PR

Pls read this, indicate that you have read and approved via submitting review

Oat++ is actually quite cool and easy to work with.

I set up a very basic but working API structure. Do not worry about the details, the only file that you should concern yourselves with is the `APIController.h/cpp`. That is where endpoints are defined exactly like in the provided example. Within those endpoints we will call game functions etc in a RESTful manner.

If you want to test it yourselves (assuming you windows weirdos are executing commands on wsl):
`sudo chmod +x utilities/oatpp-install.sh`
`sudo utilites/oatpp-install.sh`
`cmake -S . -B build`
`cmake --build build`
`cd build`
`./main`
`curl localhost:8000`

IMPORTANT

- I think we should start discussing project directory structuring (scroll down for my proposal to the "appendix" of this PR)
- I was thinking of setting up a simple docker containerization configuration for this project for the following reasons:
  - For those who have not worked with containerization before, it is the ultimate headache eliminator
  - We are using a client-server like architecure, web based frontend, c++ API engine
  - With client server architectures the issue is always the communication between them, sure you all remember xampp issues with COS216
  - If we containerize, the networking will be preconfigured and isolated, i.e. you run a simple `docker-compose up` then it spins up either two or three seperate containers
    - One for c++ API engine, oatpp. Then if you do not want to install oat++ locally you do not have to, will be installed in container.
    - One for webserver to serve frontend, comms between API and frontend made really easy and reliable via docker networking, can configure live reload as well, i.e. you change some html/css and it reflects immediately (live) in your browser
    - Finally if we are going to use DB always good idea to containerize that
  - Containerizing eliminates configuration issues and variables, everybody will have the same view of the system, it is not host dependent. Eliminates "it works on my machine" issues.
  - @aneburger my experience with docker/containerization is purely practical, know you guys study the theory and best practices in multimedia, shout at me if this is a stupid idea. The rest can also give input.
  - **IF NOBODY AGREES/DISAGREES I AM GOING TO GO AHEAD WITH THE IDEA**

Appendix

A

Proposed directory structure, done with the help of AI:

```
📂 COS214-Project-Repo
├── 📁 backend/              # Core backend logic (C++ sources, Oat++ API, DTOs, etc.)
│   ├── API/
│   ├── game/
│   ├── customer/
│   ├── transaction/
│   ├── plant/
│   ├── greenhouse/
│   └── main.cpp
│
├── 📁 frontend/             # Web-based GUI (HTML/CSS/JS/React or similar)
│   └── (placeholder – to be developed)
│
├── 📁 utils/                # Utility scripts (installers, helpers, etc.)
│   ├── oatpp-install.sh
│   └── ...
│
├── 📁 config/               # Configuration files (environment, DB, app settings)
│   ├── config.json
│   └── .env
│
├── 📁 tests/                # Unit tests and integration tests
│   └── g_test.cpp
│
├── 📁 build/                # Build artifacts (CMake outputs, binaries, libraries)
│   ├── bin/
│   ├── lib/
│   └── _deps/
│
├── 📁 docs/                 # Documentation and readmes
│   ├── README.md
│   └── BUILD_CI_TESTING_README.md
│
├── CMakeLists.txt           # Project build configuration
└── .gitignore
```

B

If you are going to install oat++ locally, add this to your `.vscode` directory to take away squigglies, name it `c_cpp_properties.json`:

```
{
    "configurations": [
        {
            "name": "Linux",
            "includePath": [
                "${workspaceFolder}/**",
                "/usr/local/include/oatpp-1.4.0/oatpp/"
            ],
```

```
            "defines": [],
            "compilerPath": "/usr/bin/gcc",
            "cStandard": "c17",
            "cppStandard": "gnu++17",
            "intelliSenseMode": "linux-gcc-x64"
        },
        {
            "name": "Main",
            "includePath": [
                "${workspaceFolder}/**"
            ],
            "defines": [],
            "compilerPath": "/usr/bin/gcc",
            "cStandard": "c17",
            "cppStandard": "gnu++17",
            "intelliSenseMode": "linux-gcc-x64"
        }
    ],
    "version": 4
}
```