17-423/723 Software System Design: Course Project

Spring 2026

Overview

The goal of this project is to help you gain hands-on experience applying design principles and techniques from this class by designing, implementing, and iteratively improving a complex software system. In particular, you will work in teams to design, implement, and deploy a **food rescue and distribution system** to support local communities in reducing food waste and improving access to surplus food.

Across many cities, campuses, and neighborhoods, food rescue initiatives help connect food donors (such as grocery stores, restaurants, cafeterias, or households) with recipients (such as food banks, shelters, community fridges, or mutual-aid organizations). While these systems are socially impactful, they are also challenging to design due to the number of stakeholders involved (e.g., food donors, recipients, volunteer drivers, health departments), the perishability of food, variations in local policies and workflows, and the need to adapt to rapidly changing conditions (e.g., seasonal surges, holidays, or emergencies). One of the main objectives of this project is to learn to design systems that are ready for these types of changes and can satisfy different types of quality attributes that are critical for the success of the application.

To give you ideas about the problem domain and the type of system that you will be building, here are some of Pittsburgh's own food sharing programs, which includes dozens of independent organizations:

- Greater Pittsburgh Community Food Bank: Warehouse-based distribution system
- 412 Food Rescue: Volunteer-driven delivery model connecting donors with recipients
- 100+ individual food pantries: Each with unique operating procedures
- Campus dining services (CMU, Pitt): Institutional surplus food

Project Mechanics

Teamwork: You will work on this project in your assigned teams. As a team, you will use a shared GitHub repository and a virtual machine to coordinate your work. Please establish a way of communication and collaboration that works for your team -- for example, a Slack channel or a Trello board. Please agree on how you take clear notes at meetings that include agreed tasks and responsibilities. We do not expect that all team members contribute equally to each part of the project, but we expect that all team members make every effort to be a good team citizen (attend meetings, prepared and cooperative, respect for other team members, work on assigned and agreed tasks by agreed deadlines, reaching out to team members when delays are expected, etc).

Milestones: There will be in total six milestones throughout this project. In the first two milestones, you will design, test, and implement an initial prototype with a basic level of functionality involving food rescue. Over the subsequent milestones, you will be asked to re-design the system to handle additional layers of complexity through the introduction of new features and quality attributes. In addition, some of the milestones will involve cross-team interactions: You will be asked to build a service that will be used by other teams; through this exercise, you will learn to develop services that are reusable and interoperable.

Infrastructure: Each team will be provided with a virtual machine, where they will deploy their system as a web app. For evaluating and testing your system throughout the milestones, the course staff will attempt to interact with the deployed application by playing the role of various types of users (e.g., a food recipient, a donor, a volunteer driver, etc.,). In a later milestone where you are asked to improve the robustness of the application, the course staff may evaluate your application through stress testing or other form of systematic testing.

Languages, tools, frameworks: Your team is free to choose any programming language (e.g., Python, Javascript, C#,...) or technology stack (e.g., Django, Flask, Express,...) for any part of this project. Each team will be assigned a virtual machine to deploy your system as a web application. You will have root access to the virtual machine and are free to install any software you deem suitable; but be responsible and careful as you also have the power to mis-configure the machine to make it inaccessible. You also may use external data and services (e.g. cloud services). For example, you can use free cloud credits that companies like Microsoft, Google, and AWS provide to students. Whenever you set up tools or services, pay some attention to configuring them with reasonable security measures; they may be vulnerable to indiscriminate attacks on the web, which could result in loss of data or Internet access for your virtual machine.

Documents and reports: Throughout most of the milestones, your team will submit reports that describe design alternatives that you considered, justification behind the final design decisions, and any lessons learned (what worked well and what did not). Please note that there's no one "correct" or "best" way to design a system; our evaluation of your reports will be based on the clarity and quality of your discussion of design alternatives, justification, and self-reflections.

Milestone 1: Domain Modeling and Initial System Design

Released: Wednesday, January 28, 2026
Due: Monday, Feb 9, 2026 11:59 pm (on Gradescope)

Learning Objectives

- Identify and specify the key entities in the problem domain and assumptions about them.
- Define a set of quality attribute scenarios that are relevant to the system.
- Apply design notations to specify and document a high-level design of the system.
- Consider alternatives for a set of design decisions and document justifications for final decisions.

Milestone Tasks

In this milestone, your team will design an initial version of a food rescue app.

Task 1: Domain Modeling

Begin by understanding the problem domain and identifying the key entities and assumptions (**ASM**) about their behaviors and properties. For this initial version of the application, focus on requirements (**REQ**) that are related to the basic workflow of rescuing surplus food. In a typical workflow, a food recipient uses the rescue app to view a list of available food donations in their area and request to pick up a donation (you may treat such a request as a claim that temporarily reserves the donation for that recipient). The recipient then directly interacts with the food donor (e.g., through a phone call) to arrange a pickup. Once the donation has been picked up, the status of the donation is then modified to “picked up”, indicating that it is no longer available to other recipients.

In this initial version, for simplicity, your food rescue software will provide the functionality for **food recipients only**, allowing them to (1) view available food donations in their area, (2) request to pick up a donation, and (3) view their previously received donations. In other words, recipients are the only direct users of your system; donors and volunteer drivers are considered part of the problem domain but are not users of your app at this stage.

Furthermore, you may assume that your app relies on an external web service (called **Food Inventory**) that can be periodically queried to (1) obtain a list of available food donations within a geographic area and (2) check the status of each food donation (i.e., whether it is still available or already picked up). This service will be designed and implemented in a later milestone; for now, this service should be treated as a part of the problem domain that is beyond your control.

For this task, document the set of relevant domain entities and assumptions that you've identified using a context model. Recall that at this stage, you should focus on describing the problem domain and its interactions with the software, and delay any decisions about the internal design of the software.

Tips: In a typical design process, understanding the problem domain and identifying assumptions will involve talking to various stakeholders and domain experts. There is no one "correct" context model that we are expecting to see for this task, and any model with a reasonable set of entities and assumptions will be acceptable. However, if you need further clarifications about the problem domain, please feel free to contact the course staff.

Task 2: Quality Attributes

Describe the most important quality attribute scenarios (between 4 to 8 would be a reasonable number) that you need to consider when designing the system and explain why they are important. Furthermore, assign rough categories of priorities to each quality attribute scenario (e.g., "high priority", "medium priority", or "low priority") to indicate which quality attributes are more important than others, and justify your prioritization.

One approach to determining priorities is to consider the level of risk that may arise if the system fails to achieve a particular quality attribute - i.e., what are possible consequences on the stakeholders? Examples of risks include food expiring before pickup, multiple recipients attempting to claim the same donation, incorrect location information leading to failed pickups, or sudden surges in available food during holidays.

Tips: Remember that quality attribute specifications should be measurable and associated with a scenario. Also note that for the same type of quality attribute (e.g., performance), you can specify multiple quality attribute scenarios (e.g., response time for one type of request and response time for another type of request).

Scope: The list of quality attributes and their specification might change throughout the project when you are learning more about the domain or new requirements. The quality attributes in this milestone should demonstrate a good understanding of the domain and be relevant for the basic functionality of food sharing, but do not necessarily have to be "complete" or "final".

Task 3. Component and Data Model Design

Once the domain model has been developed, develop a high-level design of the system, including (1) the set of major components in your system and interfaces among those components and with the domain entities and (2) a data model that captures different types of information that your system will store.

As you discuss possible design solutions within your team, consider the following questions, including (but not limited to):

- What are the major components in the system, and what are their responsibilities?
- What does each component interface look like, and what are its inputs and outputs?
- Where is each component deployed? (e.g., on the user's web browser, a server)
- What information about the domain entities does the system need to store?
- What are appropriate multiplicity constraints over those data types?
- What programming languages and web frameworks will be used?
- What is the expected sequence of interactions between the components and domain entities in typical scenarios?
- How is the system designed to achieve the quality attributes identified in Task 2?

Document your design using (1) a component diagram and (2) a data model. In addition, for **two** of the design questions listed above, (i) describe alternatives that you considered, (ii) your final decision, and (iii) justification for your decision.

Scope. Since this is your team's first design assignment, you will not be evaluated based on the quality of your design (i.e., how well it achieves different types of quality attributes that we will discuss throughout this class, such as modularity, reusability, interoperability, scalability, robustness). For this milestone, a design that achieves the basic functional requirements of food rescue for recipients will be sufficient; you will be asked to iterate on and improve this design in the future milestones.

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. **Domain model (2 pg max)**: A context model that includes the domain entities, along with a list of system requirements (REQ) and assumptions about the behaviors or properties of the entities (ASM).
2. **Quality attribute specifications (2 pg max)**: A list of quality attribute scenarios, their categories of prioritization, and justification for their prioritization.
3. **Component design (2 pg max):** (i) A component diagram that describes the set of major components in the system, (ii) a description of the interfaces among those components, and (iii) a description of the responsibilities of each component.
4. **Data design (1 pg max):** A data model that describes different types of information stored in the system.
5. **Design discussion (2 pg max)**: A discussion of two design questions, alternatives considered, and justification for the final decisions.

Grading

This assignment is out of **120** points. For full points, we expect:

- A valid context model that includes a set of relevant domain entities and interactions between them (**10 pts**), a list of requirements (**10 pts**), and a list of assumptions about the entities (**10 pts**).
- At least 4 relevant quality attribute scenarios, specified in an unambiguous and measurable way, with justified priorities (**20 pts**).
- A valid component model that contains a set of components and connections between them (**10 pts**), a description of the interfaces between the components **(10 pts**), and a description of the responsibilities of those components (**10 pts**).
- A valid data model that (1) contains information that is necessary for fulfilling system requirements (**10 pts**) and (2) includes relations between data types and valid multiplicity constraints over those relations (**10 pts**).
- A discussion of two design decisions. For each decision, (1) a description of at least two alternatives considered (**5 + 5 pts**) and (2) a justification for the final decision (**5 + 5 pts**).
- Bonus social points **(5 pts)**: See below.

Bonus social points (5 pts): Participate in an in-person social activity with your team that is not related to any coursework. This could just be an informal happy hour, playing a board game or computer game together, doing a puzzle or trivia quiz, watching a movie, or whatever you like, as long as it is not course related. After you do, post a selfie to the public Slack channel #social (including a photo of the team members) and tag all members who participated. To receive the bonus points, please post the Slack message **within 3 days** after the milestone deadline.