**Development Project I** (420-K40-HR)

**Assignment 01 – Technical Investigation & Client Meeting**

Date assigned: Monday, September 12, 2024

Date due: Final Due Date September 26, 2024, before midnight

Part A – Sept 17 2024, before midnight

Part B – Sept 19 2024, before midnight

Part C – Sept 26 2024, before midnight

Late submissions will not be accepted without prior written approval.

Contents

[Part A Section 1 - Skill Set inventory and gap analysis 2](#_Toc177022092)

[Part A Section 2 - Technical research – legacy projects 3](#_Toc177022093)

[Part A Section 3 – Authorization and Authentication 4](#_Toc177022094)

[AMS – Part 1 4](#_Toc177022095)

[AMS – Part 2 5](#_Toc177022096)

[Part A Section 4 – Technology Decisions 6](#_Toc177022097)

[.NET Framework vs .NET Core 6](#_Toc177022098)

[Webforms vs MVC 6](#_Toc177022099)

[Reflect 7](#_Toc177022100)

[Part B Section 1 – Dependency Injection 9](#_Toc177022101)

[Part B – Testing with Mock objects 10](#_Toc177022102)

[Part C Section 1 - User Meeting 13](#_Toc177022103)

[Part C Section 2 – Elevator Pitch and User Stories 14](#_Toc177022104)

[Appendix 1: Connecting your DevOps instance to Visual Studio 15](#_Toc177022105)

[Visual Studio 2019 15](#_Toc177022106)

**All student answers should be in blue text so it is easily identifiable by the instructor.**

**Learning Objectives**

Upon successful completion of these tasks, the individual students will:

* Ramp up on common skills, frameworks and technologies that will be options in the architecture and design
* Practice navigating and accessing projects and source code repositories via Microsoft Azure DevOps
* Familiarize with existing projects to use as a basis for future design choices.
* Familiarize the student with the development environment, including Microsoft Azure DevOps, and Visual Studio.
* Introduce and Understand the concepts of Dependency Injection and Mock objects

Skill set:

|  |  |  |
| --- | --- | --- |
| Skill | Options | Skill Acquisition |
| Database options | Entity Framework (model first, database first, code first)  No entity framework  *(hand coded Data Access Layer i.e. write your own CRUD or queries)*  Other | Web |
| UI options | MVC  WebForms  Other | Web, Maintenance |
| AMS | Understand the current authorization and role management systems that you must integrate into | Later |
| .Net Framework | .net core, .net Framework | Web |
| TDD | Modern test methods  *Talk about Acceptance-based TDD ( ATDD) as well.* | Web/This assignment/later |
| Dependency Injection | Modern test methods | Web/This assignment/later |
| MOQ  (testing with mock) | Modern test methods | Web/This assignment/later |

Thoroughly read this assignment. Read it alone and then compare your notes with your teammates after you’ve done a full review of the assignment. Coordinate your shared activities and plan those tasks on the Azure DevOps task board for your team.

# Part A Section 1 - Skill Set inventory and gap analysis.

Using the Skill set table (above) as a starting point.

Hint: Don’t Panic. You’re not expected to be an expert on all this technology. At the beginning of the project, it’s important to know the skill gaps that need to be filled for the project. This is just an inventory and analysis.

Add another column to indicate your experience in each skill in terms of person weeks. (i.e. if you have done 2 weeks of MVC, 8 weeks of WebForms).

Add any other rows in terms of skills/infrastructure/frameworks you think you’d need for implementing any of the projects.

|  |  |  |  |
| --- | --- | --- | --- |
| Skill | Options | Current skill level (person weeks per option) | Concerns/risks |
| Database options | Entity Framework (model first, database first, code first)  No entity framework  *(hand coded Data Access Layer i.e. write your own CRUD or queries)* | 12 weeks  10 weeks |  |
| UI options | MVC  WebForms | 20 weeks  20 weeks | Not spending too much of our time with the UI, and prioritizing functionality |
| AMS | Understand the current authorization and role management systems that you must integrate into | 0 weeks | If AMS will be accessible this semester |
| .Net Framework | .net core, .net Framework | 16 weeks |  |
| TDD | Modern test methods | 8 weeks | TDD slowing down our development |
| Dependency Injection | Modern test methods | 3 weeks |  |
| MOQ  (testing with mock) | Modern test methods | 0 weeks |  |
| Other web languages | JavaScript | 20 weeks |  |

Add any other general concerns/risks about any of the projects in general:

Worried that Portage Cybertech’s Project will be too ambitious and may not be possible for any of the teams, but also excited to be able to work with an external client and put it on my resume. Also worried that Richard’s assignment will be far too technical within a cybersecurity scope and will be out of our reach for feasibility.

# Part A Section 2 - Technical research – legacy projects

Before you start any new project, you can benefit from understanding projects that came before you so that you can learn/copy/steal their best practices.

The Computing Science department’s DevOps instance can be found at this local URL:

<https://csazure.cegep-heritage.qc.ca:8080/>

*Note that this system is not accessible via the Internet (aka remotely), and you will need to use the VPN to access it from home or outside of the campus network.*

Find the following projects and fill in the following table:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Project | URL | Programming Language | UI (WebForms/MVC) | Code snippet for AMS authentication/authorization  (provide source code filename and line number as well as code snippet) | Demonstrated DevOps / VS download and compile |
| NST | [Summary - Overview (cegep-heritage.qc.ca)](https://csazure.cegep-heritage.qc.ca:8080/NST/NST) | C# | MVC | public interface LoginSoap  {    [System.ServiceModel.OperationContractAttribute(Action="https://cegep-heritage.qc.ca/Authorize", ReplyAction="\*")]  System.Threading.Tasks.Task<LoginService.AuthorizeResponse> AuthorizeAsync(LoginService.AuthorizeRequest request);    [System.ServiceModel.OperationContractAttribute(Action="https://cegep-heritage.qc.ca/Authenticate", ReplyAction="\*")]  System.Threading.Tasks.Task<LoginService.AuthenticateResponse> AuthenticateAsync(LoginService.AuthenticateRequest request);  } |  |
| CES | [Summary - Overview (cegep-heritage.qc.ca)](https://csazure.cegep-heritage.qc.ca:8080/CES/CES) | Virtual basic, JS | n/a | <s:schema elementFormDefault="qualified" targetNamespace="https://cegep-heritage.qc.ca">  <s:element name="Authorize">  <s:complexType>  <s:sequence>  <s:element minOccurs="0" maxOccurs="1" name="username" type="s:string" />  <s:element minOccurs="0" maxOccurs="1" name="appCode" type="s:string" />  </s:sequence>  </s:complexType>  </s:element> |  |
| HERS | [Summary - Overview (cegep-heritage.qc.ca)](https://csazure.cegep-heritage.qc.ca:8080/HERS/HERS) | C#, JS | WebForms | [System.ServiceModel.OperationContractAttribute(Action="https://cegep-heritage.qc.ca/Authenticate", ReplyAction="\*")]  System.Threading.Tasks.Task<LoginService.AuthenticateResponse> AuthenticateAsync(LoginService.AuthenticateRequest request);  } |  |
| AMS | [Summary - Overview (cegep-heritage.qc.ca)](https://csazure.cegep-heritage.qc.ca:8080/AMS/AMS) | C#, JS | MVC | [System.ServiceModel.OperationContractAttribute(Action="https://cegep-heritage.qc.ca/Authenticate", ReplyAction="\*")]  System.Threading.Tasks.Task<LoginService.AuthenticateResponse> AuthenticateAsync(LoginService.AuthenticateRequest request);  } |  |

For the projects, you may run and deploy locally from Visual Studio, but be sure to use the csdev databases. You may have to change the connection strings.

Note: For each project:

1. Indicate the DevOps URL
2. You must demonstrate to the professor that you’ve been able to download, compile and view the code from the DevOps server in your Visual Studio environment.
3. Analyze the code to complete the tables above.

# Part A Section 3 – Authorization and Authentication

## AMS – Part 1

Our goal is to understand AMS from a **black box** point of view at this point.

We will learn the details how to integrate it and how the code works later on.

Your prof will give a brief walk through and demo.

Your other source of information is :

1. The Runbooks of the legacy projects will tell you how to configure the application
2. There’s help built into AMS

[http://cstest.cegep-heritage.qc.ca/AMS](http://cstest.cegep-heritage.qc.ca/AMS_Sandbox)

login with username userad, password cs@123test!

**All student answers should be in blue text so it is easily identifiable by the instructor.**

Read and understand AMS-LoginService.docx, found in Moodle for this course under reference.

There is an error in the Use Case diagram of AMS-LoginService.docx, what is the error:

Answer:

|  |
| --- |
| The error is that the use case diagram does not clearly show the interactions between the admin and the manage application roles use case |

Explain: What is the purpose of AMS? What benefit does it provide?

|  |
| --- |
| The purpose of AMS is to provide centralized login and user role management services for applications found inside the Heritage network. |

Explain: How is AMS integrated into each legacy Heritage CS application?

|  |
| --- |
| AMS is integrated by connecting them to the login service. Here, authentication and authorization occur in order to control access in the applications. |

Explain: What are the two services provided by AMS to each Heritage CS application?

|  |
| --- |
| Authentication: Checks username and password against Heritage College Active Directory |

Explain: Application, Roles and Users as modeled in AMS. How does it relate to Application codes and Role codes.

|  |
| --- |
| Application: Software that users interact with  Roles: Define permissions and access users have with each application  Users: has different permission depending on their roles within an application  Applications have unique codes, and the roles within them also have unique codes. They are used by AMS to ensure users are properly authorized based on their role for an application |

Explain: How do the applications ensure that users are constrained to the Role configured on them within AMS?

|  |
| --- |
| Applications ensure users are constrained to the role configured to them by the AMS authorize method. It checks the user’s role against applications roles. If roles do not match, the application will not allow access. |

## AMS – Part 2

**All student answers should be in blue text so it is easily identifiable by the instructor.**

Assume your team creates an application called “BeastApp”. The key users of the system are Teachers, Coordinators, Students and a new role “Beast Master”. Ron Patterson is the one and only Beast Master.

Explain: What steps would you take in AMS to configure BeastApp and ensure the correct types of users are configured?

|  |
| --- |
| 1. Add "BeastApp" to AMS by creating a new application with a unique application code. 2. Define Roles for BeastApp: Teacher, Coordinator, Student, Beast Master. 3. Give users correct roles, give Ron Patterson Beast master |

# Part A Section 4 – Technology Decisions

## .NET Framework vs .NET Core

Some of the legacy projects were implemented in .NET Framework, but MS has introduced .NET Core in recent years.

Investigate: Pros vs Cons of .NET Framework vs .NET Core

Create table below:

|  |  |  |
| --- | --- | --- |
| .NET Framework | Pros | Cons |
|  | Full support for legacy apps | Windows-only |
|  | Large number of libraries and tools | Slower |
|  | Best for legacy systems and Windows-only environments | Less optimized for microservices |
|  |  | Limited to IIS for web hosting |
|  |  |  |
| .NET Core | Cross-platform | Limited backward compatibility with .NET Framework |
|  | Faster |  |
|  | Optimized for Docker and Kubernetes |  |
|  | Fully open source |  |
|  | Best for new development, cross-platform, and cloud apps |  |

Propose: Which would you prefer to use for a new project and why?

|  |
| --- |
| I would prefer to use .NET Core because it is best tailored for modern applications. With its lightweight and modular libraries and tools, as well as it being open source, it would allow us to deploy the app on multiple Operating Systems, making it our best option for scalability. |

## Webforms vs MVC

Investigate: Pros vs Cons of WebForms vs MVC

Create table below:

|  |  |  |
| --- | --- | --- |
| Webforms | Pros | Cons |
|  | Easier for beginners | Poor separation logic |
|  | Automatic state management with ViewState | Slower |
|  |  | Difficult to test, tightly coupled with UI |
|  |  |  |
| MVC | SRP | Steeper learning curve |
|  | Faster | Manual state management |
|  | Full control over HTML, CSS, and JS  Highly testable |  |
|  | Flexible, reusable components |  |
|  | Good SEO friendliness |  |

Propose: Which would you prefer to use for a new project and why?

|  |
| --- |
| I would prefer to use MVC because it provides better control over the app, SRP is integrated., performance is better, and it’s a lot easier to test. |

## Reflect

Return to your work done for Assignment 0

What were your teams’ top and most important answers for “What went well?”

|  |
| --- |
| Communication, design, and UI were our team’s top answers. |

What were your teams’ top answers for “What didn’t go well?”

|  |
| --- |
| Time management seemed to be an issue amongst our time, as well as working with Azure. |

What were your teams’ top answers for “What questions do you still have?”

|  |
| --- |
| How could we have better estimated our time? |

Summarize the key points from your meeting minutes document, here (K40\_A0\_Minutes.docx)

|  |
| --- |
| Our what didn’t go well was larger than our what went well, as it was our first time working in groups, we ran into issues with version control and time management. Because of that, we learnt a lot of lessons that correlate to the issues we ran into. We also didn’t have many things that left us puzzled, as most of our issues were not technology related. The decisions we made corresponded to improvements we can make in the future to ensure our work is done well and on time. The actions we took and our next steps were preparation for our Dev Project, including Azure permissions and time estimation techniques. |

Refer to the lecture material, in class discussions, and WooFlash quizzes for the following answers:

|  |
| --- |
| What is a project to you?  A goal with a plan that is worked towards with a group of people  What is a project, according to Prof Chris?  A software project consists of a group of people who work together, with a plan that adheres to a fixed end goal.  What is the total current rate of project failure in Software Projects, according to Historical Trends?  19%  What is the main advantage of Agile project management in terms of outcomes?  Predictable outcomes and early issue detections  What are the phases of the Software Development Lifecycle? *(Hint: there are 6)*  Requirements, Design, Development, Testing, Deployment, Maintenance  What is a key client protocol in K40 and general projects regarding communication?  Communicate clearly and regularly  What is true about AI tools, in this program and course?  Cannot copy from students or websites, including AI tools, all work must be represented as yours, infractions will result in Academy Integrity conversations and escalation, your profs in each course will guide you more on AI tool usage.  What were Chris’ “Protips” in the Prezi presentation that you reviewed for homework?  Network, focus, reach out, get better at problem solving, take a breather |

**To submit**

Upload this completed document, ***UName\_*K40\_A1A.docx,** to Moodle with the answers filled in. i.e. celliott\_K40\_A1A.docx. *Submit this one day before the deadline, if possible, to avoid technical or user errors.*

# Part B Section 1 – Dependency Injection

On Moodle there is a sample MVC solution “HeritageCollege .NET 6 Target.zip”.

See [this tutorial](https://www.youtube.com/watch?v=9J9a77ga9R0) for samples and hints on how to implement dependency injection.

Load the solution “HeritageCollege”

Change the Student Controller to take in a IStudentRepository as a constructor.

Write out here the steps you had to do and explain each line (why are you doing this):

1. A screen shot of a computer program

   Description automatically generated
   1. Stated the reference to IStudentRepository so it is expecting the interface and not the service.
   2. Added the parameter in the constructor so it is the default, and sets the \_myRepo variable;
2. A screen shot of a computer program

   Description automatically generated  
   1. Made the interface public because it had an access modifier of internal, since the constructor is public, the parameters must also be.
3. A screen shot of a computer program

   Description automatically generated
   1. Registered the IStudentRepository and Student repository so that it is implemented when the controller is created.

Explain, what is the benefit of using Dependency Injection on the controller?

DI on the controller simplifies testing and promotes SRP and loose coupling.

Explain the different between the Singleton, Transient and the Scoped service lifetimes.

Singleton: Created once and is implemented for the entire length of the application lifetime.

Transient: New instance is created every time it is requested

Scoped: Created once per request or scope

You just created dependency injection on construction, what are all the types of dependency injection? *Hint: Read* [*this*](https://www.tutorialsteacher.com/ioc/dependency-injection)*.*

Constructor injection

Property injection

Method injection

Show an example of each highlighting the precise line that the dependency injection is happening.

CONSTRUCTOR

*public class CustomerService*

*{*

*CustomerBusinessLogic \_customerBL;*

*public CustomerService()*

*{*

*\_customerBL = new CustomerBusinessLogic(new CustomerDataAccess());*

*}*

*public string GetCustomerName(int id) {*

*return \_customerBL.ProcessCustomerData(id);*

*}*

*}*

PROPERTY

public class CustomerBusinessLogic

{

public CustomerBusinessLogic()

{

}

public string GetCustomerName(int id)

{

return DataAccess.GetCustomerName(id);

}

public ICustomerDataAccess DataAccess { get; set; }

}

public class CustomerService

{

CustomerBusinessLogic \_customerBL;

public CustomerService()

{

\_customerBL = new CustomerBusinessLogic();

\_customerBL.DataAccess = new CustomerDataAccess();

}

public string GetCustomerName(int id) {

return \_customerBL.GetCustomerName(id);

}

}

METHOD

interface IDataAccessDependency

{

void SetDependency(ICustomerDataAccess customerDataAccess);

}

public class CustomerBusinessLogic : IDataAccessDependency

{

ICustomerDataAccess \_dataAccess;

public CustomerBusinessLogic()

{

}

public string GetCustomerName(int id)

{

return \_dataAccess.GetCustomerName(id);

}

public void SetDependency(ICustomerDataAccess customerDataAccess)

{

\_dataAccess = customerDataAccess;

}

}

public class CustomerService

{

CustomerBusinessLogic \_customerBL;

public CustomerService()

{

\_customerBL = new CustomerBusinessLogic();

((IDataAccessDependency)\_customerBL).SetDependency(new CustomerDataAccess());

}

public string GetCustomerName(int id) {

return \_customerBL.GetCustomerName(id);

}

}

Further reading: [link](https://docs.microsoft.com/en-us/aspnet/core/fundamentals/dependency-injection?view=aspnetcore-3.1)

Part B – Testing with Mock objects

This section continues with the HeritageCollege solution used in Part A.

Add->New Project->Unit Test Project, “HeritageCollegeTest”

In HeritageCollegeTest, select Dependencies, right click, Add Project Reference , under projects/solution select HeritageCollege.

Add a test for the StudentController

Rename UnitTest1 to StudentControllerTest

Add the HeritageCollege namespace

Add a test to just test the Index method:

[TestMethod]

public void TestIndexStudentRepo()

{

const int OUTPUT\_OVERHEAD = 2; // output has 2 extra newlines

// Arrange

StudentRepository testRepo = new StudentRepository();

StudentController testController = new StudentController(testRepo);

// Act

string testOutput = testController.Index();

// Assert

// count lines of output

Assert.AreEqual(2+ OUTPUT\_OVERHEAD, testOutput.Split('\n').Length);

// is Jean in there?

Assert.IsTrue(testOutput.Contains("Jean"));

Assert.IsTrue(testOutput.Contains("Richard"));

Assert.IsFalse(testOutput.Contains("Jose"));

}

1. Show me a snippet of the test running successfully (screen capture of the test run results)

*A screenshot of a computer

Description automatically generated*

Test driven development: Write a test that fails first, then add code until it works.

Add a test to test the StudentController.GetStudent() (hint: it doesn’t exist yet).

[TestMethod]

public void TestStudentGet()

{

// Arrange

StudentRepository testRepo = new StudentRepository();

StudentController testController = new StudentController(testRepo);

// Act

Student testStudent1 = testController.GetStudent(1);

Student testStudent3 = testController.GetStudent(3);

// Assert

Assert.IsNotNull(testStudent1);

Assert.IsNull(testStudent3);

}

Pass 1: throw the not implemented exception, test fails

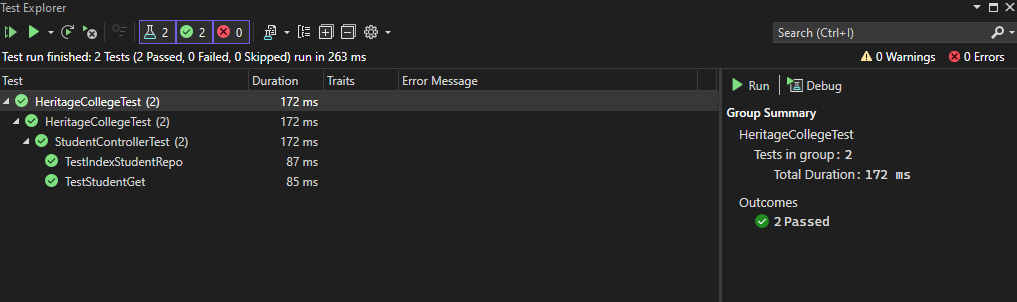
1. Show me the output of the fail on the test run

*A screenshot of a computer

Description automatically generated*

Pass 2: Implement the code so that the test passes

1. Show me the output of the pass. Show me the code snippet of the updates in the Controller, IStudentRepository, and StudentRepository classes.

**

public Student GetStudent(int v)

{

return \_myRepo.GetStudent(v);

}

Build your own Mock.

Create your own new StudentRepository class, call it MyStudentRepository that implements the IStudentRepository, but under the hood, it only has 1 student in it (your name and studentId).

Create a new Unit test that uses MyStudentRepository, modelled like TestStudentGet() that validates the existence of studentId (yours) and one that doesn’t exist.

1. Show the code for MyStudentRepository.

*A computer screen shot of a program code

Description automatically generated*

1. Show the your test code

[Fact]

public void TestMyStudentRepostory\_GetStudent()

{

// Arrange

MyStudentRepository testRepo = new MyStudentRepository();

StudentController testController = new StudentController(testRepo);

// Act

Student testStudent1 = testController.GetStudent(1);

Student testStudent3 = testController.GetStudent(3);

// Assert

Assert.NotNull(testStudent1);

Assert.Null(testStudent3);

}

*A screenshot of a computer

Description automatically generated*

Congratulations, you manually created your own mock repository

*(let’s pretend the original StudentRepository had fancy Entity Framework internals).*

Now let’s see Moq in action.

Add the following unit test and ensure it runs. You may have to import/update the Mock package, for me, it was clicking through a wizard suggestion to fix.

// mocking the repository now

[TestMethod]

public void TestStudentGetWithMoq()

{

// Arrange

var testRepo = new Mock<IStudentRepository>();

testRepo.Setup(repo => repo.GetStudent(1)).Returns(new Student() { ID = 1 });

StudentController testController = new StudentController(testRepo.Object);

// Act

Student testStudent1 = testController.GetStudent(1);

Student testStudent3 = testController.GetStudent(3);

// Assert

Assert.IsNotNull(testStudent1);

Assert.IsNull(testStudent3);

}

1. Show me the output of the test run. Everything should pass

A screenshot of a computer

Description automatically generated

Your turn, create another test method that has a mocked repository with one entry (yours).

Validate you can get your studentId and the first name matches what you expect it to be.

1. Show me the output of the pass. Show me the code snippet of the updates in the new test method

A screenshot of a computer

Description automatically generated

// mocking the repository now

[Fact]

public void TestMyStudentRepository\_GetWithMoq()

{

// Arrange

var testRepo = new Mock<IStudentRepository>();

testRepo.Setup(repo => repo.GetStudent(1)).Returns(new Student() { ID = 1, FirstName = "Sebastian", LastName = "Canales Burke" });

StudentController testController = new StudentController(testRepo.Object);

// Act

Student result = testController.GetStudent(1);

// Assert

Assert.NotNull(result);

Assert.Equal("Sebastian", result.FirstName);

}

Further reading: [link](https://docs.microsoft.com/en-us/aspnet/core/mvc/controllers/testing?view=aspnetcore-3.1)

**To submit for Part B:**

Upload this completed document, ***UName\_*K40\_A1B.docx,** to Moodle with the answers filled in.

* Ensure all of your student answers are in blue.
* Ensure you are submitting the right file.
* Download the file you uploaded to Moodle, check it for accuracy and completion. You are responsible for the accuracy and completeness of this file.
* Ensure you submit well in advance of the deadline to avoid any user or technical errors.

## Part C Section 1 - User Meeting

Multiple clients will present their proposals for a project. We will also be calling these individuals our “users”. They are our most important stakeholder.

This User Meeting will take place during our lab session times. In the scenario that this cannot be accomplished, we will meet with our clients/users outside of class hours in a reasonable time that all students are available.

You are to capture minutes and analyze what was said (captured in a memo) and submit this before the above deadline.

1. Take notes on the information gathered during the meeting with each user. In most cases, you’ll have one user/client In particular, focus on problems, user requirements, and priorities. Remember: we are problem solvers!
2. Transcribe the notes (minutes of the meeting) into a Word document which indicates the day of your meetings (i.e. named **YourUserName\_K40\_A01C\_**Meeting) and containing the following information for each meeting:

* An appropriate title
* Date and time (including both the start time and end time) of the meeting
* Place
* Attendees (list the full names of all the people who were in attendance in alphabetical order). Indicate who was the scribe (write the word “Scribe” after your name)
* Invited but did not attend (alphabetical list of full names)
* Subject (topic of the meeting)
* Items Discussed: The notes should include all the details of the user’s problems, requirements, and priorities based on what was discussed, and it should be written in a way that can be used as a reference by someone who was not in attendance. The notes can be in bullet format but should be written in full sentences.
* Decisions
* Action items – must have a description, owner, status (opened/closed) and due date
* Follow up and next steps

Tips for recording notes from a meeting:

* Don’t try to record notes word for word
* Organize your notes and write the minutes as soon after the meeting as possible while everything is fresh in your mind
* Add additional comments, or clarify what you didn’t understand right after the meeting
* Write in the same tense throughout
* Proof-read the document

## Part C Section 2 – Elevator Pitch and User Stories

**All student answers should be in blue text so it is easily identifiable by the instructor.**

Write a business memo addressed to the professor/project manager.

For the project discussed in the User Meeting, analyze and:

1. State the elevator pitch (same format as Inception Deck) for the project as you see it.
2. Identify all the roles/actors/stakeholders with a brief description
3. Provide User Stories for each type of system actor

Notes:

* Since this is an initial list, the user stories should not be detailed, as many of them will be further broken down into smaller user stories as the chosen project(s) progresses. Do not include any technical details.
* The standard user story format in Heritage CS Projects is the following:

*As a <type of system actor/user>, I want <some goal> so that <some reason>.*

Remember that <some reason> is from the point of view of that system actor, not you, the designer/developer.

**To submit**

When you have completed the assignment, upload a zip file **YourUserName\_K40\_A01C\_Px.zip** to Moodle with the following contents (See Moodle for precise submission deadline.). P*x* where {x = 1,2,3,4,5} the project number you’ve been assigned.

1. ***YourUserName*\_K40\_A01C\_P*x*\_MeetingMinutes.docx** meeting minutes.
2. ***YourUserName*\_K40\_A01C\_P*x*\_UserStories.docx** for user stories.

# Appendix 1: Connecting your DevOps instance to Visual Studio

**See the following up-to-date documentation**

[Link](https://docs.microsoft.com/en-us/azure/devops/organizations/projects/connect-to-projects?view=azure-devops&tabs=visual-studio-2019)

### Visual Studio 2019

1. Select the **Manage Connections** button in Team Explorer to open the **Connect** page. Choose **Connect to a Project** to select a project to connect to.

**Connect to a Project** shows the projects you can connect to, along with the repos in those projects.

1. Select **Add Azure DevOps Server** to connect to a project in Azure DevOps Services. Enter the URL to your server and select **Add**.
2. Select a project from the list and select **Connect**.