OlivApp - Olive Oil Mills Management App

CMPS 356 Project Phase 2 – Web UI Implementation using React, Data Management using MongoDB and Securing the application (15% of the course grade).

The project phase 1 submission is due by Project Phase 2 - Midnight Sunday 28th April 2019. Demos Tuesday 30th April during office hours.

1. Deliverables

You are to complete the implementation of the OlivApp to deliver the use cases documented in the project phase 1 requirements. Your project phase 2 deliverables include:

Part 1 - Design

1. Document in details 5 lessons learned by comparing your submitted project phase 1 with the model solution provided. You need to provide detailed reflections about the new concepts and lessons learnt when you compare your submission with the model solution.
2. Document the SPA architecture diagram for your overall design.
3. Document your database schema design in a schema diagram.

Part 2 - Implementation

Implement the server-side and client-side and Web components to deliver OlivApp use cases based on your previously developed and validated design. The expected deliverables are summarized below:

1. Design and implement the database schema to manage the data in a MongoDB database. The implementation should use mongoose library.
2. Populate the database with the data from the json files.
3. Update the *repositories* implementation to offer the same functionality as phase 1 but it should use MongoDB as the data source. **All data filtering should be done by MongoDB server and only the required data should be retrieved**. The implementation should make use of MongoDB capabilities (e.g., using aggregate query to get the data for the jobs summary report).

   Important note: When adding a Job Note you should auto-set the:
   - CreatedBy to current logged-in user
   - CreatedOn to current date and time
4. Implement the Web UI using React. Customize the application UI and behavior based on the user's role.
   - The main menu should appear on every page except the login page.
   - Same form should be used for add/update entities: Job, Product, User and Invoice.
   - All dropdown should be filled with reference data dynamically returned by Web API.

5. Implement authentication by offering the users the option to login using Google, Facebook or local authentication.
6. Secure your API using JSON Web Token (JWT) and React protected routes.

Part 3 – Testing and Documentation

1. Write a testing document including screenshots of conducted tests illustrating a working implementation of both the Web API and the Web UI.
2. Every team member should submit a description of their project contribution. Every team member should participate in solution demo and answer questions during the demo.

Important notes:

- Continue posting your questions to https://piazza.com/qu.edu.qa/spring2019/cmps356/
- Do not forget to submit your design and testing documentation (in Word format) and fill-up the Functionality column of the grading sheet provided in phase 2 Word template.
- Push your implementation and documentation to your group GitHub repository as you make progress.
- You need to test as you go!

Seek further clarification about the requirements/deliverables during the initial progress meeting with the instructor. Note that further important clarifications maybe to this document and you will be notified.

2. Grading rubric

Criteria	%	Funct Ionality*	Quality of the implementation	Score
Database Implementation and Initialization				
Design and implementation of the database schema to manage the data in a MongoDB database.	10			
Populate the database with the data from the json files.	3			
Repository implementation to read/write data from/to MongoDB	12			
Implement the Web UI using React. Customize the application UI and behavior based on the user's role.	45			
Authorization and Authentication				
Implement authentication by offering the users the option to login using Google, Facebook or local authentication.	10			
Secure your API using JSON Web Token (JWT) and React protected routes.	10			

Documentation				
*** Design documentation:** - 5 lessons learned from Phase 1 - SPA architecture diagram - Database schema diagram *** Testing documentation:** with evidence of correct implementation using snapshots illustrating the results of testing (you must use the provided template).	10			
Total	100			
Copying and/or plagiarism or not being able to explain or answer questions about the implementation.	0			

* **Possible grading for functionality**: ***Working*** (get 70% of the assigned grade), ***Not working*** (lose 40% of assigned grade and ***Not done*** (get 0). The remaining grade is assigned to the quality of the implementation. In case your implementation is not working then 40% of the grade will be lost and the remaining 60% will be determined based on of the code quality and how close your solution to the working implementation. Solution quality also includes meaningful naming of identifiers, no redundant code, simple and efficient design, clean code without unnecessary files/code, use of comments where necessary, proper white space and indentation.

Marks will be reduced for code duplication, poor/inefficient coding practices, poor naming of identifiers, unclean/untidy submission and unnecessary complex/poor user interface design.