# Assignment 2

Having just impressed the client, your next task is to join with others for a bigger project, you talk and divide the initial work amongst yourselves.

- Until the second laboratory, form two teams of equal number from the students in your subgroup. Each team will pick a team leader. A team leader will have some extra responsibilities and will be able to increase the final lab grade of any one student by +1 at the end of the semester.
- At the seminar you will negotiate a larger program to implement. After your seminar each team member will pick one module to implement, that fits with the assignment they get at the seminar. Each team member will have a different module, which they will implement it until the deadline.
- Do not use any framework functions to solve the problem entirely. It's perfectly fine to use framework functions to solve specific parts, let's say a string.Split(), but don't have your entire module be solved in 2 lines of framework calls.
- Deadline: lab 2

Data compression module: Implement string data compression using Huffman coding. Key text location should be read from a file.

Logging Module: Design a module for logging application events, errors, and user activities, crucial for debugging and monitoring. Develop a simple logging system that writes messages to a text file, including a timestamp and log level (INFO, WARN, ERROR). Info messages should be buffered and written in batches. Error level logs should be written immediately.

Data Encryption Module: Implement configurable substitution cyphers for strings. The module should be able to encrypt and decrypt.

Authentication Module: Manage user authentication with session management. Username and password will be checked against a known list. A session should expire after a set time. The module should allow the checking if a user is logged in. Passwords should be salted and hashed.

Sorting Module: Implement sorting algorithms on lists of any type. The algorithm should be configurable (Bubble Sort, Merge Sort, Gnome).

Configuration Management Module: Implement a configuration loader that reads settings from a JSON or an XML file. It needs to be able to handle different types (numbers, strings, lists) and special characters.

Caching Module: Design a module to cache frequently accessed data using an efficient memory store. On request, specific cached data must be cleared or updated. It should work with any type. Cache should expire after a while.

Internationalization Module: Create a module to support internationalization, allowing the application to cater to users from different locales with various languages. Given the language and the locale the module will provide the localized text for a given input. It should also support various date and number formats.

Feature Toggle Module: Implement a module for feature toggling, enabling, or disabling features dynamically without deploying new code. Create a basic feature flag system that allows simple on/off control over features. A feature that is turned off should throw exception or not do anything.

Basic Data Validation Module: Create a module to validate and sanitize input data for common data types and formats (email, phone numbers, dates) and sanitize inputs to remove harmful characters used in SQL injection. Phone numbers may include “+- ()” characters.

[Your idea here] Module: If you can think of any general-purpose module that you’d rather work on, write to me on teams, and if it meets [redacted criteria] I’ll add it to this list so that others may pick it also. This must be done before the second lab.