**COMP2068 – Emerging Technologies**

**Final Team Project - Value 25%**

**Part 1 (Project Concept) – Value 10%**

* **First Draft of External Document and Initial Project Structure Setup and 4 github commits**

Due Week # 10 (Monday July 15, 2019) @ midnight

**Part 2 (Finished Version) – Value 15%**

Due Week #12 (Monday July 29, 2019) @ midnight.

Final Team Project **Maximum Mark: 60**

**Overview**: Working in a group (up to 4 members) or on your own and utilizing your accumulated knowledge of the MERN Stack (MongoDB, ExpressJS and NodeJS with ReactJS as an optional front-end) create a web app from one of the following App templates:

* Survey Site
* Bracket / Tournament Web App
* Incident Management Web App

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 (30 MARKS)***

Each Web App will share these common requirements:

**(10 Marks: Site Structure, 5 Marks: Internal Documentation, 7 Marks: External Documentation, 4 Marks: Version Control, 4 Marks: Cloud Hosting)**

1. Your Web App must be designed using well-structured semantic HTML, CSS, JavaScript and implement a responsive front-end framework (e.g. Foundation or Bootstrap with ReactJS) along with a dynamic back-end (using Node, Express, and MongoDB) **(10 Marks: Site Structure):**
   1. Your Site must be **responsive** and adapt to various viewport sizes. **Note**: Using a mobile-first framework such as Bootstrap or Foundation is highly recommended (2 Marks: Site Structure).
   2. 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 (1 Mark: Site Structure).
   3. Your **JavaScript** 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 (2 Marks: Site Structure).
   4. 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 (1 Mark: Site Structure).
   5. All Your Code (HTML, CSS, JavaScript, jQuery, etc.) is error free (2 Marks: Site Structure).
   6. Your MongoDB database must be hosted online. **Note**: It is recommended that you use a provider like MongoLab, Google Cloud Platform, Amazon Web Services or Microsoft Azure (2 Marks: Site Structure).
2. Include Internal Documentation for your site **(5 Marks: Internal Documentation):**
   1. Ensure you include a comment header for your HTML, CSS, and JavaScript files that indicate: the **File name**, **Author's name**, **web site name**, **file description** (2 Marks: Internal Documentation).
   2. Ensure you include a **section headers** for all of your **HTML structure**, **CSS style sections**, and any **JavaScript functions** (1 Marks: Internal Documentation)
   3. Ensure all your code uses contextual variable names that help make the files human readable (1 Marks: Internal Documentation).
   4. Ensure you include inline comments that describe your GUI Design and Functionality. Note: please avoid “over-commenting**”** (1 Marks: Internal Documentation)
3. Create an **External Document** for your Web App that includes **(6 Marks: External Documentation):**
   1. **A company Logo** (1 Marks: External Documentation).
   2. **Table of contents** (1 Marks: External Documentation).
   3. A **Detailed description** of your Web App including its core functionality(1 Marks: External Documentation).
   4. **A Wireframes Section** that include a wireframe image and appropriate arrows and labels for each page template of your Web App (2 Marks: External Documentation).
   5. **Screen Capture Section** that includes Screen Shots (samples) of each of your site’s templates. (1 Marks: External Documentation).
   6. **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. (1 Mark: External Documentation).
4. Share your files on GitHub to demonstrate Version Control Best Practices and push your site to a cloud host **(4 Marks: Version Control, 4 Marks: Cloud Hosting).**
   1. Your repository must include **your code** and be well structured(2 Marks: Version Control).
   2. Your repository must include **commits** that demonstrate the project being updated at different stages of development – each time a major change is implemented (2 Marks: Version Control).
   3. You must deploy your site to your Cloud Server using **git** (4 Marks: Cloud Hosting).

**APP SPECIFIC REQUIREMENTS (30 MARKS)**

SURVEY SITE (OPTION)

**(15 Marks: GUI, 15 Marks: Functionality)**

1. User Management and site security **(5 Marks: GUI, 3 Marks: Functionality):**
   1. **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 (2 Marks: GUI, 2 Marks: Functionality).
   2. The user will be able to **Login, Logout** and **modify** his or her profile (1 Mark: GUI, 1 Mark: Functionality).
   3. Site security will prevent any non-registered users from creating a survey or entering secure areas of the site (2 Marks: Functionality).
2. Users can **Create** a Survey **(6 Marks: GUI, 6 Marks: Functionality):**
   1. 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.) (3 Marks: GUI, 3 Marks: Functionality).
   2. The user should be able **customize survey questions**. This includes the question text and response options (2 Marks: GUI, 2 Marks: Functionality).
   3. The user should be able to create a lifetime for the survey (i.e. when the survey becomes active and when it **expires**) (1 Mark: GUI, 1 Mark: Functionality).
3. Anonymous users can Respond to any active survey **(2 Marks: GUI, 3 Marks: Functionality):**
   1. Anonymous users should be able to select an active survey and respond to survey questions. (2 Marks: GUI, 2 Marks Functionality).
   2. Survey responses will be stored in the database for later use (1 Mark: Functionality).
4. Secure Reporting Section **(2 Marks: GUI, 3 Marks: Functionality):**
   1. 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** (1 Mark: GUI, 2 Marks: Functionality).
   2. The statistics from each survey can be **exported** in some manner (e.g. **emailed, printed**, exported to **excel**, etc.) (1 Mark: GUI, 1 Mark: Functionality).

TOURNEY BRACKETS SITE (OPTION)

**(15 Marks: GUI, 15 Marks: Functionality)**

1. User Management and site security **(5 Marks: GUI, 3 Marks: Functionality):** 
   1. **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 (2 Marks: GUI, 2 Marks: Functionality).
   2. The user will be able to Login, Logout and modify his or her profile(1 Mark: GUI, 1 Mark: Functionality).
   3. **Site security** will prevent non-registered users from posting comments in a forum or creating a new topic (2 Marks: Functionality).
2. Create A New Tournament **(4 Marks: GUI, 3 Marks: Functionality):**
   1. 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 (2 Marks: GUI, 1 Marks: Functionality).
   2. When a user creates a Tournament he is considered the owner and can choose when it becomes active and when it is completed. (2 Marks: GUI, 2 Marks: Functionality).
3. Users Can Register **Players (2 Marks: GUI, 5 Marks: Functionality):**
   1. 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) (1 Mark: GUI, 2 Marks: Functionality).
   2. An **Active Tournament** will display a list of players to Anonymous users (1 Mark: GUI, 1 Mark: Functionality).
   3. 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. (2 Marks: Functionality).
4. Tournament Management **(4 Marks: GUI, 4 Marks: Functionality):** 
   1. A Tournament can be **started** only after all players have been registered and the initial bouts determined (1 Mark: Functionality).
   2. 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) (1 Mark: Functionality).
   3. Each Tournament Round will be displayed as **a separate form page** (3 Marks: GUI)
   4. The **Tournament owner** will be able to select the winners for each bout in every round (1 Mark: Functionality)
   5. Only the **winners** will be carried forward into new bouts for each successive round (1 Mark: Functionality).
   6. When the final round is completed, a summary of the results will be displayed (1 Mark: GUI).

INCIDENT MANAGMENT SITE (OPTION)

**(15 Marks: GUI, 15 Marks: Functionality)**

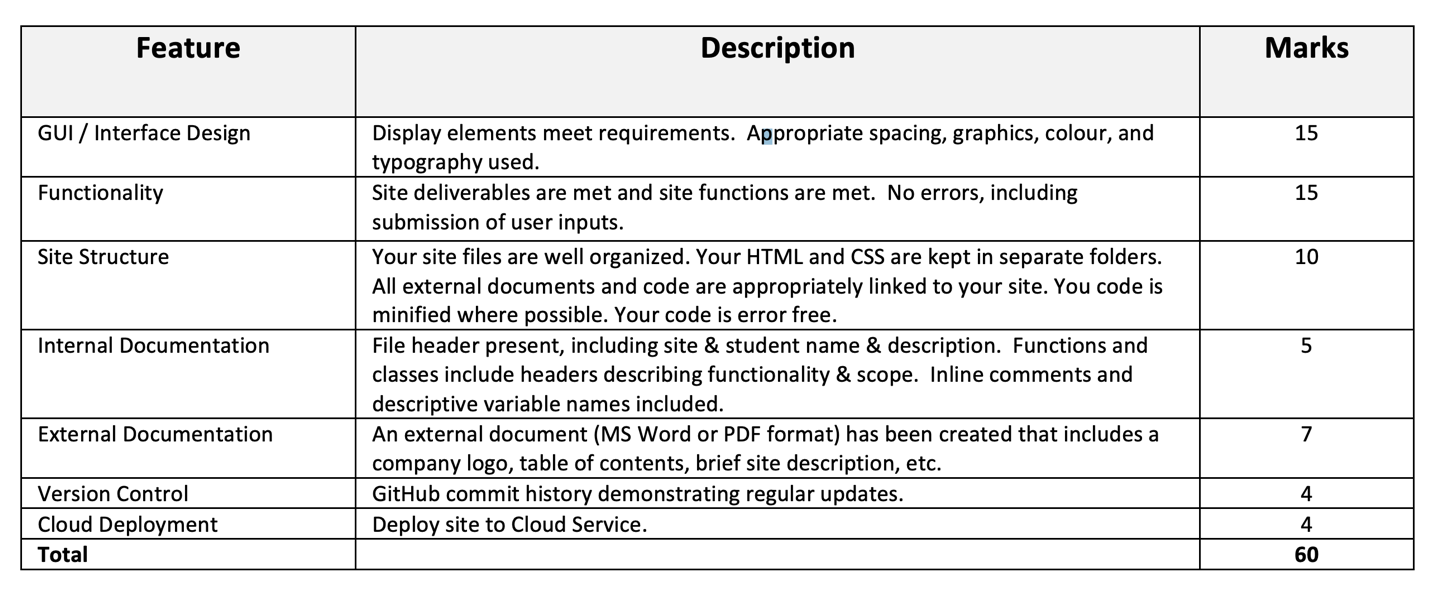
1. User Management and site security **(5 Marks: GUI, 3 Marks: Functionality):** 
   1. User Registration must be included. A form will allow the user to enter profile information (username, password, email address, user type) which will be stored in a MongoDB database table (2 Marks: GUI, 2 Marks: Functionality).
   2. The user will be able to Login, Logout and modify his or her profile (1 Mark: GUI, 1 Mark: Functionality).
   3. Site security will prevent non-registered users from creating incident records (tickets), changing ticket status, posting a comment or seeing the incident log. (2 Marks: Functionality).
2. Incident Dashboard (Log) **(3 Marks: GUI, 4 Marks: Functionality):**
   1. 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 (1 Mark: GUI, 2 Marks: Functionality).
   2. The Dashboard will include an option that allows the user to create a new incident (ticket) (1 Mark: GUI, 1 Mark: Functionality).
   3. Incidents that are closed will be initially hidden but an option on the dashboard will allow the user to view ALL incidents (1 Mark: GUI, 1 Mark: Functionality).
3. Create an Incident Record **(3 Marks: GUI, 4 Marks: Functionality):** 
   1. 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. (1 Marks: GUI, 1 Mark: Functionality).
   2. 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 (1 Mark: Functionality).
   3. 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 (1 Mark: GUI, 1 Marks: Functionality).
   4. Each Incident Record (Ticket) will have a **Status Field** associated with it. Initially the Status field will be set to NEW. (1 Mark: GUI, 1 Mark: Functionality).
4. Incident Management **(4 Marks: GUI, 4 Marks: Functionality):**
   1. 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** (1 Mark: GUI, 1 Mark: Functionality).
   2. Once the status of an Incident Record is set to CLOSED it will not accept any further modifications. **(1 Mark: GUI, 1 Mark: Functionality).**
   3. Certain fields will appear **greyed-out** and will not be modifiable (e.g. Incident Record Number, Customer Name, Incident Duration, etc.) (1 Mark: GUI, 1 Mark: Functionality).
   4. Every active Incident Record will include **an Incident Resolution Field** which must be filled out before a ticket can be officially closed (1 Mark: GUI, 1 Mark: Functionality).

**SUBMITTING YOUR WORK**

Your submission should include:

1. An external document (MS Word or PDF).
2. link to GitHub (preferable).

Please zip all files in to a single project archive.



This assignment is weighted **25%** of your total mark for this course.

All Assignments are due at the beginning of class.

Late submissions:

* 10% 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.