GRAND CANYON UNIVERSITY™

Agile and .NET Foundation Activity #1

Part 1 Tools Installation and Default App

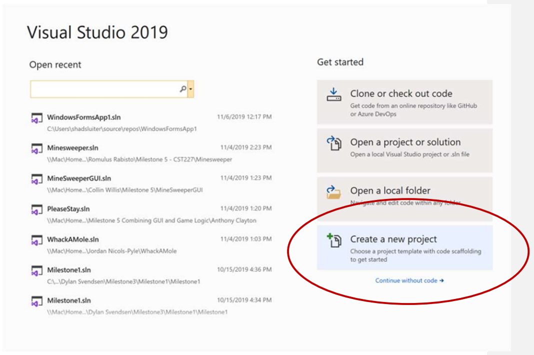
Goal and Directions:

In this activity, you will create a default .NET MVC application in Visual Studio to validate your environment and also become familiar with the .NET MVC project structure. Complete the following tasks for this activity:

Installation

1. Make sure your Visual Studio environment is installed properly:
    a. Download and install the latest version of Microsoft Visual Studio Community Edition
    b. Run the installer. Select Custom and install the Microsoft Web Developer Tools and the Microsoft SQL Server Data Tools.

The pictures here are from Visual Studio 2019.



Visual Studio 2019

Open recent

WindowsFormsApp1.sln C:\Users\shadsluiter\source\repos\WindowsFormsApp1	11/6/2019 12:17 PM
Minesweeper.sln \\Mac\Home...\Romulus Rabisto\Milestone 5 - CST227\Minesweeper	11/4/2019 2:23 PM
MineSweeperGUI.sln \\Mac\Home...\Collin Willis\Milestone 5\MineSweeperGUI	11/4/2019 1:23 PM
PleaseStay.sln \\Mac\Home...\Milestone 5 Combining GUI and Game Logic\Anthony Clayton	11/4/2019 1:20 PM
WhackAMole.sln \\Mac\Home...\Jordan Nicols-Pyle\WhackAMole	11/4/2019 1:03 PM
Milestone1.sln C:\...\Dylan Svendsen\Milestone3\Milestone1\Milestone1	10/15/2019 4:36 PM
Milestone1.sln \\Mac\Home...\Dylan Svendsen\Milestone3\Milestone1\Milestone1	10/15/2019 4:34 PM

Get started

Clone or check out code
Get code from an online repository like GitHub or Azure DevOps

Open a project or solution
Open a local Visual Studio project or .sln file

Open a local folder
Navigate and edit code within any folder

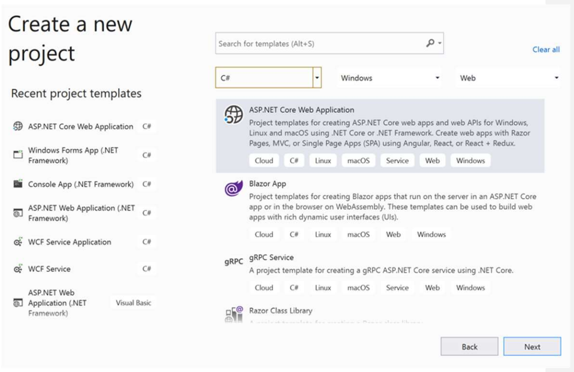
Create a new project
Choose a project template with code scaffolding to get started

Continue without code →

The Default App

2. Create a default .NET MVC application
    a. Click File → New Project or click the "Create New Project Button" on the Getting Started screen.

b. Select ASP.NET **Core Web Application** for the type of application you wish to start.



Create a new project

Search for templates (Alt+S)

Clear all

C# | Windows | Web

Recent project templates

- ASP.NET Core Web Application — C#
- Windows Forms App (.NET Framework) — C#
- Console App (.NET Framework) — C#
- ASP.NET Web Application (.NET Framework) — C#
- WCF Service Application — C#
- WCF Service — C#
- ASP.NET Web Application (.NET Framework) — Visual Basic

ASP.NET Core Web Application
Project templates for creating ASP.NET Core web apps and web APIs for Windows, Linux and macOS using .NET Core or .NET Framework. Create web apps with Razor Pages, MVC, or Single Page Apps (SPA) using Angular, React, or React + Redux.
Cloud C# Linux macOS Service Web Windows

Blazor App
Project templates for creating Blazor apps that run on the server in an ASP.NET Core app or in the browser on WebAssembly. These templates can be used to build web apps with rich dynamic user interfaces (UIs).
Cloud C# Linux macOS Web Windows

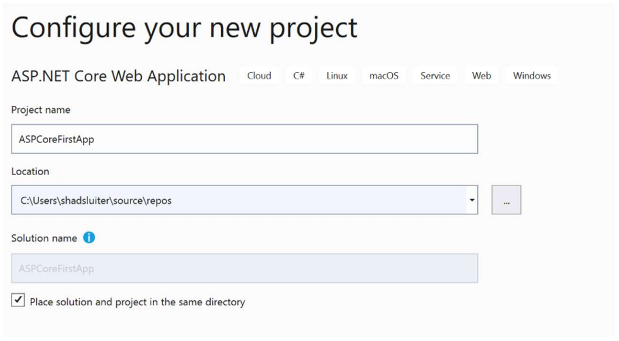
gRPC Service
A project template for creating a gRPC ASP.NET Core service using .NET Core.
Cloud C# Linux macOS Service Web Windows

Razor Class Library

c. Enter the project name, **ASPCoreFirstApp,** or something similar, in the Name field.

d. Choose a folder on your hard drive to save the project in or leave it as the default.

e. Click **Create** to continue.



Configure your new project

ASP.NET Core Web Application — Cloud C# Linux macOS Service Web Windows

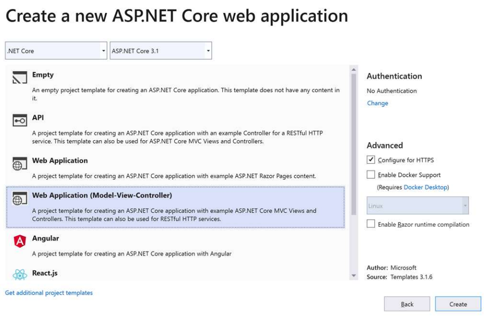
Project name: ASPCoreFirstApp

Location: C:\Users\shadsluiter\source\repos

Solution name: ASPCoreFirstApp

[✓] Place solution and project in the same directory

You will see the following dialog, which asks you to set the initial content for the ASP.NET project.



Create a new ASP.NET Core web application

.NET Core | ASP.NET Core 3.1

Empty
An empty project template for creating an ASP.NET Core application. This template does not have any content in it.

API
A project template for creating an ASP.NET Core application with an example Controller for a RESTful HTTP service. This template can also be used for ASP.NET Core MVC Views and Controllers.

Web Application
A project template for creating an ASP.NET Core application with example ASP.NET Razor Pages content.

Web Application (Model-View-Controller)
A project template for creating an ASP.NET Core application with example ASP.NET Core MVC Views and Controllers. This template can also be used for RESTful HTTP services.

Angular
A project template for creating an ASP.NET Core application with Angular

React.js

Get additional project templates

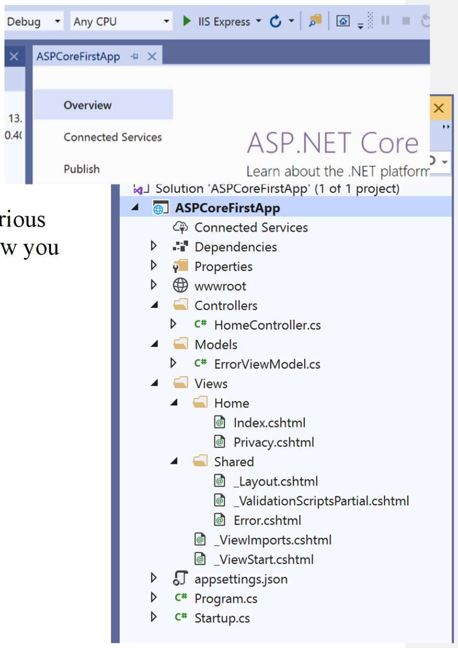
Authentication
No Authentication
Change

Advanced
[✓] Configure for HTTPS
[ ] Enable Docker Support
(Requires Docker Desktop)
Linux
[ ] Enable Razor runtime compilation

Author: Microsoft
Source: Templates 3.1.6

f. Select the **Web Application (Model-View-Controller)** template. This will create a basic MVC project with default predefined content.
g. Leave "No Authentication" selected.
h. Configure for HTTPS.
i. No Docker Support.
j. No "Enable Razor Runtime Compilation."
k. Click Create.

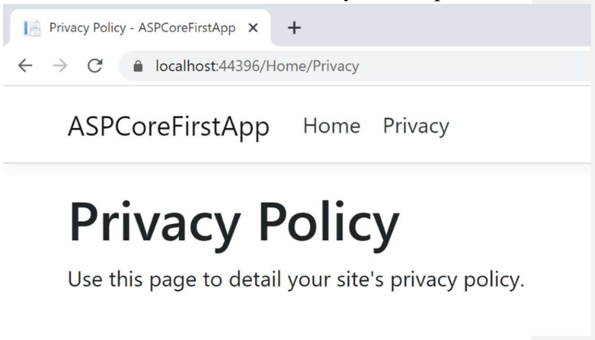
Once the project is created by Visual Studio, you will see a number of files and folders displayed in the Solution Explorer window. Expand each of the folders shown here in the Solution Explorer. Feel free to open any and all of the files to familiarize yourself with the various parts of the application. The tutorials that follow will show you how to configure and expand on each of these files.



```
Solution 'ASPCoreFirstApp' (1 of 1 project)
  ASPCoreFirstApp
    Connected Services
    Dependencies
    Properties
    wwwroot
    Controllers
      HomeController.cs
    Models
      ErrorViewModel.cs
    Views
      Home
        Index.cshtml
        Privacy.cshtml
      Shared
        _Layout.cshtml
        _ValidationScriptsPartial.cshtml
        Error.cshtml
      _ViewImports.cshtml
      _ViewStart.cshtml
    appsettings.json
    Program.cs
    Startup.cs
```

Run the Application

3. Run the application to ensure your environment is working properly. Either click the green start arrow at the top of the Visual Studio page or choose **Debug > Start Debugging** from the application menu. This will compile the code, start the IIS (Internet Information Server) application, and launch the default web browser on your computer.

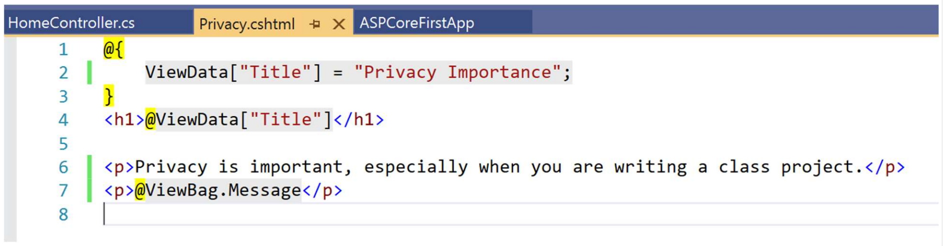

Privacy Policy - ASPCoreFirstApp
localhost:44396/Home/Privacy
ASPCoreFirstApp
Home
Privacy

Privacy Policy

Use this page to detail your site's privacy policy.

4. Click each of the three items on the navigation bar – Home and Privacy. You should notice that each of these pages are associated with a cshtml file in the Solution Explorer inside the View > Home folder.

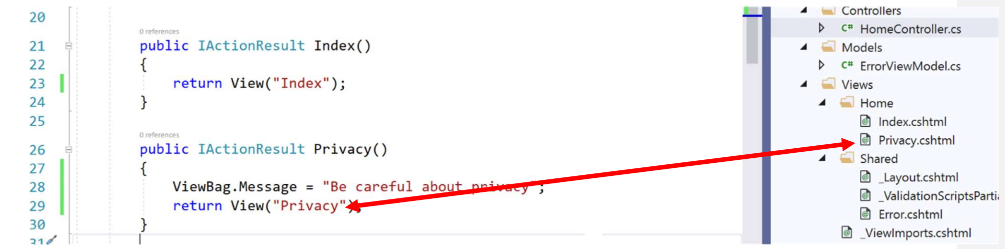
5. Modify the HTML and text of either of the View pages. I added a line with the ViewBag variable. ViewBag is an object in which you can send data from a controller to a view. It is one of several methods for sharing data to a page.
6. Capture a screenshot of the app at this stage. Put the image into a Microsoft Word document with a caption explaining what you have just demonstrated.



```
@{
    ViewData["Title"] = "Privacy Importance";
}
<h1>@ViewData["Title"]</h1>

<p>Privacy is important, especially when you are writing a class project.</p>
<p>@ViewBag.Message</p>

```

Specify the View name

It is possible to specify the name of the view being returned to the browser in a controller method. In the example below, the names "Index" and "Privacy" were specified as a parameter in the View() method. Since the name of the method matches the name of the view, the name can be omitted.



```
        0 references
        public IActionResult Index()
        {
            return View("Index");
        }

        0 references
        public IActionResult Privacy()
        {
            ViewBag.Message = "Be careful about privacy";
            return View("Privacy");
        }
```

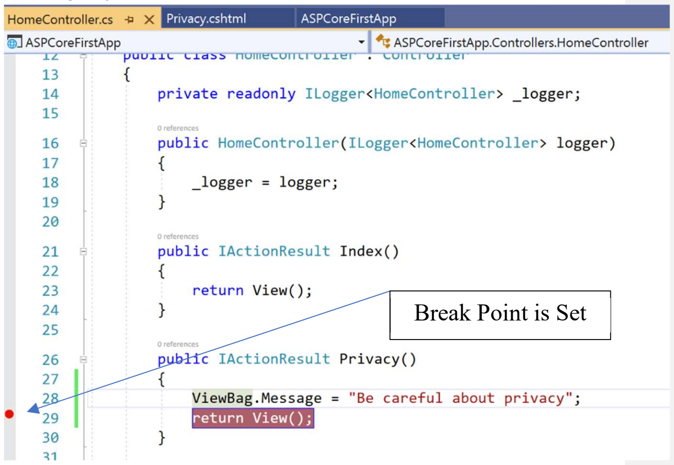
Debugging Breakpoints

Use breakpoints and the debugger to validate the code paths that handle these menu items.

1. Set a breakpoint in the HomeController.cs file on line 22, inside the Privacy() method. Click on the far-left margin and a new breakpoint stop sign will appear.
2. Add a string value to the ViewBag object.
3. Run the program.

The debugger stops the application when we click the Privacy link on the webpage. That is because the HomeController file runs when any link on the website is clicked.

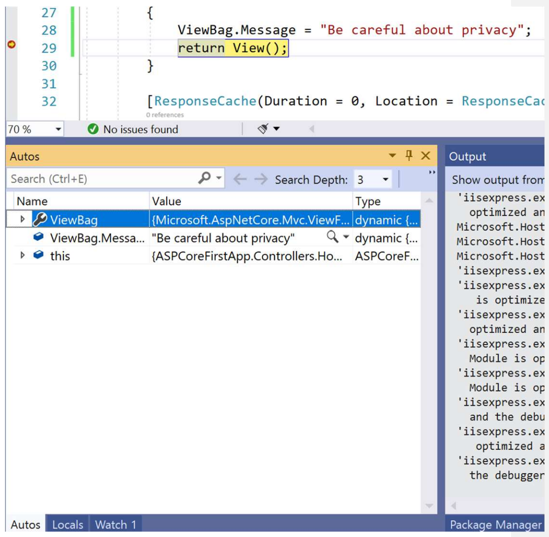
You can move forward through the code execution with the "Step Over" button on the top menu bar of Visual Studio.



```
    public class HomeController : Controller
    {
        private readonly ILogger<HomeController> _logger;

        0 references
        public HomeController(ILogger<HomeController> logger)
        {
            _logger = logger;
        }

        0 references
        public IActionResult Index()
        {
            return View();
        }

        0 references
        public IActionResult Privacy()
        {
            ViewBag.Message = "Be careful about privacy";
            return View();
        }
```

At the bottom of the debugger screen, you can see the value of all variables. In this case, you can see that the variable called ViewBag.Message is a string and it is set to “Be careful about privacy.”

Debugging is a very useful tool in the rare situation when your code doesn’t execute as you expected it would.



```
        {
            ViewBag.Message = "Be careful about privacy";
            return View();
        }

        [ResponseCache(Duration = 0, Location = ResponseCac
```

70 % No issues found

Autos

Search (Ctrl+E) Search Depth: 3

Name	Value	Type
ViewBag	{Microsoft.AspNetCore.Mvc.ViewF...	dynamic {...
ViewBag.Messa...	"Be careful about privacy"	dynamic {...
this	{ASPCoreFirstApp.Controllers.Ho...	ASPCoreF...

Output

Show output from

```
'iisexpress.ex
  optimized an
Microsoft.Host
Microsoft.Host
Microsoft.Host
'iisexpress.ex
'iisexpress.ex
   is optimize
'iisexpress.ex
  optimized an
'iisexpress.ex
  Module is op
'iisexpress.ex
  Module is op
'iisexpress.ex
  and the debu
'iisexpress.ex
   optimized a
'iisexpress.ex
  the debugger
```

4. Capture a screenshot of the app at this stage. Put the image into a Microsoft Word document with a caption explaining what you have just demonstrated.

Observations and Notables

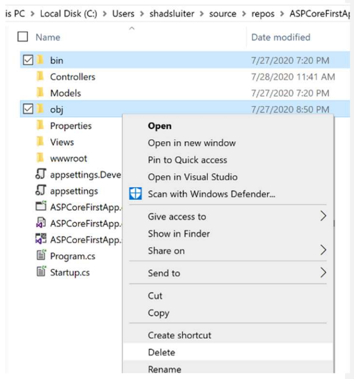
1. All **Controllers** are located in a folder called Controllers. By convention, the framework follows a Controller naming convention of **[Controller Name]Controller** and uses the a URI within the URL to resolve to the proper Controller. The method called within the Controller is also resolved via the URI with the method Index() used as a default for the root URI.
2. All **Views** are located in a folder called Views.
3. All **Models** are located in a folder called Models. In general, all Models are simple C# classes that contain nothing but a set of properties.
4. The file ‘_Layout.cshtml’ is located in the shared folder and is called by the framework as a default view prior to rendering all Views. This page often is used to reference a common/shared partial View that contains links to common JavaScript files (such as Bootstrap) and CSS files (to control a common application theme).
5. Within a partial View, the tag @RenderBody() is the used a placeholder where the desired View content is rendered.

Review the readings in this activity’s section of the syllabus for more information on the concepts presented in this lesson.

GRAND CANYON UNIVERSITY™

Deliverables:

1. This activity has multiple parts. Complete all parts before submitting.
2. Submit a Microsoft Word document with screenshots of the application being run. Show each screen of the output and put a caption under each picture explaining what is being demonstrated.
3. In the same document, in one paragraph, write a summary of the key concepts that were demonstrated in this lesson. Be sure to explain the key words introduced in this lesson.
4. Submit a ZIP file of the project file. In order to save space, you can delete the bin and the obj folders of the project. These folders contain the compiled version of the application and are automatically regenerated each time the build or run commands are executed.



is PC > Local Disk (C:) > Users > shadsluiter > source > repos > ASPCoreFirstA

Name	Date modified
bin	7/27/2020 7:20 PM
Controllers	7/28/2020 11:41 AM
Models	7/27/2020 7:20 PM
obj	7/27/2020 8:50 PM
Properties	
Views	
wwwroot	
appsettings.Deve	
appsettings	
ASPCoreFirstApp.	
ASPCoreFirstApp.	
ASPCoreFirstApp.	
Program.cs	
Startup.cs	

Part 2 How to Run a Sprint

Goals of this Lesson

This guide explains the background of the Agile method and shows how to use an Excel spreadsheet to track:

1) The project backlog of a product
2) The tasks to complete to reach intermediate goals
3) The time estimates for each task

Part A - Agile Background Information

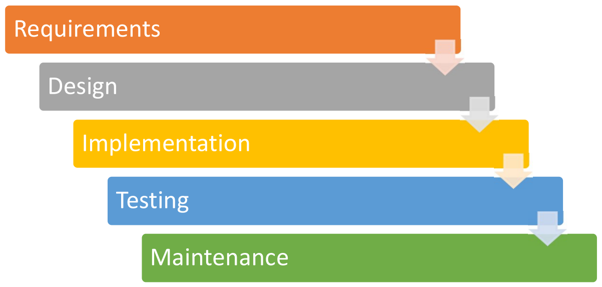
Software projects are notorious for being expensive and ineffective. Programmers don't understand what customers really want and customers themselves can't articulate what they really need. As a result, the end product is a waste of time and money

Waterfall vs. Agile

In software development, teams and organizations usually follow a variation of one of two popular design strategies, "waterfall" and "agile."

The Waterfall Design Model

The historically traditional Waterfall model follows a linear approach to designing and developing a product. The name "waterfall" comes from the cascading, sequential events that appear in the planning diagram. At first glance, this strategy seems to make perfect sense; a clear vision of the product is established, and work progresses in an efficient manner.



The Waterfall method might work well for a programmer who is writing his/her own program base. However, in the real world, programmers need to be paid by clients. Some weaknesses in the Waterfall approach have led to the rise of the Agile method. These weaknesses include:

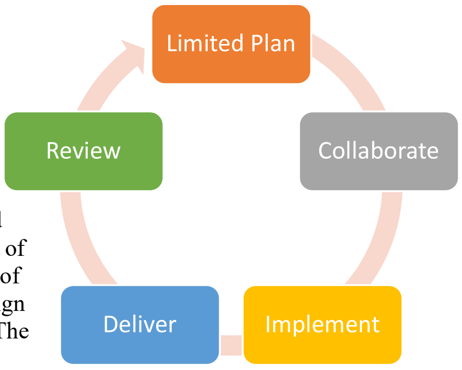
- Clients may not know exactly what their requirements are before they see working software, and so they may change their requirements after the product is finished.

- Designers may not be aware of future difficulties when designing new software. Only when the final product is delivered to the client are the difficulties apparent.

Commented [LM(1]: Sentence was incomplete verify that this wording is acceptable.

- Organizations may attempt to deal with a lack of concrete requirements from clients by employing systems analysts to examine existing manual systems for what they do and how they might be replaced. However, in practice, it is difficult to sustain a strict separation between systems analysis and programming. This is because implementing a system will expose issues and edge cases that the systems analyst did not consider.

The Agile Method



The problem with most project failures is with communication.   The “waterfall” may get what a client *said* they wanted at the beginning of the project, but it may not always get what they actually needed.

In the Agile design approach, the client is involved with the development of the product at every stage of the development. They get to see the complexities of the product, unforeseen consequences of early design decisions, and intermediate stages of the product. The more the customer can be involved in software creation, the more they take ownership of the software and, most importantly, the more the software becomes what they need.

Some principles of Agile development:

- **Collaborative** effort of self-organizing teams and their customers.
- **Adaptive** planning, evolutionary development, early delivery, and continual improvement.
- **Short term** milestone checkpoints to test limited feature sets. Regular acceptance checkpoints from the client.

Agile software development values

1) **Individuals and interactions** over processes and tools
2) **Working software** over comprehensive documentation
3) **Customer collaboration** over contract negotiation
4) **Responding to change** over following a plan

Twelve agile software development principles

1) Customer satisfaction by early and continuous delivery of valuable software.
2) Welcome changing requirements, even in late development.
3) Deliver working software frequently (weeks rather than months)
4) Close, daily cooperation between business people and developers
5) Projects are built around motivated individuals, who should be trusted
6) Face-to-face conversation is the best form of communication (co-location)
7) Working software is the primary measure of progress
8) Sustainable development, able to maintain a constant pace
9) Continuous attention to technical excellence and good design
10) Simplicity—the art of maximizing the amount of work not done—is essential
11) Best architectures, requirements, and designs emerge from self-organizing teams
12) Regularly, the team reflects on how to become more effective, and adjusts accordingly

Agile Terminology

The following terms are frequently used to describe the practice of agile software development:

Sprint – Short (two weeks is common) segment of work where a limited amount of product features is completed.

Incremental Development – A product is released with limited sets of features that can be implemented within the timeframe of a single sprint.

Minimum Viable Product (MVP) – A working version of the customer's vision of the product that can be completed using the minimal amount of effort and time. An MVP is used to test the customer's idea, refine the direction of the remaining development, and inform stakeholders, such as investors, about the development.

Product Backlog – A list of features, or User Stories, that are in the queue for current and future development.

Sprint Backlog – A subset of the product backlog. The list of features being addressed during the current sprint.

User Story – A program feature written as "As a <role>, I want to <describe action> so that I can <reason>." For example: "As an admin, I want to see a list of all customer accounts so that I can edit or delete accounts." A finished product will have dozens or even hundreds of User Stories that describe all of the features of the product.

Daily Scrum – A fifteen-minute daily team meeting to share progress, report impediments, and make commitments. During the daily scrum each team member answers three questions:

1. "What have I done since the last Scrum meeting? (i.e., yesterday)"
2. "What will I do before the next Scrum meeting? (i.e., today)"
3. "What prevents me from performing my work as efficiently as possible?"

Scrum Master – A team member who ensures that the team is following the Agile principles, working on the appropriate backlog items, and following the schedule.

Burndown Chart – A single graph that shows the estimated hours of labor required to complete a sprint, as well as the actual amount of time spent on the task. The chart shows whether the team is on schedule or not.

Milestone Retrospective – A team meets to reflect on a sprint's successes and failures. Observations and suggestions are incorporated into ongoing sprints to improve the team's progress.

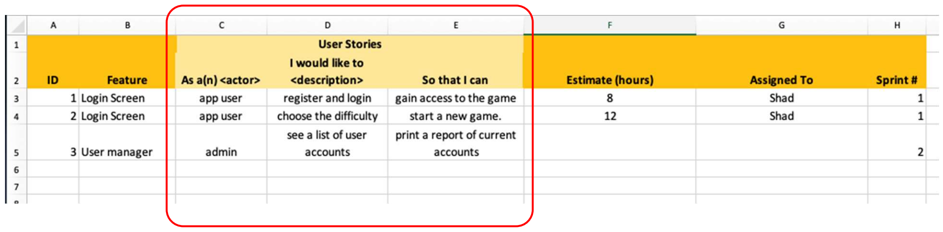
Part B – Create Documents for a Sprint

The following instructions show you how to utilize an Excel template to track the product backlog, sprint tasks, and burndown charts for each milestone of a project.

For this example, we will assume that our minimal application will have only a registration and login page.

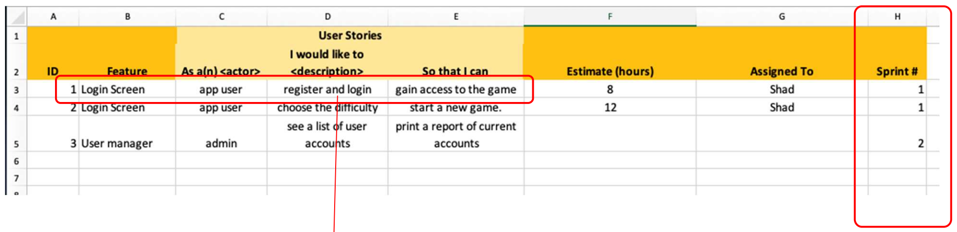
1. Using your favorite drawing tools, **create the followin**g design documents:
   a. **Wireframe** – What will the screens look like for the title, registration, and login failure and success pages?
   b. **Site Map** – What screens will be shown to the user and in what order?
   c. **Class UML** – What objects will the application need in order to handle a registration and login?
   d. **Database** – What tables will your app need in order to perform a registration and login?

2. On Day 1 of the sprint, write or update User Stories for the application. (Re)prioritize your User Stories in your Product Back Log (ideally this should be done for the next 3–4 sprints). Each feature of the application can be described as a "story" of (1) who, that will (2) do what, for (3) what reason.



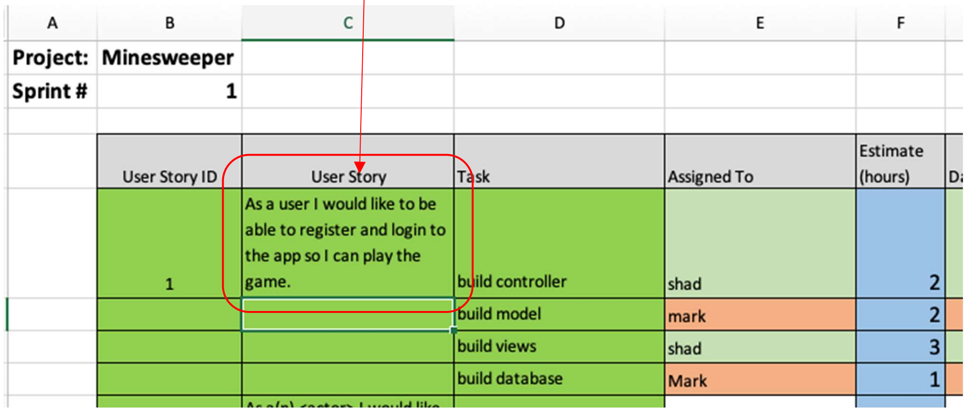
	A	B	C	D	E	F	G	H
1				User Stories				
2	ID	Feature	As a(n) <actor>	I would like to <description>	So that I can	Estimate (hours)	Assigned To	Sprint #
3	1	Login Screen	app user	register and login	gain access to the game	8	Shad	1
4	2	Login Screen	app user	choose the difficulty	start a new game.	12	Shad	1
5	3	User manager	admin	see a list of user accounts	print a report of current accounts			2
6								
7								

3. Identify the features (User Stories) that you will complete for this sprint.



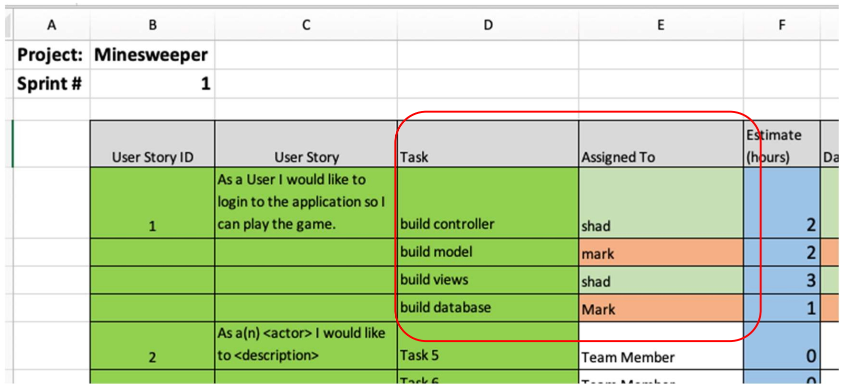
	A	B	C	D	E	F	G	H
1				User Stories				
2	ID	Feature	As a(n) <actor>	I would like to <description>	So that I can	Estimate (hours)	Assigned To	Sprint #
3	1	Login Screen	app user	register and login	gain access to the game	8	Shad	1
4	2	Login Screen	app user	choose the difficulty	start a new game.	12	Shad	1
5	3	User manager	admin	see a list of user accounts	print a report of current accounts			2
6								
7								

4. Copy each User Story to the burndown chart.



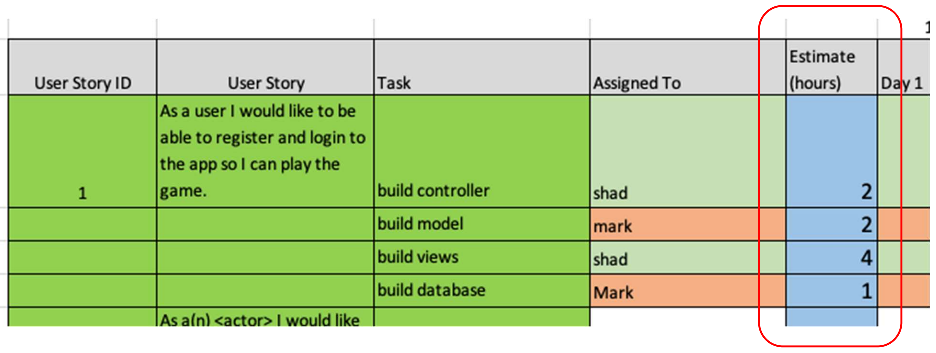
A	B	C	D	E	F	
Project:	Minesweeper					
Sprint #	1					
	User Story ID	User Story	Task	Assigned To	Estimate (hours)	Da
	1	As a user I would like to be able to register and login to the app so I can play the game.	build controller	shad	2	
			build model	mark	2	
			build views	shad	3	
			build database	Mark	1	
		As a(n) <actor> I would like				

5. Identify each task that will be done in order to complete the feature.
6. Assign the User Story to a single Team Member.



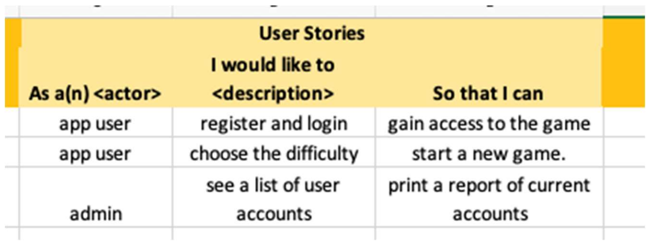
A	B	C	D	E	F
Project:	Minesweeper				
Sprint #	1				
	User Story ID	User Story	Task	Assigned To	Estimate (hours)
	1	As a User I would like to login to the application so I can play the game.	build controller	shad	2
			build model	mark	2
			build views	shad	3
			build database	Mark	1
	2	As a(n) <actor> I would like to <description>	Task 5	Team Member	0

7. Estimate the amount of work for each task (in 1, 2, 4 or 8 Effort Man Hours). If the task is larger than 8 hours, break the User Story into multiple User Stories.



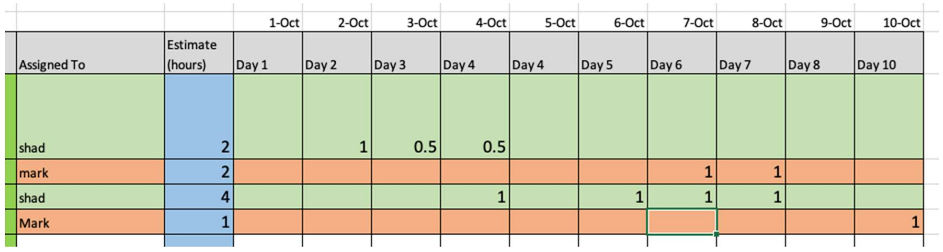
User Story ID	User Story	Task	Assigned To	Estimate (hours)	Day 1
1	As a user I would like to be able to register and login to the app so I can play the game.	build controller	shad	2	
		build model	mark	2	
		build views	shad	4	
		build database	Mark	1	
	As a(n) <actor> I would like				

8. Repeat the previous steps for each feature in the User Story list.



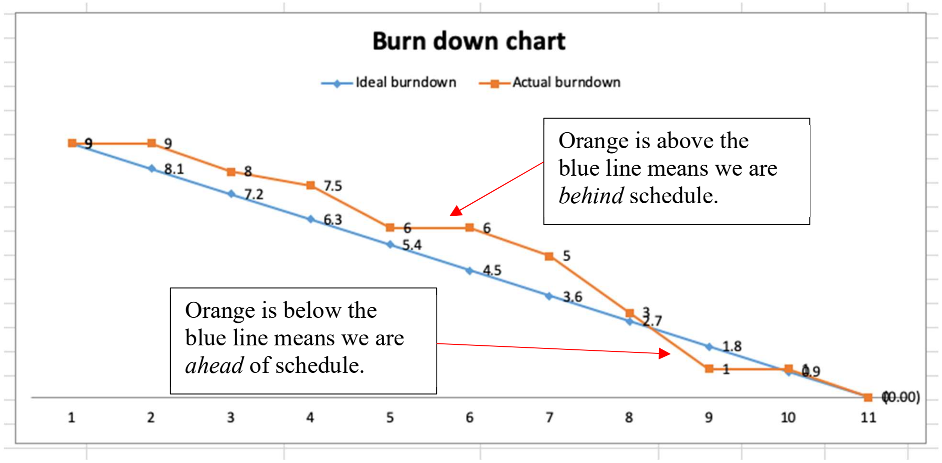
User Stories		
As a(n) <actor>	I would like to <description>	So that I can
app user	register and login	gain access to the game
app user	choose the difficulty	start a new game.
admin	see a list of user accounts	print a report of current accounts

9. At the end of each day, each team member should update the effort hours that were worked on their User Story to burn their work down. For this example, you can fill in the entire task list as if the sprint were already completed.



		1-Oct	2-Oct	3-Oct	4-Oct	5-Oct	6-Oct	7-Oct	8-Oct	9-Oct	10-Oct
Assigned To	Estimate (hours)	Day 1	Day 2	Day 3	Day 4	Day 4	Day 5	Day 6	Day 7	Day 8	Day 10
shad	2		1	0.5	0.5						
mark	2							1	1		
shad	4				1		1	1	1		
Mark	1										1

10. After each daily standup, each team member should review the burndown chart to ensure the sprint will be delivered on time. The steady downward slope of the blue line represents the goal you have set. The orange bumpy line shows the actual work accomplished. If the chart shows that the project is dangerously behind schedule, the team may have sufficient time to make adjustments such as obtaining more resources, resolving errors, and/or talking to the client about changing the goal or extending the deadline.



Deliverables

1. Microsoft Word document containing design drawings related to this sprint, including wireframes, site map, database tables, and class UMLs.
2. Excel document containing the product backlog, sprint task list, and burndown chart.