Software Design Document (SWDD)
PlayVera

Name (s):
Ivan
Ovcharov

Date: (01/10/2021)

This page is left blank on purpose

TABLE OF CONTENTS

1.0 INTRODUCTION

1.1 Purpose

This software design document describes the architecture and system design of Playvera game content webshop and its functionality.

1.2 Scope

Playvera is designed to target game content creators and game dev enthusiasts, providing a webshop where users may share questions, content and games amongst each other. The software is planned to have favourite games section, most downloaded, greenlighting and forums.

1.3 Overview

This document serves as a description of the functionality and software design for the Playvera webshop.

1.4 Reference Material

https://www.digitalocean.com/community/conceptual_articles/s-o-l-i-d-the-first-five-principles-of-object-oriented-design- SOLID principles

https://en.wikipedia.org/wiki/Interface_(computing)- Interface explanation

https://axios-http.com/docs/intro - Axios explanation

https://condor.depaul.edu/sjost/it231/documents/rest-table.htm#:~:text=Representational%20state%20transfer%20(REST)%20means,of%20a%20client%2Dserver%20architecture – REST AND CRUD OPERATIONS

https://en.wikipedia.org/wiki/Object-oriented_programming - OOP programming

https://en.wikipedia.org/wiki/HTTPS - HTTP security

https://en.wikipedia.org/wiki/Class_(computer_programming) - Class

1.5 Definitions and Acronyms

SDDW: Software Design Document

SOLID principles: S - Single-responsiblity Principle
O - Open-closed Principle
L - Liskov Substitution Principle
I - Interface Segregation Principle
D - Dependency Inversion Principle

Interface: In computing, an **interface** is a shared boundary across which two or more separate components of a computer system exchange information. The exchange can be between software, computer hardware, peripheral devices, humans, and combinations of these.

Axios: Axios is a promise-based HTTP Client for node.js and the browser. It is isomorphic (= it can run in the browser and nodejs with the same codebase). On the server-side it uses the native node.js http module, while on the client (browser) it uses XMLHttpRequests.

REST/CRUD: Representational state transfer (REST) means using the HTTP GET, POST, PUT, and DELETE operations to implement the CRUD operations: REST also assumes. use of a client-server architecture

Object-oriented programming (OOP) is a programming paradigm based on the concept of "objects", which can contain data and code: data in the form of fields (often known as attributes or properties), and code, in the form of procedures (often known as methods).

HTTP security: Hypertext Transfer Protocol Secure (HTTPS) is an extension of the Hypertext Transfer Protocol (HTTP). It is used for secure communication over a computer network, and is widely used on the Internet.In HTTPS, the communication protocol is encrypted using Transport Layer Security (TLS) or, formerly, Secure Sockets Layer (SSL). The protocol is therefore also referred to as HTTP over TLS, or HTTP over SSL.

Class: In object-oriented programming, a **class** is an extensible program-code-template for creating objects, providing initial values for state (member variables) and implementations of behavior (member functions or methods).

JSON Web Token is a proposed Internet standard for creating data with optional signature and/or optional encryption whose payload holds JSON that asserts some number of claims. The tokens are signed either using a private secret or a public/private key.

React - **React** (also known as **React.js** or **ReactJS**) is a free and open-source front-end JavaScript library for building user interfaces or UI components.

2.0 SYSTEM OVERVIEW

The functionality of Playvera consists of a backend (developed using Java Springboot) and React as the front-end UI. The backend has functionality for retrieving games, creating new content, person information and email management. Admin functionality includes managing application users and game content that is uploaded on the website.

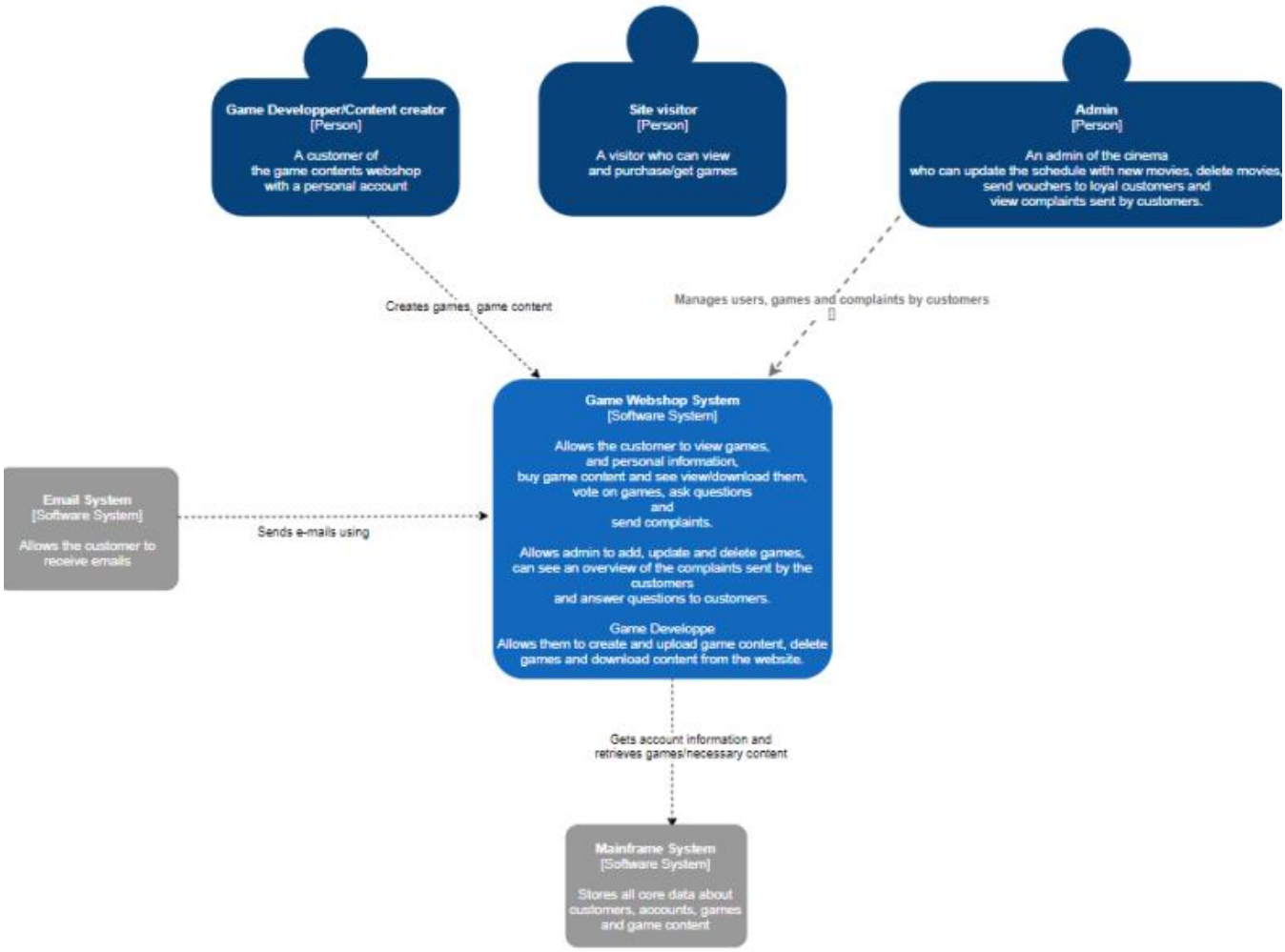
3.0 SYSTEM ARCHITECTURE

3.1 Architectural Design

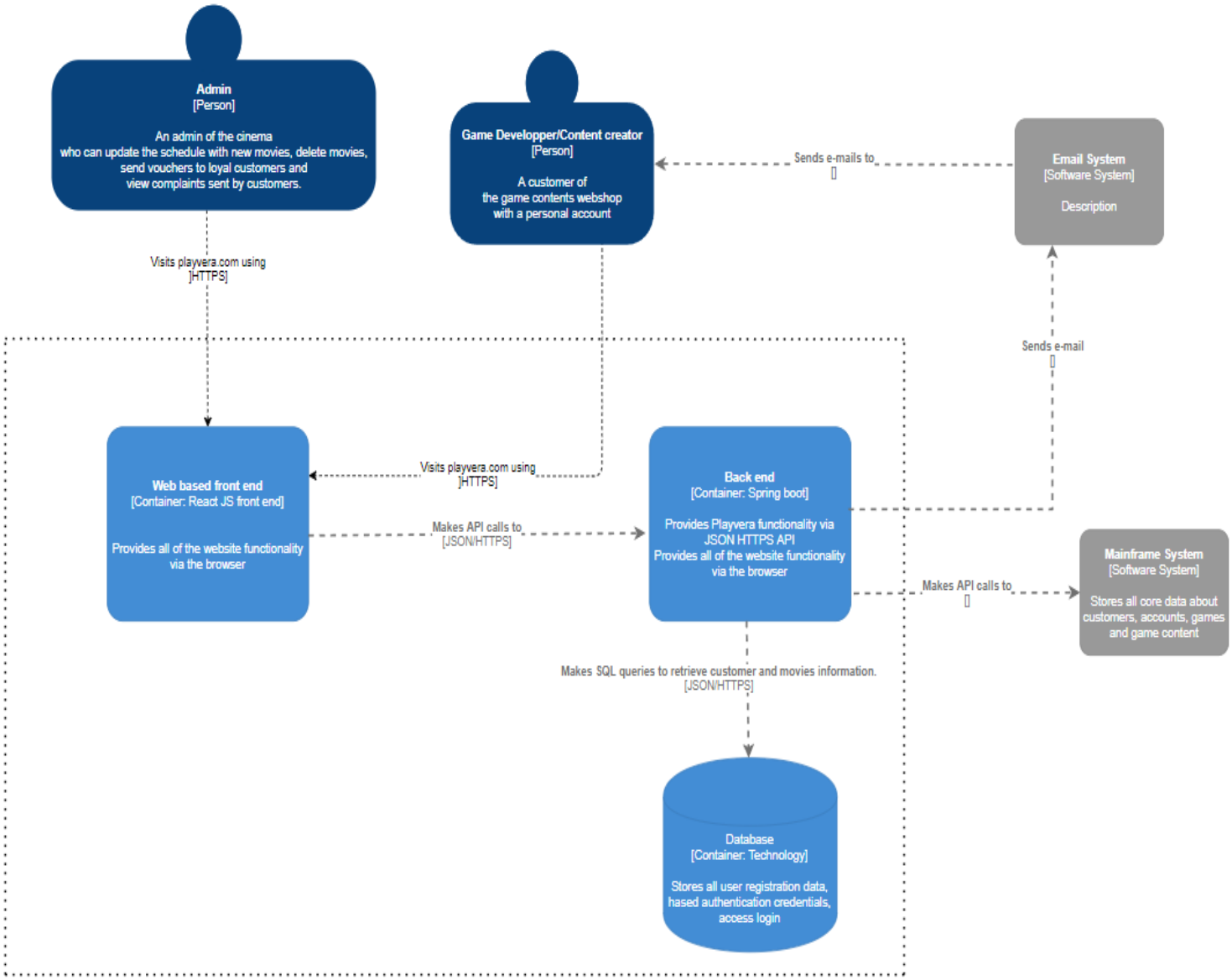
Playvera is developed by strictly following SOLID principles and ensuring that security is provided to the users. It uses interfaces to connect to the database layer, ensuring that there is no redundancy and repetition in the implementation. The backend of the application is connected with the front-end using HTTP requests with Axios to connect to REST endpoints, allowing the performance of CRUD operations.

3.2 Decomposition Description

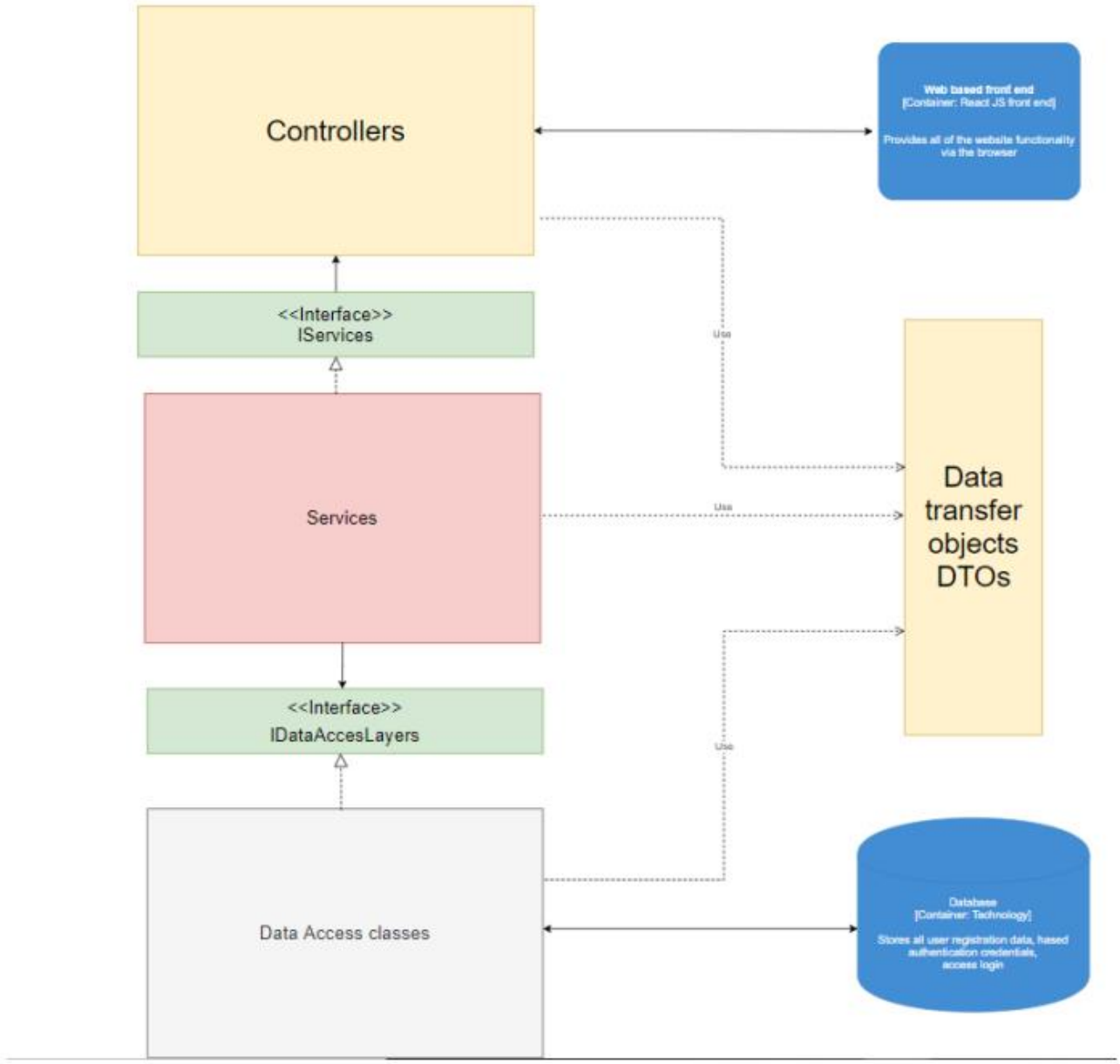
Level 1



Level 2


Admin
[Person]
An admin of the cinema
who can update the schedule with new movies, delete movies,
send vouchers to loyal customers and
view complaints sent by customers.
Game Developper/Content creator
[Person]
A customer of
the game contents webshop
with a personal account
Sends e-mails to
[]
Email System
[Software System]
Description
Visits playvera.com using
]HTTPS]
Sends e-mail
[]
Web based front end
[Container: React JS front end]
Provides all of the website functionality
via the browser
Visits playvera.com using
]HTTPS]
Makes API calls to
[JSON/HTTPS]
Back end
[Container: Spring boot]
Provides Playvera functionality via
JSON HTTPS API
Provides all of the website functionality
via the browser
Mainframe System
[Software System]
Stores all core data about
customers, accounts, games
and game content
Makes API calls to
[]
Makes SQL queries to retrieve customer and movies information.
[JSON/HTTPS]
Database
[Container: Technology]
Stores all user registration data,
hased authentication credentials,
access login

Level 3


Controllers
Web based front end
[Container: React JS front end]
Provides all of the website functionality via the browser
<<Interface>>
IServices
Uses
Services
Uses
Data transfer objects DTOs
<<Interface>>
IDataAccesLayers
Uses
Data Access classes
Database
[Container: Technology]
Stores all user registration data, hased authentication credentials, access login

4.0 DATA DESIGN

4.1 Data Description

All of Playvera's data is stored in a database that contains the following data tables:

-AppUser: Containing all of the user's information(email, password, username, id)

-Game: Containing all of the necessary information of a user created game

-TypeOfGame: Containing all of the types of games in the web application

-Role: Containing all of the roles a user may have in the application (admin, user)

-GameRating: This is used to connect a game and its rating. Once a game is deleted, the game rating linked using its' id is also automatically removed.

4.2 Data Dictionary

The following methods are used to retrieve/upload game and user data in the data storage layer of the application.
Each of the methods contains a simple explanation of its functionality in the given context.

```
Game saveGame(Game game); //Saves a game using a game instance
TypeGame saveTypeGame(TypeGame typeGame); //Saves a type of a game
void addTypeGameToGame(String gameName,String typeGame); //Adds a type to a
game
Game getGame(String gameName); //Retrieves a game using its name
List<Game> getGames(); // Retrieves all of the games in the database
```

```
AppUser saveUser(AppUser user); //Saves a user in the database using an
instance of the user itself.
Role saveRole(Role role); //Saves a role into the database(This is
preconfigured by the company itself.
void addRoleToUser(String username,String roleName); //Used to add a role to a
specific user, using it's username and the role name.
AppUser getUser(String username); //Retrieves a user given his username
List<AppUser> getUsers(); //Retrieves all users in the database
```

5.0 COMPONENT DESIGN

In this section, we take a closer look at what each component does in a more systematic way. Playvera's functionality and way of development is done using OOP. The following are objects that reside in the backend functionality of the webshop:

AppUser – An app user consists of an id, username, email, password and a collection of roles(admin rights, game developer rights and user rights)

Game – A game consists of id, game name, game type, size, rating and price.
Role – The roles are given to different users, providing them access to the application using HTTP security.
TypeOfGame – The type of game object is assigned to a game. A game may have multiple types assigned to it.

The security component of the application is done using the HTTP security class in Java. It consists of two sections:

1. A custom authentication filter that ensures a proper JWT token is given to the right user, creating a security layer.
2. A custom authorization filter that makes sure the proper token is handled correctly and a user may login using his credentials securely.

6.0 HUMAN INTERFACE DESIGN

6.1 Overview of User Interface

The user interface of the application is fully developed using React, using JavaScript with HTML and CSS For the styling. It consists of custom built components that represent the following functionalities:

Navigation bar: Used for website navigation and connection to all of the other visual components in the application

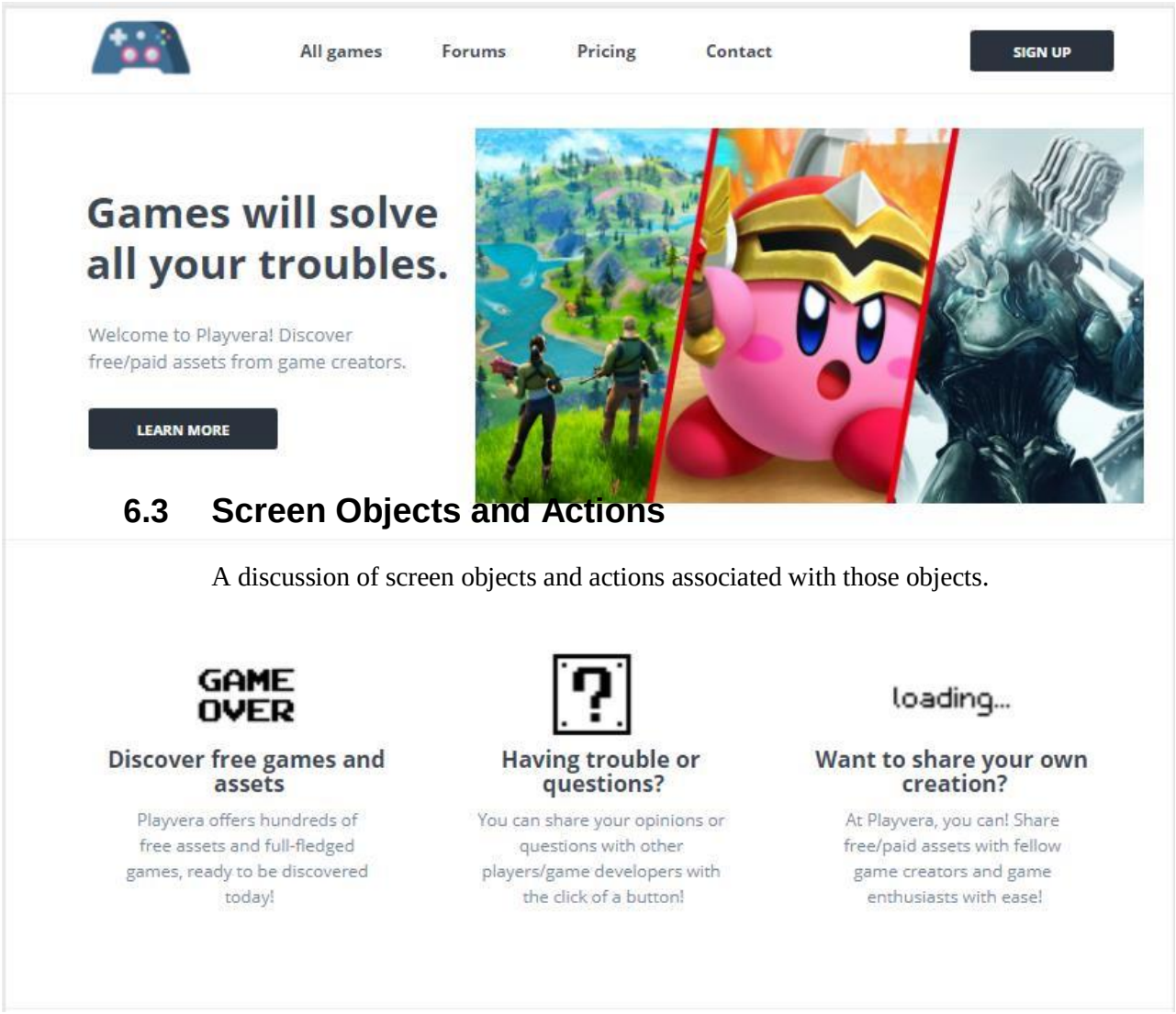
Admin panel: Used for retrieving user information, deletion of users and correcting information/general statistics

Game component: Represents a game where a picture may be uploading, pricing, size and a short description. A user may also view the rating of a game.

Comments component: Represents user-written comments that may be filtered and deleted by the administrators of the application.

6.2 Screen Images

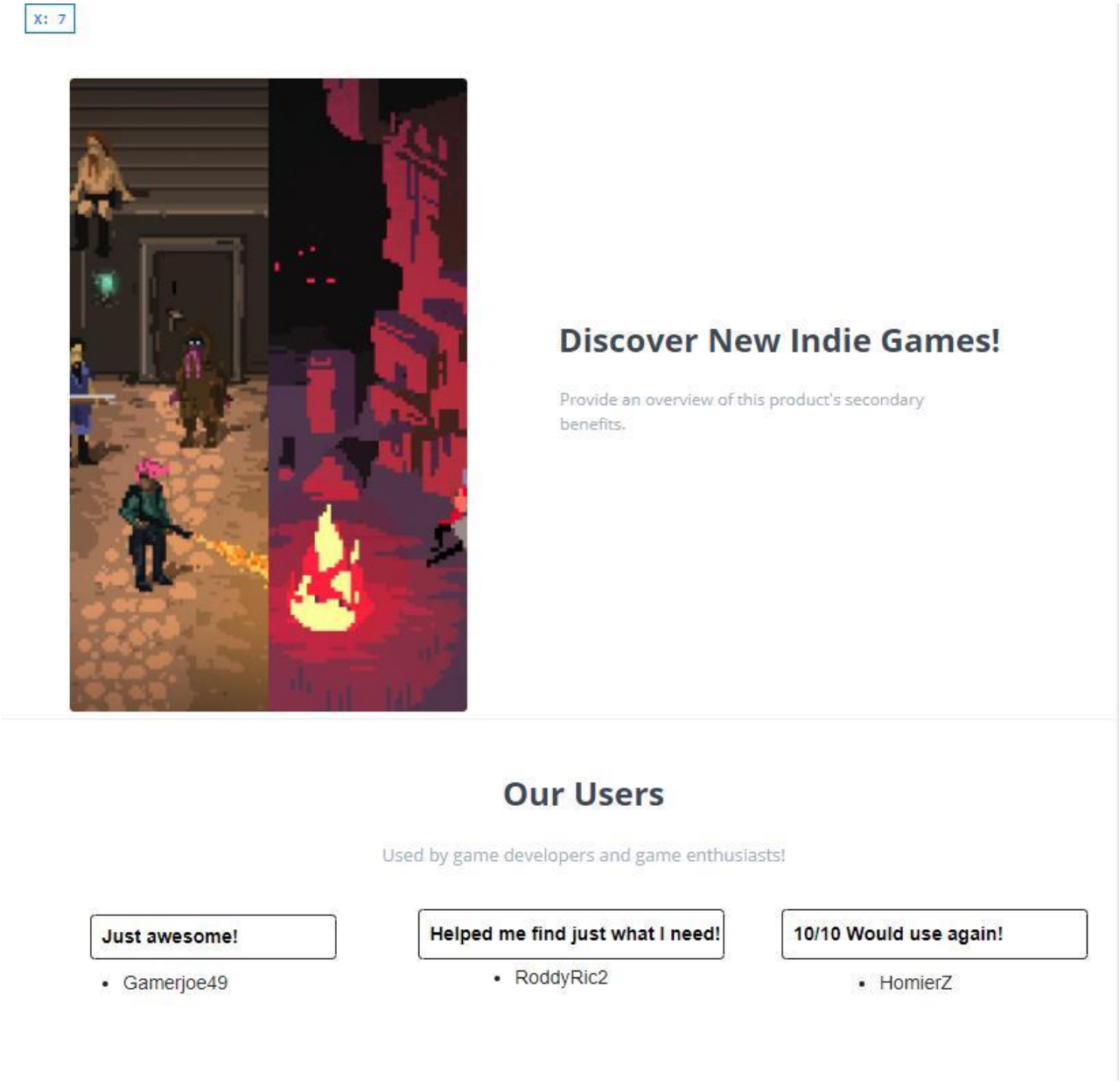
Component 1: Front page nr1. A user may sign-up/log-in, view all of the available games, forums and contact information.



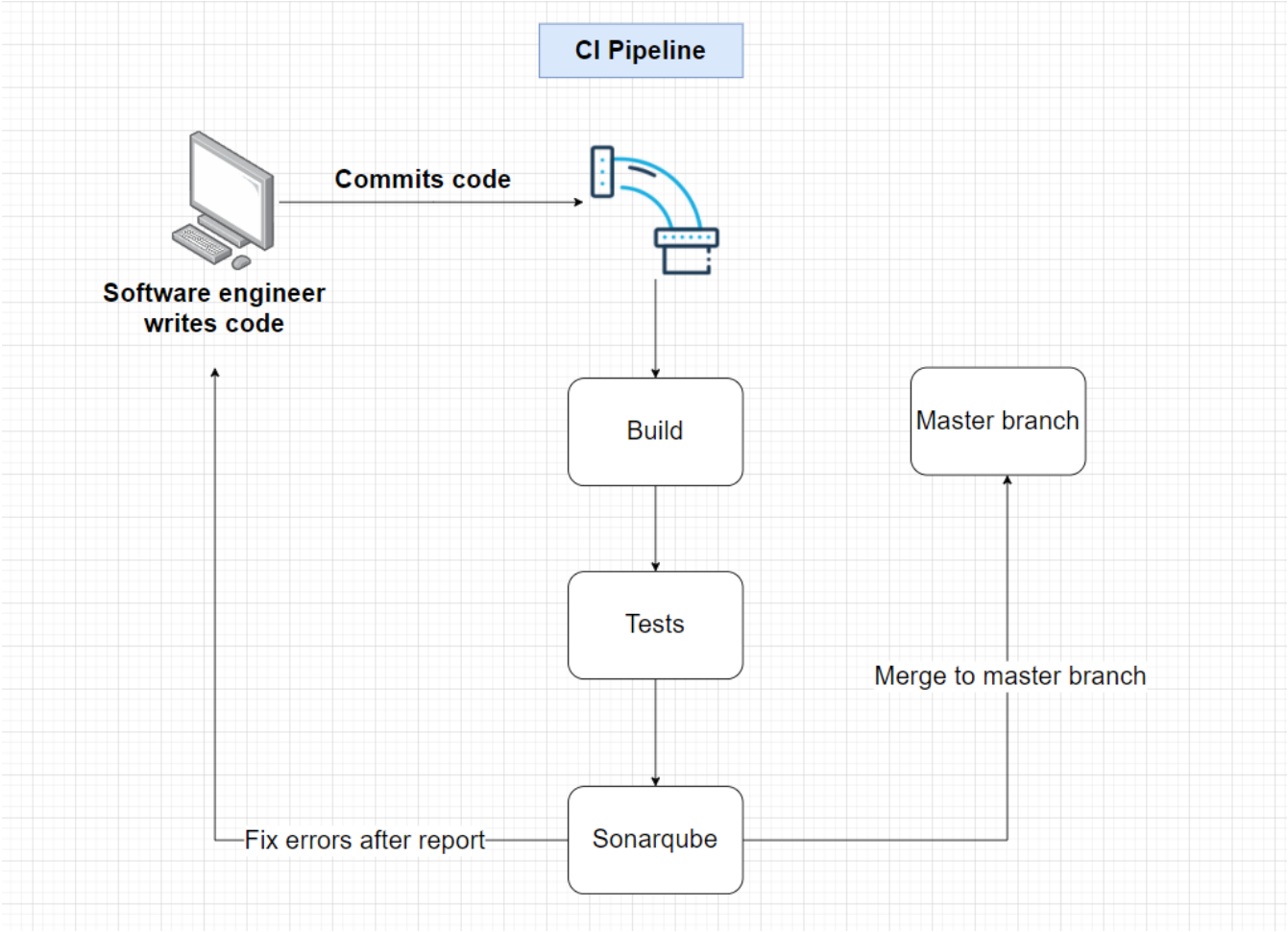
6.3 Screen Objects and Actions

A discussion of screen objects and actions associated with those objects.

Component 2: Front page nr2. A user may view different user reviews. A clickable component is added that is linked to suggested games based on user plays/downloads.



7.0 CI-CD Diagram



CI-CD Pipeline is integrated with sonarqube in order to have a clearer view of how testing works and manage to fix errors with greater ease. It also allows for running different statistics and tests on your application/code to find possible data/security leakage and prevent further mistakes.

8.0 Sonarqube report