Milestone 2: First Prototype Development

Released: Monday, Feb 12, 2024
Due: Monday, Feb 26, 2024 11:59 pm (on Gradescope)

Learning Objectives

- Develop an implementation that conforms to a design specification
- Refine and adjust earlier design decisions
- Develop tests to evaluate the end-to-end functionality and quality attributes of a system

Milestone Tasks

In this milestone, your team will develop an initial version of the scheduling application based on the design that you have in Milestone 1.

Task 1: Prototype Implementation

Implement a web-based application that supports the set of basic features that you designed for in M1: Scheduling, viewing, and modifying appointments. The user (i.e., a patient) should be able to use the application to view a list of available appointments, select an appointment at their preference, and be able to view an appointment(s) that they've previously made. In addition, the user should be able to come back to your site at a later time and modify an existing appointment (e.g., change it to a different, available time slot) or simply delete it. Your system must avoid scheduling conflicting appointments (i.e., each appointment slot should be assigned to at most one user).

Deploy your application on the VM that we have provided you with. Your application should provide a basic user interface that allows the user to perform the above tasks. Your system must support persistence of data (e.g., appointments made by users should persist in a database). The course staff will be accessing the URL for the frontend of your application and testing various scenarios that involve the scheduling features listed above.

Your team will use a private Github repository to develop and version source code artifacts for your application (instructions to follow separately). The course staff will have access to this repository and read your code as part of the evaluation.

Scope: The set of features that your application needs to support for M2 is the same as those that you were asked to design in M1. Your app does not need to provide user authentication or other security mechanisms, although the user should be able to retrieve their appointments using an identifier such as their name and/or email. In addition, your app does not need to handle issues that may arise due to concurrency (i.e., multiple users simultaneously trying to select the same appointment slot, causing race conditions).

Tips: Focus on building a first *minimal viable product (MVP)*, not a perfectly polished application. For example, you do not need to spend a lot of effort on the visual aesthetics or usability of the frontend; a bare minimum UI that supports the required functionality will be sufficient. Similarly, performance is not a quality attribute that we will be evaluating in this milestone (although your application should still be functional; i.e., it should not hang forever!).

Start early! Meet with your team and discuss (and document) clearly how you plan to divide the tasks among the team members. Do not wait until the last few days of the M2 deadline to start working on the tasks. Set intermediate milestones (e.g., every 4~5 days) and check in with the other team members frequently; this will allow you to identify and debug technical problems and issues early on. Reach out to the course staff if you face team challenges that are difficult to address internally.

Task 2: End-to-end testing

Quality attributes are important but often neglected design objectives in software development. To ensure that the implementation of your design conforms to your quality attribute requirements, you should implement end-to-end tests for **at least one quality attribute requirement** and the **system-level functionality** associated with that quality attribute requirement.

1. Pick one quality attribute scenario from your M1 submission. Describe the corresponding system-level functionality of the quality attribute scenario (e.g., the response time to what request is being measured, the availability of what functionality is measured, the functionality that should be reliably preserved under deviations from normal conditions) and describe how you can test the correctness of the functionality.
2. To measure the quality attribute, describe how your test design achieves **controllability** by outlining how you can control the system to bring it into the state that the quality attribute scenario is measuring (e.g., how to simulate the request, simulate deviations from normal conditions, simulate a high-load situation…).
3. To measure the quality attribute, describe how your test design achieves the **observability** by outlining how you are going to measure the quality attributes (e.g., execution time, availability percentage, functionality is preserved).
4. Remember that testing only a very small number of specific inputs does not give you enough confidence that your implementation satisfies your quality attribute requirement. Describe how you achieve a **medium-to-high coverage** of your quality attribute scenario. Describe your approach to increase the coverage of your quality attribute beyond a single configuration and justify why the coverage that you achieve is a good trade-off between invested resources and achieved confidence.
5. Implement the tests and include the results in your submission. If your system does not satisfy your stated objectives (i.e., it fails the quality attribute test by being too slow, not as available as you expected, or unreliable), describe an idea on how you could improve

the quality attributes of the system. Note that it is **acceptable** if your system implementation fails your quality attribute requirements.

Tip: Testing design-time quality attributes is quite hard to automate. Therefore, we recommend you to focus on run-time quality attributes, such as performance, availability, or reliability. If the quality attribute scenarios you specified in M1 are too hard to test, you are welcome to specify a new quality attribute scenario for this milestone.

Note: Although we recommend you to follow a test-driven development approach and write unit tests whenever possible, you are not required to do so. In this project, we focus on aspects of system design rather than coding practices. Therefore, we only require system-level testing that helps you assess your quality attribute requirements. You are still encouraged to use unit tests and follow practices of test-driven development where appropriate, to avoid introducing bugs as you make changes to your code.

Task 3: Design reflections

As you develop the application, you will gain new insights and knowledge about the domain or solution space that were not obvious during the design stage. As a result, you will likely refine abstract design decisions into more concrete ones, make additional decisions, or change the decisions that you made during M1. For this task, prepare a report that reflects on how the implementation process has influenced the design decisions that you made earlier in M1. In particular, the report should:

1. Discuss **at least two** design decisions that you have changed or additionally made since your initial M1 design. For each, provide a justification for the change or the need for an additional design decision.
2. Include an updated component diagram for your implementation. If there are any changes from the component diagram in M1, describe the design decisions that resulted in those changes.
3. Discuss the choice of the programming language and web framework that your team selected. Describe at least another alternative that you considered and provide justification for your final choice.
4. Discuss the choice of database engine or other form of data persistence used in your application. Describe at least another alternative that you considered and provide justification for your final choice.

Deliverables

Submit a report as a single PDF file to Gradescope that covers the following items in clearly labeled sections (ideally, each section should start on a new page). **Please correctly map the pages in the PDF to the corresponding sections.**

1. **Deployment (1 paragraph)**: Include the URL for accessing the main frontend of your application.

2. **Testing plan (2 pg max)**: A description of the design for your quality attribute tests and results.
3. **Design reflection report** (**4 pg max**): A document reflecting on how the implementation process has influenced earlier design decisions, as described above in Task 3.
4. **Team contract (1 pg max)**: A documentation of (i) the division of development tasks among team members and (ii) intermediate milestones, each with a target date and goals achieved.

Grading

This assignment is out of **125** points. For full points, we expect:

- **(40 pt)** A functional web application that provides the functionality of (1) viewing a list of available appointments, (2) making an appointment, and (3) modifying or deleting an appointment.
- **(35 pt)** A testing plan that describes how you achieve controllability **(10pt)**, observability **(10pt)** and high coverage **(10pt)** of a reasonable quality attribute scenario (functionality + quality attribute metric), and a discussion of the results **(5pt)**.
- **(40 pt)** A design reflection report with (i) a discussion of at least two new or changed design decisions along with their justifications (**10 pt**), (ii) an updated component diagram with the design decisions that resulted in changes (if any) (**10 pt**), (iii) a discussion of the choice of programming language and web framework, alternatives considered, and a final justification (**10 pt**), and (iv) a discussion of the database engine, alternatives considered, and a final justification (**10 pt**).
- **(10 pt)** A team contract that describes (i) the division of tasks among team members and (ii) a set of intermediate milestones, their target date, and expected goals.
- **(5 pt)** Bonus social points (same as M1).