# Testing Report

*TextForSale*

**Client**
Arash Fallah

**CoffeeBots**
Mehreen Awan
John Gordon
Mina Beshai
Daniel Schomisch
Daniel Kelly

12/1/2016

*TextForSale*

# Testing Report

## Table of Contents

## 1. Introduction

### 1.1 Purpose of This Document

The purpose of this document is to inform the client of the testing procedures and results conducted on the system. The testing procedures were developed ahead of time and use standard testing practices. The procedures are all outlined. The procedure then is carried out with each test having its results recorded. The tests then are interpreted to produce a meaningful analysis of the progress of the project.

### 1.2 References

Use cases from the System Requirements Specification (SRS) documentation.

## 2. Testing Process

### 2.1 Description

The process that the team followed was standard unit testing as a first phase. Each individual file and each individual component was tested upon completion and when it was in development. The unit testing included stress testing as well boundary and validation testing. Then each file was integrated into the project and tested as part of the system. It was then retested for both validation and to make sure it was error free.

The system as a whole was tested after each feature was added to make sure that each component was not breaking other features or causing errors in other parts of the system. The idea was to advance step by step, feature by feature after the original code skeleton was written. The group had a plan and followed it in a chronological order.

2.2 Testing Sessions

Testing Session :4
Date: 11/19/16
Location: ITE 240
Time Started: 1pm
Time Ended: 3pm
Performer:Mehreen, Daniel S, Daniel K, John gordon, and mina.
Use Case:Login/Register

Testing Session :3
Date: 11/15/16
Location: ITE 240
Time Started: 1pm
Time Ended: 3pm
Performer:Mehreen, Daniel S, Daniel K, John gordon, and mina.
Use Case: Search for Textbook

Testing Session :2
Date: 11/01/16
Location: ITE 240
Time Started: 1pm

Time Ended: 3pm
Performer: Mehreen, Daniel S, Daniel K, John gordon, and mina.
Use Case: View User Profile

Testing Session :1
Date: 10/18/16
Location: ITE 240
Time Started: 1pm
Time Ended: 3pm
Performer: Mehreen, Daniel S, Daniel K, John gordon, and mina.
Use Case:Payment Option

### 2.3 Impressions of the Process

The testing process made the actual development of the program easier. In testing the group changes some aspects of the code such that it runs better and smoother. While in hindsight it seems like such changes should have been the original design at the time it is impossible to see what would run best in an unknown environment. The testing process exposed bugs and logic errors, The testing process also produced missing features and made features that weren't user friendly known.

The quality of the program obviously increased dramatically as the testing process went on. The quality was not limited to errors whether logical errors and bugs but it expanded to visual aspects of the program and friendliness. It made us use our own program and as users we came to realize what was working and what wasn't, which features were useful and which weren't. The testing process made the quality of each individual feature known to the programmers and designers.

The best module has to the user registration and all its aspects. It works flawlessly as intended without any errors. It looks good and its user friendly such as when it tells the user when they did something wrong. The worst module for now is the cart module mainly because it doesn't update the session when it is cleared.  An example of this is if the user decides that they don't wish to buy what they have and deletes everything then moves on from that page to another page, when the page changes the cart goes back to having what it did before. This is a known bug, while there is a good chance it will be fixed, it has the highest chance of a bug not being fixed.

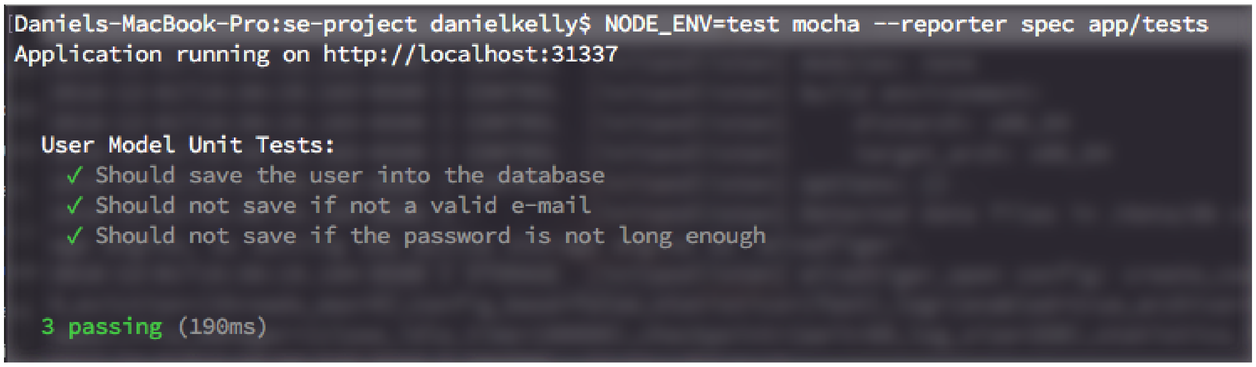
## 3. Test Results

Testing was split up between various team members and conducted by implementing use cases for various functions within the TextForSale site. Use cases were chosen to showcase the strengths of various portions of our site including, but not limited to, user authentication, handling searches, and valid page redirection.

### 3.1 Testing Suite

The unit tests were performed using a Javascript suite known as ‘Mocha’. This was utilized in conjunction with a task runner called ‘Grunt’. In addition to Mocha, various middleware was used to verify the success or error upon valid and invalid requests respectively. The standard Mocha module was used to test data upon entry to the database, or the data models in the MVC pattern. A middleware named ‘SuperTest’ was used to test the REST API route endpoints, or the controllers of the MVC pattern.

An example of a unit test for user’s is shown in the figures below.

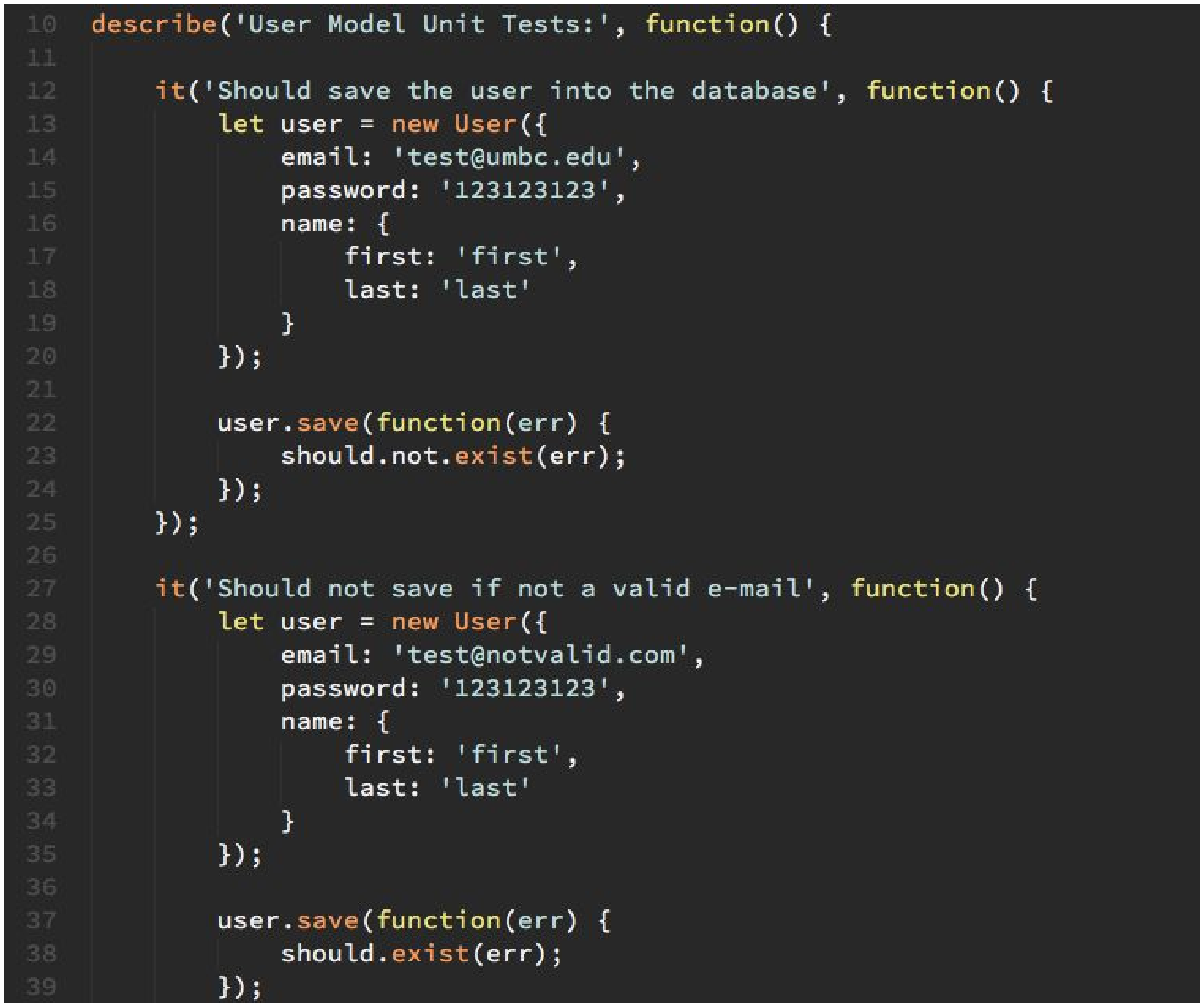


```
Daniels-MacBook-Pro:se-project danielkelly$ NODE_ENV=test mocha --reporter spec app/tests
Application running on http://localhost:31337


  User Model Unit Tests:
    ✓ Should save the user into the database
    ✓ Should not save if not a valid e-mail
    ✓ Should not save if the password is not long enough


  3 passing (190ms)
```

These unit tests can be checked upon each change to the server-side code. While the final output does not indicate the implementation of the test, the JavaScript middleware named ‘Should’ ensures that data returned from the server is as expected. A code sample of the unit test that produces this output is shown in the figure below.



```
describe('User Model Unit Tests:', function() {

    it('Should save the user into the database', function() {
        let user = new User({
            email: 'test@umbc.edu',
            password: '123123123',
            name: {
                first: 'first',
                last: 'last'
            }
        });

        user.save(function(err) {
            should.not.exist(err);
        });
    });

    it('Should not save if not a valid e-mail', function() {
        let user = new User({
            email: 'test@notvalid.com',
            password: '123123123',
            name: {
                first: 'first',
                last: 'last'
            }
        });

        user.save(function(err) {
            should.exist(err);
        });
```

For each use case, define a set of equivalence partitions and their corresponding boundary cases. Be sure to cover all classes of valid and invalid inputs.

Describe one or more test cases for each equivalence partition, including the purpose of each test, the test input data (the exact data, if possible, not a general description), any prior inputs that would have been necessary to get to a point where the test input is applicable, and the expected output.

| **Use Case** | **Login/Register** |
|---|---|
| Valid Situation | User attempts to register with valid umbc email and complies with rules for all other fields.<br>User attempts to log in with an account that was previously registered. |

| | |
|---|---|
| Invalid Situation | Attempting to register with non-umbc email or with a password less than 8 characters.<br>Attempting to log in with an account that hadn't been previously registered. |
| Purpose | To test user authentication |
| Expected Results for Valid Situation | User will click the Register button and fill in all fields to successfully register. Then clicks the Login button and uses the same info used to register. Finally is prompted with successful login message and displays "Welcome, ***" with the stars being the first name. Also has option to Logout. |
| Expected Results for Invalid Situation | User clicks Register button with invalid input in the fields.. The user is prompted error messages and information for valid inputs. User is not registered or logged in. |
| Boundary Conditions | Invalid email, password (less then 8 characters).<br>No input in a field (all fields require input). |

| **Use Case** | **Search for Textbook** |
|---|---|
| Valid Situation | Click the Search button after entering a title that matches an existing book. |
| Invalid Situation | Not entering a matching title for an existing book. |
| Purpose | To find a book the user needs or is interested in. |
| Expected Results for Valid Situation | All of the matching books are displayed to the user with their associated price. User can view the book by clicking View Book. |
| Expected Results for Invalid Situation | No books are displayed. User does not find the book he/she searched for. |
| Boundary Conditions | Input must be greater than length 0. |

| **Use Case - not fully implemented** | **View User Profile** |
|---|---|
| Valid Situation | Each user can view all parts of their own and other user profiles. |

| | |
|---|---|
| Invalid Situation | User profiles are either not visible or incomplete. |
| Purpose | To check a buyer/sellers rating before starting a transaction with them or rate another user after a transaction. |
| Expected Results for Valid Situation | User name, rating, and list of books for sale are visible. Also an option to message the user. |
| Expected Results for Invalid Situation | One or all of the above categories are not listed. |
| Boundary Conditions | User viewing a profile must also be a registered user. |

| **Use Case - not fully implemented** | **Payment Option** |
|---|---|
| Valid Situation | Users have a variety of payment options to choose from. |
| Invalid Situation | Users have no payment options available or all necessary fields in the payment form don't show. |
| Purpose | So the users can choose a payment option of their choice and make the payment in a legitimate fashion. |
| Expected Results for Valid Situation | User can pick from a list of payment options and all fields are present so the payment is valid. |
| Expected Results for Invalid Situation | User does not provide a valid payment so cannot purchase the book. |
| Boundary Conditions | User must be registered to the site and provide a valid payment from options listed. |

| **Use Case - not complete** | **Get Redirected if Book is not Available** |
|---|---|
| Valid Situation | The user is still able to purchase the book from a third party site. |
| Invalid Situation | Book is not available and the redirection to a third party site is unsuccessful. |
| Purpose | To give the user an alternative option if the book he/she is looking for is not available in the TextForSale database. |

| | |
|---|---|
| Expected Results for Valid Situation | User is sent to a trusted third party site (Amazon or the bookstore website) with the book available to purchase. |
| Expected Results for Invalid Situation | User cannot get the book he/she was looking for. |
| Boundary Conditions | 1. The book is not available in the TextForSale database.<br>2. Third party site is up and running. |

| **Use Case - not complete** | **Both Parties Verify Transaction is Complete** |
|---|---|
| Valid Situation | The buyer and seller both confirm the transaction was successful on the TextForSale application. |
| Invalid Situation | Either the buyer or seller did not confirm the transaction occurred within a timely manner. |
| Purpose | To make sure the buyer got the book(s) he/she purchased from the seller. Also for accurate user ratings. |
| Expected Results for Valid Situation | Buyer received the book(s) he/she purchased. |
| Expected Results for Invalid Situation | 1. Buyer purchased the book but never received it.<br>2. The seller gave the book away but never received payment. |
| Boundary Conditions | Both parties are registered users, book is available, valid payment option is chosen. |

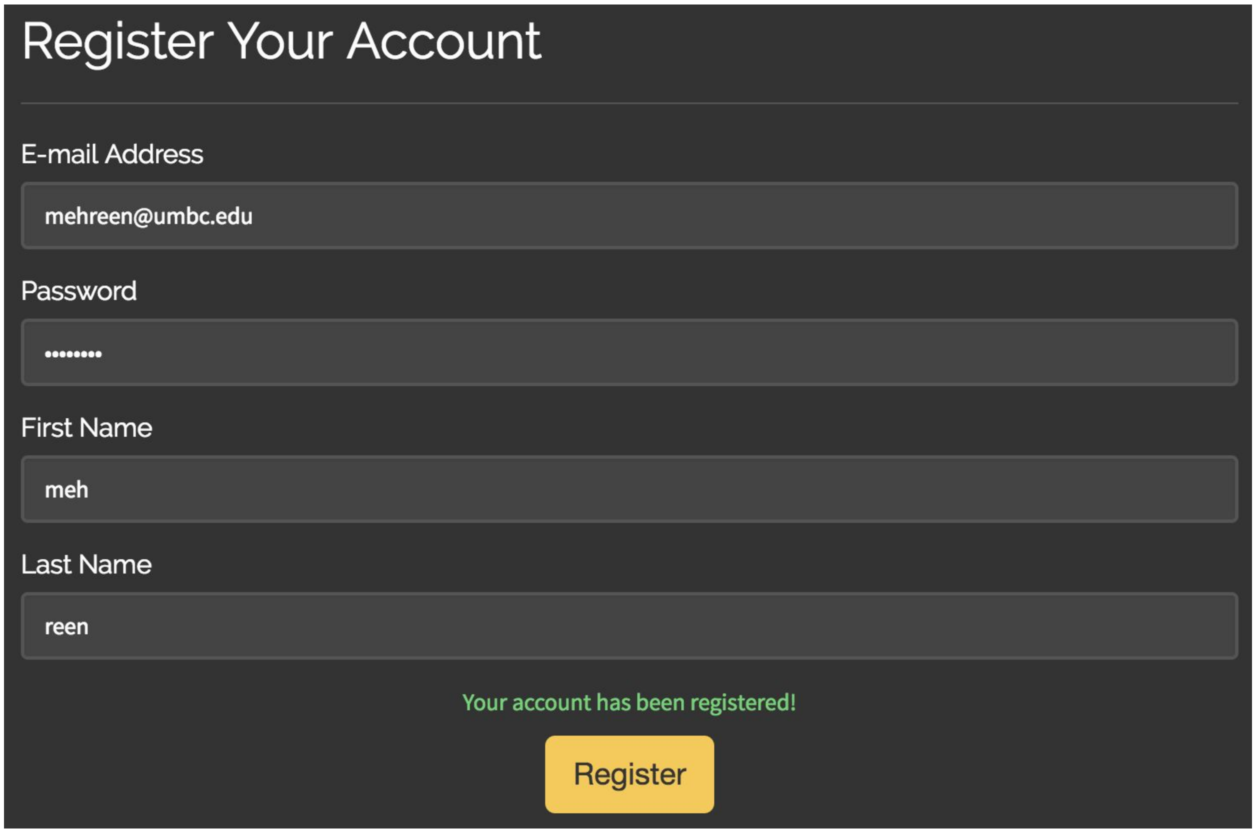
### 3.2 Test Results

State the name of the tester for the particular use case. Summarize those test cases from the above that you were not able to execute, for any reason, and briefly explain the difficulty. Summarize those test cases that ran and the actual result was identical to the expected result. Summarize those test cases that ran and the actual result was not the same as the expected result, but was nevertheless correct. Explain the discrepancy. Describe each defect detected. For each defect, outline either a suggested repair or the actual repair you made. In the cases where you successfully removed the bug, or at least tried to fix the bug but were unable to do so, briefly describe any changes you made to the program, including the modules, functions, classes, data structures, etc. affected. Try to explain the underlying logical cause of the defect in the program code.

### 3.2.1 Use Case: Login/Register

**Conducted by: Mehreen Awan**

Valid Situation:



Register Your Account

E-mail Address

mehreen@umbc.edu

Password

••••••••

First Name

meh

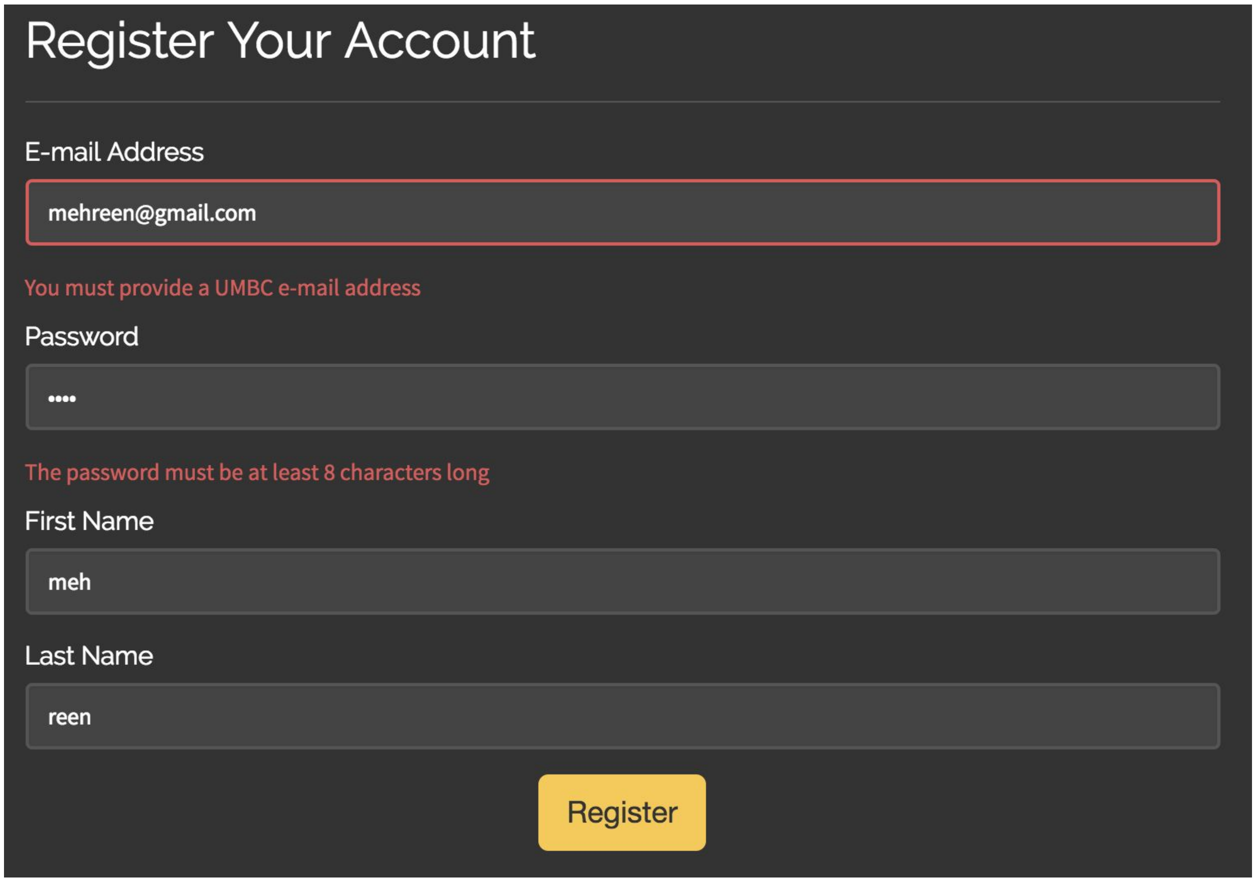
Last Name

reen

Your account has been registered!

*User fills in all fields correctly; umbc email, 8 character password. Successful registration*

Invalid Situation:



Register Your Account

E-mail Address

mehreen@gmail.com

You must provide a UMBC e-mail address

Password

••••

The password must be at least 8 characters long

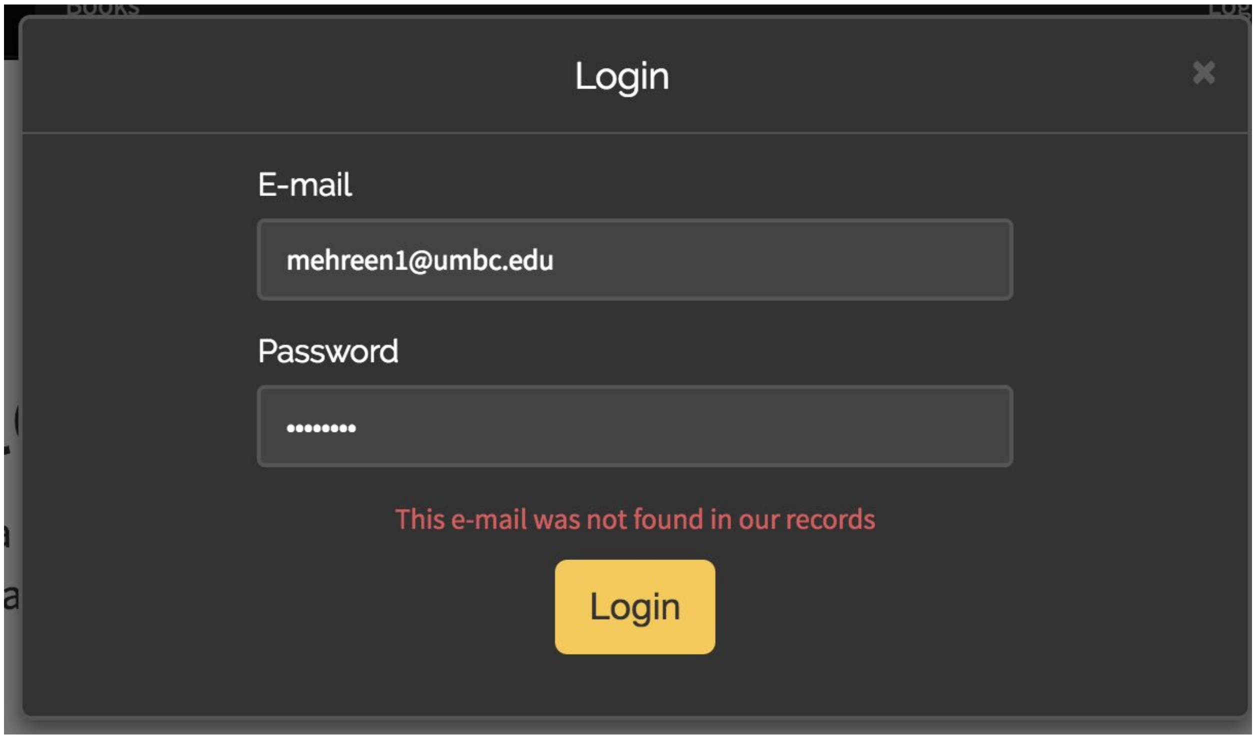
First Name

meh

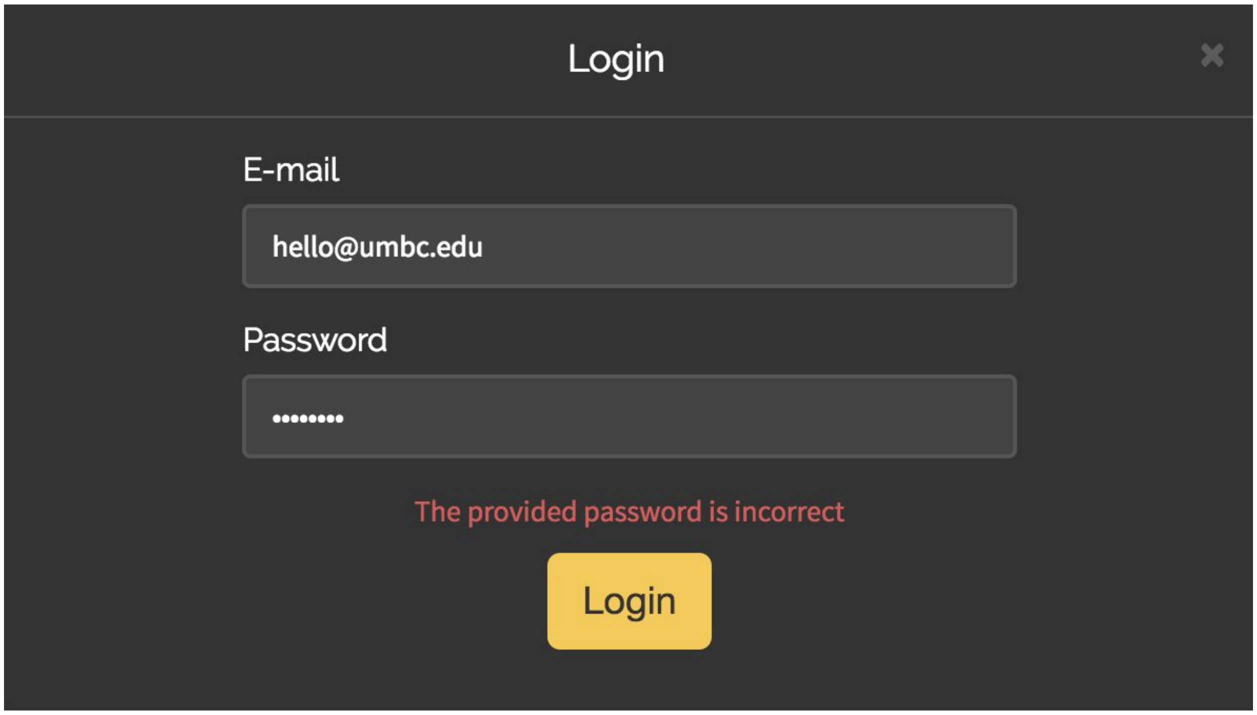
Last Name

reen

*User won't be able to register if they provide a non-umbc email or password less than 8 characters.*

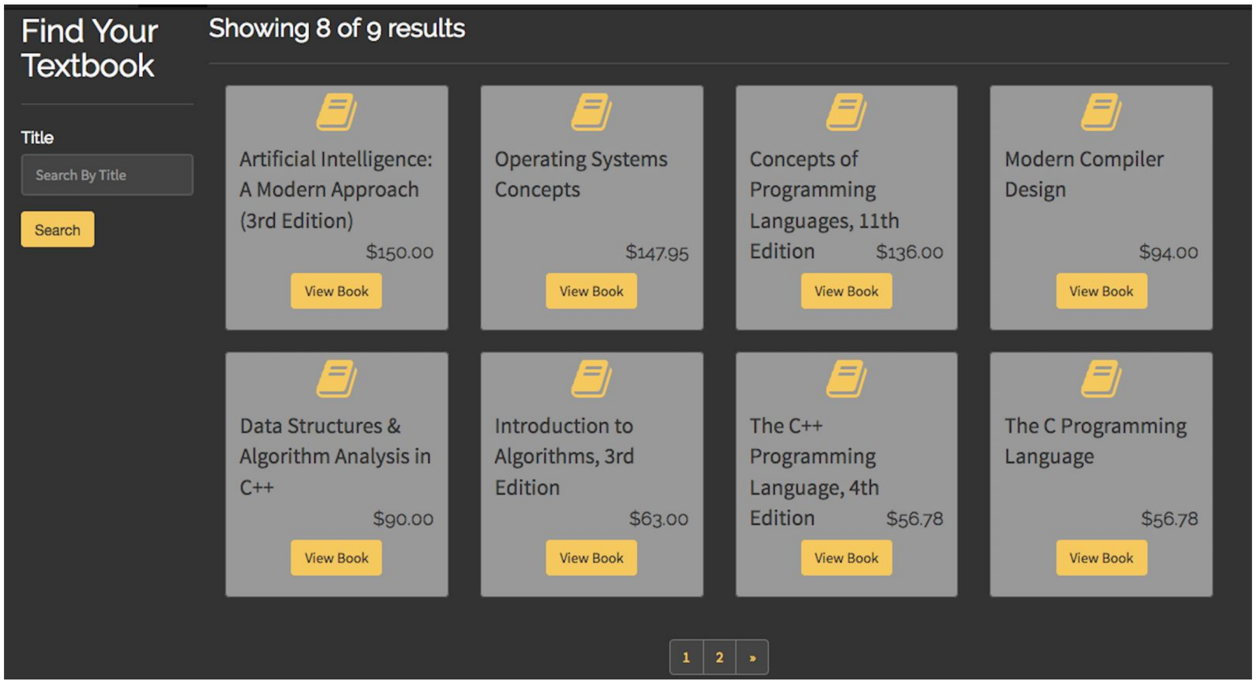


*User attempts to log in with an unregistered email. Error message pops up just like our expected result.*



*User enters invalid password - differing from the one we saved in our database associated with their email. Follows expected results.*

**3.2.2 Use Case: Search for Textbook**
**Conducted by: Mehreen Awan**
Valid Situation:


Find Your Textbook

Showing 8 of 9 results

Title

Search By Title

Search

Artificial Intelligence: A Modern Approach (3rd Edition) $150.00 View Book

Operating Systems Concepts $147.95 View Book

Concepts of Programming Languages, 11th Edition $136.00 View Book

Modern Compiler Design $94.00 View Book

Data Structures & Algorithm Analysis in C++ $90.00 View Book

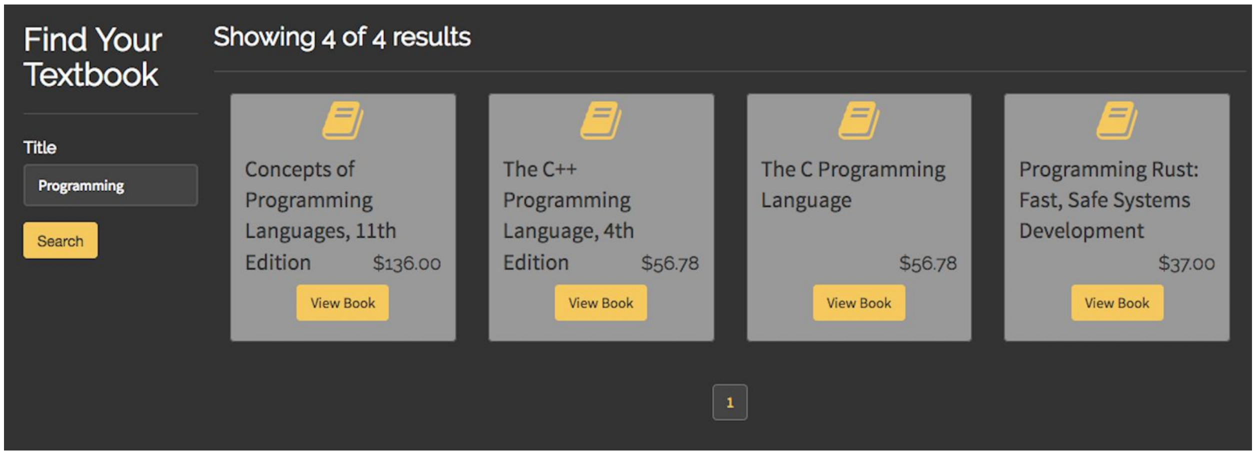
Introduction to Algorithms, 3rd Edition $63.00 View Book

The C++ Programming Language, 4th Edition $56.78 View Book

The C Programming Language $56.78 View Book

1 2 »

*Main search page lists all books available*


Find Your Textbook

Showing 4 of 4 results

Title

Programming

Search

Concepts of Programming Languages, 11th Edition $136.00 View Book

The C++ Programming Language, 4th Edition $56.78 View Book

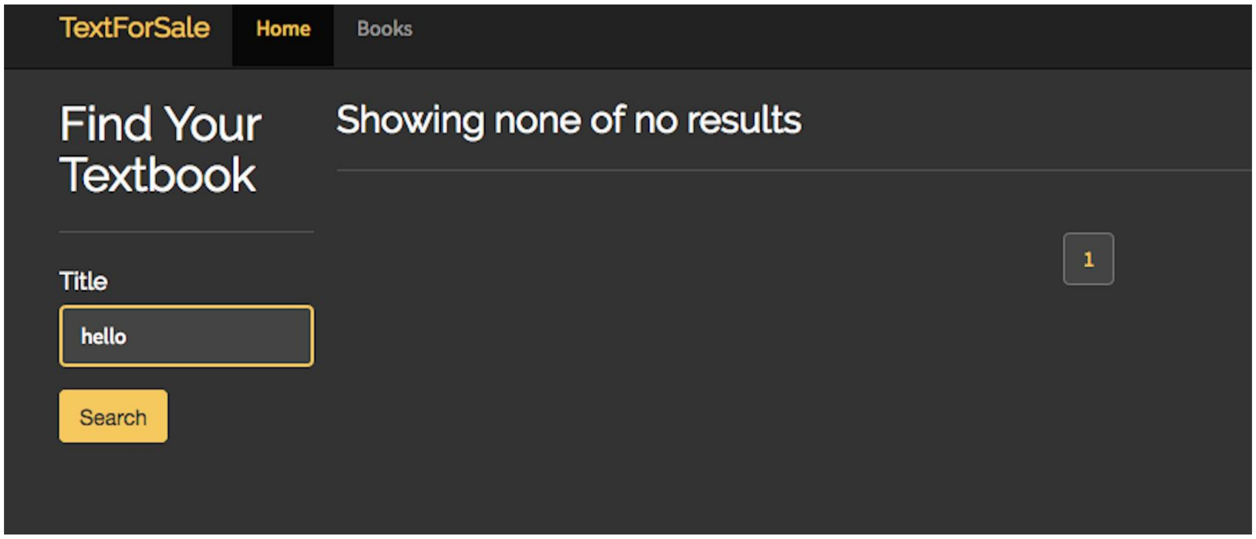
The C Programming Language $56.78 View Book

Programming Rust: Fast, Safe Systems Development $37.00 View Book

1

*User can view all books whose titles contain the keyword searched for*

Invalid Situation:



Showing none of no results

*User attempts to search for a book that doesn't exist in our database. Error message shown as expected.*

**3.2.3 Use Case: View User Profile**
*Not implemented yet*

**3.2.4 Use Case: Payment Option**
*Not implemented yet*

**3.2.5 Use Case: Get Redirected if Book is not Available**
*Not implemented yet*

**3.2.6 Use Case: Both Parties Verify Transaction is Complete**
*Not implemented yet*

**Appendix B – Team Review Sign-off**
This document has been collaboratively written by all members the team. Additionally, all team members have reviewed this document and agree on both the content and the format. Any disagreements or concerns are addressed in team comments below.

**Team**

______________________________________________
Print Name
______________________________________________ Date ____________________
Signature

Comments ________________________________________________________________________
_______________________________________________________________________________
__

______________________________________________
Print Name
______________________________________________ Date ____________________
Signature

Comments ________________________________________________________________________
_______________________________________________________________________________
__

______________________________________________
Print Name
______________________________________________ Date ____________________
Signature

Comments ________________________________________________________________________
_______________________________________________________________________________
__

______________________________________________
Print Name
______________________________________________ Date ____________________
Signature

Comments ________________________________________________________________________
_______________________________________________________________________________
__

______________________________________________
Print Name
______________________________________________ Date ____________________

Signature

Comments ______________________________________________________________________________
________________________________________________________________________________
__

**Appendix C – Document Contributions**

Jack Gordon drafted the testing suite. Mina Beshai write the description and impressions of the testing process. Daniel Schomisch came up with the use cases used for testing and appendices A and B. Mehreen Awan conducted the test results and provided screenshots. Daniel Kelly wrote the purpose of the document, the references, and assisted all other members in implementation of unit tests for the use cases.