CMPS 350 Project Phase 1 – WebApp UI Design and Implementation

Conference Management System (ConfPlus)

(15% of the course grade)

The project phase 1 submission is due by 8am Sunday 16th April 2023. Demos during the same week

1. Requirements

You are required to design and implement a Conference Management System named ***ConfPlus*** to manage the process of organizing and running an academic conference. It should be designed to meet the needs of all stakeholders involved in the conference, including **organizers**, **authors**, and **reviewers**.

You can familiarise yourself with the various notions around the management of an academic conference by visiting some important conferences such the Web Conference, the International Conference on Software Engineering (ICSE).

The app use cases include:

Use Case 1: Login

It allows users to login to use the app using their email and password. Login should be verified using the users data in **`users.json`**.

A sample list of organizers, reviewers and authors is provided in **`users.json`** file on GitHub under the *project* subfolder. To keep the app simple, there is no need for the users to register to create an account to use the app.

Also, for simplicity, upon successful login you can redirect the authors to the 'Submit paper page', redirect the reviewers to the 'Review papers' and redirect the organizers to the 'Schedule editor'.

Use Case 2: Submit a paper

Precondition: User is logged in as an author and has a paper to submit.

- Author enters the paper details including the paper title and abstract and authors (including first name, last name, email and affiliation). One of the authors should be marked as the Presenter. Note that a paper can have a **dynamic** list of authors (could be 1, 2, 3, 4 or more authors), the app should allow adding and removing the paper authors.
  The author affiliation should be a drop-down listing the available institutions from **`institutions.json`**. This file should have sample data such as 'Qatar University', 'University of Doha for Science and Technology', 'Carnegie Mellon University in Qatar'.
- Author selects the paper pdf and submits the paper.
- Upon submission, the app should upload the paper PDF to server, randomly auto assign to 2 reviewers to the paper (from the reviewers listed in **`users.json`**) and save the paper details to **`papers.json`** file.

Postcondition: Paper is submitted for review.

Use Case 3: Review paper

Precondition: User is logged in as a reviewer and has been assigned a paper to review.

- Reviewer gets the list of assigned papers from **`papers.json`**. The list should display the paper `title`, `authors`, `abstract` and a `link` to download the paper. The abstract could be made collapsible to allow hiding it.

- Reviewer can select a paper to review from the list of assigned papers. If the reviewer already reviewed the paper earlier, then the review should be loaded into the review form and allow the user to change it otherwise an empty review form should be displayed. The review form should include:

Overall evaluation:

2: strong accept
1: accept
0: borderline
-1: reject
-2: strong reject

Paper contribution:

5: a major and significant contribution
4: a clear contribution
3: minor contribution
2: no obvious contribution
1: no obvious contribution

Paper strengths: (multiline text)

Paper weakness: (multiline text)

Upon submission the review should be added to the paper and saved to **`papers.json`**.

Postcondition: Review is submitted to the system.

- **Use Case 4: Create/update conference schedule**

  Precondition: User is logged in as an organizer and has access to the accepted papers and presenters.

  First note that a conference schedule has sessions, and a section has paper presentations.

  - Organizer navigates to the schedule editor page. If a conference schedule is already defined in **`schedule.json`** file, then it should be loaded into the editor and allow the organizer to change it. Otherwise, an empty schedule editor should be displayed to allow the organizer to create the schedule.
  - Organizer can add/update/delete sessions. A session should have a *`title`*, a *`location`*, and a *`date`*.
    - The location should be a dropdown listing the available locations from ***`locations.json`*** file. You need to create this file and include some sample locations such as *Female Engineering Building C07-145, CSE Meeting Room BCR-E104, QU Library B13-201*.
    - The date should be a dropdown listing the dates from ***`conference-dates.json`***. You need to create this file and include some sample dates such as 12/06/2023, 13/06/2023, 14/06/2023.
  - Organizer assigns accepted papers and associated presenters to each session and decide the fromTime and toTime for each –presentation. The accepted papers should be loaded from **`papers.json`** and should have sum of overall evaluation >= 2.
  - Upon submission, the app should save the conference schedule to **`schedule.json`**.

  Postcondition: Conference schedule is saved and can be viewed by visitors.

Use Case 5: Get conference schedule

Allow visitors to get the conference schedule (from **`schedule.json`**). By default, all the sessions and presentations should be displayed. But provide the user the ability to filter the schedule and get the

sessions scheduled for a particular conference date. The conference date should be a dropdown listing the dates from ***conference-dates.json*** file.
For simplicity you can make the conference schedule the default home page accessible to all users without the need to login.

2. Deliverables

Seek further clarification about the requirements/deliverables during the initial progress meeting with the instructor. Note that further important clarifications maybe modified/added to the project requirements.

1. Design the App Web UI and navigation.
   You may design the UI wireframe (sketch) to decide the UI components and the layout either on paper or use a design tool such as https://www.figma.com/
2. **During the weekly office hours, you are required to present and discuss your design with the instructor and get feedback**.
3. For each use case, implement the app Web UI and navigation using HTML, CSS and JavaScript. The pages should comply with Web user interface design best practices. Also remember that 'there is elegance in simplicity'.
4. Design and implement the app navigation to allow the user to navigate from one page to another in intuitive and user-friendly way to achieve the app use cases.
5. For each use case, design and implement the Web API and the server-side data access repositories to read/write the app data from/to the data store. For phase 1, you can read/write to simple JSON files that you need to create and initialize with some sample data.
6. Application design documentation should include the Entities, Repositories and Web API class diagrams.
7. Document the app testing using screen shots illustrating the results of testing.

- Every team member should submit a description of their project contribution. Every team member should demo their work and answer questions during the demo.
- Push your implementation and documentation to your group GitHub repository as you make progress.

Note that this phase will be focused only **a fully working client-side and server-side implementation** that read/write data in json files.

3. Grading rubric

Criteria	%	Functionality*	Quality of the implementation
1) Design and implement the app Web UI and navigation using HTML, CSS and JavaScript. Including designing the App Web UI and navigation.	30		
2) Design and implement the Web API and the server-side data access repositories to read/write the app data JSON files.	60		
3) Application Design: Entities, Repositories and Web API class diagrams.	5		
4) Testing documentation using screen shots illustrating the testing results. - Discussion of the project contribution of each team member. Members should collaborate and contribute equally to the project.	5		
Total	100		
Important remark: *In case of copying and/or plagiarism or not being able to explain or answer questions about the implementation, you lose the whole grade.*			

* **Criteria for grading the functionality:**

- The functionality is working: you get 70% of the assigned grade.
- The functionality is not working: you lose 40% of assigned grade.
- The functionality is not implemented: you get 0.
- The remaining grade in all cases from above is assigned to the quality of the implementation,
- The grades are distributed on the various use cases, when the design/implementation is partial, you get only the grades of designed/implemented use cases.

Code quality criteria, include:

- Use of meaningful identifiers for variables and functions (e.g. using JavaScript naming conventions)
- Pages are responsive
- Clean code: simple and concise code, no redundancy
- Clean implementation without unnecessary files/code
- Use of comments where necessary
- Proper code formatting and indentation.

You lose marks for code duplication, poor/inefficient coding practices, poor naming of identifiers, unclean/untidy submission, and unnecessary complex/poor user interface design.