Global 01 Curriculum

🔴 **Mandatory:** A mandatory project in the main curriculum.

🟡 **Optional:** An optional bonus project, which enhances mandatory projects.

🟢 **Open:** A project which is unlocked once a defined skill or level has been reached.

🎓 *Levels: 5-18*

`go-reloaded`

A simple tool for text completion, formatting and auto-correction.

🔴 **Mandatory / Go / 1 student / 1 week**

Your very first project as a student. You will freshen up on your knowledge from the piscine by creating a very simple tool for text completion, formatting and auto-correction. You will learn to process command line arguments and the basics of string manipulation. You will learn to read from and write to the file system.

View project

`tetris-optimizer`

Arrange tetriminos into the smallest possible square.

🟢 **Open / Go / 1 student / 1 week**

You will open and parse a list of tetriminos from a file. You will need to validate that the teriminos are valid, and create an algorithm which will arrange them into the smallest possible square.

View project

`atm-management-system`

Add new features to an ATM management system, as part of an existing C project.

🟢 **Open / C / 1 student / 1 week**

You will expand an existing C project by adding features to an ATM management system. You will learn about the fundamentals of C, Makefiles, data structures, memory management, pipes and child processes. Optionally you may implement a lightweight database: SQLite

View project

`ascii-art`

Convert a string into a graphic composed of ascii characters.

🔴 **Mandatory / Go / 2-3 students / 1 week**

You will convert a command line argument into a multi-line graphic using only ascii characters. You will learn about parsing files. You may optionally accept command line flags to add bonus features to your project.

View project

`color`

Colour the ascii-art project.

🟡 **Optional / Go / 2-3 students / 1 week**

You will add a command-line flag to your ascii project which will specify the colour of the output. You'll learn about terminal formatting.

`output`

Write the ascii-art project to a file.

🟡 **Optional / Go / 2-3 students / 1 week**

You will add a command line flag to your ascii project which will specify a file path. You'll write the output to that file.

`fs`

Add some flair to the ascii-art project.

🟡 **Optional / Go / 2-3 students / 1 week**

You will add a command line flag to your ascii-art project which will specify a path. You'll read the graphic style from the file at that path to create a different appearance.

`justify`

Position the ascii-art project nicely in the terminal.

🟡 **Optional / Go / 2-3 students / 1 week**

You will add a command line flag to your ascii-art project which will specify alignment. You will need to know how wide the terminal is, so as to position the graphic correctly.

`reverse`

Reverse engineer the ascii-art project.

🟡 **Optional / Go / 2-3 students / 1 week**

You will need to reverse engineer your ascii-art project. You will accept a command line flag which specifies a path to a file. You'll open that file which contains an ascii-art project, and print its raw string equivalent to the output. Don't forget to validate the file.

`push-swap`

A sorting algorithm and checker

🟢 **Open / Go / 2-3 students / 2 weeks**

You will create a push-swap program which accepts a list of integer values. Your program will sort the values using two stacks, which can each be rotated, and values can be popped from the top of one stack onto the top of the other. Each rotation or pop action will be written to the standard output. You will also create a checker program which also accepts a list of integer values, as well as a list of actions from your push-swap program. It will simply output OK or KO depending on whether the actions properly sort the list. Your push-swap algorithm must attempt to sort the list in the smallest number of moves.

View project

`ascii-art-web`

A web version of ascii-art.

🔴 **Mandatory / Go / 2-3 students / 1 week**

Build upon your ascii-art project, by building a web server which generates HTML to display the project in a web browser. You will learn REST methods: GET and POST, and implement basic error handling with appropriate HTTP response codes. You'll describe your project in a readme with markdown.

View project

`my-ls`

A recreation of ls

🟢 **Open / Go / 2-3 students / 1 week**

You will recreate ls, which is a command for listing files in a Unix operating system. Your recreation will implement at least the -l, -R, -a, -r and -t flags. You will learn about the file system, access control lists, symbolic files, parsing command line arguments, and accurately displaying data.

View project

`groupie-tracker`

A full stack project for displaying information about musical acts.

🔴 **Mandatory / Go / 2-3 students / 2 weeks**

You will create a RESTful API with multiple endpoints, which serves information about musical acts. You will use HTML, styling, and basic JavaScript to create a website to visualise the JSON data returned by the API. You will learn about events/actions, data manipulation, and relationships. Your website must have a good user experience.

View project

`filters`

Filters for groupie-tracker.

🟡 **Optional / Go / 2-3 students / 1 week**

You will add filters to groupie-tracker, to enable users to filter by creation date, first album date, number of band members and locations of concerts. You will implement at least one range filter, and one checkbox filter.

`geolocalization`

Geolocation for groupie-tracker

🟡 **Optional / Go / 2-3 students / 1 week**

You will add a geolocation feature to groupie-tracker, which will display the locations of concerts on a map. You will convert the city to coordinates. You will learn about third-party map APIs.

`vizualisations`

Usability for groupie-tracker

🟡 **Optional / Go / 2-3 students / 1 week**

You will improve groupie tracker to ensure that its user experience follows good design principles, defined by Shneiderman's Eight Golden Rules.

`search-bar`

Search for groupie-tracker

🟡 **Optional / Go / 2-3 students / 1 week**

You will add a search bar to groupie-tracker, which enables users to find artists, concert locations, band members etc. It must be possible to find an entity based on any of its attributes.

`lem-in`

A pathfinding algorithm.

🔴 **Mandatory / Go / 3-4 students / 2 weeks**

You will create a path-finding algorithm to move ants from one part of a colony to another. You will parse a file to interpret the layout of the colony, including the position of rooms, and the connections between them. Only one ant may be in a room at a time, so you must control the flow of traffic. You must identify the most efficient route or routes. Your data structure and algorithm choices will be important, to ensure that your algorithm can scale to giant colonies. You may optionally create a visualiser.

View project

`forum`

A full stack web forum project

🔴 **Mandatory / Go / 4-5 students / 4 weeks**

You will create a forum which enables users to register, create posts, comment, like and dislike. You will manage access control to ensure that non-registered users can only view the forum. It should also be possible to filter posts. You will create a SQLite database with SELECT, CREATE and INSERT statements. An Entity Relationship Diagram is highly recommended. You must authenticate users and manage sessions. You may optionally use bcrypt to encrypt passwords, and UUID. Your project must use Docker, and you will learn about compatibility/dependency, containerizing an application and creating images.

View project

`authentication`

SSO for the forum project.

🟡 **Optional / Go / 4-5 students / 1 week**

You will implement new authentication methods to the forum project, to enable users to sign in or register with their Github or Google accounts. You may optionally add other SSO providers like Facebook, LinkedIn etc. You will learn about SSO and IAM.

`image-upload`

Image upload for the forum project.

🟡 **Optional / Go / 4-5 students / 1 week**

You will enable users to upload images along with their posts in the forum project. You must support at least JPEG, PNG and GIF. You will limit file size, and should be cautious about image dimensions.

`security`

Security for the forum project

🟡 **Optional / Go / 4-5 students / 1 week**

You will secure your forum project, by implementing HTTPS with an SSL certificate, implementing rate limiting, encrypting passwords, and persisting session states on the server. Optionally, you may decide to password protect your entire database. You will learn about cipher suites, certificate authorities, encryption, and UUID.

`moderation`

Moderation for the forum project

🟡 **Optional / Go / 4-5 students / 1 week**

You will add moderation capabilities to the forum project. You will create admin and moderator roles. Moderators will be able to approve posts before publication, and delete existing posts. Admins will be able to appoint moderators. You will learn about more advanced authorization.

`advanced-features`

Advanced features for the forum project

🟡 **Optional / Go / 4-5 students / 1 week**

You will add notification features to the forum project, to alert users when their posts have been commented on, liked or disliked. You will create an activity page to enable users to see what posts and comments they have created, and what they have liked and disliked. You will enable users to delete or edit comments and posts. You will learn some more advanced SQL.

🎓 *Levels: 17-38*

`make-your-game`

A solo game with JS and HTML

🔴 **Mandatory / JavaScript / 2-3 students / 3 weeks**

You will create a game with JavaScript and HTML, without any frameworks, canvas, WebGL etc. Your game must be performant at 60FPS without dropping frames. Your game must implement a pause menu, show a countdown clock, keep score and show remaining lives. Controls must be smooth, so that a player is not required to repeatedly press a key to move smoothly. In this project, you will learn about requestAnimationFrame, the event loop, FPS, DOM, jank/stutter, transform, opacity and dev tools.

View project

`score-handling`

Online score handling for make-your-game

🟡 **Optional / JavaScript / 2-3 students / 1 week**

You will add online scoring to make-your-game. You will implement an API in Go, which will feature GET and POST methods to get and submit scores. After each round, a paginated list of scores will be shown in ranked order. You will learn about sorting algorithms.

`history`

Story mode for make-your-game

🟡 **Optional / JavaScript / 2-3 students / 1 week**

You will add a story mode to your make-your-game project, so that the gameplay represents a storyline. Your gameplay must have a distinct introduction, middle and end. You will learn about game story theory.

`different-maps`

Efficient tile maps for make-your-game

🟡 **Optional / JavaScript / 2-3 students / 1 week**

You will optimise your tile rendering performance in make-your-game by combining your image tiles into one map. You will create a tile set, and an engine to create a map. You will learn about image manipulation

---

`real-time-forum`

A forum with real-time features

🔴 **Mandatory / JavaScript / 2 students / 2 weeks**

You will add a real-time private messaging feature to your existing forum project, and relaunch it as a single page application. There are additional requirements for how posts must be ordered. Your frontend must be implemented in JavaScript, HTML and CSS without any frameworks, and your backend will be in Golang and SQLite. You will implement websockets, and you may use gorilla for this purpose. This project will also help you to learn about the DOM, Goroutines and Go Channels.

View project

`typing-in-progress`

A typing indicator for real-time-forum

🟡 **Optional / JavaScript / 2 students / 1 week**

You will add a typing indicator to your real-time-forum project, so that the eventual recipient of a message can see that the sender is currently typing. You will need to use web sockets, and think about the logic for starting and stopping the animation.

`graphql`

A frontend project to display data from a GraphQL API

🔴 **Mandatory / JavaScript / 1 student / 2 weeks**

You will create a profile frontend, which will display your own school information like your name, XP and audits, amongst other data. This data will be obtained via an existing GraphQL API. You will display both static information and graphs using SVG. You will need to host your project, so that it is available on the world wide web. You will think about the user experience, and find a way to host your project.

View project

`social-network`

A Facebook-like Social Network.

🔴 **Mandatory / JavaScript / 4-6 students / 6 weeks**

You will create your own Facebook-like social network. Users will be enabled to create public or private profiles, publish posts with different privacy settings, and communicate via instant chat. Users will be able to join groups via invite or request, and be alerted to interesting actions with real-time notifications. You will use a frontend framework like React, Vue Svelte etc. Your backend app will be written in Go, and you must manage authentication and sessions. You will use SQLite and create db migrations. You will use websockets to enable real time features. Your frontend and backend will each be served by a docker image.

View project

`cross-platform-appimage`

A desktop messenger for social-network

🟡 **Optional / Open / 4-6 students / 1 week**

You will create a desktop application, as an instant chat for your social-network project, just like Facebook Messenger. You will enable users to sign in, register, see who is online and send/receive instant messages. You will implement an offline mode which will enable users to see their messages without internet connectivity. Your project will be implemented with Electron, and you'll learn about data manipulation, local storage and websockets

---

`mini-framework`

A mini UI framework.

🔴 **Mandatory / JavaScript / 4 students / 2 weeks**

You will create your very own mini UI framework. The framework will abstract the DOM as well as handling routing, events and state management. You will demonstrate your framework by creating a small TODO project. Your framework must have documentation to describe its usage, with examples of creating and nesting elements, and adding attributes.

View project

`bomberman-dom`

A multiplayer HTML game.

🔴 **Mandatory / JavaScript / 4 students / 4 weeks**

You will use your mini-framework project to create a multiplayer version of Bomberman. Your project must respect the same performance requirements as the make-your-game project. You must not use WebGL, canvas or any other framework. You will enable between 2 and 4 players to play on a shared map, and to communicate via a real-time chat. You will implement certain powerups, and optionally create more. In this project, you will learn about requestAnimationFrame, the event loop, FPS, DOM, jank/stutter, transform, opacity and dev tools.

View project

🎓 *Levels: 34-62*

`smart-road`

A traffic intersection algorithm for autonomous vehicles.

🔴 **Mandatory / Rust / 2-3 students / 1 week**

You will produce an algorithm to control the flow of autonomous vehicles through a 6-lane intersection, by varying the velocity of each vehicle and without traffic lights. You will also produce an animation to visualise it. It should be possible to spawn new AVs using the keyboard, and these vehicles must not overlap. You will display statistics to describe the throughput of the intersection. Image assets will be provided for you, but you may optionally choose to create your own.

View project

`filler`

A filler player.

🟢 **Open / Rust / 2 students / 1 week**

You will create a program which represents a filler player, which will attempt to place pieces on a board to increase its territory against another player. The game ends when no pieces can be placed, or an invalid piece is placed. Your program will be executed by a VM which will be provided. Your program will read the current board state and the next random piece to place from the standard output. Your program will output the best position to place that piece.

View project

`rt`

A Ray Tracer.

🔴 **Mandatory / Rust / 4-6 students / 4 weeks**

You will create a ray tracer, which will be capable of rendering a 3D scene, containing simple shapes like spheres and cubes. It must be possible to modify the scene by positioning shapes at different locations, viewing from different camera angles and adjusting light brightness. Your rendering time should be optimised. You will describe your project in a markdown readme. Your ray tracer must accept a P3 .ppm file, which you must prepare alongside your project. Optionally, you may consider textures, reflections, particles and fluids. This CGI project will test your geometry and mathematical skills.

View project

`zappy`

An automatic networked video game

🟢 **Open / Open / 2-4 students / 8 weeks**

You will create a multiplayer game over a TCP/IP network. The game represents a world which manages a population. The population consists of teams of players, each of which is an AI created by you. The game is won once the players of a team each reach the maximum level. You will create a server that will manage the game, enforce its rules, keep time and score. You will create a client which represents an AI player, and many players can connect to the server at once. You will create a graphical user interface, which will also connect to the server as a client. You will learn about client/server architecture, TCP/IP, group management and networking.

View project

`corewar`

A Corewar assembler, virtual machine and player.

🟢 **Open / Open / 4 students / 8 weeks**

Corewar is a programming game, in which two or more players compete to control a virtual machine (VM). Each player is defined in an assembly language, and you will create an assembler to compile it to byte code. You will create a VM, and the byte code of each player will be loaded into the VM's cyclical memory as processes. Players will attempt to modify the VM's memory, to corrupt their opponent's processes and ultimately control the VM. You will also create your own player, which must be competitive. Optionally, you may choose to create a visualiser, network game mode, create new instructions, or support mathematical operations in your assembly files. You will learn about assembly languages, assemblers, bytecode, virtual machines, registers, memory, program counters, endedness and encoding.

View project

`mutliplayer-fps`

A multiplayer 3D first-person shooter

🔴 **Mandatory / Rust / 4-8 students / 8 weeks**

You will create a networked 3D first-person shooter, as a recreation of Maze Wars. You'll implement your own graphical user interface, which enables players to move around a world, see their position on a mini map, destroy opponents, and view their frame rate. The client should communicate via UDP. The server and one client will be on a single machine, with a minimum of 10 other clients joining from other machines. The minimum performance is 50fps. Optionally, you may create an algorithm that creates mazes, or enable users to create their own. You may consider persisting the game's state to make it easier for clients to join, and create AI players.

View project

`0-shell`

A basic Unix shell

🔴 **Mandatory / Open / 4 students / 4 weeks**

You will create a basic unix shell, which will enable users to execute simple commands such as echo, cd, ls, pwd, cat, cp, rm, mv, mkdir and exit. The program must show a prompt ($) and handle ctrl+D interruption. You will learn about system services, command-line syntax, unix versus posix, and process creation. Optionally; you may consider Ctrl+c, auto-completion, redirection, pipes, colours, and other advanced commands. The program must be written in a compiled language like C, Go or Rust.

View project

`job-control`

Job control for 0-shell

🟡 **Optional / Open / 4 students / 2 weeks**