Milestone 2: First Prototype Development

Released: Tuesday, Feb 10, 2026
Due: Friday, Feb 27, 2026 11:59 pm (on Gradescope)

Learning Objectives

- Develop an implementation that conforms to a design specification
- Refine and adjust earlier design decisions
- Design the system for testability and changeability

Milestone Tasks

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

Task 0: Team Contract

Meet with your team and discuss 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.

A part of your deliverables will be a team contract that states the responsibility division and the intermediate milestones.

Task 1: Prototype Implementation

Viewing, requesting, and tracking food donations. The user (i.e., a food recipient) should be able to use the application to view a list of available food donations in their area, request to pick up a donation (which temporarily reserves it), and be able to view the donation(s) that they've previously requested or received. In addition, the user should be able to come back to your site at a later time and cancel an existing pickup request (returning the donation to the available pool) or mark a donation as picked up (which removes it from the available pool). Your system must avoid double-booking donations (i.e., each donation should be claimed by at most one recipient at a time).

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., donation requests 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 food rescue 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: Your app does not need to provide user authentication or other security mechanisms, although the user should be able to retrieve their donation requests 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 recipients simultaneously trying to claim a donation, causing race conditions).

You should assume that the Food Inventory web service mentioned in Milestone 1 is not yet available; instead, create a simple mock or stub for this service that returns hardcoded sample data.

Testing: As part of the codebase that you will submit for this milestone, you must provide a set of test cases that you develop and use to test the key functionality of the application. We do not impose a strict requirement on the number of test cases, but your test suite should be diverse enough to cover important scenarios that may arise in production. We also encourage you to test scenarios that involve unusual corner cases.

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!).

Sample code (optional): To help you get started, we have provided barebone code (with a React frontend, a Flask backend, and a MongoDB database) for a simple scheduling web app available at the following link:

https://github.com/cmu-swdesign/Team_Project_Boilerplate_Code

You are free to fork/clone the repository and use the code as part of your project. Alternatively, you are also welcome to build your own application from scratch, using any programming language, web framework, or libraries of your choice.

Task 2: Design for Changeability

Design your system to be ready for change. First, consider different types of changes that may occur in your application, due to changes in requirements, domain assumptions, new features, and/or quality improvement. For each change, consider the impact of the change on the rest system and identify other components that may also need to be modified.

Apply design principles (i.e., information hiding, single responsibility, interface segregation, and dependency inversion) and techniques (data abstraction, interface abstraction, and encapsulation), discussed in class, to minimize the impact of likely changes in your system. As a part of the deliverables, you will be asked to describe the design decisions that you've made to address **two** of the likely and high-priority changes.

Task 3: Design for Testability

Design your system to be testable. Identify **at least one controllability challenge** and **at least one observability challenge** for testing your application. For each of these challenges, use a concrete use case scenario that you'd like to test, to describe how the challenge makes testing difficult or require more effort.

Apply design principles discussed in class to address these two challenges. As a part of the deliverables, you will be asked to (1) describe the design decisions that you've made to improve testability and (2) include a reference to the test cases for the above scenarios in your codebase.

Task 4: Design Reflection

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 **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.

<u>Deliverables</u>

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. **Test suite (1 paragraph)**: Include the URL to the part of your team Github repository that contains the set of test cases used to validate the functionality of the system.

3. **Changeability discussion (2 pg. max)**: A document discussing possible changes to your system and the design decisions that were made to improve the changeability of the system (Task 2).
4. **Testability discussion (2 pg. max**): A document discussing (1) a controllability challenge and an observability challenge, and (2) the design decisions that were made to improve the testability of the system, along with an URL reference to the test cases in the Github codebase (Task 3).
5. **Design reflection** (**3 pg. max**): A document reflecting on how the implementation process has influenced earlier design decisions (Task 4).
6. **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 **120** points. For full points, we expect:

- **(40 pt)** A functional web application that provides the features for (1) viewing a list of available food donations, (2) requesting to pick up a donation (claiming/reserving it), and (3) viewing, canceling, or marking as picked up previously requested donations.
- **(10 pt)** A set of test cases that cover the above functionality
- **(20 pt)** A discussion of **two possible, likely changes** in your system (**10 pt)** and the design decisions made for minimizing the impact of those changes (**10 pt**)
- **(20 pt)** A discussion of at least one controllability challenge (**5 pt**) and one observability challenge (**5 pt**) in testing your application, and the design decisions made to address these challenges (**10 pt**). The solution also includes an URL to the test cases that demonstrate the effect of testability improvement.
- **(20 pt)** A design reflection 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)**.
- **(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).