

# SW Engineering CSC648/848 Fall 2018

## *EduGator* Team 03

Milestone 4  
December 2, 2018

### Team Members

James Andrews      [jandrew3@mail.sfsu.edu](mailto:jandrew3@mail.sfsu.edu) (Team Leader)  
Ian Dennis  
Ivan Varela  
Chris Johansen  
Liyao Jiang  
Shirin Namiranian  
Kenneth Surban

Revision Number	Description	Date
1.0	Initial Version	12/02/2018
2.0	Revised Version	12/10/2018

# 1. Product Summary:

**Product Name:** EduGator

**Product URL:** <https://team03-648.herokuapp.com/>

**Our Product:** We provide a marketplace for SFSU students to buy and sell general items, tutoring services, and coursebooks.

**Competitive Edge:** In addition to facilitating a marketplace for SFSU students to buy and sell items, our product also allows for SFSU students to seek and offer tutoring services from their peers, and takes advantage of the SFSU niche by mapping textbooks for courses to the appropriate course.

## **Major Committed Functions**

- Login/Registration
- Search (with pull down categories, text entry)
- Search Results
- Item details
- Seller dashboard (shows selling items and messages)
- Admin
- Messaging (buyer to seller only)

## 2. Usability Test Plan

### Test Objectives:

The objective of this test is to see how usable our current item posting interface is. We want to test how quickly a user is able to find the posting button on our home page, how easily they can navigate the posting form as they fill it out, and how understandable our error messages are in the event that a user fills out the form incorrectly.

### Test Plan:

#### A. System Setup:

- Browser: Google Chrome Browser v.71
- Operating System: macOS High Sierra 10.13
- Computer: MacBook Pro Laptop

#### B. Starting Point:

- Home Page of EduGator

#### C. Intended User:

- Any SFSU Student
  - i. Does not need to be tech savvy

#### D. Task: Create a new post for an item, and post it to Edugator.

- Completion Criteria
  - i. The user created a well-formed post that successfully posted to the database.
  - ii. The user completed this task within 2 minutes.

#### E. URL to Test System:

- <https://team03-648.herokuapp.com>

**Questionnaire:**

Question	Strongly Disagree	Mostly Disagree	Neutral	Mostly Agree	Strongly Agree
The format of the site is intuitive and easy to navigate.					
I was able to post an item on the website easily.					
Errors encountered when posting an item were explanatory.					

### 3. QA Test Plan

#### Test objectives:

The objective is to test that the posting form fields for our website function to our specifications. Inputs to the posting form should be checked for validity, and an error message should be shown if the the form field is invalid. The post form shall not allow the user to post an item with an invalid post form.

#### HW and SW setup:

- Hardware
  - MacBook Pro
- Software
  - Web browser (test on 2 latest revisions):
    1. Google Chrome
  - Test Link: <https://team03-648.herokuapp.com/post>

#### Testing Plans:

Number	Description	Test Input	Expected Results	PASS/FAIL
1	Test Empty Category Field	Do not select a category from the "Category" field drop down menu and then press the post button	Form reports an error stating "Category must be selected" in red text	PASS
2	Test Non-Numerical Price	Try to enter a non-numeric price into the "Price \$" field of the post form.	Any non-numeric characters do not appear in the price field	FAIL
3	Test Long Item Name Error	Enter more than 40 characters into the "Item Name" field.	The field becomes red, and an error is added to the form indicating that the input for "Item Name" is too long	FAIL

## 4. Code Review

### Coding Style:

- 2 space indentations
- Variables
  - Camel-Cased
  - Start with a lower-case letter
    - Exception made for module names that are being imported into a file
  - Names shall be clear and meaningful
- Header Comments at the top of each file


### Code Review:

When it comes to code review we use the tools GitHub provides. Each team member works on their own branches and once they feel that their code is ready to be merged into the master branch they create a pull request. Once the pull request has been created other team members will review the code with the final ok being given by the team lead. The following is an example of what our code reviews look like:


### Sample Code Review

`api/endpoints/login.js` Outdated

```
5 + const bcrypt = require("bcrypt")
6 +
7 +
8 + router.get('/registe', (req, res) => {
```

 **idennis7** 21 days ago + 😊 ...

We should probably remove this test route, or change the route name to it.



Resolve conversation

api/endpoints/login.js

Outdated

```
13 +  
14 +  
15 + router.post('/register', async (req, res) => {  
16 +   var saltRounds = 8;
```



idennis7 21 days ago

+ 😊 ...

As a future improvement, we should do some error checking on req.body to make sure that all the parameters were actually passed.



MrApplesnacks 20 days ago

True, we should have that as well as some form verification on the front end



Reply...

Resolve conversation

api/endpoints/login.js

Outdated

```
8 + router.get('/registe', (req, res) => {  
9 +   db.any('insert into users(email, username, password) \\  
10 +     values($1, $2, $3);', ['ema', 'name01', 'password'])  
11 +     console.log('this route is working')
```



JamesFTW 21 days ago

Lets remove the console.logs



Reply...

Resolve conversation

api/endpoints/login.js

Outdated

```
17 + var myPlaintextPassword = req.body.password  
18 + var salt = bcrypt.genSaltSync(saltRounds);  
19 + var hash = await bcrypt.hash(myPlaintextPassword, salt);  
20 + db.any("insert into users (email, username, password) values($1, $2, $3
```



idennis7 21 days ago

You don't have to, but I find using ALL CAPS for SQL keywords to be easier to read.



Reply...

Resolve conversation

## 5. Self-Check on Best Practices for Security

### Major Assets:

- Database containing user information and data
- Messages sent between users that could contain personal information
- User postings of items and tutor services
- Metrics on website usage

### Encryption of Password in Database:

Done

### Input Data Validation:

Search bar input only allows for text format and max 40 characters. We are using bootstrap for field validation.

## 6. Adherence to 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.
  - **ON TRACK**
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.
  - **ON TRACK**
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)
  - **DONE**
13. Modern SE processes and practices shall be used as specified in the class, including collaborative and continuous SW development
  - **DONE**
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).
  - **ON TRACK**