**Faculty of Engineering and Information Technology**

**School of Computer Science**

31927 – Application Development with .NET

**JULY 2024**

**ASSIGNMENT 2 – Code Design Report Template**

| **Due date** | Monday 11:00am, 19 August 2024 |
| --- | --- |
| **Demonstration** | Required in the lab/tutorial session |
| **Weight** | 35% |
| **Groupwork** | Group |
| **Submission** | Complete project folder zip |
| **Submit to** | Canvas |

**Summary**

This report template needs to be filled out by each group to receive marks for the “Code Design” section of Assignment 2. Please include as much justification & detail as you feel is necessary to receive full marks. For full marks screenshots are encouraged alongside code references. This **must be included in your zip file submission**.

| Student 1 Name & ID | Olivia Stewart 24848424 |
| --- | --- |
| Student 2 Name & ID | Oliver Warrick 24959254 |
| Student 3 Name & ID | Amber Cahill 25339906 |
| Lab Time | 7:30 |
| Lab Tutor | Davey |

Intro from each person:

# Features:

**Oliver:**

I wrote the Lobbying system and page, game sessions system and game play logic, game over page, question scaffold system, wrote all the questions, contributed significantly to the game page and code execution system and deployed the application to azure, including architectural refactors to facilitate this, I also did a large amount of work on linking together the disparate system components and fixing bugs in them.

**Olivia:**

I made a backend code runner, for actually testing the code people submit. I also made most of the db schema and db design and some of the automated deployment code. Furthermore, I did login/account and auth system. Also I wrote a question creator page. Much of the styling on all the pages was done by me. Most of the unit testing was done by me as well.

Finally, a lot of the initial project setup, (project structure etc was done by me)

**Amber:**  
I implemented syntax highlighting using GaelJ.BlazorCodeMirror6, including a settings modal which stores the user’s settings (theme, line wrapping, tab size, line numbers) in localStorage to persist between games and on the Stats page, live-typing using SignalR websockets, and the Statistics page showing win-rate and a paginated table of previous games.

**Justification**

For each criterion, include a justification as to how you have fulfilled it. Include references to your code & screenshots of how you applied the given code or database design principle.

| **Criteria** | At least 6 of the following need to be implemented across your group with each individual contributing towards at least 3:  • Polymorphism which achieves a useful purpose (Either through inheritance, interfaces or method/constructor overloading/overriding)  • File reading & writing to either JSON, XML, PDF or some other non plaintext formatted structure.  • Use of Entity Framework with the Repository & Unit of Work Patterns.  • NUnit tests on 4 different classes on all methods with mocks. (Minimum 2 methods per class).  • An event driven programming architecture with consumers & producers using delegates.  • Effective use of advanced data structures are used like Queues, Trees, Heaps etc.  • Generating & sending dynamic html emails with inline CSS.  • Effective use of different custom generic methods or classes.  • Nullable reference type checking is enabled project wide & nullable value types are used as appropriate. No nullable related warnings are present in your project.  • Effective use of extension methods or method chaining. |
| --- | --- |
| **Justification Olivia Stewart**  Include code, screenshots & any written documentation on how you met this criteria. | **FILE READING AND WRITING**  **In order to help making questions easier I made a page where you can create questions. This does two things with its api request, it will add a question to db. But it will also serialize a json file which will be used for seeding. This is so, even when recreating db, we can maintain these models**    **NUnit tests on 4 different classes on all methods with mocks. (Minimum 2 methods per class).**  I wrote a number of unit tests for the project.  They use mocks in order to effectively isolate functionality for unit testing,  **Polymorphism which achieves a useful purpose (Either through inheritance, interfaces or method/constructor overloading/overriding)**  Polymorphism is used extensively in the project. An example of this is in running code executions where the ISubmissionExecutor interface is used.This allows for potential of unique schemes of marking code submissions. For example as a base there is TestRunnerSubmissionExecutor which runs tests which are inherited by class instance execution which runs tests by creating an instance of the scaffolded class. |
| **Student 1 Recommended Mark**  Based on your justification, what mark do you believe you deserve. | 5/5 |
| **Justification**  **Oliver Warrick**  Include code, screenshots & any written documentation on how you met this criteria. | 1. Nullable type checking is enabled for the project so was thought of throughout development, I also did a cleanup towards the end of a few missed null warnings, one particular type of warning to note where it is acceptable to use the null forgiving operator (!) to ignore the warning is inclusion of an optional relation in a LINQ query used with entity framework, this is fine because it is an expression that is turned into sql that is done with a left join, so the actual query is null safe thus disabling the null warning is the recommended approach by entity framework, however when operating on the result of the query you must handle nulls as normal.    2. I used Entity framework and the unit of work pattern in multiple places, for example the data produced in a question execution is added in a single unit of work to prevent potentially incomplete data from being committed.  3. Polymorphism is used in many places, for example the database seeding process uses a polymorphic ISeedStep interface which allow for multiple implementations, for example creating question, admin users, or schema version metadata, which are all treated the same from the seed step executor.  I also created a number of extension methods such as Type.GetAliasedName and Extension.GetFullMessage and made use of event driven programming on the frontend with the use of lambdas and event delegates. |
| **Student 2 Recommended Mark**  Based on your justification, what mark do you believe you deserve. | 5/5 |
| **Justification (Student 3)**  Include code, screenshots & any written documentation on how you met this criteria. | 1. Advanced structures: live-typing messages are processed using a Queue structure to not have to realloc every time a message is taken off.      1. Nullable type checking is enabled, no nullable warnings in my files (Game.razor, GameHub.cs, Stats.razor, StatsController.cs)      1. Effective use of extension methods and method chaining |
| **Student 3 Recommended Mark**  Based on your justification, what mark do you believe you deserve. | 5/5 |

| **Criteria** | Across your group at least 2 of the following need to be implemented with each individual making a substantial contribution to at least 1:  • Use of either Blazor, ASP.NET, WPF or some other UI library instead of Windows Forms (Alternative UI libraries not listed must be confirmed with your tutor).  • Meaningful use of asynchronous coding with threads, tasks & jobs.  • Meaningful use of a REST or GraphQL API over the internet.  • Use of database joins & transactions in at least 6 different scenarios.  • Use of websockets to demonstrate realtime communication between multiple open instances of the program.  • Meaningful use of reflection & custom attributes or expression trees.  • Meaningful use of an Azure service, e.g. blob storage.  • Meaningful use of Machine Learning or AI tools.  • Use of a Mobile app framework, e.g. .NET MAUI or Xamarin.  • Use of other advanced topics agreed upon with the tutor. |
| --- | --- |
| **Justification Olivia Stewart**  Include code, screenshots & any written documentation on how you met this criteria. | The application is a client side blazor app. This uses bootstrap as a styling library.  The application also uses a REST api in order to communicate from client to server using ASP.net.  For example in one of the controllers    I also wrote the execution system which uses reflection to work.  It uses the compiled code submitted by user as an assembly which it reflects upon to create an instance of a scaffold class and invoke a method with test cases. |
| **Student 1 Recommended Mark**  Based on your justification, what mark do you believe you deserve. | 8/8 |
| **Justification Oliver Warrick**  Include code, screenshots & any written documentation on how you met this criteria. | I used asynchronous programming in many places in the project to ensure reduced thread pool usage of blocking operations and to allow for parallelism or cpu limited operations, there are also a few examples of more advanced concurrency. For example in the frontend a number of blazor components are rendered asynchronously and multiple of them may need to verify the currently logged in user, but we don’t always want to load the user and we don’t want to load the user multiple times. This is solved by using double checked locking and exposing a task for consumers to wait on.    I deployed the application to azure using a custom semi-autimated pipeline created with powershell, bash scripts and github actions, the services used are a B1 Linux VM, Azure SQL service instance and Api Application gateway, as well as supporting configuration such as a virtual network, resource group and public IP. Working out the configuration to maximise usage of the free options took some trial and error, the plan to deploy to the cloud also influenced the design of the application, for example when a user is in a game or lobby the page has a ServerId route parameter which would allow sticky routing of users to the same game server to allow for fast shared memory requests.    Additionally I worked significantly with realtime communications through writing the lobby ‘hub’ as well as contributing to game features, both of which made heavy use of websockets through SignalR. |
| **Student 2 Recommended Mark**  Based on your justification, what mark do you believe you deserve. | 8/8 |
| **Justification (Student 3)**  Include code, screenshots & any written documentation on how you met this criteria. | 1. We used the Blazor UI framework, in which I wrote most of Game.razor and all of Stats.razor 2. Use of asynchronous coding, with Timers triggering execution in Game.razor, and Stats.razor asynchronously loading paginated data from the controller. 3. Meaningful use of a REST API: we split our app into server-side controllers exposing a REST api, where I wrote StatsController.cs, and client-side Blazor pages, where in Stats.razor I made queries to that API. 4. Use of websockets to send live typing information using a SignalR Hub (GameHub.cs) and sending complex JSON-serialised objects from a SignalR HubConnection (Game.razor) |
| **Student 3 Recommended Mark**  Based on your justification, what mark do you believe you deserve. | 8/8 |

| **Criteria** | At least 6 different unique categories of UI elements have been used. e.g. buttons, headings, dropdowns, images, bar graphs, carousels, lists, context menus, modals etc. |
| --- | --- |
| **Justification Olivia View**  Include code, screenshots & any written documentation on how you met this criteria. | I mainly wrote login register view, question creator and question/test view  - Test view uses an accordion.  - Login has heading  - Login also has a spinner  - Login and register and question creator uses forms  - They all have heading etc  - They all have buttons  - Login and register have “alerts” in order to notify of status      Basically anything on those pages |
| **Student 1 Recommended Mark**  Based on your justification, what mark do you believe you deserve. | 4/4 |
| **Justification Oliver Warrick**  Include code, screenshots & any written documentation on how you met this criteria. | The UI elements which I used in the pages I worked with were:  Buttons, CheckBox, Select Dropdown, Text Input and Tables with the Lobby and Home pages which I created.  Accordion, Modal with the Game page which I contributed significantly towards and game over page which I created.  NavBars with general work on the layout and sign in. |
| **Student 2 Recommended Mark**  Based on your justification, what mark do you believe you deserve. | 4/4 |
| **Justification (Student 3)**  Include code, screenshots & any written documentation on how you met this criteria. | In my editor settings **modal**: **Heading**, **numerical input**, **drop-down**, **checkbox**, **button**    Also, in Stats.razor, a paginated **table** using **buttons** and **numerical input**. |
| **Student 3 Recommended Mark**  Based on your justification, what mark do you believe you deserve. | 4/4 |

| **Criteria** | Feature requires novel/unfamiliar technical environments that requires students to research and implement approaches they’ve not used before. |
| --- | --- |
| **Justification Olivia Stewart**  Include code, screenshots & any written documentation on how you met this criteria. | The most novel part of the project for me was the writing of the code execution system.  In order to run submissions and test them I had to learn about leveraging the Rosylyn api to compile the code into an assembly.    Then I had to write code to run test cases against this. |
| **Student 1 Recommended Mark**  Based on your justification, what mark do you believe you deserve. | 4/4 |
| **Justification Oliver Warrick**  Include code, screenshots & any written documentation on how you met this criteria. | During this project I encountered a number of technologies and techniques that were new to me. I had not used SignalR, Blazor or Azure before and leveraged all of them significantly in this project, I also learnt a lot about authentication and .NET reflection. |
| **Student 2 Recommended Mark**  Based on your justification, what mark do you believe you deserve. | 4/4 |
| **Justification (Student 3)**  Include code, screenshots & any written documentation on how you met this criteria. | The two parts of this project that required me to research and implement new approaches were the SignalR websockets and the highly-asynchronous Blazor UI framework. I learned a lot about establishing real-time communication, batching updates, and having data be loaded on the server without slowing down the UI and while updating related fields. For example, in Stats.razor, I couldn’t simply @bind PageNumber to a property and use a setter to fetch the data, since property setters can’t contain async code, so I had to use one-way value binding and @oninput/@onclick events. |
| **Student 3 Recommended Mark**  Based on your justification, what mark do you believe you deserve. | 4/4 |