Revision #: 1 Date: December 5th, 2019
SW Engineering CSC 648 / 848 Fall 2019

Milestone 4:

GatorDater

Team 204 (local):

Tejas Vajjhala (tvajjhala@gmail.com): Team Lead, Github Master

Sanil Rijal (srijal0509@gmail.com): Front-end Lead

Thomas Zhang(t.zhang70@gmail.com): Front-end assist

Aaron Li (ali16@mail.sfsu.edu): Back-End Lead, Database Master

Milestone	Version	Date Submitted For Review
Milestone 4	1	12/05/2019
Milestone 3	2	11/21/2019
Milestone 2	2	10/24/2019
Milestone 1	2	

Table of Contents

Product Summary

Name of Product: GatorDater

Current project URL: http://35.203.131.184/m3/Index_Page.html

All Major Committed Functions:

- Users shall be able to make an account
- Users shall be able to login
- Users shall be able to access the office hour scheduling page
- Users shall be able to reserve a time slot with a professor
- Professors shall be able to specify their free time slot
- Users shall be able to update their name
- Users will be able to update their password

What is unique about GatorDater:

Currently, there is nothing like GatorDater that has been implemented for SFSU. For scheduling office hours, students need to either email the professor and try to come up with a time to meet, or sign up via a paper list posted outside the teachers room. This is very inconvenient as its difficult to sync up with professors and stay organized.

GatorDater allows users and professors to schedule appointments via an easy to use web interface and displays upcoming appointments, along with the ability to cancel, and reschedule them seamlessly.

Usability Test Plan

Test Case: Search for a professor

- Test Objective:
    - The professor search feature is a major component of our application. This is the first step to scheduling office hours. Without this feature, the rest of the application would not be usable.
    - Our goal is for the user to be able to search for the professor via the search bar, select the professor that matches the input, and then get redirected to the professor's scheduling page where the user can then schedule office hours with them.
    -
- Test Description:
    - System setup:
        - The system will be set up with the latest version of Chrome, Safari, or FireFox. The operating system will be either OSX, Windows, or Linux.
    - Starting point:
        - The starting point of the test case will be at the user's home page which is located at /homepage.html. This is the page that includes the professor search bar, which a user can use to find a professor to schedule an appointment with.
    - Intended users:
        - The intended user for this operation would be a student, as they would be the ones who are searching for a professor and attempting to schedule office hours.
    - URL of the system:
        - The url of the system where the test will begin is /homepage.html.
    - What is to be measured:
        - We are measuring the ability to fetch results from the database that match the inputted search term.
- Usability task description:
    - The user will begin at their homepage and locate the search bar in the center of the page.
    - The user will then enter a professor's name, and will press the search button or hit enter.
    - Based on whether the professor is in our system, the user will be presented with a list of professors that match the search term, or be

presented with text informing them that no professor was found for the entered name. When clicking on a professor's name, the user is then redirected to the professor's scheduling page.

- Questionnaire:
    - Action: The user enters a professor's name and then clicks search or presses enter to submit the query.
    - Question: I was satisfied with the speed of the search bar.
        - Strongly Disagree
        - Disagree
        - Agree
        - Strongly Agree

    - Action: After submitting a search query, the user receives a list of professors to choose from, or no list if no matching professor exists.
    - Question: I was satisfied with the results of the search bar.
        - Strongly Disagree
        - Disagree
        - Agree
        - Strongly Agree

    - Action: The user finds the search bar on their homepage and utilizes it to find a professor.
    - Question: I was able to easily use the search bar without any confusion.
        - Strongly Disagree
        - Disagree
        - Agree
        - Strongly Agree

QA Test Plan

Test objectives:

Ensure the professor search feature is functional and returns a list of professor names that match or partially match the search query.

HW and SW setup (including URL):

The user will be using the latest build of GatorDater and load their homepage at /homepage. The user will be using the latest version of Chrome, Firefox, or Safari to conduct the test.

Feature to be tested:

The user will test the professor search functionality of our application.

QA Test plan:

Number	Description	Test Input	Expected Output	Result
1	Test % in LIKE professor_name field	“Jose”	Get the professors that have “Jose” in their name.	PASS
2	Test ability to load professor page from search result	Clicking on “Jose Ortiz-Costa” after searching “Jose”	Redirect to the professor’s page.	PASS
3	Test ability to display no professors found.	“Emily”	“No matches found”	PASS

Code Review

From: Aaron Li
Date: Sun, Dec 1, 2019 at 4:01 PM
Subject: Prof Search Code Review
To: tvajjhala@gmail.com

Hi Tejas,
The following is a snippet of the professor search code for review. I've hooked it from the homepage search Sanil made with the backend function you made for the VP.

HTML:

```
$(document
).ready(func
tion(){
                    $('.search-box input[type="text"]').on("keyup input",
            function(){
                                /* Get input value on change */
                                var inputVal = $(this).val();
                                var resultDropdown =
            $(this).siblings(".result");
                                if(inputVal.length){
                                $.get("vp/backend-search.php",
            {term: inputVal}).done(function(data){
                                                                    //
            Display the returned data in browser

            resultDropdown.html(data);
                                                                    });
                                } else{
                                resultDropdown.empty();
```

```
                                        }
                                        });

                // Set search input value on click of result item
                $(document).on("click", ".result p", function(){

$(this).parents(".search-box").find('input[type="text"]').val($(this).text()
);
                        //sets search field val

                        $(this).parent(".result").empty();
                        //clears dropdown
                        username = $(this).text().replace(/\s/g,'');


window.open("vp/user-page.php?user="+username);



                        });
                });
</script>
</head>
<body>
<p>
</p>
<div class="search-box">
<input type="text" autocomplete="off" placeholder="Start Typing
Professor Name" />
```

```
<div class="result"></div>
</div>
</body>
```

--Aaron

Response:

From: Tejas Vajjhala
Date: Sun, Dec 1, 2019 at 4:45 PM
Subject: Re: Prof Search Code Review
To: liaaron727@gmail.com

Hey Aaron,
Thanks for sending me the code. Just a couple of changes I would make.
For the following:

```
$.get("vp/backend-search.php", {term: inputVal}).done(function(data)
```

I would change the path of the backend search php file to something that is NOT in the vp folder, as this will be deleted in production. We should keep the vp code separate from the production code. But not a huge change.

Second:

```
resultDropdown.html(data);
                                                                    });
                                        } else{
                                        resultDropdown.empty();
                                        }
                                        });
```

I would have the search be redirected to a new page rather than do a drop down of results. It kind of breaks the continuity/theme of the site as the drop down looks out of

place. I think a good thing to do rather than a drop down is have the actual search results be displayed ON the page below the search bar (rather than a drop down field that originates from the search bar if that makes sense).
But the code looks solid, this is basically all we need for the search functionality. We'll just clean up how the output is displayed/formatted.

Thanks,
Tejas

Self-check On Best Practices For Security

- List of major assets that are being protected
    - User passwords
    - UserIDs
    - User IP addresses
    - User email addresses
- We are encrypting user passwords using bcrypt.
- Search input data is validated via character check to make sure all characters are letters, hyphens, or spaces. Therefore only valid names are searchable for the professor search. We are also preventing MySQL injection by using prepared statements in conjunction with MySQLi. For example
    - $stmt = $dbConnection->prepare('SELECT * FROM professors_list WHERE name = ?'); $stmt->bind_param('s', $name); // 's' specifies the variable type => 'string' $stmt->execute(); $result = $stmt->get_result(); while ($row = $result->fetch_assoc()) { // Do something with $row }

Self-check: Adherence to original Non-functional specs

Security:

1. Login shall be mandatory for student and professor - ON TRACK
2. Username shall be the user's registered email - DONE
3. Password shall be encrypted before saving it in the database - DONE
4. User's session shall be ended after 30 minutes of inactivity - ON TRACK
5. User's session shall only be ended by code design - ON TRACK
6. This site shall not accept third-party cookies - ON TRACK

Audit:

1. The site admin shall be the only person authorized to delete users - DONE
2. Users shall not be allowed to modify any web configuration files - DONE
3. Users shall not be allowed to login into the admin page - DONE
4. Unregistered users shall not be able to access anything other than the login/signup page - On Track

Performance:

1. The site loading time shall be less than 1 second for all the screens - DONE
2. “Search” shall be executed in a background thread for improving performance - ON TRACK
3. Query shall be executed in a background thread for improving performance - ON TRACK

Capacity:

1. The total data storage allowed by the web site shall not exceed 80 % of the server capacity for this site - DONE
2. The web site shall be prepared to support scalability for adding future new features - DONE
3. The web site shall be capable to handle at least 50 users - DONE

Reliability:

1. Downtime for maintenance shall be less than 1 hour per month - DONE
2. Downtime for maintenance shall not affect the main functionality of the site - DONE

3. In all cases, downtime for maintenance shall be informed to the users either by email - ON TRACK
4. In all cases, downtime for maintenance shall be informed to the users by publishing an announcement in a visible place of the main page. - ON TRACK

Recovery:

1. In a total failure case, the whole site should be put down to revision - DONE
2. If broken, the mean time to recovery shall not exceed one day - DONE

Data Integrity:

1. Database tables shall be backed up every week - ON TRACK
2. Admin shall be able to execute a recovery when needed - ON TRACK
3. Images size shall be limited up to 1 megabyte - ON TRACK
4. Images shall be uploaded in the correct format (jpg or jpeg) - ON TRACK

Compatibility:

1. The site shall be compatible with the last version of Safari browser - DONE
2. The site shall be compatible with the last version of Firefox browser - DONE
3. The site shall be compatible with the last version of Chrome browser - DONE
4. The site shall be compatible with at least an old version of all the browsers listed above - DONE
5. Third party applications shall not be able to modify any content that may affect the site compatibility - DONE
6. The site shall be ready to support with any or minimal changes any other compatibility that may be added in future versions - DONE
7. The site should be compatible to escalate to new databases - DONE

Conformance with Coding Standards:

1. Architecture and design standards shall meet all the requirements listed under the High-Level Architecture section of this document - DONE
2. Only working code that meets all the code standards shall be submitted to the project repository - DONE
3. Any working code shall be tested and debugged before being considered to be working code - DONE

4. Any internal errors or exceptions returned by the code shall be stored in a log - DONE
5. Any error that may affect the functionality of the site shall be reported to the user - ON TRACK
6. Any error shall be handled in a way that does not affect the functionality of the site - DONE
7. This site shall not be launched without all the priority one featured finished and tested - ON TRACK
8. This site shall be tested and debugged as a whole 2 weeks before the delivery due date - ON TRACK

Look and Feel Standards:

1. The application and its layouts shall look professional - DONE
2. The site shall be enough simple to handle by all the parties involved - DONE
3. Elements on the screen shall have the correct density to meet the compatibility standard of the browsers - DONE
4. Elements on the screen shall have the correct size to meet the compatibility standard with all mobile devices - ON TRACK
5. Elements on the screen shall have rich and beautiful colors for user delight - DONE
6. The site shall be able to work correctly without mouse interaction - ON TRACK
7. The site shall be able to work correctly without keyboard interaction - ON TRACK
8. Elements in screen shall be resized automatically without user interaction when being loaded in all the different platforms supported by the site - DONE

Internationalization / Localization Requirements:

1. The default language of this site shall be English - DONE
2. The site shall support scalability to add more supported languages in future versions - ON TRACK