# Real World Interaction - final project

The RWI assessment has two components:

- Labs - those you pass by doing the weekly assignments, there's no grade for it
- The final project and an oral assessment - those are graded and contribute to the final grade (80%-20% weights).

The final project can be done individually or in pairs. **Doing the project in pairs requires an explicit permission of a teacher**. That's to prevent piggy-backing that happens way too often in group projects. The final project application must be written in C# and contain at least the following elements:

- A GUI - We only allow the *WPF* or its open-source alternative, the *Avalonia* framework. (Really, no other GUI frameworks are allowed.)
- A networking component - your application has to communicate over the network with external open services or with peers (p2p) or be a client/server combo.
- Perhaps not an obvious one: for the purpose of the project, a GUI implies user interaction.
- A design with UML class diagrams.

Moreover, the application:

- **Must use exceptions** to handle failure and communicate errors to a user. It must not not crash.
- Should use asynchronous tasks - in order to make your application responsive, long-lasting tasks need to be performed asynchronously. That's not a hard requirement.

## Final project

There are two types of applications you can select from:

1. A **GUI application** that uses an external RESTful (free) web service(s) to gather data and display it in a graphical form to the user. This can be for instance (those are only examples):
    - A weather forecast application, allowing a user a selection of a city, etc.
    - A stock market application, allowing a user to select a stock and a time period, etc. and see the data for this period
    - A cryptocurrency application, allowing a user to select a cryptocurrency and a time period, etc. and display the associated information
    - A news application, allowing a user to follow news sources or subscribe to interest areas.
    - A sport statistics application (e.g. tracking of Formula 1 or NBA results).
    - A game application (not easy to find existing RESTful web services for this), in this case your app fetches a game from the server and allows the user to play it. This can be e.g. a sudoku puzzle or some other simple single-player game.
2. A TCP client-server application combo. The client is a GUI application that connects to a server (this can be a Console app) and sends data to it. The server should allow multiple clients to connect at the same time. Examples of such applications are:
    - A chat application, allowing multiple users to connect and chat with each other. The chat client is a GUI app.
    - A file exchange application, allowing multiple users to share files with each other. Every connected user offers a list of files they want to share. Other users can browse those lists and download the files.
    - A game application, allowing multiple users to connect and play a game **with each other** (think easy, tic-tac-toe, battleship, etc., something where it's simple to check who won). The server should potentially keep high scores etc.
    - A game applications, allowing multiple users to connect and play a game **alone** (think easy, minesweeper, sudoku, etc., something where it's simple to check the game end condition and program the game logic or use an existing library for it). The server should potentially keep high scores etc.

Discuss your application idea with a teacher - what you want to do and if you'll need any external services for it. Make sure that you find an external service (in the case of consuming a REST API) or an existing library that will make your program possible (e.g. if you want to do a client-server application for

playing chess) **before you pitch your idea**. The teacher will also approve doing a project in pairs if it's sufficiently complex and allows for splitting up tasks between the two.

### List of public REST API

You can find a list of free, public REST APIs at this link https://github.com/public-apis/public-apis. If that one doesn't work, use the fork: https://github.com/public-api-lists/public-api-lists.

## Final project submission

A final submission must contain the following:

- A zipped project, including all the assets.
- UML class diagrams of the design. As a guideline: we'd like to see classes, inheritance, and interface implementations in the diagrams. When applicable also aggregations/ compositions. (If the project is really big, documentation of only a part of it is sufficient - 10-15 fully documented entities is enough.)
- Any instructions that help running and testing the program, if necessary. Those must be stored in a README.md markdown document.
- If the project has been done by a pair a file TASKS.md must be included. This is a markdown document with only a single table inside that documents the division of tasks between the contributors. An example of such a document will be posted.
- A signed own work declaration, in which a student declares that they've done the assignment on their own and that all sources that were used are properly referred. This form will be made available.

**If some of the required items are missing from a submission, it will be directly marked as not meeting the requirements. The same grade will be filled in Bison as the first attempt's grade.**

## Project's grading

A final project program must have a functioning, interactive GUI, use networking and compile/ run. We'll try to build and run your project on a Windows machine with .NET 6.0 installed. We'll be able to use either the dotnet command or Visual Studio or Rider to do so.

**The project will be automatically graded as not meeting the requirements if there is no GUI or no networking used in it.** If the program cannot be compiled and run (possibly following extra steps in README.md) the project will be graded as not meeting the requirements. In both cases the same grade (Doesn't meet the conditional requirements) will be also filled in Bison as the first attempt's grade.

If the program can be built and run, it will be **graded using the rubrics visible in the final project's submission form**.

The points received for each item are added, giving a maximum of 50. An attempt is considered successful if it scored at least 25 points. Authors of unsuccessful attempts will get a grade Doesn't meet the conditional requirements.

## Assessment

Authors of programs that were awarded at least 25 points will participate in a final assessment. During the assessment each student will be asked two questions, one about a piece of the program she has written and one related to the content taught in the module. Answers will be graded using the scale:

- 0 - unsatisfactory: wrong or no answer.
- 1 - below expectations: an attempt at an answer was made, however, despite multiple prompts the answer missed the point.
- 2 - sufficient: an answer provided some evidence that the topic is known to a student, albeit superficially. Multiple hints were needed to arrive to it.
- 3 - moderate: correct, however somewhat brief answer with some omissions (despite hints).
- 4 - good: correct answer with possible minor mistakes and few, if any prompts.
- 5 - very good: correct, detailed answer without hints, prompts and help.

## Final grade

The final grade is calculated as:

$$Grade = (0.8 \cdot P + A)/5$$

Rounded to the nearest integer. $P$ stands for the project points (50 max), $A$ for the assessment (10 max)