Milestone 4: Service Development & Integration

Released: Monday, Mar 25, 2024

Learning Objectives

- Develop a large-scale software system in multi-team settings.
- Apply design principles for: Design with reuse, design for reuse, design for interoperability, design for testability, and design for change.
- Integrate and test services developed by other teams.

Milestone Tasks

In this milestone, your team will develop an initial version of the service assigned to your team in Milestone 3, integrate the services developed by other teams, and help other teams integrate your service. This milestone has two parts with separate deadlines, where (A) the first week and a half will be allocated for implementing the shared service that has been assigned to your team and (B) the second week will involve integrating and testing all of the services. Only part B includes a report submission.

Part A: Service Development

Due: 11:59 pm Wednesday, **Apr 3**, 2024

Task 1: Service Implementation

Implement the web service assigned to your team. Please ensure that your service conforms to the API you shared with other teams and try to not change your API, as the other teams depend on it. In the exceptional case where it is absolutely necessary to modify the API, please contact the course staff as well as the other teams as soon as possible. When implementing your service, please carefully consider applying the design principles taught in the lectures, including Design for Change, Design for Reuse, and Design for Interoperability, to ensure that the implementation is modular, understandable, and compatible with other services.

You may use any available resources for your implementation, including code generation, generative AI, frameworks, libraries, and code snippets from the web. Please document when you use external tools and code from other sources, applying the design principles taught in the Design With Reuse lecture where possible.

Task 2: Functional Testing

Similar to Milestone 3, create test double components for the APIs of other services that your service interacts with and test your service using the test doubles (see *Test Stub*, *Test Spy*, and *Mock Component* from the Testability lecture). You may have already developed test doubles for some of these services during Milestone 3. Your tests should verify that your service handles

indirect inputs (see Testability lecture slide 8) sent by other services correctly and that it sends the correct indirect outputs to other services. Your tests do not need to cover all types of inputs and outputs but should cover the most important use cases. Please ensure that your service handles errors gracefully and does not crash on erroneous input, as this will make integration easier!

Task 3: Service Deployment

After you have implemented and tested your service, deploy the service as an HTTP endpoint on the VM that has been assigned to your team. You will be asked to share the URL and port number for your service with the other teams, so that they can start accessing the service from their own services. To avoid delays in the following integration process, your service must be up and running by the above stated deadline for Part A.

Part B: Service Integration
Due: 11:59 pm Wednesday, **Apr 10**, 2024

Task 4: Service Integration

Provide your service to other teams by deploying the service in your VM and providing teams with the URL and most recently API specification of your service.

Replace the test doubles that you used in Part A with the actual services that have been deployed by the other teams. In theory, this step should be as simple as replacing the invocation of a service from a test double to the deployment endpoint for the service; however, there might be unexpected incompatibility issues that arise (possibly due to the changes in the APIs or implicit assumptions about how the other services work). Keep track of these issues as they arise and coordinate with the teams to resolve them. At the end of this milestone, all of the appointment scheduling apps and shared services must be fully functional.

Task 5: Integration Testing

The goal of this task is to develop and run additional tests to ensure that your team's implementation (i.e., the scheduling app and the assigned shared service) integrates correctly with the other services. Identify a set of use case scenarios that involve your scheduling app, your shared service, and other team's services as different components in the overall system. Document these scenarios and develop tests to ensure that the integration works correctly. Sequence diagrams are a good way to document these scenarios, although you do not need to create them for every integration test. You may automate these tests if possible, but a precise sequence of instructions to follow how to run the tests manually is sufficient as well.

Important notes: Since some of the services are shared by multiple teams, please **take extra caution when testing stateful operations**; i.e., those that involve changes to the state of the system (e.g., modify an existing appointment that was created by another team's scheduling

app). Doing so may interfere with and cause another team's tests to fail. If you are testing a shared service that involves state changes, make sure that those changes involve data that was created by your team.
In addition, please avoid overloading the shared services with too many requests at once, as doing so may introduce delays and interfere with another team's usage of the service.

Finally, during the course of the integration process and the future milestones, your team's service may behave unexpectedly or even possibly fail (e.g., when given an unexpected input from another input). Please try to monitor your service from time to time and ensure that it is available throughout the remainder of this project. If, at some point, your service experiences a failure, please try to re-deploy it and make it available to the other teams as soon as possible.

Task 6: Design Reflection

Reflect on your experience implementing, (possibly) re-designing, and integrating your service with the scheduling app and the other teams' services. Please discuss the **design principles** you applied when designing your service (see Design for Change, Design for Interoperability, and Design for Reuse lectures), how you applied them, and how they impacted your design decisions.

Furthermore, discuss **at least two** other topics from this list:

- How you decided to **reuse** existing libraries, frameworks, or code snippets from other sources. Explain the reuse candidates you considered and why you chose to reuse or not to reuse them (see Design with Reuse lecture for guidance)
- How **cross-team collaboration** affected your design decisions, interfaces, and system design. E.g., what considerations did you make to avoid potential integration issues with services developed by other teams, how did you structure your communication with other teams, and how did you review their design (see Review Designs lecture for guidance)
- How **starting from your fixed interface description** impacted your implementation. E.g., did it feel easier to implement a service with already existing interfaces, or did it constrain the implementation too much?

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 (1) the URL for accessing the main frontend of your application and (2) the URL for the shared service that your team has implemented.
2. **Mock testing report (1 pg max)**: A description of how you used test double components to test your service implementation in Part A, including links to the test artifacts (e.g., parts of the source code that contains the test doubles).

3. **Integration testing report (2 pg max):** A description of at least **three use case scenarios** that you developed for testing the integration of your service with the other services. If your test artifacts (scenarios or implementation) are not included in the report, please include links.
4. **Design reflection report** (**2 pg max**): A document reflecting on the process of implementing and integrating your service with the other parts of the system, as described above in Task 6.
5. **Team contract (1 pg max)**: A documentation of (i) the division of development tasks among team members (including the interface person) 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 service that implements the functionality assigned to your team and integrates with all appointment scheduling apps and a functional appointment scheduling system that integrates with all other services.
- **(10 pt**) A mock test report of your approach for implementing test doubles for other services.
- **(20 pt)** An integration testing report outlining how you test the integration of other services.
- **(40 pt)** A design reflection report with (i) a discussion of design principles you applied (**10 pt**), (ii) at least two topics of the list above (**20 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 the previous milestones).