## **1. Promotional Material**

Our promotional material included an A2 poster, leaflets and a website to host both our promotional content as well as link to the interactive prototype.

Links:

* [Poster](https://github.com/deco3500-2017/BBD/blob/master/Promotional%20material/Poster.pdf)
* [Leaflet](https://github.com/deco3500-2017/BBD/blob/master/Promotional%20material/Leaflet.pdf)
* <http://bbd.uqcloud.net> - Promotional website

## **2. Link to the prototype**

<http://bbd.uqcloud.net/#buy>

## **3. Summary**

The initial idea behind Codeverter was to make learning how to code within classrooms a more social experience for not only the students but also the teachers with the aim to boost interest and participation.

In 2013 over 99% of Australian students had access to a computer at home and 98% had access to the internet *(statistics may variate for different age groups)*[2]. Statistics as such suggest that a web application would be an appropriate platform for Codeverter to run upon.

The web app targets primary school students from years 3 through to 6 and teachers of all technical skill levels. With a younger and older student age range being a potential stretch goal.

Basically a teacher creates their own profile on Codeverter and creates a “classroom” that their students will sign-on to with their own accounts. Codeverter allows the teachers to then assign tasks for the students to work upon.

There are different types of activities that require different levels of teamwork and collaboration.

* Individual activities: Activities that requires students to work independently.

For example:

* Activities for students to complete at home as a form of homework
* Individual learning, assessment and practice mode
* Group activities: Activities that require students to form small teams/pairs and work together to learn and solve problems.
* Class activities: Activities that require students to form small teams and work together or compete with other teams.

For example:

* All teams work together to solve a bigger problem.

For example each team works on a coding challenge and upon completion they add a puzzle piece to the classroom’s jigsaw board. The class works together to finish the jigsaw puzzle.

* Teams compete against each other to quickly finish coding challenges to for example win the car race.

We also understand that different year levels will require the content to be delivered in different ways and to be of certain levels of difficulty.

Years 3-4

* + Content will focus more on learning the basic concepts and developing a certain way of thinking and problem solving that will prepare students for when they actually start learning code.
  + A majority of the challenges will use visual representations of code such as building blocks and by the end students will be translating the visuals to pseudo code.

Years 5-6

* Content will be harder and go into more detail as appropriate for this older age range of students.
* Will start off with pseudo code however will then progress into translating that into real programming languages.

Trying to integrate Codeverter into classes also means that we need help support the teachers too. On their profile a teacher can indicate their own programming skill level. Based on this information Codeverter will provide a certain amount of support to the teacher and take over a certain amount of control over the classroom. So for teachers that aren’t so proficient at coding Codeverter can take more control over the teaching and also provide more support and guidance to the teacher in relation to their lesson and activity plans on the site. A possible form of support would be provide easy and quick learning material for teachers to study beforehand and give more suggestions for their lesson plans.

## **4. Process**

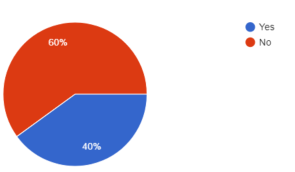
4.1 Initial Requirements Gathering

A more detailed breakdown of these results can be found here:

<https://github.com/deco3500-2017/BBD/blob/master/Tests/Initial%20Requirements%20Gathering.pdf>

The key points from the initial requirements gathering are listed below:

* 23-27 students in each classroom on average
* Computer labs widely available with 35-40 computers
* Groups of 2-4 work best for students
* Students work best when using visual learning material.
* Emphasis was made on presenting colour, interactive and creative forms of learning.
* Teachers also said that learning visually off the whiteboard and big display screen were some of the best ways to deliver content to the entire class.



How many teachers are teaching students how to program

Yes - 60% → Are interested in continuing to teach.

No - 40% → 100% of those teachers said they’d be interesting in teaching programming within their classrooms.

Learning to code isn’t a regular class activity.

However they have had exposure through:

**Scratch**

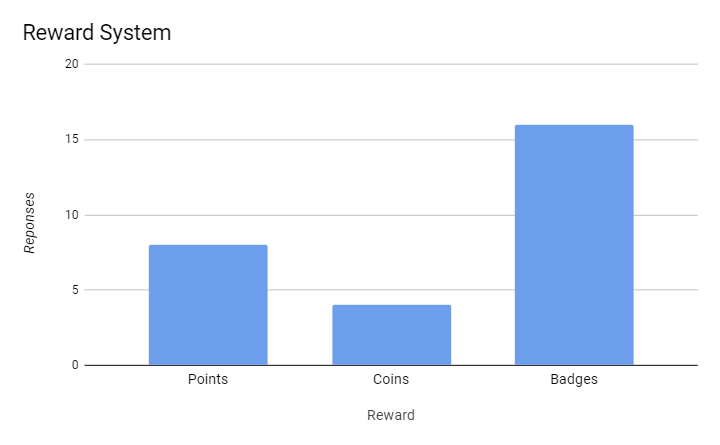
Key Comment: “There is a lot of restrictions. This also causes the planned activities to lack variety.”

**Python Turtle**

Key comment: “I am teaching the students turtle (with python) by setting exercises myself and showing examples on my computer and individually helping kids when they run into problems/have questions. It's good that it allows me to address each of the students problems individually but it is a fair amount of work coming up with the problems myself”

**Robogals workshops (lego mindstorms)**

Key Comment: “Students that have attended many workshops have said that the activities get quite repetitive and students lose interest to return.”

****

Testing what reward system will work best

4.2 Initial Design

The initial idea for Codeverter was a tool aimed at school students to assist with learning code that simply converted english sentences into lines of code. We felt that this lacked a social aspect and also did not fully fulfil “teaching children code” as they would only be writing english and not getting practice at the real thing.

We decided to use the idea of a tool that will assist teachers and students to learn programming in a way that is both engaging and social.

The design process started by discussing how to make learning code a more social experience. Our solution was to “gamify” the experience and utilise teamwork.

Games:

* As a team of 3, we initially brainstormed many possible games for the website, then each got assigned one to further develop, paper prototype and test
* We then narrowed it down to two games to fit in the time constraints and prototype nature of this project. The details for each game are below:

Maze Game:

In this game mode, students will race each other (as individuals or in teams) through a maze trying to reach a goal in the maze. Each student starts at a different location in the maze and is given steps (to use for moving through the maze) when they answer questions correctly.

On the student screen they can only see the questions and a close up of their player in the maze.

The teacher’s screen shows the entirety of the maze and allows everyone to see where everyone else is at any given time. This allows the teachers screen to be a point on which everyone is converging.

Spaceship Game:

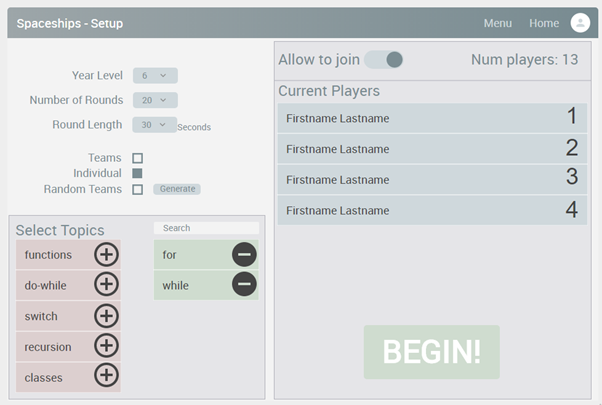
The spaceship game gives students the keys to their own spaceship that they will use to help fight aliens from invading earth. In this game mode students will work together to create a fleet of spaceships which they will program the behaviour for. The teacher will be able to specify the age level and this will change the amount of control that the students are given. The game is played in rounds and players will unlock new code snippets (that can be used to upgrade their ship - if they can code it) based on how well they do in the preceding round. Students can play as individuals or in “Crews” (teams) where they will write code collaboratively.

On the student screen they will only be able to see their ships statistics and the code they have written for their spaceship.

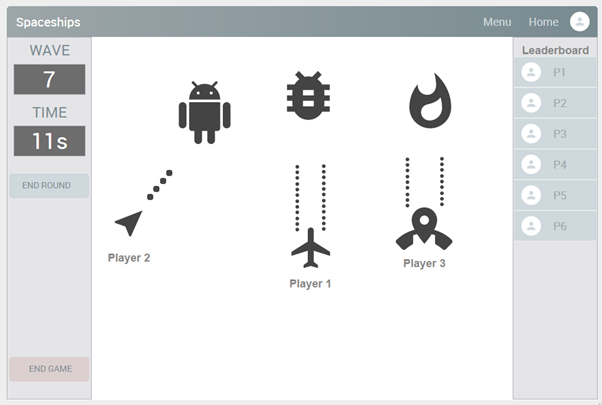
The teacher’s screen will show where everyone's spaceship is and show the battle taking place along with a leader board and other information about the game (round number, time, etc)

4.3 Paper Prototyping

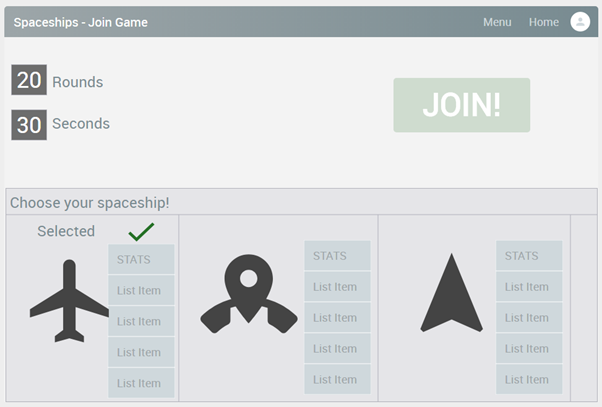
Spaceship Game:



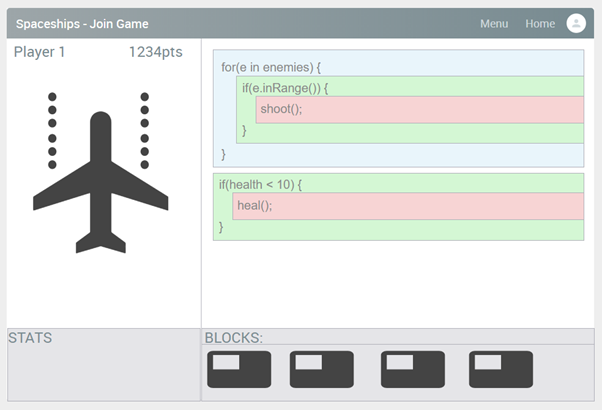
Option screen/setup for teacher



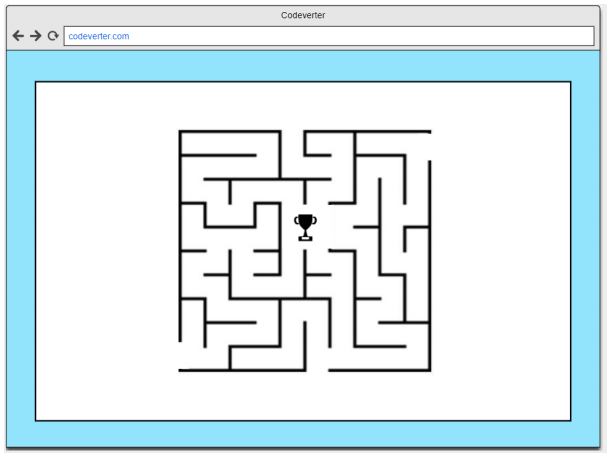
Game screen for teacher



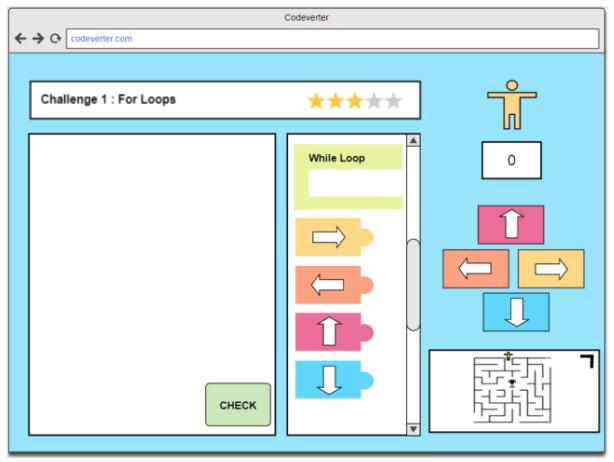
Start screen for students

Game screen for students

Maze Game:

****

Teacher’s game screen



Student game screen

More about paper prototypes can be found at the following links:

Space ship game: <https://github.com/deco3500-2017/BBD/tree/master/PaperPrototype/SpaceGame>

Maze game: <https://github.com/deco3500-2017/BBD/tree/master/PaperPrototype>

In these folders can be found detailed reports on user testing that was conducted on the above prototypes.

The findings of these tests helped inform a number of important decisions ;

Teachers were concerned over the amount of control they would have with content, they wanted to be able to tailor each lesson to their teaching styles and students. This lead to focus on the lesson plan concept and being able to choose the topics covered in each game.

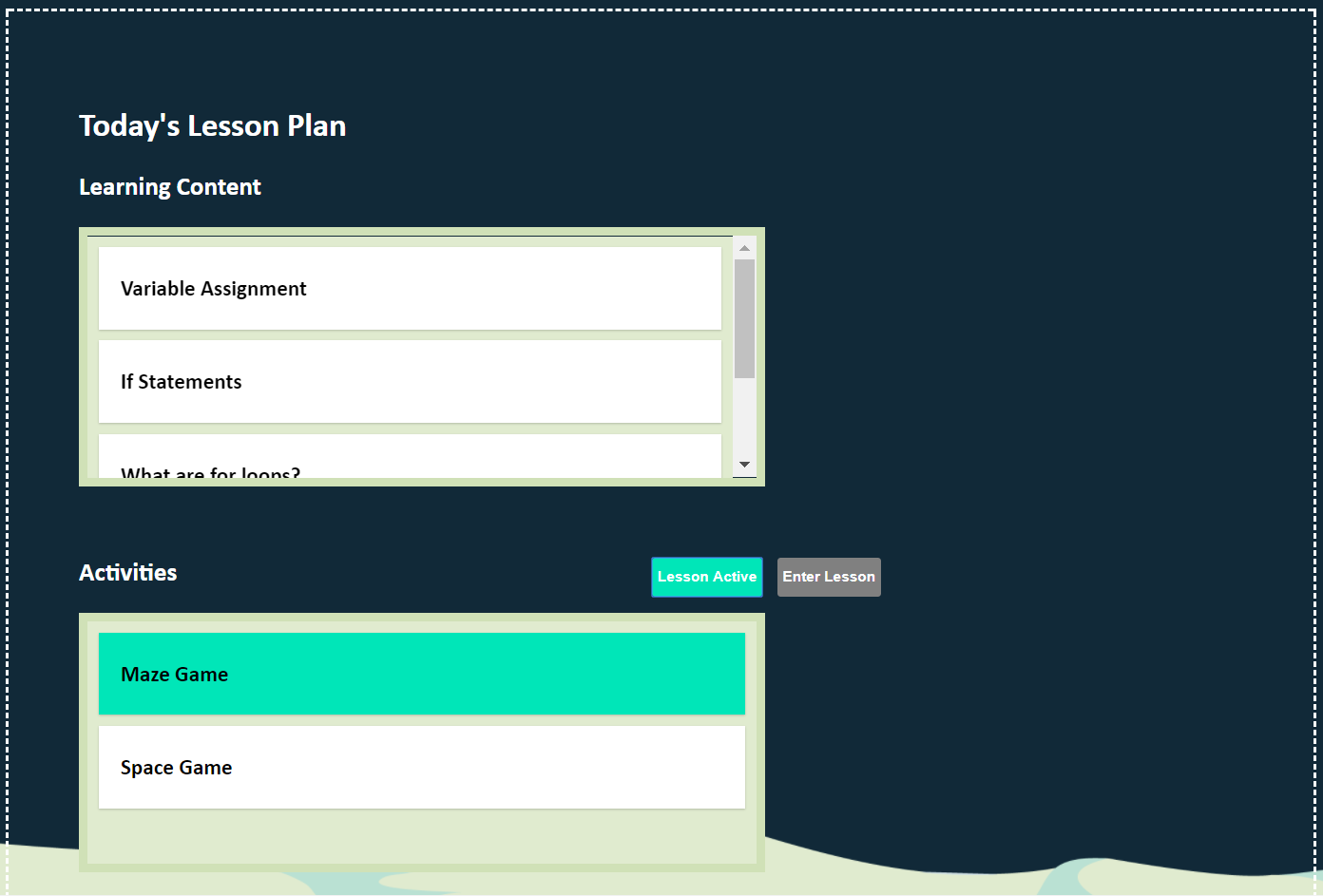
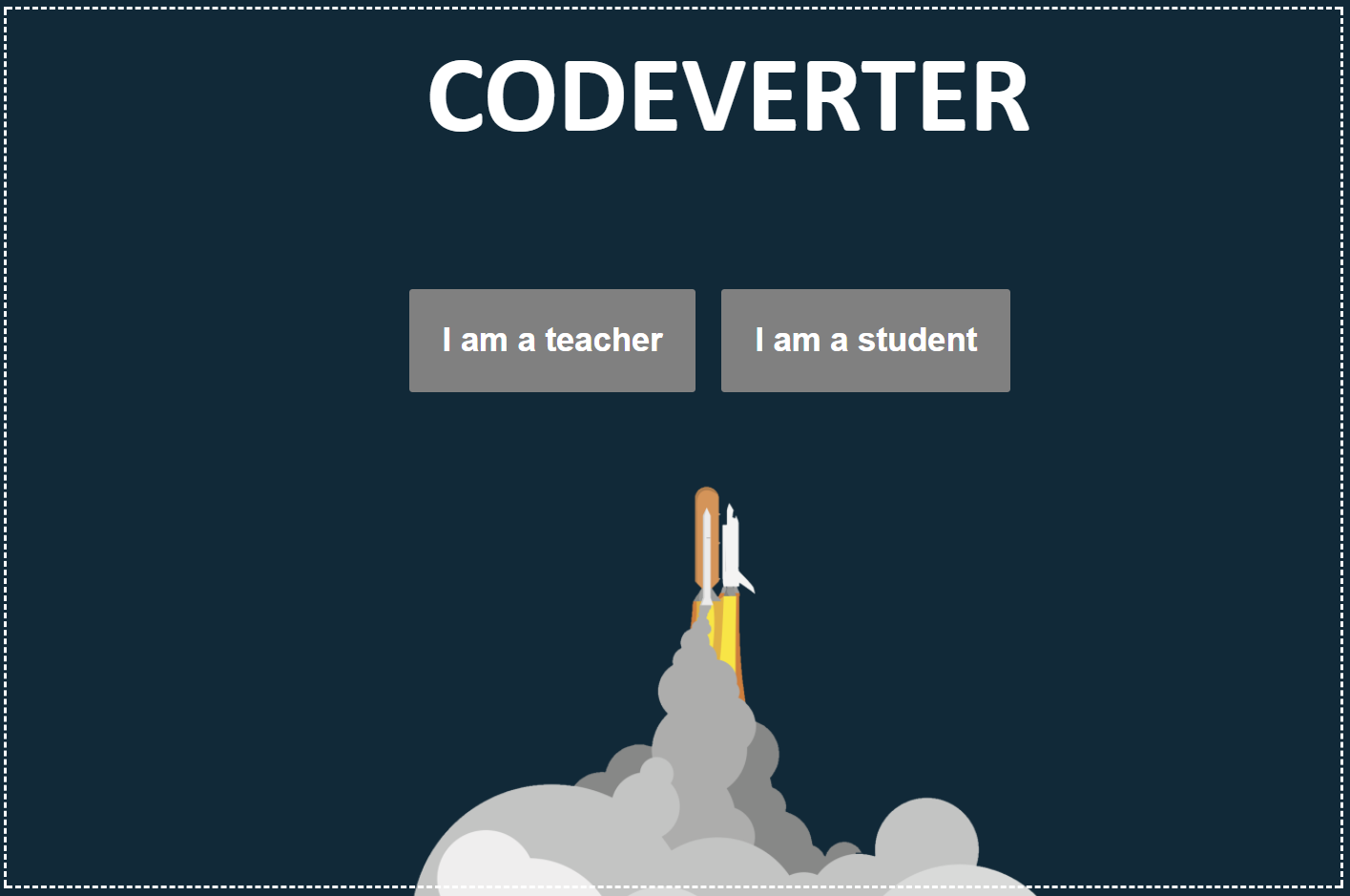
Students found the games enticing but the lack of colour in the initial prototypes detracted from their engagement. To combat this we added more style into the games and general design. (Background images, textures instead of flat colours, etc).

To make the whole tool more social we decided to add a “Dashboard” that would be used to celebrate the classes achievements and to highlight students that have been doing really well over the term. This includes things such as *most improved* and *best player in the last round of \_\_\_*.

Evidence of these changes can be seen in the figures from the Electronic Prototype section below.

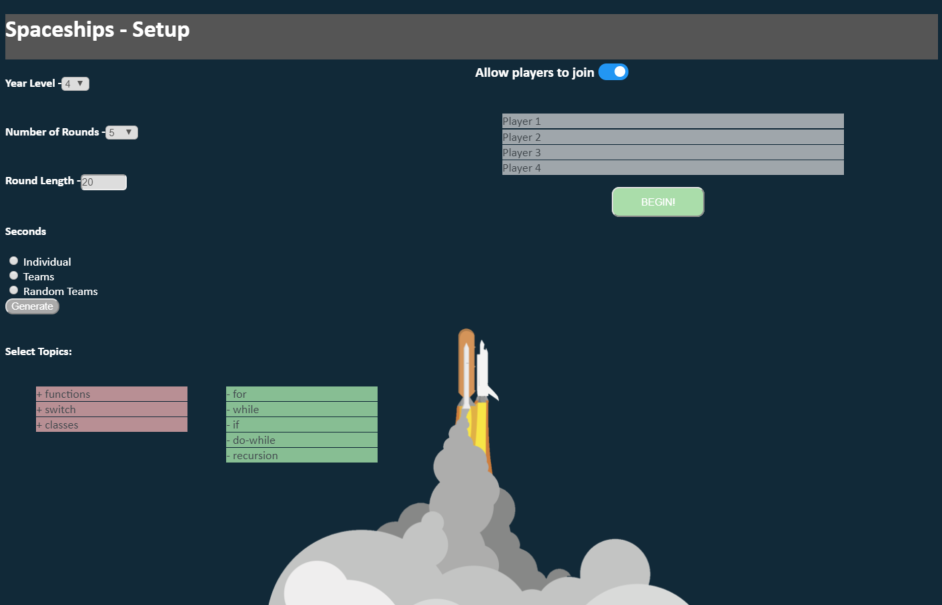
4.4 Electronic Prototype

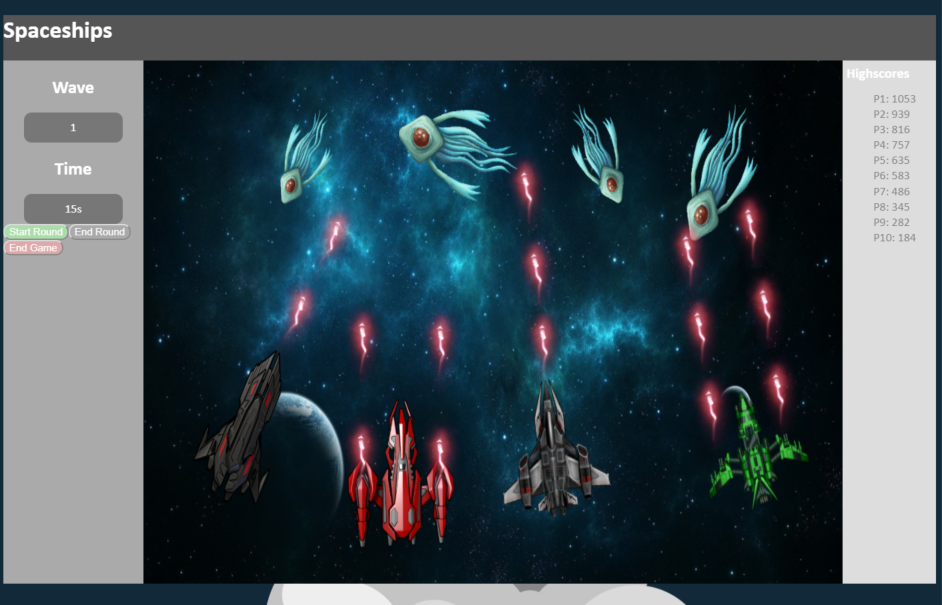
Main screen:



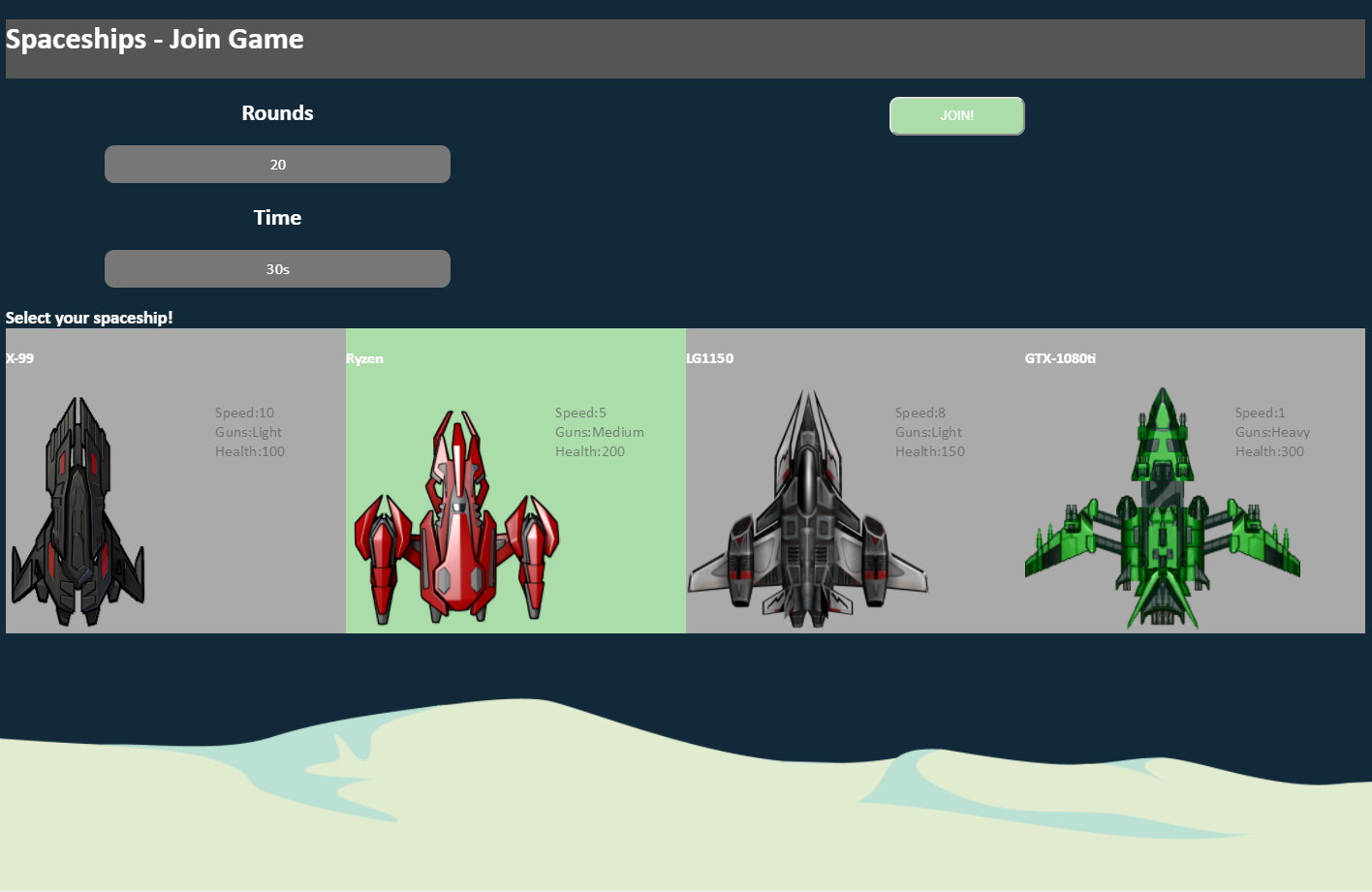
Lesson plan (viewable by teachers and students) that can be editted by teachers

Spaceship game:

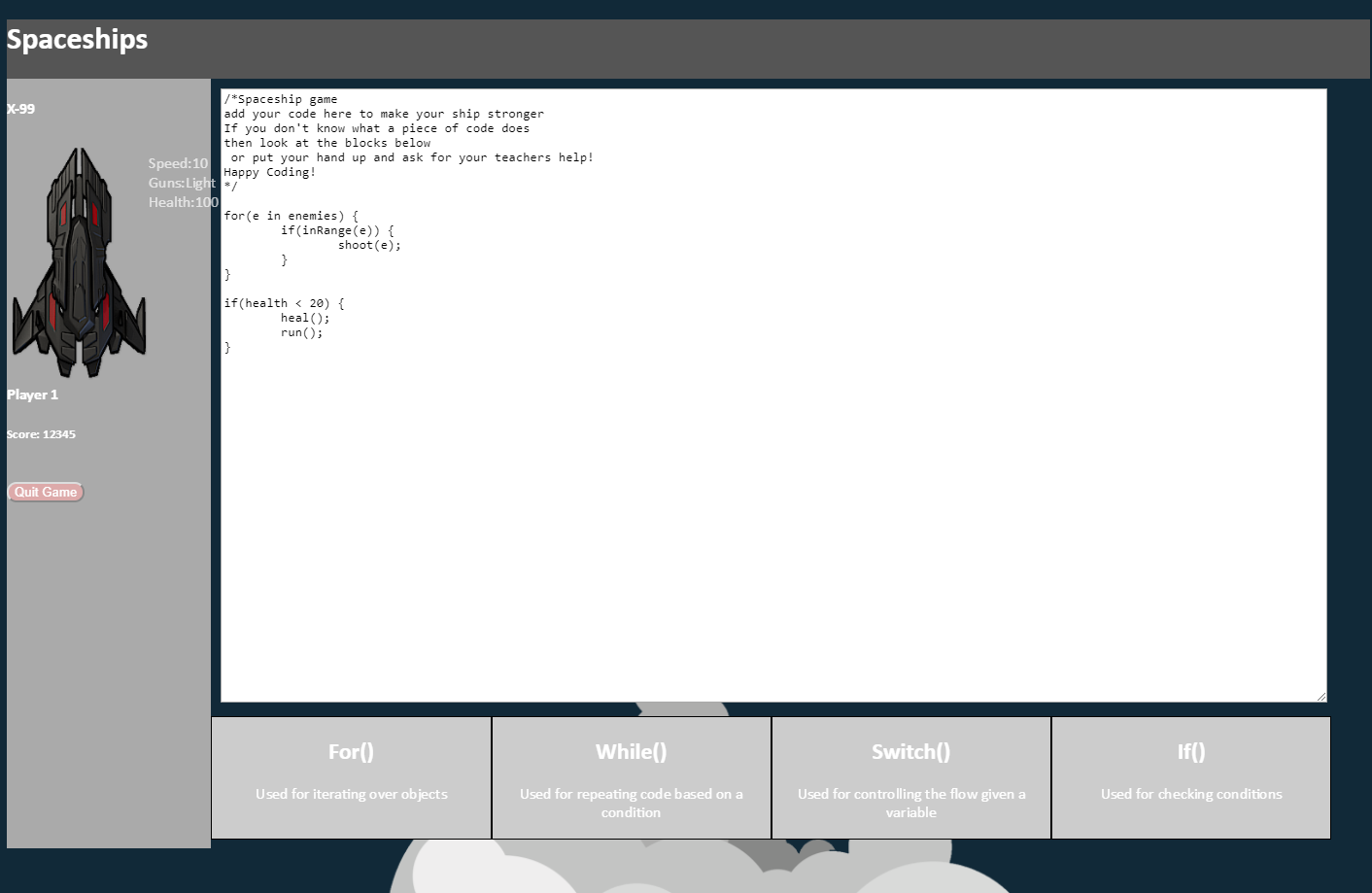


Teacher setup screen

Redone teacher game screen to be more engaging/colourful



Student start screen, allows student to select their spaceship

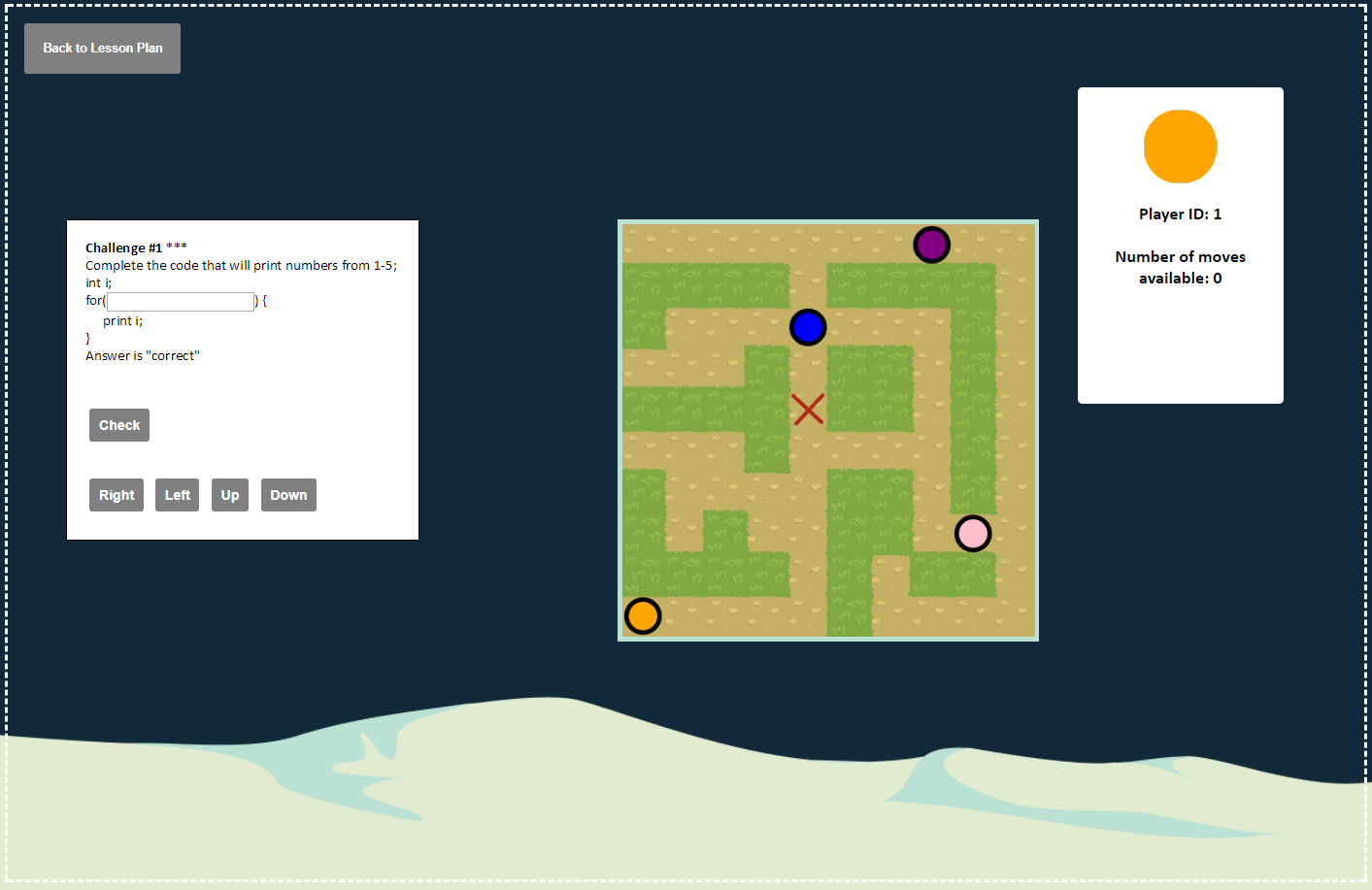


Student game screen, students can code their ships behaviour and can use code chunks that they unlock throughout the game.

Maze game:

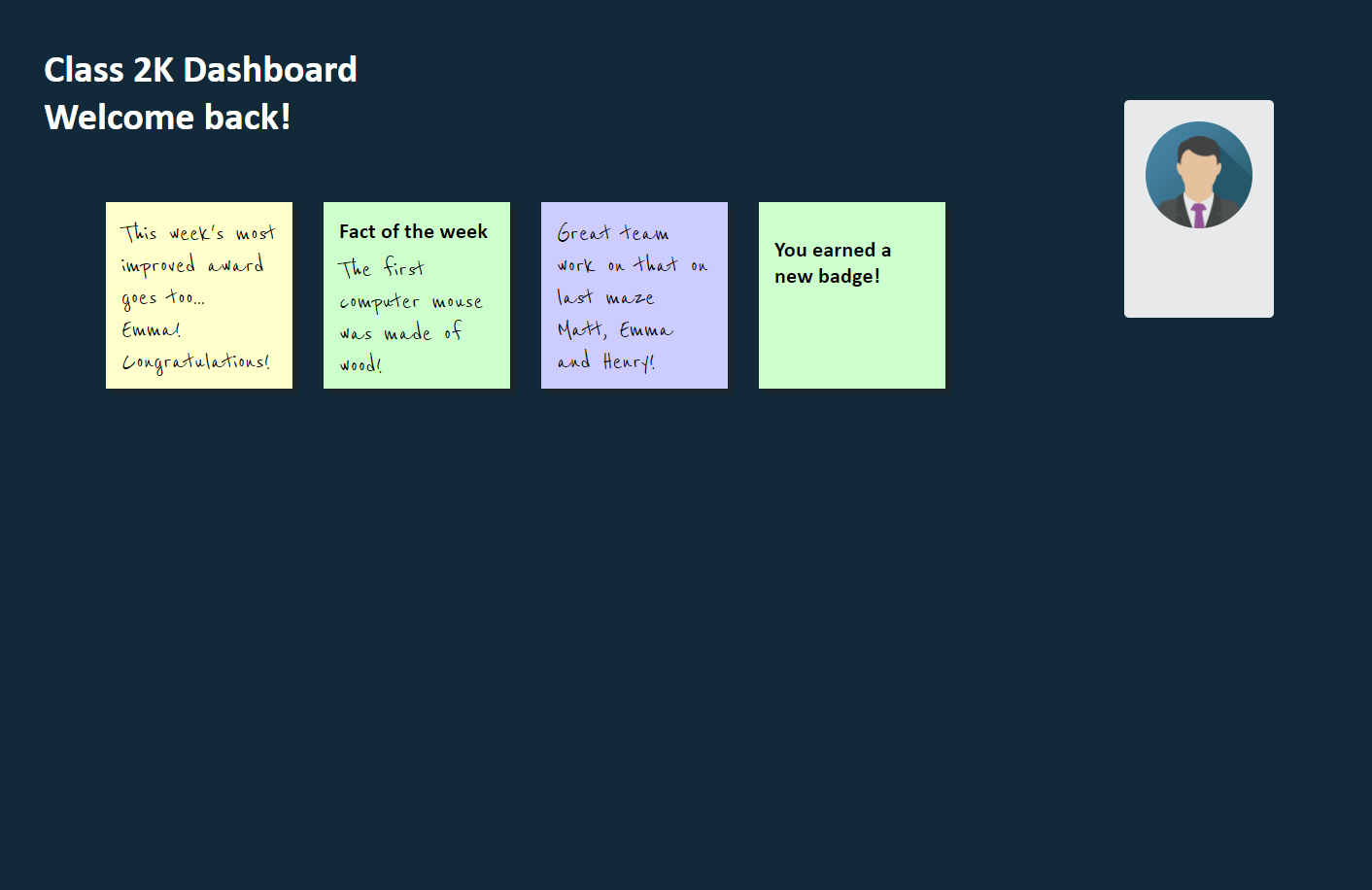


Teacher screen showing the maze and all players



Student screen showing maze and questions that grant moves

Dashboard:

Dashboard that is a central place for the class to celebrate each other's achievements.

## 5. Roles

You can download our planning table for who did what during the project here -

<https://github.com/deco3500-2017/BBD/blob/master/List%20of%20what%20each%20person%20did.docx>