COMP229 – Web Application Development

Final Team Project - Value 55%

Part 1 – Team Contract – Value 5%

- Teams Formed – 4 to 6 members
- Specialty Roles Selected
  - Project Manager
  - Lead Software Engineer
  - UI Programmer
  - Security Programmer
  - Database Programmer
  - Web Designer
- Project Selection (Survey Site, Tournament Site, Incident Management)
- Agile Tracking Tool Selected (Trello or Jira)

Due end of Week 8 @ midnight

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)
- External Design Document v1
- Agile Tracking – Backlog Snapshot
- Demo Video of First Release (10 minutes)

Due end of Week 9 (End of week) @ midnight

Part 3 – Authentication Release – **Value 10%**

- Site Security Functional
- Authentication Navigation added
- External Design Document v2
- Agile Tracking – Backlog Snapshot
- Demo Video of this Release (10 minutes)

Due end of Week 11 @ midnight

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)
- External Design Document v3
- Agile Tracking – Backlog Snapshot and reporting

Due end of Week 13 @ midnight

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 @ midnight.

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 as an optional frontend) create a web app from one of the following App templates:

- Survey Site
- Bracket / Tournament Web App
- Incident Management Web App
- Another approved project (please contact the instructor)

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 front-end framework (e.g. Bootstrap or Angular Material) along with a dynamic back-end (using Node, Express, and MongoDB or Firebase)
    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. **Note:** It is highly recommended that you use CDNs where you can in the development release of your App
    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
    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 humanreadable
    d. Ensure you include **inline comments** that describe your GUI Design and Functionality. **Note**: please avoid "over-commenting"
3. Create an **External Design Document** for your Web App that includes
    a. A **company Logo**
    b. **Table of contents**
    c. A **Detailed description** of your Web App including its core functionality
    d. A **Wireframes Section** that includes a wireframe image and appropriate arrows and labels for each page template of your Web App
    e. **Screen Capture Section** that includes Screen Shots (samples) of each of your site's templates.
    f. **Potential Future Functionality –** a section that describes features that could be added to your app but do not as yet exist given the time constraints.
4. Share your files on **GitHub** to demonstrate Version Control Best Practices and push your site to a cloud host
    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**
5. 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
    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 **15 minutes**

APP SPECIFIC REQUIREMENTS (60 MARKS)

SURVEY SITE (OPTION)

(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 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** a survey or entering secure areas of the site
2. Users can **Create** a Survey
    a. After a user is **registered** and **logged in**, he or she can create a survey based on 1 or 2 possible **survey templates** (e.g. Multiple Choice, Agree/Disagree, Short Answer, etc.)
    b. The user should be able **customize survey questions**. This includes the question text and response options
    c. The user should be able to create a **lifetime** for the survey (i.e. when the survey becomes **active** and when it **expires**)
3. Anonymous users can **Respond** to any active survey
    a. Anonymous users should be able to **select** an active survey and respond to survey questions
    b. Survey responses will be stored in the database for later use
4. Secure Reporting Section
    a. A **registered user** will be able to get simple analysis for any survey that he or she **owns** including **number of respondents** and survey answer **statistics**
    b. The statistics from each survey can be **exported** in some manner (e.g. **emailed**, **printed**, exported to **excel**, etc.)

TOURNEY BRACKETS SITE (OPTION)

(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 posting comments in a forum or creating a new topic
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 name fields. A fixed number of players will be allowed to be registered (I suggest 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 bouts initially (4 bouts for 8 players and 8 bouts for 16 players). Opponent selection for each bout 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 bouts determined
    b. A tournament will include a **fixed number of rounds** depending on how many players are registered (3 Rounds for 8 players and 4 rounds for 16 players)
    c. Each Tournament Round will be displayed as a **separate form page**
    d. The **Tournament owner** will be able to select the winners for each bout in every round
    e. Only the **winners** will be carried forward into new bouts for each successive round
    f. When the final round is completed, a summary of the results will be

INCIDENT MANAGMENT SITE (OPTION)

(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 incident records (tickets), changing ticket status, posting a comment or seeing the incident log.
2. Incident Dashboard (Log)
    a. After a user is **registered** and **logged in**, he or she can view an Incident **Dashboard** that will display all open Incidents (tickets) in a clickable list format
    b. The Dashboard will include an option that allows the user to **create a new incident (ticket)**
    c. Incidents that are closed will be initially hidden but an option on the dashboard will allow the user to view ALL incidents
3. Create an Incident Record
    a. When a new incident is created a form will be displayed that will require the user to select and/or complete several fields including: Incident Description, Incident priority, Customer information and Incident Narrative.

b. The new incident record will be stored in numerical order and the incident **record number** will be generated based on the incident date (e.g. 130418-0000001). This is typically the number provided to the customer as a reference
c. Each Incident Record (Ticket) will include an **Incident Narrative** that will be **timestamped** with every status change or Incident modification. This will provide detailed incident information as well as an audit trail
d. Each Incident Record (Ticket) will have a **Status Field** associated with it. Initially the Status field will be set to NEW.

4. Incident Management
    a. A registered user can change the **Status Field** of an Incident by first selecting it on the Dashboard and then selecting the appropriate Status (e.g. In Progress, Dispatched, Closed, etc.). The user **must** then enter a comment in the **Incident Narrative**
    b. Once the status of an Incident Record is set to CLOSED it will not accept any further modifications
    c. Certain fields will appear **greyed-out** and will not be modifiable (e.g. Incident Record Number, Customer Name, Incident Duration, etc.)
    d. Every active Incident Record will include an **Incident Resolution Field** which must be filled out before a ticket can be officially closed

SUBMITTING YOUR WORK

Your submission should include:

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

Feature	Description	Marks
GUI / Interface Design	Display elements meet requirements. Appropriate spacing, graphics, colour, and typography used.	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.	10
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
External Documentation	An external document (MS Word or PDF format) has been created that includes a company logo, table of contents, brief site description, etc.	7
Version Control	GitHub commit history demonstrating regular updates.	4
Cloud Deployment	Deploy site to Cloud Service.	4
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.