SOEN343: Software Architecture and Design I
Term project

Instructor: Dr. C. Constantinides

8 September, 2017

Contents

1 Introduction

This is a group project which weighs a total of 75% of your overall grade. Please read the entire document very carefully. I will not respond to questions that can be answered by the current document.

In this project you will adopt an iterative process to develop a web application. The details are described in the current document.

You are currently provided with a set of requirements, described in Section 2. Some of these requirements are highlighted. The importance of highlighting will be discussed later in the document.

Note that the current set of requirements is not frozen. The set will change during development. In the schedule given, you will notice a 2-day gap between iterations. During this period, we may post new requirements, modify existing requirements, or eliminate existing requirements. Your ability to adapt to change is vital to the learning process and to the objectives of this project and it should be demonstrated in your artifacts.

You are expected to form teams of 9-10 members, and assign a team leader. The position of the team leader may be rotated among team members, but not during a development iteration.

2 Description

This section contains the initial set of functional and non-functional requirements of the project.

2.1 The system

The system you will develop will be a web application that allows clients to view a retailer's catalogs and make purchases.

2.2 The database

A database holds records of electronics sold by a company as well as records of registered clients and administrators. Electronics can be television sets or computer systems. Televisions can be categorized into high-definition, (HD) LED, Smart or 3D. Computer systems can be desktops (display units are sold separately), laptops, or tablets.

1. A record for a television contains the detailed dimensions (in centimeters), the weight, the model number (alphanumerical, unique), the brand name and the price.
2. A record for a desktop computer contains the detailed dimensions (in centimeters), processor type, the RAM size, the weight, the number of CPU cores, the hard drive size, the brand name, the model number (alphanumerical, unique), and the price.
3. A record for a monitor display contains the size (in inches), the weight, the the brand name, the model number (alphanumerical, unique), and the price.
4. A record for a laptop contains the display size (in inches), the processor type, the RAM size, the weight, the number of CPU cores, the hard drive size, battery information, the brand name, the model number (alphanumerical, unique), the built-in operating system, and the price. Laptops may additionally contain a camera. A laptop may also be of a touch-screen type.
5. A record for a tablet computer contains the display size (in inches), the detailed dimensions (in centimeters), the weight, the processor type, the RAM size, the number of CPU cores, the hard drive size, battery information, the brand name, the model number (alphanumerical, unique), the built-in operating system, camera information, and the price.
6. A client record consists of a first and last name, a physical and email address, and a phone number. An administrator is a client to the database with privileges. All administrators and clients have a unique id.

2.3 Administering the database

An administrator can use a graphical interface to enter new records into the database, to modify existing records, or to delete existing records. An administrator can additionally view the contents of the database.

There must be at least one registered administrator in the system. An administrator may additionally hold a regular account (in order to be able to conduct transactions).

We can divide system operations into two categories:

Write operation A write operation modifies the contents of the database, and it includes tasks undertaken by an administrator such as adding a new record, or modifying or deleting an existing record. Another type of write operation is performed upon successful termination of a purchase transaction, whereby one or possibly several records are erased from the database, or during a return whereby one or possibly several records are added to the database (See subsections 2.5 - 2.7).

Read operation A read operation views (but does not modify) the contents of the database.

As the system is expected to provide shared resources, it must be built with the following properties:

Safety A table must not be simultaneously accessed by a writer and reader. Furthermore, a table must not be simultaneously accessed by more than one writer.

Liveness A client who wishes to obtain access to a table will eventually be allowed access.

Fairness Requests for access must succeed infinitely often. Nobody will wait for ever to be serviced. Furthermore, administrators who wish to write, should have priority over regular clients (who may write or read).

No regular client may hold more than one account, and no client or administrator may be logged in more than once at any time. Additionally, clients and administrators who log in are monitored by the system, i.e. the system maintains a registry of its active users through a collection of pairs of id and timestamp.

2.4 Viewing the catalogs

Clients should be able to chose an item category, and view the contents of the corresponding catalog, either in a random order, or by creating a result set through a selection of filtering criteria. This implies that characteristics of each category are broken down into ranges of values and are available for selection, e.g. “View all monitors of size 26 inch and above, *and* manufactured by Dell.”

Clients may additionally chose the order by which they view a result set: random order, or sort according to criteria, e.g. “View sorted by (increasing or decreasing) price.”

From a given result set, a client may chose an item to view in detail. The client can subsequently proceed to the next item in detail view, or go back to the (possibly filtered) initial result set view.

2.5 Purchases

A purchase can only be associated with a single client and any client who is pursuing a purchase transaction must already be successfully logged in as a regular client. This implies that administrators are not eligible to make purchases through their administrative account. Upon initiation, a purchase transaction must be either completed or canceled. All completed purchases are logged by the system by a collection of pairs of client id and the associated timestamp. No null purchases are recorded. A transaction is considered *null* if it has been initiated but canceled before completion.

Upon logging into the system, clients who wish to make a purchase can place items in a shopping cart, remove items from the cart, or view the contents of the cart while being able to continue browsing the catalog.

An item cannot belong to more than one purchase transaction. An item which is placed in a shopping cart is locked, i.e. it is not any more available for purchase by another client unless it is removed from the cart. Items are removed from the cart either explicitly, or implicitly upon cancellation of the transaction, or upon reaching a time limit of being held by the cart. This implies that a shopping cart cannot lock items for ever. There must be an upper limit (in the order of a few minutes) by which items held in a cart are released to the inventory. In other words, at any point in time all items held by a shopping cart are locked, but each item inside a cart may be associated with a different time limit.

For the purpose of this exercise we will not implement any payment functionality. For each client that wishes to perform a sale transaction and upon a successful completion of such transaction we will keep items in a collection owned by the client.

2.6 Shopping cart

Upon initiating a purchase transaction, a shopping cart is empty and can hold up to 7 items. A cart can only be owned by one (regular) client. The contents of a cart must always correspond to records held by the retailer's database. An item can only be placed in a cart provided it is not locked by another purchase that takes place at that time. All items in a cart have unique alphanumeric id's. This means that if a client chooses to purchase two items of the same type, the transaction entails two different id's of the corresponding items.

One can place one or more items into a cart provided there is enough space available and upon successful termination of the operation, the size of the cart is incremented accordingly. Additionally, one can remove one or more items from a non-empty cart and upon successful termination of the operation, the size of the cart is decremented accordingly. Finally, upon successful completion of a purchase, items from a cart are not any more part of the retailer's database.

2.7 Returns

For one or more items that a client has in their purchased collection, they should be able to return them to the store. When initiated, a return transaction must be either completed or canceled. Upon successful completion of a return transaction, all its line items become part of the store's inventory (i.e. the system enters the corresponding records into the database).

3 What to do: "A little analysis, a little design and a little coding: Repeat 5 times."

This section describes in detail what needs to be done: A high-level development plan, the artifacts to produce (including the running application), and finally some more technical considerations such as object-relational mapping and the deployment of formal methods.

3.1 Plan

There will be five (5) iterations in the project, the dates of which are listed in Section 4.4. The plan for the completion of requirements is as follows:

Iteration 1 Creation of the database. At least one administrator must register with the system. Administrators can enter items into the database (persistent memory) and may view its inventory.

Iteration 2 Administrators can modify or delete records from the database.

Iteration 3 Regular clients may register with the system and may view its inventory.

Iteration 4 Regular clients may perform transactions with the system.

Iteration 5 Reserved for the implementation of a new requirement, to be added after the completion of iteration 4. **Well beforehand, you must study the material on aspect-oriented programming (AOP) and become familiar with some technology that can support AOP in your chosen server-side programming language**. A list of AOP technologies is provided by wikipedia at (`https://en.wikipedia.org/wiki/Aspect-oriented_programming#Implementations`).

3.2 Artifacts and running application

Your software system will consist of the following artifacts that must be developed in line with an iterative process:

- A *Software Requirements Specification* (template available on course website),
- A *Software Architecture Document* (template available on course website),
- Code repository (contains application code, as well as testing, configuration, installation and other supporting implementation), and finally
- A running version, available as a web application.

Your code must be readable, and must follow the naming conventions of the language(s) involved. You may consider the following references:

- Top 15+ Best Practices for Writing Super Readable Code (`https://code.tutsplus.com/tutorials/top-15-best-practices-for-writing-super-readable-code--net-8118`).

- Code Conventions for the Java Programming Language (`http://www.oracle.com/technetwork/java/javase/documentation/codeconventions-135099.html`).
- PHP Coding Standards (`https://make.wordpress.org/core/handbook/best-practices/coding-standards/php/`).
- PEP 8 – Style Guide for Python Code (`https://www.python.org/dev/peps/pep-0008/`).
- JavaScript Style Guide and Coding Conventions (`https://www.w3schools.com/js/js_conventions.asp`).
- C# Coding Conventions (C# Programming Guide) (`https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/inside-a-program/coding-conventions`).

3.3 Object-relational mapping

You must develop this system from scratch and in an object-oriented way. You must implement the object-relational mapping manually, i.e. you may not deploy some framework for this task.

3.4 Formal methods

Extended finite state machines The manipulation of the shopping cart is among the critical requirements of this project. Develop an extended finite state machine (both as a set of mathematical rules and as a UML state diagram) to formally capture the corresponding use case. (destination: SRS)

Concurrent properties Capture the concurrent properties of the system in temporal logic statements. (destination: SRS)

Object Constraint Language (OCL) Capture all highlighted requirements in OCL expressions. In your implementation model, you must provide internal documentation (in the form of comments) to indicate where these requirements are addressed. (SAD)

Contracts Deploy contract programming (also known as "design by contract") for the implementation and manipulation of a) a purchase transaction, b) a return transaction and c) a shopping cart. A list of technologies (language and third party support) is provided by wikipedia at (`https://en.wikipedia.org/wiki/Design_by_contract#Language_support`).

4 Organization and management

This section describes the expectations on the team formation, the preparation of your development platforms, development process, technologies to be used, and your overall responsibilities.

4.1 Development platforms

You must organize your development as follows:

Google Docs For the preparation of the SRS and SAD you must use **Google Docs** that supports **Collaborative Document Development**. Team leaders should create a folder in their Google drives entitled `SOEN343`, in which to maintain both reports. This folder should be kept with Link Sharing set to off; this prohibits anonymous changes. All team members should be invited to the folder via an email invite (with permission settings set to organize, add, & edit) and should be logged in to their accounts when editing to properly log all changes. Please invite the instructor to the folder as well.

GitHub For implementation, you need to deploy **Version Control** and **Issue Tracking** which can be provided by **GitHub**. Your GitHub repository must be private (GitHub provides free private repositories when you request their free student plan). Please create an account with editing rights for the instructor.

Resources

- Google Docs (`https://www.google.ca/docs/about/`
- GitHub (`https://github.com/`
- Student Developer Pack (`https://education.github.com/pack`)

4.2 Preparation and submission of credentials

Please email your your team information together with your credentials to the instructor by **Friday 15 September at 17:00**. Your email should be in plain text, and formatted as follows:

```
Subject:  [SOEN343] Our team

Syd Barrett (*) syd@pink-floyd.com
Pink Anderson
Floyd Council

Google Docs: ...
Version control / issue tracking: ...
Application: http:// ...
```

```
--
Name and id of team leader.
```

where the asterisk indicates that Syd Barrett is the team leader. Please do not include any additional information or formatting (email information of team members, id's, bullet points, numbering, extra spacing, tabs, etc.). Upon submission of this information, team formations will be posted on the course website and each team will be assigned a number. Please use this number when you contact the instructor. **After you submit your team's credentials to the instructor, you may not change a team and you may not break your team or restructure teams, unless you discuss the matter and obtain permission from the instructor**.

4.3 Cannot find a team? Looking for a team member?

If the deadline is approaching to form a team and submit its credentials and you still do not have a team, please write your name down on a list that I will place outside my office door. If you are a team leader and you are looking for team members you may consult this list. In the case where you recruit someone for your team, please make sure you scratch that name from the list. On Friday 15 September, I will collect any names on this list and either form a team, or (in the case of a very small number of people) assign you to teams.

4.4 Process

You must follow any iterative process of your choosing, with iterations timeboxed into two weeks (ending at 17:00 on the last day) as follows:

Iteration 1 (weeks 3 - 4) : Mon 18 Sep - Fri 29 Sep.

Iteration 2 (weeks 5 - 6) : Mon 2 Oct - Fri 13 Oct.

Iteration 3 (weeks 7 - 8) : Mon 16 Oct - Fri 27 Oct.

Iteration 4 (weeks 9 - 10) : Mon 30 Oct - Fri 10 Nov.

Iteration 5 (weeks 11 - 12) : Mon 13 Nov - Fri 24 Nov.

4.5 Technologies

At the back-end, you must maintain a relational database management system such as one of the following:

- SQLite (`https://www.sqlite.org/`
- MySQL (`https://www.mysql.com/`
- PostgreSQL (`https://www.postgresql.org/`

To develop your server-side application, you have the choice of the following languages, listed below together with recommended frameworks:

- Java with Spring (`https://spring.io/`).
- C# with ASP.NET (`https://www.asp.net/mvc`).
- Python with Django (`https://www.djangoproject.com/`).
- PHP with Laravel (`https://laravel.com/`).
- node.js (Javascript) with Express (`https://expressjs.com/`).

Please note that any built-in object-relational mapping technologies that come with some of these frameworks must not be used.

4.6 Your responsibilities throughout the project

As this is an academic exercise (as opposed to an industrial assignment) where the objective is to learn, **you are all expected to participate in all activities (requirements analysis, architecture and design, coding, testing) on a relatively equal weight**. Failure to do so will result in penalties to the individual(s) involved as well as to the team leader.

For each member of your team, we will monitor the following: a) number of commits, b) number of assigned tasks, c) the total number of tasks, and d) the degree of importance (of committed tasks), where the latter is defined as follows:

0: Unimportant commits, e.g. comments.

1: Minor commits, e.g. changing the names of functions or variables.

2: Important commits, e.g. adding new functions into classes.

3: Very important commits, e.g. such as adding a class.

In other words, the notion of "relatively equal weight" of workload does not exclusively depend on the number of your commits, but on all four factors (a - d) described above.

5 Assessment

This section provides all necessary information about your assessment: Marking, monitoring of your progress, dealing with slacking cases, due date of the final version and late submissions and important dates.

5.1 Marking

A marking scheme is made available to you in an accompanying document. Note that even though this is a group project, marking is not assigned collectively. Based on the criteria described in this document, individual penalties will result in different people from the same team possibly receiving a different project mark.

5.2 Monitoring your progress

Our markers will monitor your account and obtain an image of your workspace at the end of each iteration. On the due date and time we will obtain a final image of your workspace. Please do not submit either a hard copy or an electronic copy of your artifacts by using the ENCS electronic submission system.

Each team member will have to submit a peer review at the end of each iteration, due on a Friday at 19:00. This peer review would include only the current iteration and it should be emailed as an attachment to the instructor. Your email should be in plain text, and formatted as follows:

```
Subject:  [SOEN343] Peer review by <name, id> (Team X, Iteration Y)
```

where `id` is your Concordia id, `X` is your team number, and `Y` is the current iteration. For example:

```
Subject:  [SOEN343] Peer review by Syd Barrett, 12345 (Team 1, Iteration 2)
```

5.3 Slacking off

Accommodating a slacker will result in the team leader receiving a penalty. As a team, you have the power to vote a team member out by majority vote. In the case of a tie vote, the team leader has the privilege of a casting vote. In the case where the alleged slacker is the current team leader, then the team leader loses their privilege of a casting vote and you (the entire team) must arrange a meeting with the instructor.

5.4 Due date of final version and late submissions

The final version of your project is due by **Friday 24 November at 17:00**. Any late submission by 24 hours will get a 50% penalty and it will subsequently receive a 10% penalty per day. What is the meaning of "late submission"? If by the due date your project is not ready, please inform the instructor. We will then not download your workspace until you tell us that it is ready.

5.5 Important dates

- 15 Sep (by 17:00): Team leaders should send their team's credentials to the instructor: Team formation, URL's of collaborative document development account, version control, and issue tracking, and URL of where application will be hosted.
- 29 Sep, 13 Oct, 27 Oct, 10 Nov, 24 Nov (by 17:00): Your iteration is concluded and your artifacts are under assessment.
- 29 Sep, 13 Oct, 27 Oct, 10 Nov, 24 Nov (by 19:00): Peer review form by all.