetDegrees has the ability for approved users to send messages to sellers through their listings.
- Users, who are approved users, that have received the message have the ability to reply to the messages that are received.
- Approved users who initiated conversation shall be able to view the conversation.
- Admin in CsGetDegrees have the ability to add departments to the database. Admin have also the ability to add courses to the database as well. Admin can approve product listings.
- With the ability to approve, Admin can also reject any product that is created. Admin can remove users as well, if they deem it necessary.

The uniqueness of CsGetDegrees, starts with the ability to have services that are uniquely tailored to San Francisco State students. Services that include tutoring, babysitting and all others that are approved by the admin. The front end team will work on displaying all the features,

getting the information and passing it to the backend team. Back end team will specifically handle all the data passed by our front end team. Some of the features go hand in hand with front end implementation and design with back end data management.

2.Usability Test Plan

I. Test Objective:

For our team usability test plan, we've decided to go with our search functionality as the main subject to carry out the test. We believe this is our most important functionality within the application. The purpose of this test is to verify how effectively the search function can retrieve a certain product desired by the user. The search function is how the user will be able to see the appeal of our application. By implementing the search functionality hand in hand with a filtering system, users will be able to search for desired products as well as narrow down the result list to the perfect item. For this very reason search functionality must be rigorously tested for it's usability to the users. The test is for us developers to identify factors that can cause unpleasantness in a user's experience with their searches, such as difficult to navigate interface or inaccurate search result items. Through consistently providing users with accurate products that meet their standards along with an easy to navigate user interface, this will prove to be a major selling point of our application.

II. Test Background and setup:

Evaluator of the test will be provided with a link to the beta version of the web application. Using the link provided, users can access the testing environment through most

common web browsers such as chrome, firefox, microsoft edge, and internet explorer. Evaluators will be a representation of the personas described in the first milestone documentation, which narrows down to SFSU students that are struggling to find affordable class materials and services. Once at homepage, evaluators will perform certain assigned tasks to verify the clarity of the application's user interface. The tasks are to determine how the application can adjust to improve the user's experience upon navigating through search for their intended product.

Usability task description: Starting from the home page, proceed to search for available textbooks offered by the application within the computer science department. Narrow down further to products associated with course 648, then attempt to look for the cheapest available products. Second task, look for available notes associated with Asian American study, specifically Chinese-American, where products prices are below $50. Third task, attempt to look for tutoring services offered for any biology courses, check if any services available for less than $20/hour with higher than 3 stars rating. Fourth task, proceed to search for available babysitting services, check the service if they're available on Monday, Wednesday, and Friday. Fifth Task, look for car pool services offering to students commuting from East Bay to SFSU between 8am-10am, with available seats.

The effectiveness of the search functionality can be measured by how accurate the filter function transcribes user's needs into the desired product. For that reason, categories and departments filters are necessary to be visually clear for their intended purpose.

In order to measure the efficiency of the search functionality, evaluators are given five search tasks, with each involving a single main step of clicking the search button, then clicking twice to navigate through the categories filter, and/or navigate through the department filter..

Finally a single filter click to confirm the filter function is working properly. A successful search task on average will take about 4-6 clicks.

Lickert questions:

1. I am confident that I can find a desired product within 6 clicks of the search process.
2. I find the filter offers well defined category options of my intended search.
3. I find the filter options easy to understand and navigate through.

1-Strongly disagree 2-Disagree 3- Neither agree or disagree 4- Agree 5- Strongly agree

3. Quality Assurance Test Plan

Our Quality assurance test shall be revolving around the Search functionality of the application. The more accurate the search functionality can provide users with closely matched results to their intended product, the more likely the user will stay for future searches. This test shall be performed within the back end of the server, where search and filter algorithms are presented with string input that matches how the server would receive the user's search input, after being broken down by the front end of the application into a request.

For this test the result can be found by inserting test input in a proper format after this URL: http://34.94.123.246:5000/search/?queries=all where all is replaced with the test input. We are breaking this test into three parts, addressing three features of the search functionality: search field text input, filtering category and price range filter.

Test #	Test Title	Description	Browser	Input	Expected Result	Result
1	Test for closely match of search field	Verify that search can return matched products	Firefox	Babysitting	A List of available "Babysitting" Products	
2	Test for closely match of	Verify that search can return matched	Firefox	Notes	A list of available "Notes" products	

	search field	products				
3	Test for closely match of search field	Verify that search can return matched products	Firefox	Car Pool	A list of available "Car Pool" products	
4	Test for Price range filter	Verify that filter can return correct price range	Firefox	Min$ 0 - Max$ 50	A list of products that have prices less than or equal to 50$	
5	Test for Price range filter	Verify that filter can return correct price range	Firefox	Min$ 0 - Max$ 20	A list of products that have prices less than or equal to 20$	
6	Test for Price range filter	Verify that filter can return correct price range	Firefox	Min$ 15 - Max$ 50	A list of products that have prices between 15$ and 50$ inclusive	
7	Test for Department Filter	Verify that filter can return correct department	Firefox	Business	A list of products within "Business" department	
8	Test for Department Filter	Verify that filter can return correct department	Firefox	Science & Engineering	A list of products within "S.T.E.M" department	
9	Test for Department Filter	Verify that filter can return correct department	Firefox	Ethnic Studies	A list of products within "Ethnic" department	
10	Test for closely match of search field	Verify that search can return matched products	Chrome	Babysitting	A List of available "Babysitting" Products	
11	Test for closely match of search field	Verify that search can return matched products	Chrome	Notes	A list of available "Notes" products	
12	Test for closely match of search field	Verify that search can return matched products	Chrome	Car Pool	A list of available "Car Pool" products	
13	Test for Price range filter	Verify that filter can return correct price range	Chrome	Min$ 0 - Max$ 50	A list of products that have prices less than or equal to 50$	
14	Test for Price range filter	Verify that filter can return correct price range	Chrome	Min$ 0 - Max$ 20	A list of products that have prices less than or equal to 20$	
15	Test for Price range filter	Verify that filter can return correct price range	Chrome	Min$ 15 - Max$ 50	A list of products that have prices between 15$ and 50$ inclusive	
16	Test for Department Filter	Verify that filter can return correct department	Chrome	Business	A list of products within "Business" department	

17	Test for Department Filter	Verify that filter can return correct department	Chrome	Science & Engineering	A list of products within "S.T.E.M" department	
18	Test for Department Filter	Verify that filter can return correct department	Chrome	Ethnic Studies	A list of products within "Ethnic" department	

4. Code Review

***Below is the transcript of a code review session between Timothy Wells and Russel Azucenas.**

Hey Russ,

All and all excellent code. Python coding styles of line spacing and indentation were respected and the functions were conceptual in their implementation. Comments were made in the code and were clear in explaining code. Here are some of the changes that were made in my review which mostly have to do with homogenization of the project for the sake of readability. I took a look at your code and made the following adjustments:

- Your function for class creation has been restructured such that it is uniform with the method of user creation and cohesive with the method of receiving posts requests. This may need adjustment later depending on how it is implemented in the entry point. This function's naming convention was also homogenized
- Restructured RemoveClassByID, FetchAllClassesByDeptID, FetchClassByID, FetchClassByCourseNum, and FetchClassByCourseName such that query variables match userAPI.py naming conventions and database execution works. Return values are also assimilated to one another.
- setupConnection function was added. This is to best homogenize database access and ensure functionality
- structureIntoValidJson has replaced applyjson for the sake of function continuity
- changeclassdepartmentid function was removed for now until we can find a purpose for this function
- JSONformat was removed

Please adjust your other API to match this. I will attach the new classAPI for reference. But! Excellent code. Thank you Russel. Glad to have you on the team and I look forward to finishing this project with your expertise.

classAPI:

```
import mysql.connector
from flask import jsonify
import json
from flask import Flask
from flask import request
application = Flask(__name__)

def setupConnection():
   conn = mysql.connector.connect(
       user='aramis',
       password='CsGetDegrees648!',
       host='localhost',
       database='CsGetDegrees',
       auth_plugin='mysql_native_password')
   cursor = conn.cursor()
   return conn, cursor

# Adds a new class into the database table
def new_class_creation():
   if request.method == 'POST':
      n_course_number = request.get_data('course_number')
      n_name = request.get_data('name')
      n_desc = request.get_data('desc')
      n_department_id = request.get_data('department_id')
      new_user_insert_query = "INSERT INTO Class (course_number, name, desc, department_id) VALUES (%s, %s,
%s, %s)"
      new_user_values = (n_course_number, n_name, n_desc, n_department_id)
      conn, cursor = setupConnection()
      cursor.execute(new_user_insert_query, new_user_values)
      conn.commit()
      conn.close()
      return structureIntoValidJson([json.dumps({"class_created": True})])
   else:
      return structureIntoValidJson([json.dumps({"class_created": False})])


# Deletes an existing class from the database table
def RemoveClassByID(cid):
   class_removal_query = "DELETE FROM Class WHERE cid = \'" + cid + "\'"
   conn, cursor = setupConnection()
   cursor.execute(class_removal_query)
   conn.commit()
   conn.close
   return structureIntoValidJson([json.dumps({"class_created": True})])


# Fetch all classes that are under a specific Department
def FetchAllClassesByDeptID(did):
   class_from_dept_query = "SELECT * FROM Class USE INDEX(department_id) WHERE did = \'" + did + "\'"
   conn, cursor = setupConnection()
   cursor.execute(class_from_dept_query)
   classes_data = cursor.fetchall()
```

```
    jsonArray = []
    for element in classes_data:
        jsonArray.append(json.dumps({'cid': element[0], 'course_number': element[1], 'name': element[2], 'desc':
element[3], 'department_id': element[4]}))
    return structureIntoValidJson(jsonArray)


def FetchClassByID(cid):
    class_by_id_query = "SELECT * FROM Class WHERE cid = \'" + cid + "\'"
    conn, cursor = setupConnection()
    cursor.execute(class_by_id_query)
    class_data = cursor.fetchall()
    jsonArray = []
    for element in class_data:
        jsonArray.append(json.dumps(
            {'cid': element[0], 'course_number': element[1], 'name': element[2], 'desc': element[3],
             'department_id': element[4]}))
    return structureIntoValidJson(jsonArray)


def FetchClassByCourseNum(course_number):
    class_by_coursenum_query = "SELECT * FROM Class WHERE course_number = \'" + course_number + "\'"
    conn, cursor = setupConnection()
    cursor.execute(class_by_coursenum_query)
    class_data = cursor.fetchall()
    jsonArray = []
    for element in class_data:
        jsonArray.append(json.dumps(
            {'cid': element[0], 'course_number': element[1], 'name': element[2], 'desc': element[3],
             'department_id': element[4]}))
    return structureIntoValidJson(jsonArray)


def FetchClassByCourseName(name):
    class_by_name_query = "SELECT * FROM Class WHERE name = \'" + name + "\'"
    conn, cursor = setupConnection()
    cursor.execute(class_by_name_query)
    class_data = cursor.fetchall()
    jsonArray = []
    for element in class_data:
        jsonArray.append(json.dumps(
            {'cid': element[0], 'course_number': element[1], 'name': element[2], 'desc': element[3],
             'department_id': element[4]}))
    return structureIntoValidJson(jsonArray)

def structureIntoValidJson(jsonArray):
    jsonStructure = "{\n\"data\": ["
    for i in jsonArray:
        jsonStructure = jsonStructure + "\n"
        jsonStructure = jsonStructure + i + ", "
    jsonStructure = jsonStructure[:-2]
    jsonStructure = jsonStructure + "\n]\n}"
    return jsonStructure
```

5. Security practice

In our application, we seek to protect the users confidentiality within our system. Only bare minimum of the users information shall be accessible by other users. In order to review a user's information, you must also be an approved user, this is to ensure any information about the user is not available to the public. Data such as user's logins are specifically not allowed to be sent to the front end of the application. Functions within the back end of the application shall not be permitted such data to be included as return values. All passwords, when sent to the back end of the application, must go through an encryption and decryption function before being usable or storable to the system. Admin accounts have no power or access to a user's private information, the only purpose of admins is to manage the traffic of contents in the application. Any user input is required to be less than 40 characters or shall be considered invalid request, with few exceptions such as product's descriptions and messages.

6. Progress self-check

1. Application shall be developed, tested and deployed using tools and servers approved by Class CTO and as agreed in M0: Done

2. Application shall be optimized for standard desktop/laptop browsers e.g. must render correctly on the two latest versions of two major browsers: On Track

3. Selected application functions must render well on mobile devices: Issues: due to lack of experience with react language, the rendering on mobile app will not be rendered well or get to.

4. Data shall be stored in the team's chosen database technology on the team's deployment server: Done

5. No more than 50 concurrent users shall be accessing the application at any time: On Track

6. Privacy of users shall be protected, and all privacy policies will be appropriately communicated to the users: Done

7. The language used shall be English: Done

8. Application shall be very easy to use and intuitive. On Track

9. Google analytics shall be added: Issue: We might not be able to commit to this requirement due to time and priority constraints.

10. No email clients shall be allowed: Issue: the application will have it's own messaging system, but we are a little behind so this function might not be committed to the end product.

11. Pay functionality, if any (e.g. paying for goods and services) shall not be implemented nor simulated in UI: Done

12. Site security: basic best practices shall be applied (as covered in the class): On track

13. Modern SE processes and practices shall be used as specified in the class, including collaborative and continuous SW development: On Track

14. The website shall prominently display the following exact text on all pages "SFSU Software Engineering Project CSC 648-848, Spring 2020. For Demonstration Only" at the top of the WWW page. (Important so not to confuse this with a real application). On Track