SW Engineering CSC648/848 Fall 2018

SW Engineering CSC648/848 Fall 2018

Milestone 4

Team 15 (global)
Jonas Kühle, Eric Groom, Mariko Sampaga, Nay Lin Min
jkuehle@mail.sfsu
12/14/2018

Revision History Table		
Version 0.1	**Submitted for Review**	**12/02/2018**
Version 0.2	**Revised**	**12/14/2018**

1. Product Summary

Product Name: Gator Goods
Product URL: http://ec2-18-237-111-132.us-west-2.compute.amazonaws.com/
Itemized List of All Major Committed Functions

1. Home page
2. Search (including search field validation)
3. Detail
4. Messaging/contact seller
5. Post Upload
6. Login (Registration)

The “GatorGoods” project is a project started by a student startup team from San Francisco State University and Fulda University in Germany. The “GatorGoods” project is a buy/sell website for students and faculties at San Francisco State University. The motivation behind this project is to build a website that is not only simple to use but also safe and secure for the users. We wanted a place for students and faculties to be able to post items they wish to sell with ease and potential buyers to be able to browse through those post and buy them safely and securely.

Unlike other buy/sell websites out there, ours allow buyers to engage with seller safely. Our website has a build in messaging features that will allow buyers to ask questions about the product directly with the seller. The “GatorGoods” project will also allow buyers and sellers to arrange a meet up spot that is around the campus for safer transactions. Another thing that separates us from other buy/sell websites is that we make the browsing easier for buyers who are looking to purchase books for a certain course. Our sellers can input the SFSU course number for their books they are selling so that buyers can easily find it. Our website will also have a moderator that has the ability to approve/deny posts and blacklist users as they seem fit from their moderator dashboard. This will keep our website safe and secure from spammers and inappropriate posts.

The “GatorGoods” project is started by a team of four students from SFSU and three students from Fulda University in Germany. Our team is diverse and everyone has their own unique skill sets. You can read more about our team at our “About Us” page on the website.

2. Usability Test Plan

Test Objective

The test objective of the usability test plan is to understand user's behaviour and to improve our design. It helps the developer to identify where users struggle with our website.

Test Plan

The system setup will have the evaluator with a computer laptop opened up to the homepage of the Gator Goods website, the starting point will be at URL provided in section 1. The observer will be removed from the test environment. The intended user would be a university student. The evaluator will be tasked with searching for a certain telescope. Successful completion would be in 1 item showing the certain telescope.

Task for the usability tester

1. Open the webpage
2. Search for a certain telescope
3. Pick an appropriate item
4. Contact the seller

Questionnaire

I found the search bar easy to use
__Strongly Disagree __Disagree __Neutral __Agree __Strongly Agree
Comment:
_____
_____
_____
___

The results were relevant to my search
__Strongly Disagree __Disagree __Neutral __Agree __Strongly Agree
Comment:
_____
_____
_____
___

It was easy to find the details of the telescope
__Strongly Disagree __Disagree __Neutral __Agree __Strongly Agree
Comment:
_____
_____
_____
___

3. QA Test Plan

Test objectives

The test objectives of our QA test are to prevent bugs, mistakes and defects. It is part of the whole development process.

HW and SW setup

http://ec2-18-237-111-132.us-west-2.compute.amazonaws.com/

The two latest Mozilla Firefox (latest version: 63.0.3) and Google Chrome (latest version is 70.0.3538.110) versions are used for testing. Additional to that, the Mobile Device Emulators of Firefox and Chrome are used in order to test the responsive web design. The emulator makes it possible to easily test several common screen sizes and resolutions for both mobile and custom devices.
Test automation with selenium in progress.

Feature to be tested

The search bar functionality is tested. It is tested whether a certain telescope is shown if a string with an accurate product description is entered into the search bar. In addition, the behavior of the web page is tested if an empty or invalid string is entered. Items (and no blank page) should be shown to the user if no appropriate item matches the input.

Test plan

Number	Description	Test Input	Expected Output	PASS/FAIL
1	Test relevance of results	Enter “SCT XLT” in search bar	Get 1 result, check that it is a telescope model “SCT XLT”	PASS
2	Test empty input	Enter an empty string in search bar	Get 6 items	PASS
3	Test invalid input	Enter “-1” in search bar	Show error message “-1 is invalid input. Try something else.” and get 6 items	FAIL

4. Code Review

The coding style we have chosen is to have brief header comments and camel case implemented into our project.

Example for coding style in PaginationProvider.js. In this case, the header comment is exceptionally more comprehensive than usual.

```
import React from "react";
import PropTypes from "prop-types";
import apiFactory from "../api";

/**
* This component uses some more advanced concepts of React, I'll try to summarize here,
* but this is by no means comprehensive and you should read more about Component Lifecycle
* Methods and Render Props to gain a better understanding.
*
* Public API for this component:
* If you look at the render method, you can see that the only thing this component does
* rendering-wise is return the results of calling `this.props.children`. Children is a
* special prop in React, it is whatever is in between the start tag and end tag of your
* component. For example, if we rendered <PaginationProvider><p>hi</p></PaginationProvider>,
* the `children` prop in `PaginationProvider` would be `<p>hi</p>`. We take advantage of this
* by instead passing a function. This way, PaginationProvider can provide functionality to
* it's children without needing to know the structure of it's children.
*
* The easy way to use this component is to pass the info from the render prop function to
* `PageInfo`. This component also needs to be passed a `resource` prop which should be the
* endpoint you are trying to access such as `/items` (note: you do not need to provide the
* hostname as this is provided by the api module)
*/
class PaginationProvider extends React.Component {
 static propTypes = {
   children: PropTypes.func.isRequired,
   resource: PropTypes.string.isRequired
 };
 state = {
   hasNextPage: false,
   hasPrevPage: false,
   nextPage: null,
   prevPage: null,
   data: null,
   currPage: 1,
   loading: false,
   count: 0
 };
 async componentWillMount() {
   await this.setup();
 }
 async componentDidUpdate(prevProps) {
   if (prevProps !== this.props) {
     await this.setup();
   }
 }
```

```
setup = async () => {
  this.setState({ loading: true });
  const api = apiFactory(fetch);
  const res = await api(this.props.resource);
  const json = await res.json();
  this.setStateWithResults(json);
};
setStateWithResults = data => {
  const { next, previous, count, results } = data;
  this.setState({
    count,
    data: results,
    nextPage: next,
    prevPage: previous,
    currPage: 1,
    loading: false,
    hasNextPage: next !== null,
    hasPrevPage: previous !== null
  });
};
nextPage = async () => {
  if (this.state.hasNextPage) {
    const url = this.state.nextPage;
    const res = await fetch(url);
    const json = await res.json();
    this.setStateWithResults(json);
  }
};
prevPage = async () => {
  if (this.state.hasPrevPage) {
    const url = this.state.prevPage;
    const res = await fetch(url);
    const json = await res.json();
    this.setStateWithResults(json);
  }
};
render() {
  const {
    count,
    hasNextPage,
    hasPrevPage,
    data,
    loading,
    currPage
  } = this.state;
  return this.props.children({
    count,
    hasNextPage,
    hasPrevPage,
    data,
    loading,
    currPage,
    nextPage: this.nextPage,
    prevPage: this.prevPage
```

```
  });
 }
}

export default PaginationProvider;
```

b) Peer review example. Since we don't have any code under review that currently refers to the search bar a current example for peer-review via email is provided instead.
In this case, It wasn't necessary to comment the code since it was ok. The only thing missing was a header comment. The emails are provided below.

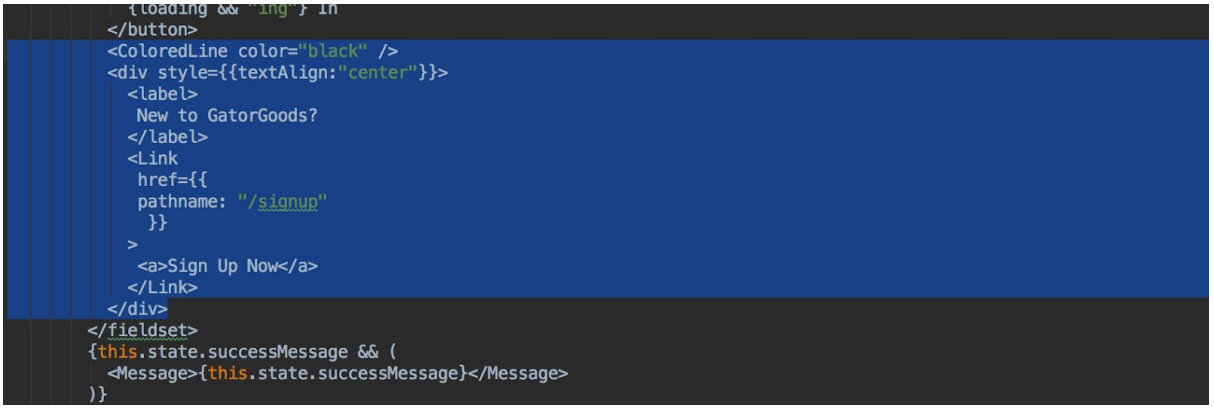
Email from Naylin:
Hey guys,

I have separated the login/registration page so that they are showing on different tabs. I used the same logic as before but added a new page for sign up. Also added pattern to the email field to look for sfsu emails.
Please review it.

Answer from Jonas:
Hey Naylin,
your code looks great. Good job! But can you make sure to add header comments with a brief summary? It is very important for a better understanding.
Keep up good work!

Cheers,
Jonas

Screenshots of the code:



```
    return (
      <Form onSubmit={this.signin} method="POST">
        <h1>Sign up</h1>
        <Error error={this.state.error} />
        <fieldset aria-busy={loading} disabled={loading}>
          <label htmlFor="email">
            Email:
            <input
              type="email"
              name="email"
              id="email"
              pattern=".+@mail.sfsu.edu"
              title = "Please use your school email address."
              placeholder="Email"
              value={this.state.email}
              onChange={this.handleChange}
            />
          </label>
          <label htmlFor="password">
            Password:
            <input
              type="password"
              name="password"
              id="password"
              placeholder="Password"
              value={this.state.password}
              onChange={this.handleChange}
            />
          </label>
          <label htmlFor="confirmPassword">
            Confirm Password:
```

```
import Signin from "../components/Signin";
import Grid from "../components/styles/Grid";
import Centered from "../components/styles/Centered";

const SigninPage = () => (
  <Centered>
    <Grid>
      <Signin />
    </Grid>
  </Centered>
);

export default SigninPage;
```

```
1   import Signup from "../components/Signup";
2   import Grid from "../components/styles/Grid";
3   import Centered from "../components/styles/Centered";
4
5   const SignupPage = () => (
6     <Centered>
7       <Grid>
8         <Signup />
9       </Grid>
10    </Centered>
11  );
12
13  export default SignupPage;
14
```

5. Self-Check on Best Practices for Security

Fulda team confirmed that passwords are saved encrypted from the beginning and all input data is validated in the server side against the database field types.
Additionally to that, some data validation is done in front end.
Fulda team also confirmed that images can’t be changed by unauthorized persons.

6. Self-Check: Adherence to Original Non-Functional Specs

1. 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
2. Application shall be optimized for standard desktop/laptop browsers e.g. must render correctly on the two latest versions of all major browsers: Mozilla, Safari, Chrome. DONE
3. Selected application functions must render well on mobile devices. ON TRACK
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 ON TRACK
10. No e-mail clients shall be allowed DONE
11. Pay functionality, if any (e.g. paying for goods and services) shall not be implemented nor simulated. 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-Fulda Software Engineering Project CSC 648-848, Fall 2018. For Demonstration Only"* at the top of the WWW page. (Important so as to not confuse this with a real application). DONE