**Group Project – Developing Apps Using Emerging Web Technologies**

**Due Date:** Group presentation and demonstration in Week 14

Purpose: The purpose of this project is to:

1. Design and code web apps using emerging web frameworks
2. Build a **Graph QL API using Express** or Next.js
3. Build a Front-End (React or Svelte) that utilizes the Graph QL API
4. Apply appropriate design patterns and principles
5. Use **Deep Learning to make intelligent use of data**

References: Read the reference textbooks, lecture slides, class examples, and additional references provided here. This material provides the necessary information that you need to complete the project. You may need to read and use more materials and tools to implement a good solution.

Be sure to read the following general instructions carefully:

* This Project **may be completed in groups of 5-6 students**.
* This project can be **replaced with your capstone project** (COMP-231 or COMP-313), if you use and implement the **same front-end/back-end technologies shown in this document**.
* You will have to **present and demonstrate your solution in Week 14** and upload the solution on eCentennial through the assignment link on D2L. Bonus marks will be given if you also **publish the app on Heroku, Microsoft Azure, Amazon, or another Cloud platform.**
* **Your VS Code project should be named “YourGroupNameCOMP308Project” and should be zipped in a file YourGroupNameCOMP308Project.zip**.

**Project Specifications**

Your client needs an application to help nurse practitioners to **monitor patients during the first weeks of their release from the hospital and also help the patients to monitor their daily activities**. Develop a modern web app that implements the following functionalities:

1. User **registration**/**login**
2. If the user is a **nurse**:
   1. Allow the user to **enter vital signs**: *body temperature*, *heart rate*, *blood pressure*, or *respiratory rate*.
   2. Allow the user to **access information** captured during a previous clinical visit, vital signs: *body temperature*, *heart rate*, *blood pressure*, or *respiratory rate.*
   3. Allow the user to **send daily motivational tips** to the patient (by storing them in the database and providing a *daily tips* page for the patient to view, etc.).
   4. Allow the user to generate a list of possible medical conditions, and advise the patient to see a doctor if necessary - **intelligent use of symptoms** or other data using deep learning and publicly available datasets.
3. If the user is a **patient**:
   1. Allow the user **to create and send an emergency alert** to first responders (by storing this in a separate collection)
   2. Allow the user to access **fitness** **games** **page** designed to encourage patients to exercise at home. The **Gaming students** are encouraged to design/incorporate their own games/interactive pages.
   3. Allow the user to **enter daily information as specified by the nurse practitioner** (for example *pulse rate*, *blood pressure*, *weight*, *temperature*, *respiratory rate*).
   4. Allow the user to use a **checklist of common signs and symptoms (COVID-19 or RSV for example)**, and submit the choices.
4. Use **MongoDB** for storing the information.
5. Use Express to implement Graph QL API. Alternatively, use Next.js to implement the Graph QL API.
6. Choices for front-end frameworks:
   1. React 18.2 or higher, using functional components
   2. Next.js 14 or higher (https://nextjs.org/docs)
   3. Remix 1.17.0 or higher (https://remix.run/docs/en/1.17.0)
   4. Svelte 3.5 or higher (<https://v2.svelte.dev/>)
7. Apply Micro frontends architecture for UI and Microservices architecture for the backend.
8. Using **TypeScript** is optional.

Apply **responsive** web design principles. Use **CSS and React Bootstrap or Tailwind** to create a nice look and feel of your app. Display the **logo for the application**, other **images**, **game objects**, etc. **(100 marks)**

**Evaluation of software solution for each component (all items need to be shown during the group**

**presentation):**

|  |  |
| --- | --- |
| **Evaluation Component** | **Weight** |
| User registration/login | 10 |
| Nurse: Enter vital signs | 5 |
| Nurse: Access previous clinical visit information | 5 |
| Nurse: Send daily motivational tips to patients | 5 |
| Nurse: Intelligent use of symptoms (deep learning) | 15 |
| Patient: Create and send emergency alert | 5 |
| Patient: Access fitness games page | 5 |
| Patient: Enter daily information | 5 |
| Patient: Use checklist of common signs and symptoms | 5 |
| MongoDB database (proper use of document structure/model) | 5 |
| Correct Graph QL API implementation (proper use of design patterns with Express or Next.js) | 10 |
| Correct Front End implementation (proper use of architecture/libraries/frameworks: React, Next.js, Remix, Svelte) | 10 |
| Friendliness of UI (using CSS, React Bootstrap, etc.), correct use of naming guidelines for functional components, variables, methods, comments.. | 5 |
| Project Presentation according to presentation guidelines | 10 |
| Total | 100 |

**Evaluation rubric** with criteria and level of achievement:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Criteria** | Level of Achievement | | | |
| Failure to Minimal: 0 - 59% | Satisfactory: 60% - 69% | Good to Excellent: 70% - 79% | Excellent to Outstanding: 80 -100% |
| MongoDB database (proper use of document structure/model) | Incorrect model, errors. | Model has some missing or incorrect fields. | Model has the correct fields, some constraints are missing | Excellent design/implementation of the document model |
| Graph QL API Design and implementation | Incomplete design/implementation of most CRUD functionalities, errors. | The design/implementation of some functionalities is missing or has errors | The design/implementation of functionalities is correct | Excellent design Graph QL API as per specs.  Excellent use of design patterns. |
| Front End Design (proper use of architecture/libraries/frameworks) and Implementation | Incomplete front-end, most components are not implemented, errors | The design/implementation does not follow the specs and UI is not friendly. Some elements are not aligned. Some components have errors. | The design/implementation mostly follows the specs. UI is not very friendly. | Excellent design/implementation of components, very friendly UI |
| Friendliness and use of naming guidelines for functional components, variables, methods, comments. | No use of naming guidelines in most components | Some functional components, methods, variables are not named correctly | Most functional components, methods, variables are named correctly | Excellent use of naming guidelines |
| Intelligent use of symptoms or other data using deep learning | Not implemented | Does not work properly | The AI implementation is mostly correct. | Correct implementation and excellent use of AI, efficient model and high accuracy. |

**References:**

1. <https://docs.mongodb.com/manual/data-modeling/>
2. <https://expressjs.com/en/5x/api.html>
3. <http://mongoosejs.com/docs/guide.html>
4. <https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs>
5. <https://react.dev/>
6. <https://svelte.dev/>
7. <https://nextjs.org/>
8. <https://www.tensorflow.org/js/>
9. <https://www.tensorflow.org/js/tutorials>
10. https://archive.ics.uci.edu/datasets
11. <https://vitalflux.com/key-deep-learning-techniques-medical-disease-diagnosis/>
12. <https://link.springer.com/article/10.1007/s00521-021-06426-4>
13. Scalable and accurate deep learning with electronic health records, <https://www.nature.com/articles/s41746-018-0029-1>
14. Using recurrent neural network models for early detection of heart failure onset, <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5391725/>
15. Deep patient: An unsupervised representation to predict the future of patients from the electronic health records., <https://www.nature.com/articles/srep26094>
16. Deep learning augmented ECG analysis to identify biomarker-defined myocardial injury, https://pubmed.ncbi.nlm.nih.gov/36849487/