SW Engineering CSC 648/848 Spring 2024

Section 02

Team 05

Team Members	
Team Lead	Johnny Kwon
Database Engineer	Abby Lin
Frontend Engineer	Zabiullah Niemati
Frontend Engineer	Zizo Ezzat
Github Master/Frontend	Ethan Ho
Backend Engineer	Nichan Lama
Backend Engineer	Fadee Ghiragosian

Team Lead email: hkwon4@sfsu.edu

Github Page:
https://github.com/CSC-648-SFSU/csc-648-02-spring24-team05

Date: 4/23/2024

Document History	
Date Submitted	4/23/24
Date Revised	4/30/24

1. Product Summary

1.1. Name of product: OlymPick™

1.2. List of committed functions:

1.2.1. All users shall be able to filter searches based on specific sporting events and relevant athletic accomplishments.

1.2.2. Searches can be done both through keywords and user manual input.

1.2.3. Users shall be able to register for an account using an email address and a secure password.

1.2.4. University Admin shall be able to post and update profiles.

1.2.5. University Admin shall be provided with a reporting feature to generate reports on user engagement, application numbers, and platform usage statistics.

1.2.6. Faculty shall be able to initiate contact with users for recruiting purposes with the option to offer positions.

1.3. Product Uniqueness: OlymPick serves as an optimized search platform for Olympic sports up-and-comers planning their collegiate sports futures. With on-site communication and customizable user profiles, OlymPick also serves as a specialized networking platform between users and university recruiters.

1.4. Website URL: http://35.212.255.5/

2. Usability Test

2.1. Test Objective:

2.1.1. The search function will be tested. As it is the primary purpose of OlymPick, the search functionality must be flawlessly operational, efficient, and easy to use. As a user, they should have a wide range of search filters, especially to accommodate searching through the many different disciplines of the Olympic games and the numerous colleges spread across the US. Having effective search filters ensures that users streamline their search results without overloading them with too much information.

2.2. Test Background and Setup:

2.2.1. System Setup: Desktop or mobile platform with internet access

2.2.2. Intended Users:

2.2.2.1. High-School to College level athletes

2.2.2.2. University Faculty

2.3. Function Testing URL: http://35.212.255.5/

2.4. Testing metrics:

2.4.1. Ease of Use: Is the UI easy to understand and operate?

2.4.2. Aesthetics: Are elements of the UI positioned conveniently and effectively?

2.4.3. Performance: Is the speed of the search adequate to the user's needs?

2.5. Task Description

2.5.1. After clicking the link provided, the test will begin at the search bar's default filter set to "University." The search filter is the gray box located to the right of the search bar and has the following options: University, Sports, and State. While the search filter is set to "University," type in the search box "San Francisco State University," and click on the blue

“Search” box. The results should produce a list of Athlete names, gender, sporting category, event name, and number of medals they possess. Afterward, continue searching by either using the search bar above the results listed, using your internet browser’s back navigation button or click on the OlymPick logo at the top center of the page to get back to the search page. Once back at the search function, change the filter to “Sports” and type “100m freestyle”. A result page will appear but with the list of Universities that have athletes that have attended that sport along with performance information. Continuing the search, change the search filter to “State” and type “CA.” A similar list to the “Sport” filter should appear but only list universities with Athlete data in California. This concludes the test

2.6. Lickert Subjective Test

2.6.1.

The search function is intuitively designed				
Very Disagree	Somewhat Disagree	Neutral	Somewhat Agree	Very Agree
Elements of the UI positioned conveniently and effectively				
Very Disagree	Somewhat Disagree	Neutral	Somewhat Agree	Very Agree
The time it takes to perform searches is fast				
Very Disagree	Somewhat Disagree	Neutral	Somewhat Agree	Very Agree

3. QA Test

3.1. Test Objectives:

3.1.1. The QA test plan aims to verify the functionality and reliability of the search feature on the system. It ensures that users can search for other users effectively and receive relevant search results.

3.2. Hardware & Software:

3.2.1. HW- Standard desktop/laptop

3.2.2. SW- Flask/NGINX on a Google server

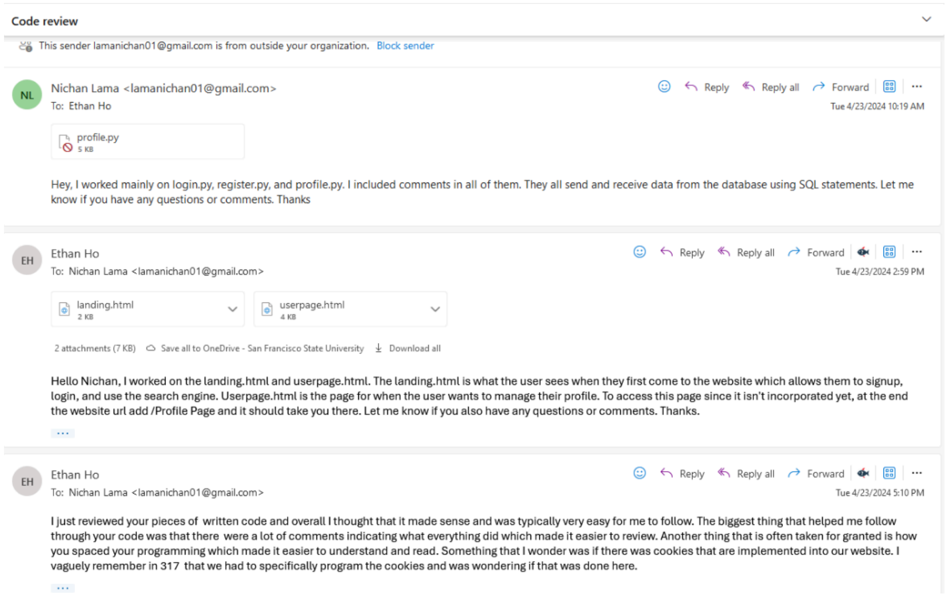
3.2.3. URL- https://35.212.255.5

3.3. QA Test Plan Table:

Test #	Test Title	Description	Input	Expected output	Google Chrome	Firefox
1	Search by University	Verify Users can search by school	Enter a university's name in the search bar and submit the query	Display the relevant data for the specific university	Pass	Pass
2	Search by State	Verify Users can search by state	Enter a State by its initials in the search bar and submit the query	Display the data for universities located in the inputted state	Pass	Pass
3	Search by Sport	Verify Users can search by sport	Enter a sport in the search bar and submit the query	Display the universities that have the inputted sports program	Pass	Pass

4. Code Review

4.1. Email Correspondence:



Code review

This sender lamanichan01@gmail.com is from outside your organization. Block sender

Nichan Lama <lamanichan01@gmail.com>
To: Ethan Ho
Tue 4/23/2024 10:19 AM

profile.py
5 KB

Hey, I worked mainly on login.py, register.py, and profile.py. I included comments in all of them. They all send and receive data from the database using SQL statements. Let me know if you have any questions or comments. Thanks

Ethan Ho
To: Nichan Lama <lamanichan01@gmail.com>
Tue 4/23/2024 2:59 PM

landing.html
2 KB

userpage.html
4 KB

2 attachments (7 KB) Save all to OneDrive - San Francisco State University Download all

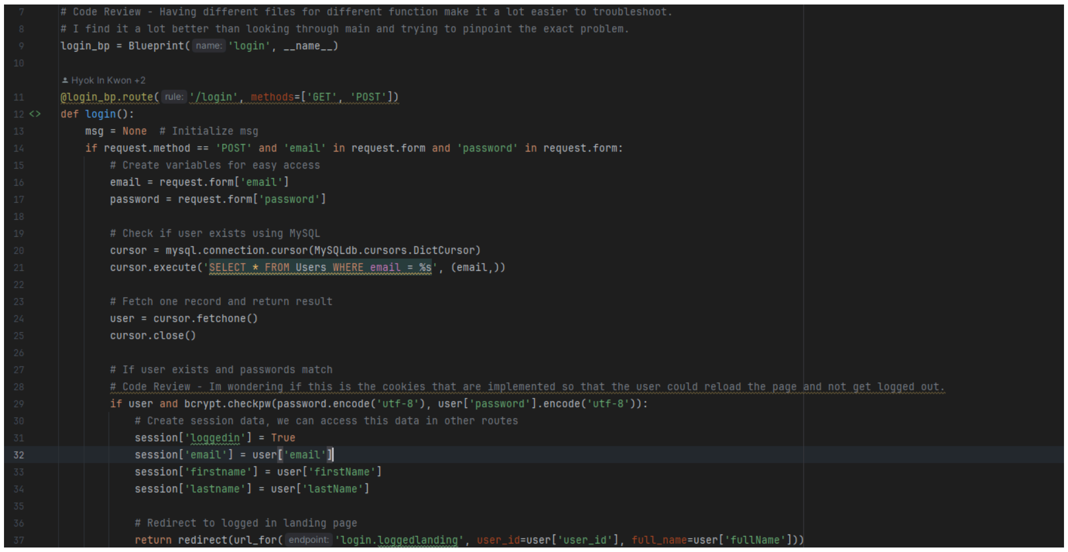
Hello Nichan, I worked on the landing.html and userpage.html. The landing.html is what the user sees when they first come to the website which allows them to signup, login, and use the search engine. Userpage.html is the page for when the user wants to manage their profile. To access this page since it isn't incorporated yet, at the end the website url add /Profile Page and it should take you there. Let me know if you also have any questions or comments. Thanks.

Ethan Ho
To: Nichan Lama <lamanichan01@gmail.com>
Tue 4/23/2024 5:10 PM

I just reviewed your pieces of written code and overall I thought that it made sense and was typically very easy for me to follow. The biggest thing that helped me follow through your code was that there were a lot of comments indicating what everything did which made it easier to review. Another thing that is often taken for granted is how you spaced your programming which made it easier to understand and read. Something that I wonder was if there was cookies that are implemented into our website. I vaguely remember in 317 that we had to specifically program the cookies and was wondering if that was done here.

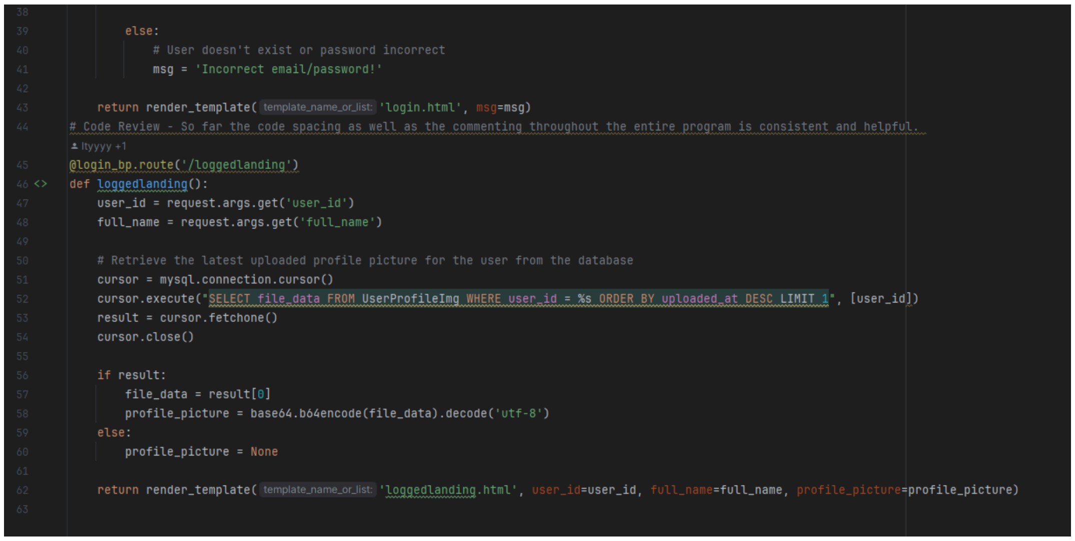
4.1.1.

4.2. Code that was reviewed (within login.py):



```python
# Code Review - Having different files for different function make it a lot easier to troubleshoot.
# I find it a lot better than looking through main and trying to pinpoint the exact problem.
login_bp = Blueprint( name: 'login', __name__)

Hyok In Kwon +2
@login_bp.route( rule: '/login', methods=['GET', 'POST'])
def login():
    msg = None  # Initialize msg
    if request.method == 'POST' and 'email' in request.form and 'password' in request.form:
        # Create variables for easy access
        email = request.form['email']
        password = request.form['password']

        # Check if user exists using MySQL
        cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
        cursor.execute('SELECT * FROM Users WHERE email = %s', (email,))

        # Fetch one record and return result
        user = cursor.fetchone()
        cursor.close()

        # If user exists and passwords match
        # Code Review - Im wondering if this is the cookies that are implemented so that the user could reload the page and not get logged out.
        if user and bcrypt.checkpw(password.encode('utf-8'), user['password'].encode('utf-8')):
            # Create session data, we can access this data in other routes
            session['loggedin'] = True
            session['email'] = user['email']
            session['firstname'] = user['firstName']
            session['lastname'] = user['lastName']

            # Redirect to logged in landing page
            return redirect(url_for( endpoint: 'login.loggedlanding', user_id=user['user_id'], full_name=user['fullName']))
```

4.2.1.

4.2.2. 

```python
        else:
            # User doesn't exist or password incorrect
            msg = 'Incorrect email/password!'

    return render_template('login.html', msg=msg)
# Code Review - So far the code spacing as well as the commenting throughout the entire program is consistent and helpful.
@login_bp.route('/loggedlanding')
def loggedlanding():
    user_id = request.args.get('user_id')
    full_name = request.args.get('full_name')

    # Retrieve the latest uploaded profile picture for the user from the database
    cursor = mysql.connection.cursor()
    cursor.execute("SELECT file_data FROM UserProfileImg WHERE user_id = %s ORDER BY uploaded_at DESC LIMIT 1", [user_id])
    result = cursor.fetchone()
    cursor.close()

    if result:
        file_data = result[0]
        profile_picture = base64.b64encode(file_data).decode('utf-8')
    else:
        profile_picture = None

    return render_template('loggedlanding.html', user_id=user_id, full_name=full_name, profile_picture=profile_picture)
```

5. Security Measures

5.1. Assets protected:

5.1.1. SQL database

5.1.1.1. All inputs that access the database have SQL injection protection implemented through prepared statements

5.1.2. User login credentials:

5.1.2.1. All user password information is encrypted

5.1.3. Production Server:

5.1.3.1. Only specified SSH keys and users are allowed onto the server

6. Non-functional Specs

6.1.

Specification	DONE/ON TRACK
Application shall be developed, tested and deployed using tools and servers approved by Class CTO and as agreed in M0 (some may be provided in the class, some may be chosen by the student team but all tools and servers have to be approved by class CTO).	DONE
Application shall be optimized for standard desktop/laptop browsers e.g., must render correctly on the two latest versions of two major browsers	DONE
Selected application functions must render well on mobile devices (this is	DONE

a plus)	
Data shall be stored in the team’s chosen database technology on the team’s deployment server.	DONE
The language used shall be English.	DONE
Application shall be very easy to use and intuitive.	DONE
Google maps and analytics shall be added	ON TRACK
No e-mail clients shall be allowed. You shall use webmail.	ON TRACK
Pay functionality, if any (e.g. paying for goods and services) shall not be implemented nor simulated in UI.	DONE
Site security: basic best practices shall be applied	ON TRACK
Modern SE processes and practices shall be used as specified in the class, including collaborative and continuous SW development	ON TRACK
The website shall prominently display the following exact text on all pages "SFSU Software Engineering Project CSC 648-848, Spring 2024. For Demonstration Only” at the top of the WWW page. (Important so not to confuse this with a real application).	DONE