CSYE 6202 Final Project Instructions

Last Updated: November 18, 2015

General Information

The objective of this final project is to give the students to demonstrate and apply what they learned from the course so far.

The expectation from you is to apply all the knowledge and good software programming practices where you see fit.

Both projects outlined here are individual efforts and you will be responsible from start to finish. This means you are responsible from all the stages of developing a software which includes designing the software (using UML diagrams and Balsamiq to communicate your overall design), implementing it using the C# programming language and .NET Framework, providing unit tests (using NUnit) where applicable, and eventually presenting your solution/ideas to your fellows.

There are couple project ideas outlined down below. You need to select one of these project ideas and implement a software for it. If you would like to work on a project idea of your own, then you need to discuss this with me so we can make sure that the complexity and the scope of the project is appropriate for a Final Project.

Airline Reservation System

Clinic Management System

CSYE 6202 Final Project Instructions

Airline Reservation System

Overview

This project will focus on the workflows and operations of an Airline Reservation System. The system will allow the end user to book/unbook and see information on selected bookings as well as all current flights.

Outlined below is the minimum functionality that is required from this software. There are no data restrictions, and no detailed explanations on how to handle data or on how the forms or your data should look like. You are free to implement your ideas as soon as you explain and present them to the crowd using UML and plain documents.

Required Functionality

1. Users should access to the application through a login screen
2. There can be multiple flights
    a. Each flight has a
        i. Unique name
        ii. Unique date/time
        iii. Unique destination/arrival
        iv. Unique carrier
        v. Economy/Economy Plus/Business class seats and associated prices
    b. You decide on how this information looks like, but it needs to be consistent throughout the application
3. End user should be able to see all the flights in one location
4. End user should be able to see all detailed info about the flight – This includes
    a. Carrier info
    b. Flight crew info
    c. All passengers booked for that flight so far
    d. How much each user paid for that flight (Dependent on the seat level mentioned above)
5. End user should be able to book a flight for at least one customer at any given point
6. End user should be able to unbook a flight for at least one customer at any given point
7. The app must provide proper logging (can be plain file, a table in a DB, or even a XML file) to trace what the app is doing at any given point (Log file can be saved in any location, but needs to be mentioned in design documents)
8. The app should be recording all data using a persistence layer. This can be either an XML file or a DB

Extra Credit [25 points extra]

1. Users have different access levels:
    a. Administrator – Can also execute the workflows mentioned below
    b. Regular – Restricted to the workflows mentioned above only
2. Administrator can introduce/update/delete a flight
3. Administrator can introduce/update/delete a flight carrier
4. Administrator can introduce/update/delete flight crew
5. Instead of XML, use ADO.NET as the backend/persistence layer

Deliverables

1. UML documents to communicate the overall design for your implementation (You can use http://yuml.me/ or even a plain Word document would suffice)
2. Balsamiq documents to show your design docs
3. Code implementation (As always your implementation should apply all software programming best practices, as well as OOP practices where applicable)
4. NUnit tests where applicable
5. Backend/persistence layer implementation using either a XML file or a DB
6. Your presentation slides/documents

CSYE 6202 Final Project Instructions

Clinic Management System

Overview

This project will focus on the workflows and operations of a Clinic Management System. The system will allow the end user to do the CRUD operations (Create, Read, Update, and Delete) for scheduling appointments for patients. The user should be also able to see detailed information on selected appointments such as patient info, appointment date/time, visit reason, doctor/nurse info etc.

Outlined below is the minimum functionality that is required from this software. There are no data restrictions, and no detailed explanations on how to handle data or on how the forms or your data should look like. You are free to implement your ideas as soon as you explain and present them to the crowd using UML and plain documents.

Required Functionality

1. Users should access to the application through a login screen
2. Each appointment has a
    a. Unique name
    b. Unique date/time
    c. Unique doctor/nurse assigned to it
    d. Patient info

    You decide on how this information looks like, but it needs to be consistent throughout the application

3. End user should be able to see all the appointments in one location
4. End user should be able to see all detailed info about each scheduled appointment. This includes
    a. Patient info
    b. Assigned healthcare professional info
    c. Date/time of the appointment
    d. Visit reason/type
5. End user should be able to schedule an appointment for at least one patient at any given point
6. End user should be able to remove a scheduled appointment for at least one patient at any given point
7. The app must provide proper logging (can be plain file, a table in a DB, or even a XML file) to trace what the app is doing at any given point (Log file can be saved in any location, but needs to be mentioned in design documents)

8. The app should be recording all data through a persistence layer. This can be either an XML file or a DB (extra credit)

Extra Credit [25 points extra]

1. Users have different access levels:
    a. Administrator – Can also execute the workflows mentioned below
    b. Regular – Restricted to the workflows mentioned above only
2. Administrator can introduce/update/delete a healthcare professional (doctor or nurse)
3. Administrator can introduce/update/delete a visit reason/type
4. Administrator can introduce/update/delete a visit cost
5. Instead of XML, use ADO.NET as the backend/persistence layer

Deliverables

1. UML documents to communicate the overall design for your implementation (You can use http://yuml.me/ or even a plain Word document would suffice)
2. Balsamiq documents to show your design docs
3. Code implementation (As always your implementation should apply all software programming best practices, as well as OOP practices where applicable)
4. NUnit tests where applicable
5. Backend/persistence layer implementation using either a XML file or a DB
6. Your presentation slides/documents

Additional Notes

- Any additional functionality on top of the minimum required functionality will be awarded with extra points given that it is working and well documented (This holds true even if it is a new feature that you introduced and it is not mentioned here)
- Please check all the deliverables to the related GitHub branch (CSYE-6202/FinalPro) as well as send a zip file to me @ barisyanmaz@gmail.com

Deadlines

- For this assignment the deadline is **12/11/2015 @ 4pm** and hard-deadline is **12/18/2015 @ 4pm.** Unfortunately projects sent over after this deadline will not receive any credit
- You are supposed to give a small presentation and showcase your ideas and solutions to your fellow developers. This will take place on **12/11/2015** during our regular class hours and unfortunately there is no extension time for this. Definitely let me know beforehand if there is an emergency situation that would prevent you giving the presentation on that day