

CS 320 Course Project Final Report

for

Event Center Scheduling System

Prepared by

Group Name: Scofield-Eavenson Ltd.

Matt Scofield
Benjamin Eavenson

11666037
11561401

matthew.scofield2@wsu.edu
benjamin.eavenson@wsu.com

Date: 12/11/2019

Table of Contents

1	Introduction	3
1.1	Project Overview.....	3
1.2	Definitions, Acronyms and Abbreviations	3
1.3	References and Acknowledgments	3
2	Design.....	4
2.1	System Modeling	4
	4
2.2	Interface Design.....	5
3	Implementation.....	7
3.1	Development Environment.....	7
3.2	Task Distribution	7
3.3	Challenges.....	7
4	Testing	8
4.1	Testing Plan.....	8
4.2	Tests for Functional Requirements.....	8
4.3	Tests for Non-functional Requirements	10
4.4	Hardware and Software Requirements.....	10
5	Analysis	11
6	Conclusion	12

1 Introduction

1.1 Project Overview

This product is an event schedule database web page intended for an event center office. It should be used to schedule and view events that are in the database in an easy to understand user interface. Users can first add a username and password and immediately begin adding/modifying/deleting events in the system.

1.2 Definitions, Acronyms and Abbreviations

None needed

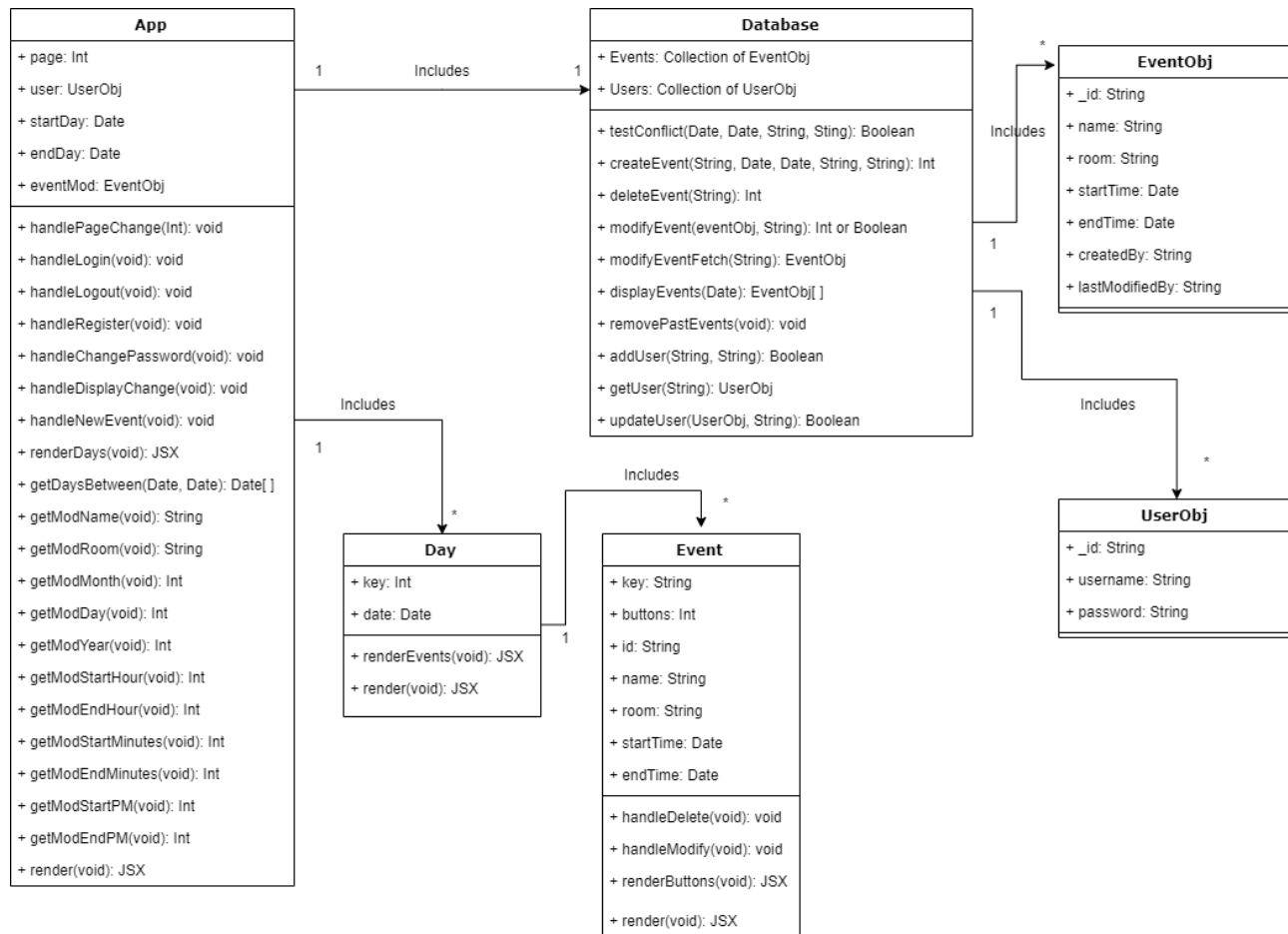
1.3 References and Acknowledgments

None needed

2 Design

2.1 System Modeling

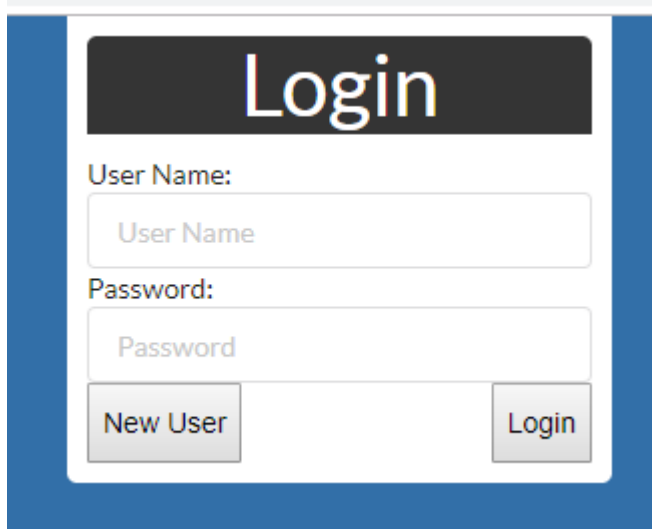
Class Diagram(updated)



Our implementation strictly follows the design document in milestone 2, except for the class diagram which changed quite a bit.

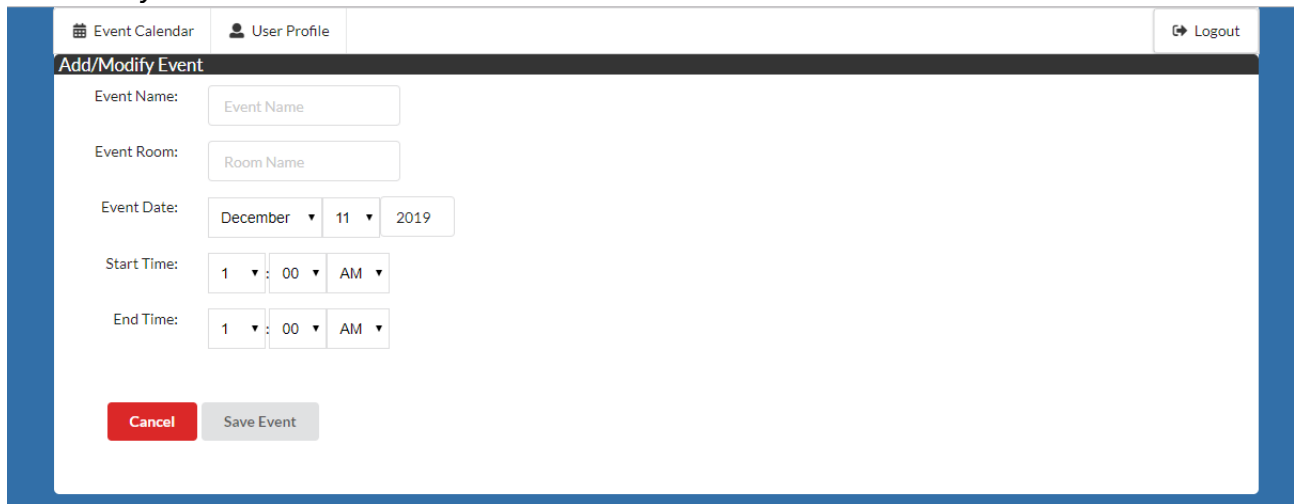
2.2 Interface Design

Login Screen



A login screen mockup with a blue border. At the top, a dark grey rectangle contains the word "Login" in white. Below this, the text "User Name:" is followed by a text input field containing the placeholder "User Name". Underneath, the text "Password:" is followed by a text input field containing the placeholder "Password". At the bottom, there are two buttons: "New User" on the left and "Login" on the right.

Add/Modify Screen



An "Add/Modify Event" screen mockup with a blue border. At the top, a navigation bar contains "Event Calendar" (with a calendar icon) and "User Profile" (with a user icon), and a "Logout" button (with a door icon) on the right. Below the navigation bar, a dark grey header reads "Add/Modify Event". The main form area contains the following fields:

- "Event Name:" followed by a text input field with placeholder "Event Name".
- "Event Room:" followed by a text input field with placeholder "Room Name".
- "Event Date:" followed by three dropdown menus showing "December", "11", and "2019".
- "Start Time:" followed by three dropdown menus showing "1", "00", and "AM".
- "End Time:" followed by three dropdown menus showing "1", "00", and "AM".

At the bottom left, there are two buttons: a red "Cancel" button and a grey "Save Event" button.

Display Screen

[New Event](#) [User Profile](#) [Logout](#)

Display Events

From December 12 2019 To December 13 2019 [Display](#)

Thu Dec 12 2019

test

Start Time: 1:00:00 AM
End Time: 2:00:00 AM
Room: test

[Modify Event](#)
[Delete Event](#)

test

Start Time: 1:00:00 AM
End Time: 2:00:00 AM
Room: test2

[Modify Event](#)
[Delete Event](#)

Fri Dec 13 2019
No Events are Scheduled for this day.

Add User Screen

New User Creation

User Name:

Password:

3 Implementation

3.1 Development Environment

For our project we used IntelliJ as our IDE. Our code was done using Meteor, Javascript, React and MongoDB. For our version control, we used git.

3.2 Task Distribution

Both Matt and Ben worked on the user facing web page design and layout, however final touch ups on the web pages themselves were implemented by Ben. Matt worked on the database interface and its testing, the back end. Ben worked on the front facing web pages such as React, user input and user display pages.

3.3 Challenges

This was the first time either of us had worked on a web based project with a database. The database was rather frustrating to set up in the beginning but went smoother than we expected. Testing posed a new set of problems as tests would often break as the product went through changes.

4 Testing

4.1 Testing Plan

The testing should be done as follows:

1. Run meteor testing command before each build, this will test the interfaces to the database which includes conflict detection, database adding, database modifying, and database deleting. This also includes several error handling cases.
 - a. Command: meteor test --full-app --driver-package meteortesting:mocha
2. Test that UI reacts and updates accordingly to requirements outlined in the Software Requirements Specification. UI testing should be performed for the final build, and for each component as it is completed.

4.2 Tests for Functional Requirements

1. **Database:**
 - a. Adding an event; passed
 - b. Modifying event; passed
 - c. Deleting event; passed
 - d. Start time after endtime for event add should result in error; passed
 - e. Attempting to modify event that does not exist should result in error; passed
 - f. Attempting to delete event that does not exist should result in error; passed
 - g. Attempting to add event before current date/time should result in error; passed
 - h. Attempt to add an event that conflicts with another should result in an error; passed
 - i. Attempt to modify event to conflict with another should result in an error; passed
2. **UI Testing:**
 - a. Login screen displays correctly on launch; passed
 - b. Attempting to login with a username that doesn't exist should result in an alert; passed
 - c. Attempting to login to a user account with an incorrect password should result in an alert; passed
 - d. Attempting to login with a valid username and matching password should result in the calendar page being displayed; passed
 - e. Clicking the New User button should display the New User page; passed
 - f. Attempting to register a new user with a username that already exists should result in an alert; passed
 - g. Attempting to register a new user with a password and confirmation password that don't match should result in an alert; passed
 - h. Attempting to register a new user with a valid username and valid password should result in the login screen being displayed; passed
 - i. Clicking the logout button anywhere should display the login screen;
 - i. display page; passed
 - ii. user profile page; passed
 - iii. new event page; passed
 - j. Clicking the new event button should display the add/modify page; passed
 - k. Clicking the event calendar button anywhere should display the event calendar;
 - i. User Profile Page; passed

- ii. Event add/modify page; passed
- l. Clicking the user profile button anywhere should display the user profile page;
 - i. Display page; passed
 - ii. Event add/modify page; passed
- m. Clicking the delete event button on an event should prompt the user to confirm deletion; passed
- n. Confirming deletion of an event should remove it from the calendar; passed
- o. Denying deletion of an event in the confirmation window does not delete the event; passed
- p. Clicking the modify event button on an event should display the add/modify page, with input fields pre-populated with the event's info; passed
- q. Clicking the display button should automatically refresh the display; passed
- r. Clicking the display button should update the display to display only events that are in the specified date range; passed
- s. Attempting to display a range of events that is greater than 5 years will result in an alert; passed
- t. Attempting to display a day that doesn't exist will result in an alert; passed
- u. Attempting to display a range of events with a start date that is after the end date will result in an alert; passed
- v. Attempting to add or modify an event with a start time that is after the end time will result in an alert; Passed
- w. Attempting to add or modify an event that shares a room and time with a pre-existing event will result in an alert; Passed
- x. Attempting to add or modify an event with a start time that is in the past will result in an alert;
 - i. Modifying: Passed
 - ii. Adding: Passed
- y. Attempting to add or modify an event with a non numerical value in the year field will result in an alert;
 - i. Modifying: Passed
 - ii. Adding: Passed
- z. Attempting to add or modify an event to have a date that doesn't exist will result in an alert;
 - i. Adding: Passed
 - ii. Modifying: Passed
- aa. Attempting to add or modify an event and leaving a field blank will result in an alert;
 - i. Event Name:
 - 1. Adding: Passed
 - 2. Modifying: Passed
 - ii. Event Room:
 - 1. Adding: Passed
 - 2. Modifying: Passed
 - iii. Year:
 - 1. Adding: Passed
 - 2. Modifying: Passed
- bb. Attempting to add or modify an event to have valid values will result in the calendar page being displayed, and the event being added to its appropriate spot;
 - i. Adding: Passed
 - ii. Modifying: Passed
- cc. Attempting to change a user password with an incorrect current password will result in an alert; Passed

- dd. Attempting to change a user password with a new password that doesn't match the confirmation password will result in an alert; Passed
- ee. Attempting to change a user password with valid values will result in a success alert; Passed

4.3 Tests for Non-functional Requirements

- 1. *All functions involving the web interface should take no longer than 5 seconds; passed*

4.4 Hardware and Software Requirements

Testing requirements:

- 1. Chrome browser
- 2. Meteor and its require packages installed on system

5 Analysis

Milestone 1:

Matt: 4 hours

Ben: 4 hours

Milestone 2:

Matt: 2 hours

Ben: 2 hours

Milestone 3:

Matt: about 20 hours

Ben: about 20 hours

Milestone 3 took the most amount of time because we both had to learn web frameworks and tools to implement the final product.

6 Conclusion

During our time working on this project, we learned many things about web app development. We used meteor for our database framework which took some time to learn. We also used react for our front end development which came with its own challenges. However, more importantly than learning the frameworks themselves, we developed the understanding on how to read documentation on frameworks to utilize them correctly and effectively.

Appendix A - Group Log

We met at least twice a week before and after class to discuss and implement our program. Most issues and questions were solved through this communication. Smaller points of interest were discussed over discord as they cropped up. This was effective enough to get the project into a form we both agreed with.