**Indiana University Southeast**

**2021-22 CSCI Capstone Project**

**Workshop Management Web Application**

**Sponsor: Dr. Suranga Hettiarachchi**

**Programmer’s Manual**

**James Schlesener**

**Vision Statement**

The Workshop Management Web Application allows users to manage workshops, the participants attending the workshops, and any equipment that will be assigned to participants.

**Introduction**

The new 2020 Indiana Education Preparation Standards along with the 2018 state STEM Strategic Plan initiative require new programming for the Indiana University Southeast (IUS) campus. A partnership between the School of Natural Science and the School of Education of IUS will help to fill the need created with the new standards. This partnership focuses on creating opportunities for current elementary teachers to learn new technologies and practices in programming and robotics and on sharing ideas and teaching strategies through professional development and curriculum alignment to the new standards. As part of this initiative, workshops will be held for teachers and students. This web application helps manage these workshops by tracking the workshops, participants, and equipment.

**Component Overview**

**CLIENT**

1. **Login**: Default screen where the user logs into the web application. A JSON Web Token (JWT) to authenticate the user is received from the server, and the client routes to the Workshops screen.
2. **Register**: Accessible from the Login screen. The user can register a new username and password. A JWT to authenticate the user is received from the server, and the client routes to the Workshops screen.
3. **Workshops**: Displays a list of all the workshops from the database. A drop-down menu is available to show all, past, current, or future workshops. A link to add a new workshop is provided on the navigation bar.
4. **Participants**: Displays a list of all participants from the database. A drop-down menu is available to show all, assigned, and unassigned participants. A link to add a new participant is provided on the navigation bar.
5. **Equipment**: Displays a list of all equipment from the database. A drop-down menu is available to show all, checked out, and not checked out equipment. A link to add new equipment is provided on the navigation bar.
6. **View Workshop**: Displays an individual workshop along with any participants enrolled and any equipment that is checked out to participants of the workshop. Links to edit the workshop and add a participant are provided on the navigation bar. A trash can icon is available to delete the workshop from the database.
7. **View Participant**: Displays an individual participant along with their workshop they are enrolled in and any equipment they have checked out. Links to edit the participant and add equipment are provided on the navigation bar. A trash can icon is available to delete the participant from the database if the participant doesn’t have any equipment checked out.
8. **View Equipment**: Displays an individual piece of equipment along with the participant who has the equipment checked out and the workshop they are enrolled in. A link to edit the equipment is provided on the navigation bar. A trash can icon is available to delete the equipment from the database.
9. **Add Workshop**: A reactive form with input validation that allows the user to add a workshop to the database.
10. **Edit Workshop**: A reactive form with input validation that allows the user to edit the details of a workshop.
11. **Add Participant**: A reactive form with input validation that allows the user to add a participant to the database. The user can assign the participant to any workshop or have them be unassigned.
12. **Edit Participant**: A reactive form with input validation that allows the user to edit the details of a workshop. The user can reassign a participant to a different workshop or have them be unassigned.
13. **Add Equipment**: A reactive form with input validation that allows the user to add equipment to the database. The use can assign the equipment to any participant or have it not checked out.
14. **Edit Equipment**: A reactive form with input validation that allows the user to edit the details of a piece of equipment. The user can change the image, assign the equipment to any participant, or have it not checked out.
15. **Navigation**: Creates the header and navigation bar that is on every screen. Communicates with the Navbar service to display active options.
16. **App Routing Module**: Routing to direct the client to a component based on the URL.
17. **App Component**: Sets up the HTML structure for the header/navigation bar and the app components that are driven by the App Routing Module.
18. **Index**: The header and body for the HTML page.
19. **Environment**: Stores the URL for the server and the name of the folder on the server where the uploaded images of the equipment will be stored.
20. **Services**
    1. **Server**: The components call this service, and the service sends the http calls to the server.
    2. **Auth-guard**: The service checks to see if the user is logged in.
    3. **Auth**: This service gets, stores, and clears the JWT from local storage.
    4. **Navbar**: The components call this service, and the service sets the corresponding option on the navigation bar to active or inactive.
21. **Pipes**
    1. **Convert2ampm**: Converts a 24 hour time to a 12 hour time for display purposes.
    2. **Phone**: Converts a 10 digit string to a (000) 000-0000 format for display purposes.

**SERVER**

1. **Auth**: Verifies that the user’s JWT is valid.
2. **Events**: Handles the client’s http requests.
3. **Index**: Starts the express server.
4. **Uploaded\_images**: Folder on the server where the uploaded images are stored.

**Tool Overview**

1. Angular: The client side of the web application was created using Angular. Angular is an open-source web application framework developed by Google used to implement well-designed and structured webpages and applications. It provides all the functionality needed to handle user input in the browser, manipulate data on the client side, and control how elements are displayed in the browser view.
2. Express: The web server was created using Express. Express is a Node.js module that provides several key components for handling web requests. It provides the ability to set up destination routes for users to connect to and provides great functionality on working with HTTP request and response objects.
3. Node.js: The server side of the web application was created using Node.js. Node is a website/application framework used to write backend services, server-side scripts, and any supporting web application functionality.
4. Typescript: A strongly typed, object-oriented, compiled language. It is a super set of Javascript.
5. MySQL: The database for this web application was created using MySQL. MySQL is an open-source relational database management system distributed and supported by Oracle Corporation. It stores all the data for the users, workshops, participants, and equipment that are used by the Workshop Management web application.
6. Multer: A node.js middleware for handling multipart/form-data; used for uploading files.
7. MD5: A hashing algorithm that uses a complex mathematical formula to create a hash. It is used to hash the user password.
8. Cors: A node.js package for providing a Connect/Express middleware that can be used to enable Cross-Origin Resource Sharing with various options. It is a browser security feature that restricts cross-origin HTTP requests with other servers and specifies which domains access the resources.
9. BodyParser: A node.js middleware for parsing incoming request bodies.
10. Jsonwebtoken: A compact token format intended for space constrained environments such as HTTP Authorizations headers. It is used to provide user authentication.
11. Rjxs: Reactive Extensions for JavaScript is a reactive library used to implement reactive programming to deal with async implementation, callbacks, and event-based programs.

**Project Repository**

1. **Software**: <https://github.com/Schlesener73/Capstone-Workshop-Management>
2. **Test Cases**: <https://github.com/Schlesener73/Capstone-Workshop-Management/blob/main/Docs/Reports/RS-7%20Test%20Case%20Report.docx>
3. **Documentation**: <https://github.com/Schlesener73/Capstone-Workshop-Management/tree/main/Docs/Reports>
4. **Test platform description**: <https://github.com/Schlesener73/Capstone-Workshop-Management/blob/main/Docs/Reports/RS-6%20Software%20Test%20Plan.docx>

**Installation for a New Install**

**DEVELOPMENT**

Database

1. Install MySQL for Windows: <https://dev.mysql.com/doc/refman/8.0/en/windows-Installation.html>
2. Update the server index.js file with your host, user, password, and database information.
3. Create a folder on your server and name it “uploaded\_images”. It should correspond to the folder specified in the server event.js file (assigned to the fileDir variable).
4. Create the following tables in your database:

CREATE TABLE users

(

id SMALLINT UNSIGNED NOT NULL AUTO\_INCREMENT PRIMARY KEY,

username VARCHAR(50) NOT NULL,

password VARCHAR(50) NOT NULL

);

CREATE TABLE workshops

(

id INT unsigned NOT NULL AUTO\_INCREMENT, # Unique ID for the record

start DATE NOT NULL, # Workshop start date

end DATE NOT NULL, # Workshop end date

meet TIME NOT NULL, # Meeting time

location VARCHAR(100) NOT NULL, # Meeting location

numofpart INT NOT NULL, # Number of participants

frequency VARCHAR(100) NOT NULL, # Frequency of meetings

PRIMARY KEY (id) # Make the id the primary key

);

CREATE TABLE participants

(

id INT unsigned NOT NULL AUTO\_INCREMENT PRIMARY KEY,

first\_name TEXT NOT NULL,

last\_name TEXT NOT NULL,

address VARCHAR(100),

city TEXT,

state TEXT,

zip INT unsigned,

email VARCHAR(100),

phone VARCHAR(15),

workshop\_id INT unsigned,

CONSTRAINT fk\_workshop

FOREIGN KEY (workshop\_id)

REFERENCES workshops(id)

ON DELETE SET NULL

);

CREATE TABLE equipment

(

id INT unsigned NOT NULL AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(100) NOT NULL,

storage\_loc VARCHAR(100),

year INT unsigned,

image TEXT,

eq\_condition VARCHAR(100),

participant\_id INT unsigned,

CONSTRAINT fk\_participant

FOREIGN KEY (participant\_id)

REFERENCES participants(id)

);

1. Start MySQL locally by entering ‘mysqld –console’ from your MySQL bin directory.

Server

1. Install node.js for Windows: https://nodejs.org/en/download/
2. From the Windows Command Prompt:
   1. Install Express: npm install express
   2. Install Multer: npm install multer
   3. Install MD5: npm install md5
   4. Install Cors: npm install cors
   5. Install BodyParser: npm install body-parser
   6. Install Jsonwebtoken: npm install jsonwebtoken
   7. Download the WM-server files.
   8. From the WM-server directory, enter ‘node src/index.js’ to start the server locally.

Client

1. From the Windows Command Prompt:
   1. Install TypeScript: npm install -g typescript
   2. Install Angular CLI: npm install -g @angular/cli
   3. Install Rjxs: npm install rxjs
   4. Download the WM-client files.
   5. From the WM-client directory, enter ‘ng serve’ to start the client locally
2. Navigate to ‘http://localhost:4200’ in your web browser to view the Workshop Management Web Application.

**PRODUCTION**

Server

1. Update the server index.js file with your host, user, password, and database information.
2. Install node.js on your production server.
3. Upload WM-server code to the web server.
4. Run ‘npm install’
5. Create a folder on your server and name it “uploaded\_images”. It should correspond to the folder specified in the server event.js file (assigned to the fileDir variable).
6. Start the web server with the NODE\_ENV=production parameter.
7. Create the following tables in your MySQL database:

CREATE TABLE users

(

id SMALLINT UNSIGNED NOT NULL AUTO\_INCREMENT PRIMARY KEY,

username VARCHAR(50) NOT NULL,

password VARCHAR(50) NOT NULL

);

CREATE TABLE workshops

(

id INT unsigned NOT NULL AUTO\_INCREMENT, # Unique ID for the record

start DATE NOT NULL, # Workshop start date

end DATE NOT NULL, # Workshop end date

meet TIME NOT NULL, # Meeting time

location VARCHAR(100) NOT NULL, # Meeting location

numofpart INT NOT NULL, # Number of participants

frequency VARCHAR(100) NOT NULL, # Frequency of meetings

PRIMARY KEY (id) # Make the id the primary key

);

CREATE TABLE participants

(

id INT unsigned NOT NULL AUTO\_INCREMENT PRIMARY KEY,

first\_name TEXT NOT NULL,

last\_name TEXT NOT NULL,

address VARCHAR(100),

city TEXT,

state TEXT,

zip INT unsigned,

email VARCHAR(100),

phone VARCHAR(15),

workshop\_id INT unsigned,

CONSTRAINT fk\_workshop

FOREIGN KEY (workshop\_id)

REFERENCES workshops(id)

ON DELETE SET NULL

);

CREATE TABLE equipment

(

id INT unsigned NOT NULL AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(100) NOT NULL,

storage\_loc VARCHAR(100),

year INT unsigned,

image TEXT,

eq\_condition VARCHAR(100),

participant\_id INT unsigned,

CONSTRAINT fk\_participant

FOREIGN KEY (participant\_id)

REFERENCES participants(id)

);

Client

1. Install node.js for Windows: https://nodejs.org/en/download/
2. From the Windows Command Prompt:
   1. Install TypeScript: npm install -g typescript
   2. Install Angular CLI: npm install -g @angular/cli
   3. Install Rjxs: npm install rxjs
   4. Download the WM-client files.
   5. From the WM-client directory, enter ‘ng build –prod’ to build the production files for the client.
   6. Navigate to the ‘dist’ directory. Upload these production files to your client.

**Installation for a New Platform**

To install the software on a new platform, follow the same steps that were followed for the Windows version. However, instead of installing the windows version, find the equivalent downloads for the new platform.

**Further Development Statement**

If I had another year to do this, I would further develop the software. I would upgrade the login/registration feature by verifying if the user is already registered, storing the user’s name and email address, and adding the ability to recover a password. I would enhance the user interface by changing the navigation menu so that menu items don’t move depending on which screen the user is on, displaying the user’s name once logged in, and showing equipment image thumbnails on the view all equipment screen. I would add a confirm delete feature to make sure the user doesn’t accidentally delete an item. I would allow participants to be added to multiple workshops. I would create an admin screen where the user could backup the database and view logs of all actions done and by whom.