Project: Implementation Instructions

**Due date:** Wednesday, June 12.

## Implementation Overview

In the time remaining in the course each team is tasked with the completion of the project. By this point, each team should have the following project-related assets:

- Complete database design: E-R model and the relational model.
- List of use cases mapped to API calls specified in the application's WSDL.
- Implementation of the data-loading API calls.

The key remaining task each team has to complete is **implementation of the system**.

### Timeline

The implementation shall be conducted outside of classroom/lab periods, except when noted. Each team is responsible for setting up and maintaining a realistic implementation schedule and for all communications between team members concerning the project matters.

To ensure that the work on the project is spread relatively evenly over time, and to provide you the opportunity to solicit and obtain additional customer comments, you will have one official customer visit:

**Thursday, May 16 9:30am - 11:00am** David Cumberland.

Each team will have an opportunity to demonstrate the progress and to discuss outstanding issues.

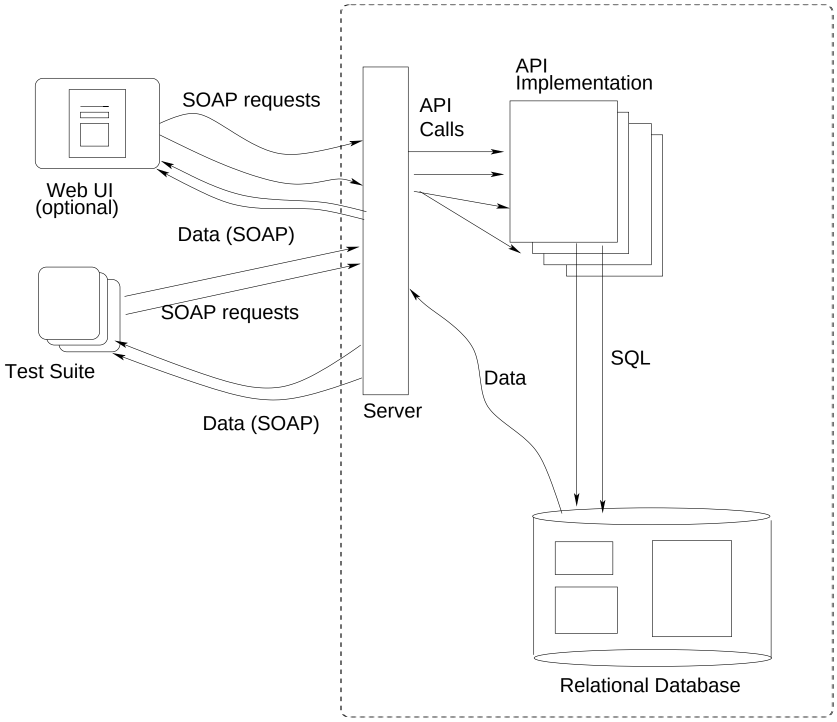


Figure 1: Architecture of the inventory request processing system.

## Extra Credit

The project will have a number of extra credit oppotunities related to development of specific extra components of the system. The extra credit opportunities will be officially announced on May 16. Most of the opportunities can be pursued by multiple teams. The only exception is the construction of the web UI for testing the software you are building. Only one team will be asked to work on it, as the intent is to make all your projects work with a single Web UI.

## System Implementation Notes

Figure 1 shows the architechture of your system. A brief overview of the components is below.

**Input.** The input to your system is SOAP requests that follow the WSDL specification you have been working with. Each SOAP request will contain a request for your system to perform a single action. There are two possibile ways to initiate the SOAP requests: the test suite/harness, each team will develop, and the Web UI, which may be developed for the use of all teams.

**Test Suite/Harness.** Each team needs to develop a test suite/test harness which will originate the SOAP requests. This is necessary in order for each team to be able to test their system throughout the remainder of the development cycle. The test suite must adequately test all 16 API calls requested by the project's WSDL (see Table 1).

**Web UI.** To simplify demonstration of your work, a Web UI may be developed as an extra credit assignment by one of the teams. The Web UI will allow each team to provide the credentials of its server and use the UI's SOAP request generation facilities to pass the data between the UI and the team's implementation. The integration with the Web UI may require the server for each team to abide by some additional specifications, that will be provided to you. It is not expected that these specs will be difficult to imlement.

**Core System.** The core system you are implementing consists of three components: *thin web server*, *thick API implementation* and the *relational database*. The roles of each of the components are clarified below.

**Server.** The web server has two core tasks:

1. Listen for SOAP requests. Upon receiving a request, parse it, determine which API functionality needs to be engaged, construct the API call and commence its execution in a separate thread.
2. Receive results of execution an API call, envelop them in SOAP syntax and pass them back to the sender of the initial SOAP request.

The server is considered to be *thin*, because outside of the implementations of the API, it does not work with any data stored in the database. It role is to convert SOAP requests into API calls, and convert outputs of the API calls back into SOAP.

**API Implementation.** At present, the WSDL specification asks you to implement the methods 16 for interacting with data in the database enumerated in Table 1. Your system must implement all 16 methods. Please note that except for the three operations described in **Lab 5**: database creation, database clearing and **database removal from DBMS, all other interactions with the database must take place from within the implementation of these methods**.

**Relational database.** The database supporting your application. You are allowed to choose the flavor of the DBMS (Oracle, MySQL, etc.) for the database. The only two requirements are: (a) the database must be relational and store data in the tables that match your logical database design, and (b) all interactions with the database must go through SQL commands.

| |
|---|
| `createFulfiller` |
| `getFulfillerStatus` |
| `createFulfilmentLocation` |
| `getFulfillmentLocations` |
| `getFulfillmentLocationTypes` |
| `allocateInventory` |
| `deallocateInventory` |
| `fulfillInventory` |
| `getItemLocationsByFulfiller` |
| `createBin` |
| `getBins` |
| `getBinTypes` |
| `getBinStatuses` |
| `geadjustInventorytInventory` |
| `adjustInventory` |
| `refreshInventory` |

Table 1: List of methods from the WSDL document.

.

## Final Deliverables and Submission

Each team will need to submit the final set of deliverables by or on **Wednesday, June 12** (regardless of which day the team's demo will take place). Detailed instructions and the list of required deliverables will be issued separately closer to the end of the quarter.

## Project Demo Notes.

In addition to softcopy deliverables, each team will demonstrate its project to the instructor and the customer.

Each demo will take about 30 mins. We will try to schedule all demos back-to-back in order to reduce the travel of the customers. We will reserve one of the CS labs for the duration of the demos. The demos will take place on **Wednessday, June 12** and **Thursday, June 13**.

Each team must delegate at least two people to participate in the demo. Presence of all people in the team is not required, although may be desired (as each team member will be in the best position to answer questions about his/her direct contribution to the project).

In its allotted time, each team will conduct two independent and separate presentation. One presentation will be done for the benefit of the customer (David Cumberland), the other - for me. At least one student should be present at each of the two presentations (hence we need at least two to be present for the demo).

Both the instructor and the customer will have (somewhat different) evaluation sheets that they will be filling out during the demo. These evaluation sheets will be later used to assign the numeric score to the project.

Each team is encouraged to arrive to the lab at least 20 mins prior to their time slot and to set up demonstrations on two workstations in the lab. The workstations should not be immediately adjacent to each other (allow some physical space between the two demos to keep the noise from one from sipping into the other).

If your project contains known bugs or unimplemented (or underimplemented) use cases, these need to be declared by the demonstrators. Additionally, your application documentation deliverable should list any such known bugs, incompletenesses and/or other deficiencies.

## Grading.

Your project grade will be based on the following:

1. Correctness of the final E-R model and relational database design.
2. Timeliness and quality of the submission of supplemental documentation and code.
3. Assessment of completeness/quality of your project implementation performed by the instructor (during the demo).
4. Opinion of the customer based on the demo.

Note, that while there is a single project score, **it does not override** Lab 2 — Lab 5 scores for each team.

**GOOD LUCK!**