Battleship Technical Documentation

Purpose: This document contains technical documentation that explains the structure of the Battleship game and the different folders used. Additionally, a section of the documentation is dedicated to showing how to set up a development environment where the game can be deployed to and ran from.

# Software Used:

**Java JDK 17.0.6**

<https://www.oracle.com/java/technologies/downloads/#java17>

**Microsoft SQL Server Express 2022**

[SQL Server Downloads | Microsoft](https://www.microsoft.com/en-us/sql-server/sql-server-downloads)

**Spring Boot 3.0.X:**

<https://start.spring.io/>;

Using Gradle Groovy and language set to Java

Text

Description automatically generated with low confidence

# Folder Structure

Graphical user interface, text, application, email

Description automatically generated

## Battleship Folder

Within the src folder there are 3 other folders

Graphical user interface, application

Description automatically generated

**Core Folder:** Keeps track of the actual game logic, creates the different Player classes, stores the logic for the CPU Algorithms, along with the different events for striking the board as well as when a ship is destroyed.

Graphical user interface, text, application

Description automatically generated

**Game Folder:** Logic for controlling which scenes are loaded based on user input, contains the different models for working the database, a services folder for calling the Battleship API. The Board class controls which scenes are created and disposed of. Game Constants class is used for storing all the resource asset locations and can easily be called. Main is the entry point for the Battleship game.

Graphical user interface, text, application, email

Description automatically generated

**AudioPlayer Class**: Class for playing the different music, the individual scenes control which music is played and what resources to load

**GameConstants Class:** tracks the resource files used by the scenes and the AudioPlayer class

**Main Class:** The entry point to the Battleship game

**TestBoard Class:** Used for testing the different positions to ensure the logic for hitting locations works as expected, the ships sink as expected, and the ships can be positioned correctly.

Within the scenes folder is each scene along with its corresponding events, drawers, and listeners.

Graphical user interface, text, application, email

Description automatically generated

To create a new scene, a new drawer, event, and listener class must be created. All scenes must implement the BaseScene Interface. Below is an example diagram using the SinglePlayerScene that illustrates what classes need to be added/modified in order to create a new scene

Diagram

Description automatically generated

**Resource Folder:** All the external resources we are using for images and music files to load in the game.

Graphical user interface, table

Description automatically generated with medium confidence

## Database Scripts Folder

Contains the script for creating the Sql Server Database to use on the server. Refer to the Microsoft SQL server Dev setup instructions in this document on how to setup the database. Once the database is setup for the first time, the script needs to be run to create all the necessary tables, procedures, etc.

### Database structure:

Diagram

Description automatically generated

**Game Table:** Tracks the different game instances, this is how different players can join the games

**Player Table:** Contains all the players ships positions and whether or not the ship is sunk yet or not

**Board Table:** Tracks all the positions on the board, whether they have been hit, and whether a players ship exists there or not.

**Position Table:** Contains all the positions of the game to be used, the script will initialize all the positions to be used in the procedure for creating games

### Stored Procedures

The following are the stored procedures created when running the database setup script

Text

Description automatically generated

**CREATE\_NEW\_CPU\_GAME:** Calling this will create a new CPU game, used by the Spring Boot Api. A new entry is added to the Game table and two Players are created in the Player table with the matching game id.

**CREATE\_NEW\_MULTIPLAYER\_GAME:** Calling this procedure will create a New Multiplayer game. A new entry is added to the Game table and two Players are created in the Player table with the matching game id.

**CREATE\_USER:** Used directly in the Database setup script, creates a database user that is used in the Spring Boot API for the database connection

**TABLE\_CREATION:** This is used to create the base tables that are used for storing the Battleship game data

## SpringBootApi-Battleship Folder

Within the **./src/main/java/com/sprinbootapi/battleship/demo** folder is the main code for making changes to the Spring Boot Api:

Graphical user interface, text, application, email

Description automatically generated

**Controllers:** This is what contains the different controllers with the API endpoints, this is done in the Game Controller class. The GameController class uses the MsSqlWrapper to connect to the database and the BattleshipQueries class to run the different SQL queries against the database. The different model classes are used to send and receive data between the database.



**Datasource**: contains BattleshipQueries and MsSqlWrapper classes. If the database connection string needs to be edited, the MsSqlWrapper is where it would need to be done.

Graphical user interface, application, background pattern

Description automatically generated with medium confidence

**Models:** These are the models used by GameController to read and write data to the database using its different endpoints: Graphical user interface, text, application, email

Description automatically generated

When the api is running, the swagger page can be accessed at

[http://localhost:8081/swagger-ui/index.html#](http://localhost:8081/swagger-ui/index.html)

This lists all the different API endpoints used as well as allows for testing in the browser. This will show the structure of the different Models as well in the different endpoints. Normally, these would only be called by the game itself and not in the browser. It is advised to run the Database Setup script again to ensure that any database writes done can be reset.

Development Environment Setup Instructions

Purpose: This portion of the document will show how to setup a server where the game and the database can be deployed to. Note that these instructions assume the machine being used is a Windows machine

* Programming Language: Java
* IDE: Visual Studio
* Database: Microsoft SQL Server
* Database Manager: Microsoft SQL Server Manager

GitHub repository link: <https://github.com/bg1302023/COSC625Project.git>

# Visual Studio Setup:

## 1) Install Visual Studio Code at:

<https://code.visualstudio.com/Download>

Website, timeline

Description automatically generated

## 2) Install the Coding Pack for Java at:

<https://code.visualstudio.com/docs/java/java-tutorial#_coding-pack-for-java>

Graphical user interface, text, application, email, website

Description automatically generated

### Run the installer through to completion, accept the license agreement and click next

Graphical user interface, text, application, email

Description automatically generated

### Select Install and wait for it to finish:

Graphical user interface, text, application, email

Description automatically generated

## 3) Install the Extension Pack for Java at:

<https://code.visualstudio.com/docs/java/java-tutorial#_coding-pack-for-java>

Graphical user interface, text, application, email

Description automatically generated

### Clicking the link will open this prompt, select “open”:

Graphical user interface, text, application, email

Description automatically generated

## 4) Inside Visual Studio Code, Click install for the Extension pack for Java

A screenshot of a computer

Description automatically generated with medium confidence

# Creating a local repo from the remote GitHub

This is important as this will allow you to work locally on making code changes to the repository.

## Instructions from Microsoft on how to clone a GitHub repo into VS Code

<https://learn.microsoft.com/en-us/azure/developer/javascript/how-to/with-visual-studio-code/clone-github-repository?tabs=create-repo-command-palette%2Cinitialize-repo-activity-bar%2Ccreate-branch-command-palette%2Ccommit-changes-command-palette%2Cpush-command-palette>

## This is the link to our Project to clone, which will be used in the GitHub cloning steps:

<https://github.com/bg1302023/COSC625Project.git>

# Java JDK to install after VS Code setup

JDK 17 needs to be installed to ensure that all the Spring Boot and LibGDX applications work correctly

## 1) Navigate to: <https://www.oracle.com/java/technologies/javase/jdk17-archive-downloads.html>

## 2) Select the installer based on the OS you want it for:

Graphical user interface, application

Description automatically generated

## 3) Check the Java version at the command line and ensure it says 17.x.x. Open command prompt and type: java –version

Text

Description automatically generated

If other versions exist, continue on to the remaining steps, otherwise you are done installing the JDK

## 4) We need to ensure that JDK 17 is the only one on the machine. Open Control panel and select “Uninstall a program”

Graphical user interface, text, application

Description automatically generated

## 5) Look for the Java(TM) Software Development Kit, ensure that only version 17.x.x is installed. If it is not, uninstall the others.

Graphical user interface, text, application

Description automatically generated

# Running the Spring Boot API locally

## 1) Right click on the “SpringBootApi-Battleship” folder and select Open in Integrated Terminal

Graphical user interface, text

Description automatically generated

## 2) Run the following command in the open terminal: ./gradlew bootRun

Graphical user interface, text

Description automatically generated

Text

Description automatically generated

### This will start the API at: <http://localhost:8081>

Graphical user interface

Description automatically generated with low confidence

### A specific page to navigate to would be: <http://localhost:8081/swagger-ui/index.html>

Graphical user interface, text, application, email

Description automatically generated

## 3) To stop the API, enter ctrl-C in the terminal. It will give you a confirmation to end the batch job, type “Y” and press enter.

A screenshot of a computer

Description automatically generated with medium confidence

# Microsoft SQL server Dev setup

This is the most complex part of the setup as there are several configuration changes that must be made

## 1) Install the Microsoft SQL server Express edition at:

Direct Download:

<https://go.microsoft.com/fwlink/p/?linkid=2216019&clcid=0x409&culture=en-us&country=us>

OR

Look for SQL Server Express on this page

[SQL Server Downloads | Microsoft](https://www.microsoft.com/en-us/sql-server/sql-server-downloads)

## 2) After successful installation you will be presented with a success page, click the Install SSMS button

Graphical user interface, application

Description automatically generated

## 3) Once Downloaded, open the SSMS file and click install

Text

Description automatically generated

## 4) Open SQL Server 2022 Configuration manager, find the TCP/IP protocol for the SQL server and Enable them.

Graphical user interface, application

Description automatically generated

## 5) Then, Open the Properties of the TCP/IP protocol and find all instances of the ipv4 addresses and enable them.

Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated

## 6) Open the SQL Server Management studio you will be presented with this prompt which will have your machine name followed by \SQLEXPRESS. Click on Options

Graphical user interface, text, application, email

Description automatically generated

## 7) under the Connection Properties Tab, Uncheck “Encrypt Connection”

Graphical user interface, application

Description automatically generated

### It should be set to this:

Graphical user interface, application

Description automatically generated

## 8) Connect to the database and Right click on the Connection name and select properties

Graphical user interface, application

Description automatically generated

## 9) Click on Security and change the authentication to SQL Server and Windows Authentication Mode

Graphical user interface, text, application, email

Description automatically generated

## 10) Open Services and restart SQL Server(SQLEXPRESS)

Graphical user interface, application

Description automatically generated

## 11) In the Repo, under the folder Database Scripts, select the file BattleshipDB\_Setup.sql, Copy the contents of the file (ctrl+A and then ctrl+C)

A screenshot of a computer

Description automatically generated

## 12) In Sequel Server Management Studio, create a new query and paste the script in

Graphical user interface, text, application

Description automatically generated

## 13) Select the entire query and click execute

Graphical user interface, application

Description automatically generated

### You should get this as your result:

Graphical user interface, text, application, email

Description automatically generated

# Testing the database connection in Spring Boot

## 1) Open spring boot API following the steps under “Running the Spring Boot Api Locally”

## 2) Click on the /api/databasetest endpoint under the swagger ui page

Graphical user interface, text, application, email

Description automatically generated

## 3) Click on Try out

Graphical user interface, application

Description automatically generated

## 4) Click Execute

Graphical user interface, application, Word

Description automatically generated

## 5) If everything is set up correctly, you should see this result:

Graphical user interface, application

Description automatically generated