MSSA Cloud Application Development Class Project:

Overview

Adapted from Charles C. Carter's 22 Nov 2017 proposal

OVERVIEW

Introduction

The MSSA Cloud Application Development Class project spans the entire 18 weeks of the MSSA program, with weekly student activities producing incremental deliveries to achieve the project's overall outcomes. The table below summarizes the project. It consists of four phases: inception, data component, logic component, and interface component.

Learning Objectives. Upon successful completion of the Class Project, the student:

1. Appreciates what a typical, real-world software development project entails by applying concepts throughout the Cloud App Dev program to a selected project's creation.

2. Demonstrate software project management by constructing a portfolio of results that illustrate, document, and exhibit the project they've worked upon

Week	Phase	Topic	Deliverables	Graded In
1	Inception	Explore characteristics of proposed projects	1 page write-up	ISTA 220
2	Inception	Specific project description	2 page write-up	ISTA 220
3	Inception	Presentation	PowerPoint	ISTA 220
4	Data	Conceptual design of database	ERD	ISTA 420
5	Data	Logical design of database	ERD	ISTA 420
6	Data	Physical design of database	Database diagram	ISTA 420
7	Data	Implementation	SQL source listing	ISTA 420
8	Data	Presentation	PowerPoint	ISTA 322
9	Logic	Requirements analysis	SRS	ISTA 322
10	Logic	Design	UML diagrams	ISTA 421
11	Logic	Implementation	C# source listing	ISTA 421
12	Logic	Testing	C# source listing	ISTA 421
13	Logic	Presentation	PowerPoint	ISTA 421
14	Interface	Wireframe	Image file	ISTA 421
15	Interface	Prototype	Semi-functional layout	ISTA 421
16	Interface	Mockup	Image file	ISTA 421
17	Interface	User Interface	Source listing	ISTA 421
18	Interface	Presentation	PowerPoint	ISTA 421

The project spans the entire MSSA Cloud App Dev program experience; as it progresses, its focus changes, and this is reflected in the way that the weekly graded elements of the project are booked into different courses.

Let's look at the phases of the project:

Inception

The first three weeks of the MSSA program will consist of the selection of the Class project. Students are not expected to have the technical skills necessary to begin the project, but are expected to have a layman's grasp of software applications and an interest in some knowledge domain that they can leverage throughout their course.

1.1. Preliminary research

Students will begin thinking about the kind of project they would want to showcase for their Class. They should list the specific functionalities of their chosen kind of project, and so some preliminary investigation toward implementing the project. All projects should include a data component and an interface component. The deliverable should be a one-page document listing several specific projects they would be interested in doing, and a short description of the characteristics of each project.

1.2. Project selection

Students will select one project from their list to focus on for their Class project They should make a formal investigation of the requirements of the project. If possible, they should identify similar applications that have been written. The deliverable is a two-page project proposal. The project proposal should contain a sufficient description of the project that would allow construction of the project to begin.

1.3. Proposal presentation

Presentation: students will prepare a slide presentation and make an oral presentation of the project to the class. Deliverables include the slide presentation and a video of their oral presentation.

Data

This phase concentrates on using persistent data using a relational database (SQL Server). The outcome of this phase will be a fully functional database.

1.4. Conceptual design

Students will discover the nature of conceptual database design. The deliverable will be an entity-relationship diagram showing the conceptual design of their project database.

1.5. Logical design

Students will discover the nature of logical database design. The deliverable will be an entity-relationship diagram showing the logical design of their project database.

1.6. Physical design

Students will discover the nature of physical database design. The deliverable will be an database diagram showing the physical design of their project database.

1.7. Implementation

Students will write a SQL source listing implementing their database. The deliverable will be the SQL source listing.

1.8. Presentation

Presentation: students will prepare a slide presentation and make an oral presentation of the database to the class. Deliverables include the slide presentation and a video of their oral

presentation.

Logic

The logic component includes the business rules for the application. This should be accessible from a console interface. It should be a fully functional version of their implemented project.

1.9. Requirements analysis

Students will create a software requirements specification (SRS) of the functional requirements of their project. (The Class project does not include non-functional requirements.) The deliverable will be the project SRS.

1.10. Design

Students will create appropriate design artifacts. These will include a class diagram, a component diagram, a system sequence diagram, and other documents as necessary (e.g., data flow diagrams, state models, etc.) The deliverable will be a collection of design diagrams.

1.11. Implementation

Students will implement the logic component of their project. The deliverable will be the source code listings of their implementation. These will not include auxilliary files, such as container folders, configuration files, etc.

1.12. Testing

Students will write automated tests for their application using both white box and black box techniques. All test code should be automated. The deliverable will be the source listiong for the test suite.

1.13. Presentation

Presentation: students will prepare a slide presentation and make an oral presentation of the logic component to the class. Deliverables include the slide presentation and a video of their oral presentation.

Interface

This phase concentrates on a graphical user interface. Students will use either UWP or ASP as they choose.

1.14. Wireframe

A wireframe is a sketch of the system to be built. Wireframes should clarify exactly what elements realize the different features on all pages or screens of a future product but without full details. The actual screen design will be produced at a later stage by referencing the wireframe. Deliverable will be an image file.

1.15. Prototype

A prototype, in this case, is a semi-functional layout that can give a high-fidelity preview of the actual app or website user interface (front-end) functionality. While prototype might not have full functionality, it generally gives customers and/or end-users an ability to click around the elements of the interface and simulate the way the app will actually work. Deliverable will be a source listing.

1.16. Mockup

The mockup demonstrates all the graphics, typography, colors and other page elements. Mockups are about presenting how what the final product would look like. Deliverable is an image file.

1.17. Implementation

Students will implement the user interface of their project. The deliverable will be the source code listings of their implementation. These will not include auxilliary files, such as container folders, configuration files, etc.

1.18. Presentation

Presentation: students will prepare a slide presentation and make an oral presentation of the user interface to the class. Deliverables include the slide presentation and a video of their oral presentation.

Conclusion

Throughout the project, the instructor will provide students with additional guidance for each phase. For example, during the Data phase, he might discuss the database design process including normal forms and integrity constraints. During the Logic phase, she might discuss iterative, incremental processes and UML. During the Interface, principles of user interface design might be further explored.

Instructors should use the Class project as an opportunity to add value to the curriculum by exposing the students to topics not expressly included in the curriculum, such as version control, software engineering, software quality control (testing), security, design patterns, and so on. This should be at the instructor's discretion.

The Class project is a *supplement* to the official curriculum. As such, it should enhance the student's experience, and not detract from it by overloading the student with an amount of work that is impossible to do. The emphasis is on understanding the topics presented and building a portfolio, not building a completely functional application.

Project Assignment, Step 1

Preliminary research

Students will begin thinking about the kind of project they would want to showcase for their project. They should list the specific functionalities of their chosen kind of project, and some preliminary investigation toward implementing the project. All projects should include a data component and an interface component. The deliverable should be a two page document listing several specific projects they would be interested in doing, and a short description of the characteristics of each project.

You may choose your own project. Here is a list of sample projects for you to consider.

School system

This may consist of a student module, an instructor module, a curriculum module, and provides for the assignment of instructors to classes, students to classes, course management, student management, an d instructor management. As example would be the CSU student information system.

Electronic learning system

This may consist of a collection of courses. Each course may allow for tests, quizzes, exercises, research papers, discussions, etc. As example would be Cougarview.

An auction system.

This my consist of a sellers module, a bidders module, a purchase module, and provide for different kinds of sales (i.e., open bidding, bidding with reserve, buy now, etc.). It would include user registrations and appropriate databases of auctions, transactions, etc. An example would be eBay.

Messaging system

This would include user registrations, posting messages, sending messages, composing groups, and similar functionality. An example would be Twitter.

Encyclopedia

This would include editor registrations, posting entries, editing entries, search functions, analytical functions, and similar functionality. An example would be Wikipedia.

Electronic voting system

This would include voter registrations, candidate qualification, casting ballots with appropriate authentication, and calculation of results. An example would be the current voting system in Georgia.

Networking system

This would include user registrations, posting employment data, school data, certification data, allow the posting of jobs, etc. An example would be LinkedIn.

Online store
This would implement the display of merchandise, a shopping cart, a payment module, and inventory control mechanism, etc. An example would be Amazon.

Database GUI
This would implement a graphical front end to a database. Modules would (1) create or drop databases, (2) create, drop, and alter tables, (3) insert, update, or delete data, and (4) run simple queries. An example would be TOAD.

Organization system
This might include member registration, a member directory, an event calender, a newsletter, a photo album, a FAQ, or other functions suitable for organizations. Examples would include a social organization, a service organization, a band or orchestra, a church, an athletic team, etc.

Automated teller machine
This might include customer authentication, checking account status, acceptance of deposits, and dispensing cash.

A strategy game
At a minimum, this would consist of a user interface, a data source, and a logic engine implementing the game rules.

Deliverable

Your deliverable will be a two page discussion of three or four proposed projects. Your description of each proposed project should contain enough information so that readers can understand the nature of the project. You may also give examples of similar, existing implementations.

> The format of the paper will eventually use *Markdown*. We will not cover this until the third week. For now, please write your paper in plain ASCII text with no formatting, perhaps using Microsoft Notepad or a similar application. In general, your papers should consist of a short introduction, a short conclusion, and a body containing the substance of your discussion.
>
> Your paper should have a title, author, and date. The subdivisions of your paper should have appropriate section and subsection headings.

Version Control

The topic for this week will be version using the Git Version Control System. The following will be covered.

- The rationale for version control in software development
- Installation of the Git SCM software
- The command `git config` for the user name, user email, and core editor
- The command `git init`
- The command `git add` with the appropriate arguments
- The command `git status`
- The command `git commit` with the appropriate arguments

- The command `git log`

Project Assignment, Step 2

Preliminary selection

Students will select one project from their list to focus on for their project They should make a formal investigation of the requirements of the project. If possible, they should identify similar applications that have been written. The deliverable is a six to eight paragraph project proposal. The project proposal should contain a sufficient description of the project that would allow construction of the project to begin.

Deliverable

Your deliverable will be a six to eight paragraph discussion of your proposed project. Your proposal should begin with a short introduction and conclude with a short conclusion. It should include discussions of the purpose of the software, an overall description of the high-level functional requirements of the software (that is, what the software will actually do), a survey of the relevant literature available that will assist you in completing your project, a short discussion of similar software, and a brief discussion of your project plan. Your writeup is not limited to two pages, and can include other sections that you feel will help the reader in understanding your proposal.

The format of the paper will eventually use *Markdown*. We will not cover this until the third week. For now, please write your paper in plain ASCII text with no formatting, perhaps using Microsoft Notepad or a similar application. Your paper should have a title, author, and date. The subdivisions of your paper should have appropriate section and subsection headings.

Distributed Version Control

The topic for this week will be version control using the Github Distributed remote repository. The following will be covered.

- The rationale for distributed version control in software development
- Signing up for a Github account
- Creating your first remote repository
- The command `git remote add` with the appropriate arguments
- The command `git remote -v`
- The command `git push` with the appropriate arguments
- a `.gitignore` file
- a `README.md` file

Markdown

Markdown is a very simple, human readable, formatting system. A Markdown document is a simple text document, with no binary codes or special formatting instructions. It contains only printable characters, such as alphabetical characters, punctuation, and digits. The file extension is

`.md`. This is half of all the Markdown you will need to know. You can pick up the other half just as easily.

- Headings
- Paragraphs
- Itemized lists
- Enumerated lists
- Hyperlinks
- Source listings

Project Assignment, Step 3

Introductory Presentation

For this week's assignment, you will prepare and create a short (five minute) video presentation and a short written presentation. In your presentation, youu should (1) introduce yourself, (2) talk about your participation in the MSSA program, (3) and introduce your project. When you introduce your project, you should generally adapt your deliverable from last week. You will not necessarily read your deliverable from last week (although you could if you really wanted to), but explain your project as if you were giving a short oral presentation to an employer. In fact, you could treat your presentation as if it were part of an employment interview.

Deliverable

You have two deliverables, one written and one oral. You should first prepare your written presentation. The format of the paper will use *Markdown*. Your paper should have a title, author, and date. The subdivisions of your paper should have appropriate section and subsection headings.

The second deliverable is a video presentation, This should be a short video, not more than five minutes, prepared using your laptop camera and software. If your laptop did not come with video software, VSDC is free and works well.

You should upload both your written presentation and your video presentation to your project account in Github.

Software Life Cycle

A brief introduction to the software life cycle:

- Inception
- Elaboration
- Construction
- Transition
- Maintenance
- End of life

Software Development Cycle

A brief introduction to software development workflows:

- Requirements
- Analysis
- Design
- Implementation
- Testing

Software Process Models

A brief introduction to software process models:

- Incremental development
- Iterative development
- Spiral models
- Waterfall
- Rational Unified Process
- Agile processes

Project Assignment, Step 4

Database Conceptual Design

Typically, a database project consists of five or six phases: requirements engineering, conceptual design (analysis), logical design, physical design, implementation, and testing. For the CAD project, we will only cover conceptual design, logical design, physical design, and implementation. We will not cover requirements engineering nor testing. We do not cover the requirements phase because, presumably, since you are your own client/customer, you will already know your requirements. We do cover testing, but not in the database phase of the project. During the later phases of the project, you will be able to test your database design and revise it as necessary.

Generally, the conceptual design phase of a database project requires developers to break down the problem domain into relevant objects, called entities, and identify the relationship between objects. For example, a school might have courses, teachers, and students: teachers *teach* courses, students *enroll in* courses, and teachers *grade* students. The entities are TEACHER, STUDENT, and COURSE. The relationships are TEACH, ENROLL IN, and GRADE.

For this week's assignment, you will prepare a conceptual design of your project database and create an *entity-relationship diagram*. This process requires you to identify the entities that your project deals with and the relationship between the entities. This process also requires you to identify important attributes of your entities, and (if necessary) of your relationships. Examples of important attributes for STUDENT would be ID and NAME, and for COURSE would be CRN and TITLE.

We will also discuss the Dia Diagram Editor, https://sourceforge.net/projects/dia-installer/.

Deliverable

Your deliverable is an Entity Relationship Diagram (ERD). Your ERD is an image, and you should prepare it in some graphical format. The preferred format is PDF, but other image formats are acceptable, such as SVG, JPEG, GIF, PNG, EPS, etc.

Entities, Relationships, and Attributes

The following topics will be discussed:

- What is *conceptual database design*
- What is an *entity*
- What is an *entity attribute*
- What is a *relationship*
- What is a *relationship type*
- What is a *relationship degree*
- What is a *relationship attribute*
- What is a *candidate key*
- What is a *super key*

Project Assignment, Step 5

Database Logical Design

Typically, a database project consists of five or six phases: requirements engineering, conceptual design (analysis), logical design, physical design, implementation, and testing. For the CAD project, we will only cover conceptual design, logical design, physical design, and implementation. We will not cover requirements engineering nor testing. We do not cover the requirements phase because, presumably, since you are your own client/customer, you will already know your requirements. We do cover testing, but not in the database phase of the project. During the later phases of the project, you will be able to test your database design and revise it as necessary.

Generally, the logical design phase of a database project requires developers to decompose the entities identified by the conceptual design and create a logical schema. This means that the entities become tables (relations), and the attributes are expanded and identified as fields. During this process, the database is normalized and integrity constraints are realized. Logical design is much more of an art than a science, and is notoriously difficult.

For this week's assignment, you will prepare a logical design of your project database and create an *database diagram*. This process requires you to identify the relations (tables), that you normalize your database, and that you specify all fields (attributes) of your tables.

Deliverable

Your deliverable is an Database Diagram. Your database diagram is an image, and you should prepare it in some graphical format. The preferred format is PDF, but other image formats are acceptable, such as SVG, JPEG, GIF, PNG, EPS, etc.

Normalization and integrity constraints

The following topics will be discussed:

- What is an *first normal form*
- What is an *second normal form*
- What is an *third normal form*
- What is an *entity integrity*
- What is an *domain integrity*
- What is an *referential integrity*

Thought Exercise

You are given an assignment to build a database that models student enrollments in courses. In your conceptual design phase, you identify two entities, students and courses, and one relationship, `[Students] Enroll in [Courses]`. In your first attempt, you see immediately that your database is not in first normal form in that multiple students enroll in the same course and multiple students enroll in the same course. It's not in second normal form because student attributes do not depend on the course primary key, and course attributes depend do not on the student primary key. You see that it also (probably) is not in third normal form because some non-key attributes uniquely identify other non-key attributes.

Complete a logical design of this problem. We will discuss this in class. In my solution to this exercise, I designed five different tables. You may or may not have the same number of tables in your solution.

Project Assignment, Step 6

Database Physical Design

Typically, a database project consists of five or six phases: requirements engineering, conceptual design (analysis), logical design, physical design, implementation, and testing. For the CAD project, we will only cover conceptual design, logical design, physical design, and implementation. We will not cover requirements engineering nor testing. We do not cover the requirements phase because, presumably, since you are your own client/customer, you will already know your requirements. We do cover testing, but not in the database phase of the project. During the later phases of the project, you will be able to test your database design and revise it as necessary.

Physical design targets a particular relational database management system, such as SQL Server or SQLite. Logical designs target an abstract database, but physical designs must be specific to the particular database system. All elements must be specified, such as the constraints we will discuss. You should be able to hand your physical design to a SQL programmer, and he should be able to write the implementation code without reference to anything other than your physical design.

For this week's assignment, you will prepare a physical design of your project database and create a *database diagram*. This process requires you to create tables, create columns, specify

Deliverable

Your deliverable is a Database Diagram. Your database diagram is an image, and you should prepare it in some graphical format. The preferred format is PDF, but other image formats are acceptable, such as SVG, JPEG, GIF, PNG, EPS, etc.

Database objects and constraints

The following topics will be discussed:

- What is are *column level constraints* and *table level constraints*
- What is a *primary key*
- What is a *foreigh key*
- What are *insert, delete, and update anomolies*
- What is a *nullability constraint*
- What is a *uniqueness constraint*
- What is a *datatype*
- What is an *index*
- What is an *enumeration*
- What is a *check constraint*

Project Assignment, Step 7

Database Implementation

Your assignment is to implement the database you designed in your last deliverable. You should create a new database, define your tables, and fully implement the tables. You should also insert a sufficient amount of data into your tables to allow for testing, and develop a sufficient number of queries to test your database.

Your deliverable is a SQL file containing your implementation code. You should also deliver a TXT file containing the results of the execution of your SQL file.

Project Assignment, Step 8

Database Presentation

For this week's assignment, you will prepare and create a short (five minute) video presentation and a short written presentation. In your presentation, you should (1) introduce yourself, (2) give a short summary of your project, (3) discuss the information architecture of your project, and (4) discuss the design and implementation of your database. You can use the deliverables from the last four weeks as building blocks for your presentation.

Deliverable

You have two deliverables, one written and one oral. You should first prepare your written presentation. The format of the paper will use *Markdown*. Your paper should have a title, author, and date. The subdivisions of your paper should have appropriate section and subsection headings.

The second deliverable is a video presentation, This should be a short video, not more than five minutes, prepared using your laptop camera and software. If your laptop did not come with video software, VSDC is free and works well.

You should upload both your written presentation and your video presentation to your project account in Github.

Software Development Cycle

A brief review to software development workflows:

- Requirements
  - requirements engineering
  - collaboration with stakeholders
  - use cases (user stories)
- Analysis
  - functional requirements
  - non-functional requirements
  - Software Requirements Specification (SRS)
- Design
  - Unified Modeling Language (UML)
  - static design diagrams
  - dynamic design diagrams

- Implementation
- Testing

Object Oriented Development Principles

A brief review to software development principles for OOAD:

- Single responsibility principle (SRP): This principle states that software component (function, class or module) should focus on one unique tasks (have only one responsibility).
- Open/closed principle (OCP): This principle states that software entities should be designed with the application growth (new code) in mind (be open to extension), but the application growth should require the smaller amount of changes to the existing code as possible (be closed for modification).
- Liskov substitution principle (LSP): This principle states that we should be able to replace a class in a program with another class as long as both classes implement the same interface. After replacing the class no other changes should be required and the program should continue to work as it did originally.
- Interface segregation principle (ISP): This principle states that we should split interfaces which are very large (general-purpose interfaces) into smaller and more specific ones (many client-specific interfaces) so that clients will only have to know about the methods that are of interest to them.
- Dependency inversion principle (DIP): This principle states that entities should depend on abstractions (interfaces) as opposed to depend on concretion (classes).

Project Assignment, Step 9

Programming — Requirements Analysis

The Cloud Application Development Project consists of four parts, an introductory part, the database part, the software development part, and the interface part. All these parts can be considered integral to software development, but they are approached differently. Last week, we concluded the database portion of the project. This week, we begin five week looking specifically at programming development. We can call this *software emngineering* in a narrow sense.

The discipline of software engineering includes various workflows in building software applications. These workflows are the same as engineering workflows in other engineering disclipines, such as civil and automotive engineering. These workflows consists of the following:

- Requirements
- Analysis
- Design
- Implementation
- Testing

Software engineers use many different processes, among them waterfall, the Rational Unified Process, and various agile processes, including Scrum and Kanban. Different processes order and emphasize the workflows differently, but despite their differences, all processes use the same workflows. The process we will use in this project is a modified version of Extreme Programming (XP). If you are interested in XP, you can read about it offline.

In this project, we will not use the *requirements* workflow. In gathering requirements, developers try to understand the tasks that the application is to model. Developers engage in many different kinds of activities, such as reviewing paper files and other business documents, interviews with employees and other stakeholders, observation, and so on. The object of the requirements phase is to discover what it is that the software should actually do. The requirements workflow is incredibly important. After all, how can you build an application if you do not know what it will do? In fact, the largest number of software defects are failures in requirements. We build error-prone software because we fail to understand what it's supposed to do.

Despite this, we will not be doing the requirements workflow in this project. We are constrained by time, and since your project is *your* project, we can assume that you know what you want it to do, (This assumption may not be justified, but that's another issue.) Our first software workflow for this project will be the *analysis* phase. In the next three weeks, we will do design, implementation, and testing.

The object of analysis is to develop a *software requirements specification* (SRS) from the requirements. A SRS is a formal written document that derives directly from the requirements and addressed to software designers. Generally, requirements can be seen as *functional* and *non-functional* requirements. Functional requirements consist of discrete objectives stating exactly what the software will do. Non-functional requirements consist of other requirements, such as machine capability, availability and uptime, GUI requirements, and other requirements that are necessary for the software to perform but do not describe the functions that the software should perform. In this project, we will only consider fumctional requirements.

We will base our discussion on IEEE Std 830-1998, *IEEE Recommended Practice for Software Requirements SpeciÞcations*, June 25, 1998. This document has been superseded, but purchase is required for the current version. You can find copies of Std 830-1998 freely available. From the introduction of Std 830-1998:

This recommended practice describes recommended approaches for the specification of software requirements. It is based on a model in which the result of the software requirements speciÞcation process is an unambiguous and complete specification document. It should help

i. Software customers to accurately describe what they wish to obtain;

ii. Software suppliers to understand exactly what the customer wants;

iii. Individuals to accomplish the following goals:

    1. Develop a standard software requirements specification (SRS) outline for their own organizations;

    2. Define the format and content of their specific software requirements specifications;

    3. Develop additional local supporting items such as an SRS quality checklist, or an SRS writer's handbook.

We will focus our study on sections 4.3, *Characteristics of a good SRS*, and 5.3, *Specific requirements (Section 3 of the SRS)*.

Deliverable

You deliverable is a formal SRS, written using Markdown formatting. You should upload your SRS to your project account in Github.

TERM 2, Steps 10-18