CS-499-T4251 Computer Science Capstone

Professor Goggin

Carlos Aguirre

Southern New Hampshire University

March 22, 2023

Milestone Three: Enhancement Two: Algorithms and Data Structure

- **Algorithms and Data Structure**

For this first category as part of my final project, I also decided to work with an artifact that was completed during **CS-360 Mobile Architect & Programming.**

Artifact: Event tracking App – Built in Java and Android Studio.

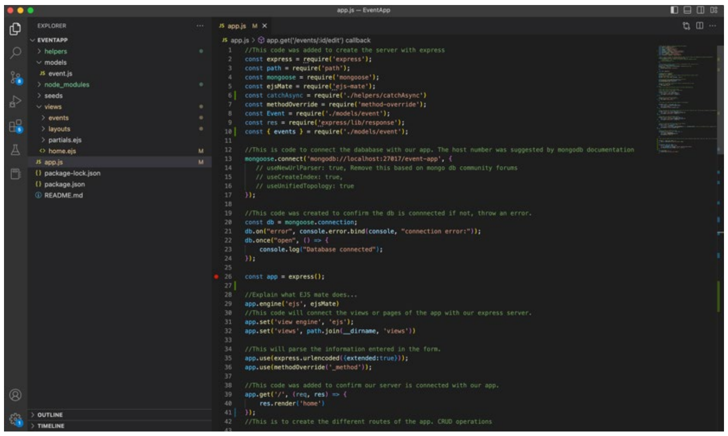
As mentioned before, the artifact was created as part of the CS-360 Mobile Architect & Programming course and was an Android Studio and Java-based event tracking app. For adding, removing, updating, and deleting Events, the app provides CRUD actions. The software was also connected to a database to simplify user account setup and data maintenance.

I made the decision to incorporate my intention to entirely rebuild an application using Javascript, another programming language, in my ePortfolio. I can also demonstrate that I have the abilities required to function as a full-stack engineer, a software engineer who can develop, testing, and utilizing a range of software applications. In this section of the project, I will be able to show that I understand various Javascript data structures like arrays, objects, stacks, and queues. A data structure is a grouping of data items, their connections, and the possible functions or operations on the data. These Javascript data structures are formats that make it easier to retrieve the data and make changes as needed.

Enhancements:

Both primitive (built-in) and non-primitive (not built-in) data structures are available in JavaScript. The programming language includes primitive data structures by default, and you can use them right away (like arrays and objects). For this project, many classes and methods are divided into the source code based on their purpose and mode of operation. This is required to keep our code organized, to make it easier to access the data, and to group data items, their relationships, and potential functions or actions.

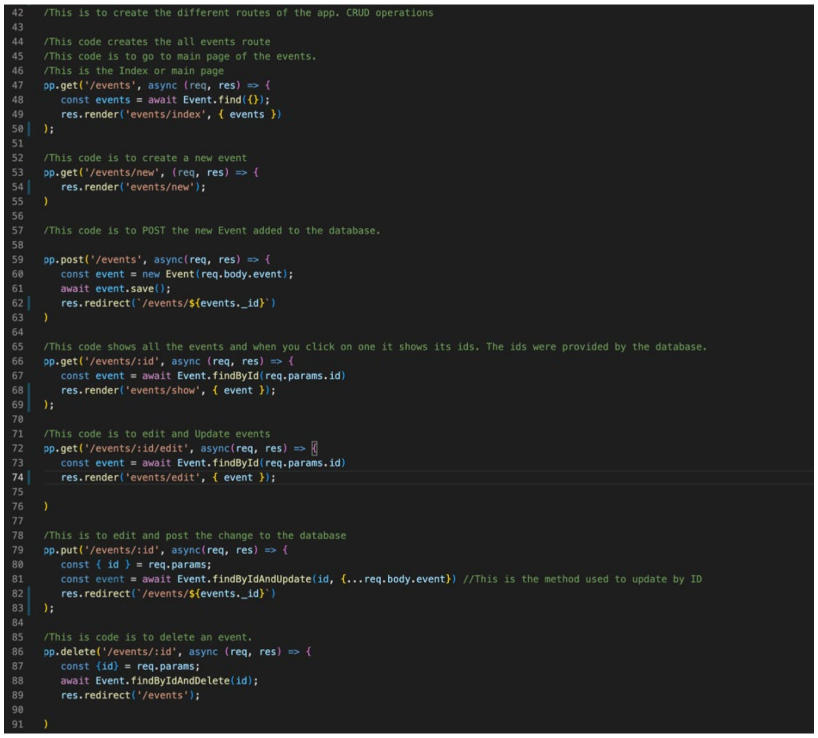
Milestone Three: Enhancement Two: Algorithms and Data Structure



```javascript
//This code was added to create the server with express
const express = require('express');
const path = require('path');
const mongoose = require('mongoose');
const ejsMate = require('ejs-mate');
const catchAsync = require('./helpers/catchAsync')
const methodOverride = require('method-override');
const Event = require('./models/event');
const res = require('express/lib/response');
const { events } = require('./models/event');

//This is code to connect the dababase with our app. The host number was suggested by mongodb documentation
mongoose.connect('mongodb://localhost:27017/event-app', {
   // useNewUrlParser: true, Remove this based on mongo db community forums
   // useCreateIndex: true,
   // useUnifiedTopology: true
});

//This code was created to confirm the db is connnected if not, throw an error.
const db = mongoose.connection;
db.on("error", console.error.bind(console, "connection error:"));
db.once("open", () => {
    console.log("Database connected");
});

const app = express();

//Explain what EJS mate does...
app.engine('ejs', ejsMate)
//This code will connect the views or pages of the app with our express server.
app.set('view engine', 'ejs');
app.set('views', path.join(__dirname, 'views'))

//This will parse the information entered in the form.
app.use(express.urlencoded({extended:true}));
app.use(methodOverride('_method'));

//This code was added to confirm our server is connected with our app.
app.get('/', (req, res) => {
    res.render('home')
});
//This is to create the different routes of the app. CRUD operations
```

The code has been divided into many classes and methods, as seen in the figure above. Developers maintain the database's data internally by implementing Create, Read, Update, and Delete after designing a Data Structure with the ability to write to the database.

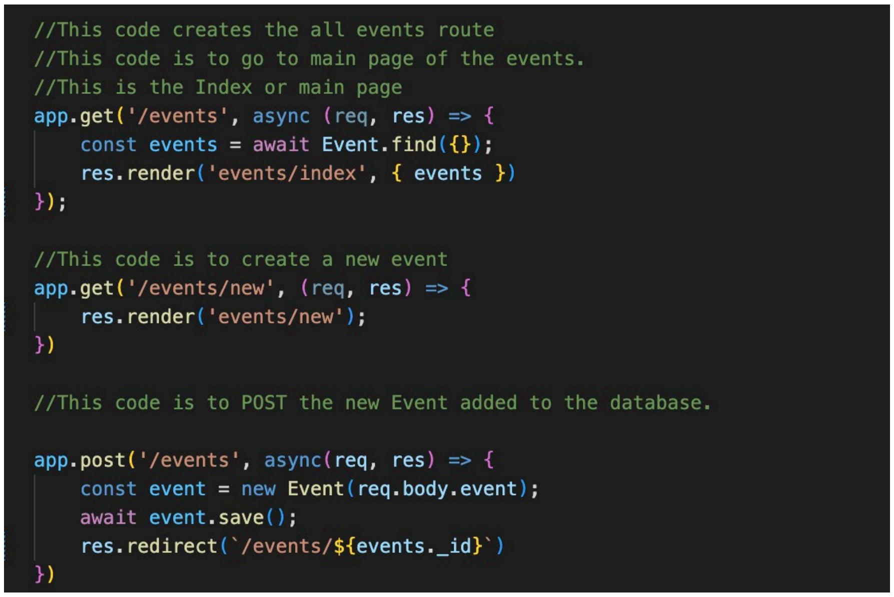
As I mentioned in my last milestone, the database was created with the name Event-App, and it was verified via a connected terminal. The database was then seeded after that. This is completed in order to begin developing, testing, and making sure that the CRUD functions within the web application are functioning properly.



```javascript
/This is to create the different routes of the app. CRUD operations

/This code creates the all events route
/This code is to go to main page of the events.
/This is the Index or main page
pp.get('/events', async (req, res) => {
   const events = await Event.find({});
   res.render('events/index', { events })
);

/This code is to create a new event
pp.get('/events/new', (req, res) => {
   res.render('events/new');
)

/This code is to POST the new Event added to the database.

pp.post('/events', async(req, res) => {
   const event = new Event(req.body.event);
   await event.save();
   res.redirect(`/events/${events._id}`)
)

/This code shows all the events and when you click on one it shows its ids. The ids were provided by the database.
pp.get('/events/:id', async (req, res) => {
   const event = await Event.findById(req.params.id)
   res.render('events/show', { event });
);

/This code is to edit and Update events
pp.get('/events/:id/edit', async(req, res) => {
   const event = await Event.findById(req.params.id)
   res.render('events/edit', { event });


)

/This is to edit and post the change to the database
pp.put('/events/:id', async(req, res) => {
   const { id } = req.params;
   const event = await Event.findByIdAndUpdate(id, {...req.body.event}) //This is the method used to update by ID
   res.redirect(`/events/${events._id}`)
);

/This is code is to delete an event.
pp.delete('/events/:id', async (req, res) => {
   const {id} = req.params;
   await Event.findByIdAndDelete(id);
   res.redirect('/events');

)
```

Milestone Three: Enhancement Two: Algorithms and Data Structure

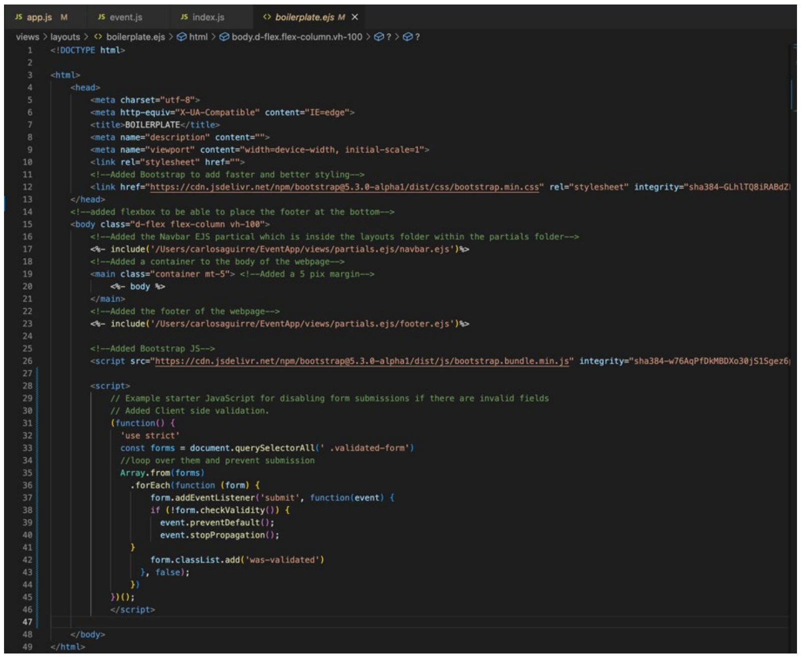
The ability to transfer the system requirements and design into code depends on an understanding of the algorithms required by the program. I utilized the proper function declaration throughout my code, which includes the function name, a list of the function's parameters contained in parentheses, and commas to denote separation. The curly bracketed JavaScript statements that define the function. Besides using Asynchronous programming is an approach that allows your software to begin a work that could take a while to complete while still being able to respond to other events without having to wait until that task is complete. When that task is complete, the outcome is displayed to your software.



```
//This code creates the all events route
//This code is to go to main page of the events.
//This is the Index or main page
app.get('/events', async (req, res) => {
    const events = await Event.find({});
    res.render('events/index', { events })
});

//This code is to create a new event
app.get('/events/new', (req, res) => {
    res.render('events/new');
})

//This code is to POST the new Event added to the database.

app.post('/events', async(req, res) => {
    const event = new Event(req.body.event);
    await event.save();
    res.redirect(`/events/${events._id}`)
})
```

The source code is neatly structured and consistently indented and line broken. We employ the appropriate syntax and conventions in accordance with best practices and applications in programming. The saved variable values can be used effectively in methods for other classes thanks to the data structures that were constructed through programming. Method names are verbs because they describe activities being performed on something.

Also, I used boilerplates, which is a term for code that needs to be used again with little to no modification. It is frequently used to describe programming languages that are thought of as lengthy, requiring a lot of code to accomplish simple tasks.


app.js M
event.js
index.js
boilerplate.ejs M
views > layouts > boilerplate.ejs > html > body.d-flex.flex-column.vh-100 > ? > ?

```
<!DOCTYPE html>

<html>
    <head>
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <title>BOILERPLATE</title>
        <meta name="description" content="">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <link rel="stylesheet" href="">
        <!--Added Bootstrap to add faster and better styling-->
        <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-GLhlTQ8iRABdZ
    </head>
    <!--added flexbox to be able to place the footer at the bottom-->
    <body class="d-flex flex-column vh-100">
        <!--Added the Navbar EJS partical which is inside the layouts folder within the partials folder-->
        <%- include('/Users/carlosaguirre/EventApp/views/partials.ejs/navbar.ejs')%>
        <!--Added a container to the body of the webpage-->
        <main class="container mt-5"> <!--Added a 5 pix margin-->
            <%- body %>
        </main>
        <!--Added the footer of the webpage-->
        <%- include('/Users/carlosaguirre/EventApp/views/partials.ejs/footer.ejs')%>

        <!--Added Bootstrap JS-->
        <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/js/bootstrap.bundle.min.js" integrity="sha384-w76AqPfDkMBDXo30jS1Sgez6

        <script>
            // Example starter JavaScript for disabling form submissions if there are invalid fields
            // Added Client side validation.
            (function() {
              'use strict'
              const forms = document.querySelectorAll(' .validated-form')
              //loop over them and prevent submission
              Array.from(forms)
                .forEach(function (form) {
                    form.addEventListener('submit', function(event) {
                    if (!form.checkValidity()) {
                      event.preventDefault();
                      event.stopPropagation();
                  }
                    form.classList.add('was-validated')
                  }, false);
                })
            })();
            </script>

    </body>
</html>
```

The application is currently working as expected, there a few bugs I need to fix but before I can say is perfect. Since last time I presented this web app. It was completed with all the main logic, I already started styling the application to be ready for user interaction.

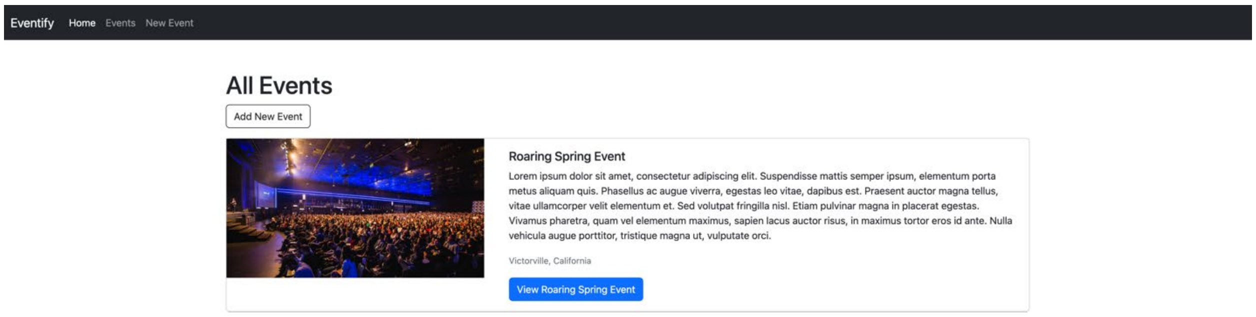
There are a few problems I need to fix before I can declare the application is perfect, but it is currently functioning as planned. From the last time I showed this web application. The application's primary functionality was finished, and I had already begun styling it in preparation for user interaction.

In essence, EJS partials are views designed to be used inside other views, and I used them. For sharing the same markup across numerous layouts, partials, and views, they are extremely useful. Instead of generating style sheets from scratch, I was able to use bootstrap to speed up the development process and yet create a high-quality webapp.

Milestone Three: Enhancement Two: Algorithms and Data Structure

The Webapp now offers the ability to CRUD events, but it also offers the option to give a thorough description, add a pricing, and submit a picture or even a flyer, all of which were not included in the original application created back in CS-360.

The following are screenshots if the current views of the app:



Eventify Home Events New Event

All Events

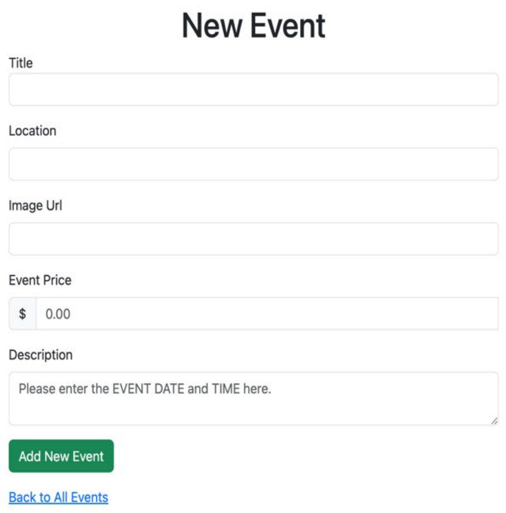
Add New Event

Roaring Spring Event

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Suspendisse mattis semper ipsum, elementum porta metus aliquam quis. Phasellus ac augue viverra, egestas leo vitae, dapibus est. Praesent auctor magna tellus, vitae ullamcorper velit elementum et. Sed volutpat fringilla nisl. Etiam pulvinar magna in placerat egestas. Vivamus pharetra, quam vel elementum maximus, sapien lacus auctor risus, in maximus tortor eros id ante. Nulla vehicula augue porttitor, tristique magna ut, vulputate orci.

Victorville, California

View Roaring Spring Event



New Event

Title

Location

Image Url

Event Price

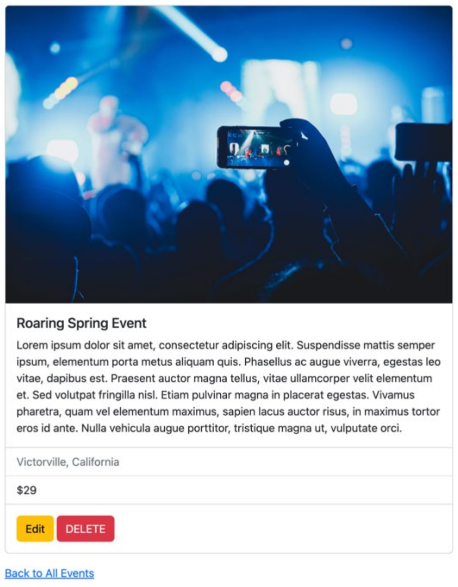
$ 0.00

Description

Please enter the EVENT DATE and TIME here.

Add New Event

Back to All Events



Roaring Spring Event

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Suspendisse mattis semper ipsum, elementum porta metus aliquam quis. Phasellus ac augue viverra, egestas leo vitae, dapibus est. Praesent auctor magna tellus, vitae ullamcorper velit elementum et. Sed volutpat fringilla nisl. Etiam pulvinar magna in placerat egestas. Vivamus pharetra, quam vel elementum maximus, sapien lacus auctor risus, in maximus tortor eros id ante. Nulla vehicula augue porttitor, tristique magna ut, vulputate orci.

Victorville, California

$29

Edit DELETE

Back to All Events

Finishing the application is the next phase. I'm currently working on refining the style and perfecting the authentication and authorization process. I also want to be able to add client-side validations, which can significantly alter the user experience, as well as error handling. I would also like to be able to apply the review mechanism if I have the time.

Milestone Three: Enhancement Two: Algorithms and Data Structure

It's excellent to see that the artifact is almost complete. I appreciated seeing how well the application matched the system's criteria and the initial idea that I had in mind. Working closely with the online documentation for the technology once more is something that has really benefited me. Other than time, the only challenge I've faced is debugging. Debugging takes time even though it is a skill that everyone should have. This method aids in locating and eliminating flaws or faults in any software's source code**.** Software developers examine the code to ascertain the cause of any faults that may have happened when software does not function as planned. A few problems I encountered were mostly the result of typos.

References:

"JavaScript Data Types and Data Structures." MDN Web Docs, 23 Oct. 2019, https://developer.mozilla.org/en-US/docs/Web/JavaScript/Data_structures