COMP229 – Web Application Development

Final Team Project - Value 55%

Part 1 – Team Contract – Value 5%

- ✦ Teams Formed – 4 to 6 members
- ✦ Suggested Roles:
  - ☐ Project Manager
  - ☐ Lead Software Engineer
  - ☐ UI Programmer
  - ☐ Security Programmer
  - ☐ Database Programmer
  - ☐ Web Designer
- ✦ Project Selection (Sell Used Products Site, Tournament Site, Ticket Management)
- ✦ Agile Tracking Tool Selected (Trello or Jira)

Due end of Week 8 @ 11:59 PM

Part 2 – First Release – Value 10%

- ✦ Initial Project Structure and Landing Pages (top level)
- ✦ NoSQL Database structure and Model classes in place
- ✦ Team GitHub Repo Setup
- ✦ Cloud Provider Setup (Heroku Recommended)
- ✦ Cloud Database Setup (MongoDB Atlas or Firebase)
- ✦ Agile Tracking – Backlog Snapshot
- ✦ Demo Video of First Release (10 minutes)

Due end of Week 9 (End of week) @ 11:59 PM

Part 3 – Authentication Release – Value 10%

- ✦ Site Security Functional
- ✦ Authentication Navigation added
- ✦ Agile Tracking – Backlog Snapshot
- ✦ Demo Video of this Release (10 minutes)

Due end of Week 11 @ 11:59 PM

Part 4 – Final Release – Value 15%

- ✦ Final Touches and Code Features
- ✦ Improve Visual Appeal
- ✦ Cloud Provider Deployment (Heroku Recommended)
- ✦ Cloud Database Deployment (MongoDB Atlas or Firebase)
- ✦ Agile Tracking – Backlog Snapshot and reporting

Due end of Week 13 @ 11:59 PM

Part 5 – Final Presentation and Demo – Value 15%
✦ Final Version of External Design Document
✦ Video Demo of the Final Presentation
Due end of Week 14 @ 11:59 PM.

Final Team Project   Maximum Mark: 100

Overview: Working in a group (up to 6 members) and utilizing your accumulated knowledge of the MEAN Stack (MongoDB, ExpressJS and NodeJS with AngularJS) create a Web App from one of the following type of projects (details below):

- Sell Used Products Site

- Bracket / Tournament Web App

- Help Desk Management Web App

- Another type of project previously approved.

    o **Note:** project not approved will be graded with zero (0).

Your Web App **must be hosted on a live site** that supports a **Node server**. You will have core functionality requirements as well as specific criteria for your type of Web App.

Instructions:

COMMON REQUIREMENTS (40 MARKS)

Each Web App will share these common requirements:

1. Your Web App must be designed using well-structured semantic HTML, CSS, JavaScript and implement a responsive frontend framework (e.g. Bootstrap or Angular Material) along with a dynamic backend (using Node, Express, and MongoDB or Firebase)   (5 marks):

    a. Your Site must be **responsive** and adapt to various viewport sizes. **Note:** Using a mobile-first framework such as Bootstrap, Angular Material or Foundation is highly recommended

    b. Your **CSS** rules reside in separate file(s) in their own folder and adhere to best practices. You may use SASS and Compass for additional functionality

    c. Your **JavaScript, Typescript** files, libraries (and other external code) are contained in their own folder and are appropriately linked to your site.

    d. Your **images and multimedia assets** are contained in their own folder and appropriately linked to your site. **Note:** You may need to provide additional versions of your multimedia assets to accommodate various viewport sizes

    e. All Your Code (HTML, CSS, JavaScript, jQuery, etc.) is error free

    f. Your MongoDB database must be hosted online. **Note:** It is recommended that you use a provider like MongoDB Atlas or Google Firebase

2. Include **Internal Documentation** for your site (5 marks):
    a. Ensure you include a **comment header** for any **HTML**, **CSS, and JavaScript files** that you create or modify that include: The **Student IDs** of all Developers, **web app name**, **brief description**
    b. Ensure you include a **section headers** or **brief comments** for all of your **HTML structure, CSS style sections,** and any **JavaScript functions, TypeScript Classes, etc.**
    c. Ensure all your code uses **contextual variable names** that help make the files human readable.
    d. Ensure you include **inline comments** that describe your GUI Design and Functionality. **Note**: please avoid "over-commenting"

3. Share your files on **GitHub** to demonstrate Version Control Best Practices and push your site to a cloud host (20 marks):
    a. Your repository must include **your code** and be well structured
    b. Your repository must include **commits** that demonstrate the project being updated at different stages of development – each time a major change is implemented
    c. You must deploy your site to your Cloud Server using **git** (10 marks).

4. Create a Short Video presentation on **YouTube** or another streaming provider. You must include a short **PowerPoint** (or Google Slides) Slide Deck that includes a **single slide** to start your video (10 marks):
    a. The **first** (and only) **Slide** of your Slide Deck must include a **current image** of you (no avatars allowed) that is displayed appropriately on the page. You must also include your **Full Name**, **Student ID**, the **Course Code**, **Course Name,** and your **Assignment** information.
    b. You will **demonstrate** your site's functionality. You must show each page working properly
    c. You will **describe** the code in your app.js file that drives the functionality of your site
    d. Sound for your Video must at an appropriate level so that your voice may be **clearly heard,** and your screen resolution should be set so that your code and site details are **clearly visible**
    e. Your Short Video should run no more than **10 minutes.**

APP SPECIFIC REQUIREMENTS (60 MARKS)

SELL USED PRODUCTS SITE (OPTION 1)
(30 Marks: GUI, 30 Marks: Functionality)
1. User Management and site security

a. **User Registration** must be included. A form will allow the user to enter profile information (**username**, **password, email address, etc.)**, which will be stored in a database structure

b. The user will be able to **Login**, **Logout** and **modify** his or her profile

c. **Site security** will prevent any non-registered users from **creating, editing, and deleting** an ad or entering secure areas of the site.

2. Users can **Post an Ad:**

a. After a user is **registered** and **logged in**, he or she can post an advertisement.

b. The user should be able to create a **lifetime** for the advertisement (i.e. when the it becomes **active/enabled** and when it **expires**).

3. Anonymous users can **send questions** to any active ad:

a. Anonymous users should be able to **select** an active ad and send questions.

b. Only the user who published the ad should be able to post an answer for a question.

c. Questions and answers will be stored in the database for later use.

d. Questions and answers must be available for anonymous visualization.

4. Secure Section

a. A **registered user** will be able to **edit** any advertisement that he or she **owns**.

b. Even a **registered user** will **NOT** be able to **delete** any advertisement. In that case, he or she must **disable or expire** the ad.

TOURNEY BRACKETS SITE (OPTION 2)
(30 Marks: GUI, 30 Marks: Functionality)

1. User Management and site security

a. **User Registration** must be included. A form will allow the user to enter profile information (**username**, **password, email address, etc.),** which will be stored in a MongoDB database table

b. The user will be able to **Login**, **Logout** and **modify** his or her profile

c. **Site security** will prevent non-registered users from **creating, editing, and deleting** tournaments or entering secure areas of the site.

2. Create A New Tournament

a. After a user is **registered** and **logged in**, he or she can create a **new Single Elimination Tournament.** Each Tournament will have a name and include a short description

b. When a user creates a Tournament, he is considered the **owner** and can choose when it becomes **active** and when it is **completed.**

3. Users Can Register Players

a. Users can register players by completing a form that includes several player names fields. A fixed number of players will be allowed to be registered (suggested 8 or 16 players)

b. An **Active Tournament** will display a list of players to Anonymous users

c. Depending on the number of players there will be a fixed number of Tournament matches initially (4 matches for 8 players and 8 matches for 16 players). Opponent selection for each match will be decided in player-registration order.

4. Tournament Management

   a. A Tournament can be **started** only after all players have been registered and the initial matches determined.

   b. Each Tournament Matches will be displayed, and its result if it is completed in the page of the tournament.

   c. The **Tournament owner** will be able to select the winners for each match.

   d. Only the **winners** will be carried forward into new matches for each successive match.

   e. When the final match is completed, a summary of the results will be also displayed, and the status of the tournament changes to **completed.**

HELP DESK SITE (OPTION 3)
(30 Marks: GUI, 30 Marks: Functionality)

1. User Management and site security

   a. **User Registration** must be included. A form will allow the user to enter profile information (**username**, **password, email address, user type)** which will be stored in a MongoDB database table

   b. The user will be able to **Login**, **Logout** and **modify** his or her profile

   c. **Site security** will prevent non-registered users from **creating, editing, and deleting** ticket records, changing ticket status, posting a comment, or seeing the ticket log.

2. Ticket Dashboard (Log)

   a. After a user is **registered** and **logged in**, he or she can view a Ticket **Dashboard** that will display all open tickets in a clickable list format

   b. The Dashboard will include an option that allows the user to **create a new ticket.**

   c. Tickets that are closed will be initially hidden but an option on the dashboard will allow the user to view ALL tickets.

3. Create a Ticket Record

   a. When a new ticket is created a form will be displayed that will require the user to select and/or complete several fields including Ticket Description, Ticket priority, Customer information and Ticket Description.

   b. The new ticket record will be stored in numerical order and the ticket **record number** will be generated based on the ticket date (e.g. 20220701-0000001). This is typically the number provided to the customer as a reference.

   c. Each Ticket will include a **Ticket Description.**

   d. **Each iteration** will be registered with the username, the date and time, and a comment. It can be a status change or a ticket modification. This will provide detailed ticket information as well as an audit trail.

e. Each Ticket will have a **Status Field** associated with it. Initially the Status field will be set to **NEW**.

4. Ticket Management

a. ADMIN users are who work on the tickets.

b. ADMIN users can change the **Status Field** of a Ticket by first selecting it on the Dashboard and then selecting the appropriate Status (e.g. **In Progress, Dispatched, Closed**, etc.). The admin **must** then enter a comment in the **Ticket Iteration.**

c. Once the status of a Ticket Record is set to CLOSED it will not accept any further modifications.

d. Certain fields will appear **greyed-out** and will not be modifiable (e.g. Ticket Record Number, Customer Name, and Ticket Duration)

e. Every active Ticket Record will include a **Ticket Resolution Field** which must be filled out before a ticket can be officially closed.

f. Neither the user nor the admin can delete a ticket. The ticket must change its status to **Cancelled**.

SUBMITTING YOUR WORK

Your submission should include:

1. A link to your GitHub Repo
2. A link to your cloud provider live site
3. A zip archive of your website's Project files submitted.
4. A link to you demo video on YouTube (or another streaming provider)

Feature	Description	Marks
GUI / Interface Design	Display elements meet requirements.	30
Functionality	Site deliverables are met, and site functions are met. No errors, including submission of user inputs.	30
Site Structure	Your site files are well organized. Your HTML and CSS are kept in separate folders. All external documents and code are appropriately linked to your site. You code is minified where possible. Your code is error free.	5
Internal Documentation	File header present, including site & student name & description. Functions and classes include headers describing functionality & scope. Inline comments and descriptive variable names included.	5
Version Control	GitHub commit history demonstrating regular updates.	10
Cloud Deployment	Deploy site to Cloud Service.	10
Video Presentation	Your short video must demonstrate your site and describe your code	10
Total		**100**

This assignment has a combined weight of **55%** of your total mark for this course.
All Assignment components are due on the designated dates Late submissions:

   • 20% deducted for each day late.

External code (e.g. from the internet or other sources) can be used for student submissions within the following parameters:

1. The code source (i.e. where you got the code and who wrote it) must be cited in your internal documentation.
2. It encompasses a maximum of 10% of your code (any more will be considered cheating).
3. You must understand any code you use and include documentation (comments) around the code that explains its function.
4. You must get written approval from me via email (jazeved5@my.centennialcollege.ca) .