Project Phase 2

Last Modified: Friday 13 Nov 2020
Due: Monday 7 Dec 2020 at 11 pm

How to start Phase 2 (https://q.utoronto.ca/courses/180703/pages/how-to-start-phase-2) <-- Setup Instructions and Tips

The primary goal of Phase 2 is to give you a chance to see how extensible your code is, by extending it! There are three categories of extensions that you will make for Phase 2, before submitting your code and accompanying documentation. Here they are:

1. Mandatory extensions

- There will now be many types of events. A one-speaker event is the same as a "talk" from Phase 1. You can have multi-speaker events, like a panel discussion, and no-speaker events, like a party. Events can last as long as you want. You can measure the duration of an event in hours, or minutes. You get to decide.
- Events can be canceled by at least one organizer.
- At least one more type of user will be included in your program. For example, an Admin user who can delete messages or events with no attendees, or a VIP user who can access VIP-only events.
- Organizers can also create Speaker accounts, Attendee accounts, and any other type of user accounts that are in your program.
- Each event has a maximum number of people who can attend it. This amount can be set when the event is created and also changed later, by an organizer. Your program should check the maximum capacity for the room where the event will be held, and prevent the number of attendees from going above the room's capacity.

2. Optional Extensions (Choose 4*)

You will implement four of these, unless you implement a GUI. Please see the note below the list. Also, if your group has fewer than 7 members, please see an instructor during an office hour or after lecture to discuss how many of the features in this section you are required to implement.

- Allow the same users to log in and select which conference they want to participate in. Here, participation means viewing and signing up for events. The inbox can be conference-specific, or one general inbox for all messages from all conferences to that user. You decide which one.
- Enhance the user's messaging experience by allowing them to "mark as unread", delete, or archive messages after reading them.
- Have the program produce a neatly formatted program or schedule for the conference that users have the option of "downloading" (outputting it as html, pdf, or similar). Alternatively, if you just want the program to print the schedule to the screen, then users should be able to request a schedule by at least three of: day, by speaker, by time (all 3-4 pm talks on all days), or just the ones they have signed up for, or "liked" events (where you have to enable users to "like" events).
- Add additional constraints to the scheduling for various types of events (e.g. technology requirements for presentations, tables vs rows of chairs). When organizers are creating events, they can see a suggested list of rooms that fit the requirements of their event.
- Allow the system to support additional user requests (e.g. dietary restrictions, accessibility requirements) where organizers can tag a request as "pending" or "addressed". All organizers see the same list of requests, so if one tags a request as addressed, all other organizers can log in and see that.
- Use a database to store the information from your program through specific gateway class(es) so that you do not violate Clean Architecture and still have an Entity layer.
- Expand the menus available to organizers to allow them to get useful summary stats about the conference. Include app traffic statistics, event enrollment statistics, and top-five lists (or something similar). You should include enough statistics to make this option as complicated as the other ones on this list.
- Replace your text UI with a Graphic User Interface (GUI), which should follow the Model-View-Presenter architecture. See the note below.

***Note:** Implementing a GUI with JavaFx, java.swing, java.awt, or Android counts as two features. We will not be teaching GUIs in this course, so you will have to refer to online tutorials and examples.

3. Create your own new features

You are encouraged to invent new features that you would want to use in this program. The purpose of these invented features is to give you something to brag about during job interviews and make your program a bit different than everyone else's in the class.

These features must be as complicated as the mandatory features, listed above. You can do this by implementing one big new feature or several smaller ones. If you are having difficulty thinking of new features, you can discuss this part with your TA, during office hours, or just implement more of the Optional (Type 2) Features.

4. What you will submit

- The code
- A README file with instructions for running your program and any other information we need to know to test its functionality.
- A list of features that you implemented for Phase 2, so we don't miss any.
- (optional) A video showing how your program works (and that it works). This is non-optional if we are required to install anything in order to run your program. **In the absence of a video, you are expected to demonstrate how your code works to your TA during your last weekly meeting.**
- A single UML diagram for the entire program or partial diagrams for different parts of your program. The single file should be called `design.pdf`. If you use multiple files, you can call them `design1.pdf`, `design2.pdf`, etc.

- A list of design patterns that you used and why/how. Please include the names of all classes involved in the implementation of the pattern. The goal is to convince us that you understand the course material.
    - You can also include reasons for not using certain design patterns, but that part is optional.
- (optional) A list of design decisions and explanations about how your code has improved since Phase 1.
- It is possible that you will receive an email from your TA during the 10 days immediately after the submission deadline with requests to meet with individual group members and/or the entire group. The email will include the questions that the TA will ask you, so you can prepare for the meeting. We will only email you if we need your help to assign a fair grade to each group member. The group mark is based on the code. **Your individual mark will only be the same as the group mark if we can find evidence in the git logs that you contributed approximately equally to the coding of your project.** This does **NOT** mean that you should commit every few minutes. We can see what changes were made inside each commit. Many small commits will be counted as a few larger commits. It **DOES** mean that you should contribute code and not just Javadoc or just move existing code around.