

**Software Design Document**

**for**

# Minigame Website

**Version 1.0 approved**

**Prepared by Andrew Fales, Jacob Lin, Selena Sat, Tyler Anton**

**Instructor: Gordon Gul1**

**Course: CS410 Software Engineering**

**Date: April 17th, 2024**

## Table of Contents

## Revisions

## 1 Introduction

## 1.1 Purpose

## 1.2 Scope

## 1.3 Document Overview

## 1.4 Definitions, Acronyms and Abbreviations

## 1.5 References and Acknowledgments

## 2 System Overview

## 3 System Architecture

## 3.1 Architectural Design

## 3.2 Design Decomposition

## 3.3 Design Rationale

## 4 Data Design

## 4.1 Data Description

## 4.2 Data Dictionary

## 5 Component Design

## 6 Human Interface Design

## 6.1 Overview of User Interface

## 6.2 Screen Images

## 7 Requirements Traceability Matrix

## Revision History

| **Primary Authors** | **Date** | **Reason For Changes** | **Version** |
| --- | --- | --- | --- |
| Tyler Anton, Andrew Fales, Selena Sat,  Jacob Lin | 04/17/24 | Initial version | 1.0 |
| Tyler Anton,  Andrew Fales,  Selena Sat,  Jacob Lin | 05/06/24 | Mid-quarter project scoping | 1.1 |

1. INTRODUCTION
   1. Purpose

This software design describes the system design and architecture of the Mini Games project.

* 1. Scope

Our project is a mini game website which includes a user account system and high score table. Our goal is to entertain users and provide a persistent database tracking user account information. The benefit of our project is accessibility to the user including the ability to play games with or without an account.

1.3 Overview

We are a team of students enrolled in Bellevue College’s computer science program. This document provides a project overview of our mini game website, including design considerations, decisions and its implementation structure. In addition, this document will detail our references for the project as well as our plans for the scope maintainability of the minigame website.

* 1. Reference Material

N/A

* 1. Definitions and Acronyms

SQL - Structured Query Language, a coding language to organize and parse data.

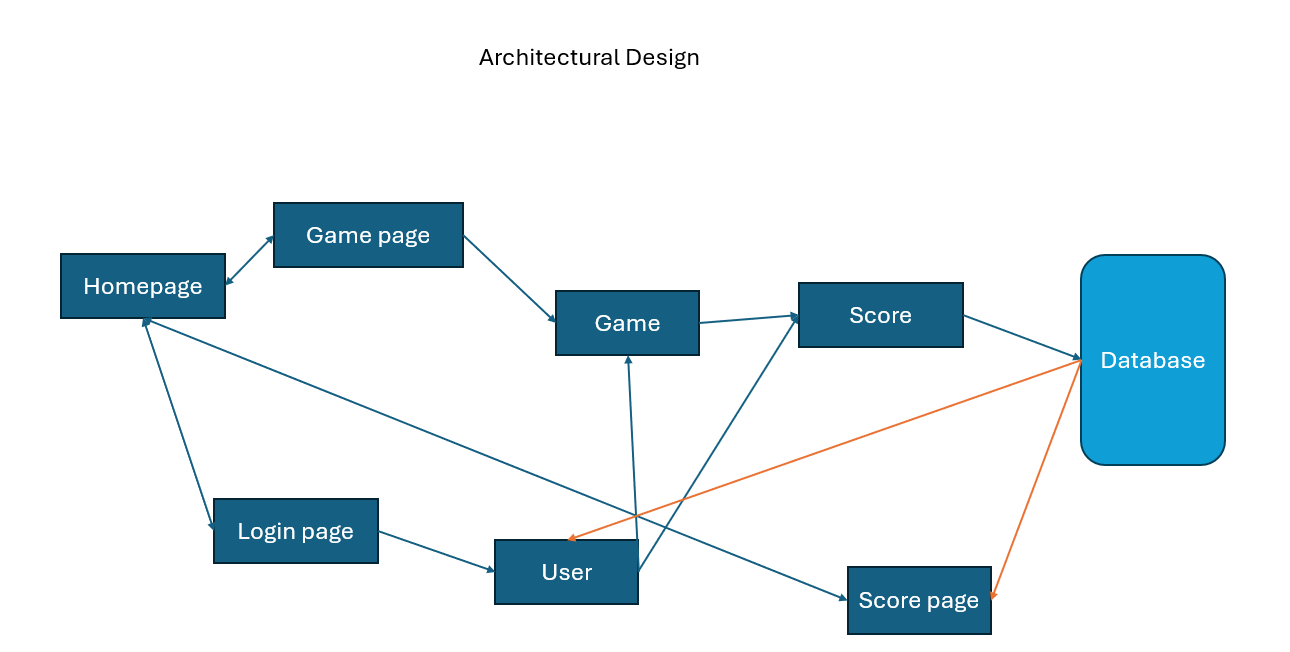
PHP - PHP: Hypertext Preprocessor

Cloud hosting - Using a company to host our website and database on the internet.

1. SYSTEM OVERVIEW

Retro inspired themed website which will include the typical gaming components such as a leader scoreboard, friends list, retro mini-games/browsing (minesweeper/dino run/tetris/pacman), log in/log out, user registration(username/password), contact page (? for bugs).

1. SYSTEM ARCHITECTURE
   1. Architectural Design



From the homepage you can go to the game page or login page or high score page.

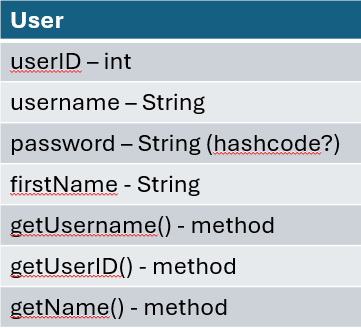
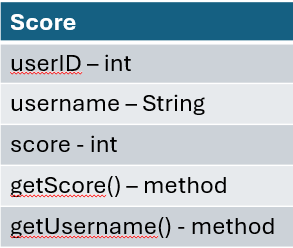
From the game page, you can access the game. The game takes the logged in user and the score they generate and creates a score object that will then be sent to the database.

From the login page, you can log in, accessing the database to pull from and create a user object to be used for the whole website.

Accessing the high score page will pull the top score objects from the database.

We are going to use standard html/css for our frontend website without the use of a web app framework like Angular or React. For building our minigames, we are going to use Godot due to its ease of use and export to HTML5 function, so we are depending on any libraries needed for things such as Pygame. We will use a simple backend, with robust data handling so we will be using a SQLite3 database to handle user and score data. The website/backend will be hosted using Amazon AWS EC2 and S3 services. We will code our user account system using PHP and Javascript.

* 1. Decomposition Description

See 3.1 for additional design decomposition diagram.

* 1. Design Rationale

Our architecture was selected for simplicity and ease of use for the user. Critical issues considered were user experience, lightweight web programming, scalability and supported devices/browsers. Some trade offs considered were lightweight vs scalability and device support vs project maintainability. We decided to target getting a responsive website built rather than focus on scalability from the ground up, due to the time limitations on the project. In addition, we are focusing on supporting modern browsers on desktop only so we can keep the project maintainable and lightweight. NoSQL databases were considered but traditional relational databases were chosen for ease of implementation and straightforwardness. Particularly SQLite3 was chosen because it is designed to be lightweight. Unity and other paid game dev architectures were considered but not chosen due to the learning curve, cost, heavyweight nature, and difficulty running natively as a web application.

1. DATA DESIGN
   1. Data Description

The database will consist of two tables: the User table, and the Score table. The User table will have the fields: UserID (primary key), Username, Password, FirstName. The Score table will have the fields: Game, UserID (hybrid key), score (hybrid key).

These records of these tables can easily be fetched and turned into objects (and arrays), with the objects looking like what is described in Section 3.2.

* 1. Data Dictionary

Score Object:

| Field Name | Data Type | Data Format | Description | Example |
| --- | --- | --- | --- | --- |
| userID | int | int | unique user ID | 100001,  100002, etc |
| username | String | varchar | user chosen username | TheLegend27 |
| score | long | bigint | score from game | 999999 |

User Object:

| Field Name | Data Type | Data Format | Description | Example |
| --- | --- | --- | --- | --- |
| userID | int | AUTO\_INCREMENT  NOT NULL | unique user ID | 100001,  100002, etc |
| username | String | varchar | user chosen username | TheLegend27 |
| password | ??? (research how to store passwords) | ??? | user chosen password | hunter2 |
| firstName | String | varchar | user first name (optional) | Jacob |

1. COMPONENT DESIGN

All member functions described in 3.2 are getters. There are no complex functions that are to be defined.

Pseudocode would look like:

function getFirstName(): return firstName

1. HUMAN INTERFACE DESIGN
   1. Overview of User Interface

The user will come to our website and be greeted by our homepage. From the homepage, there will be links to log in, various minigames, and the scoreboards for all the minigames.

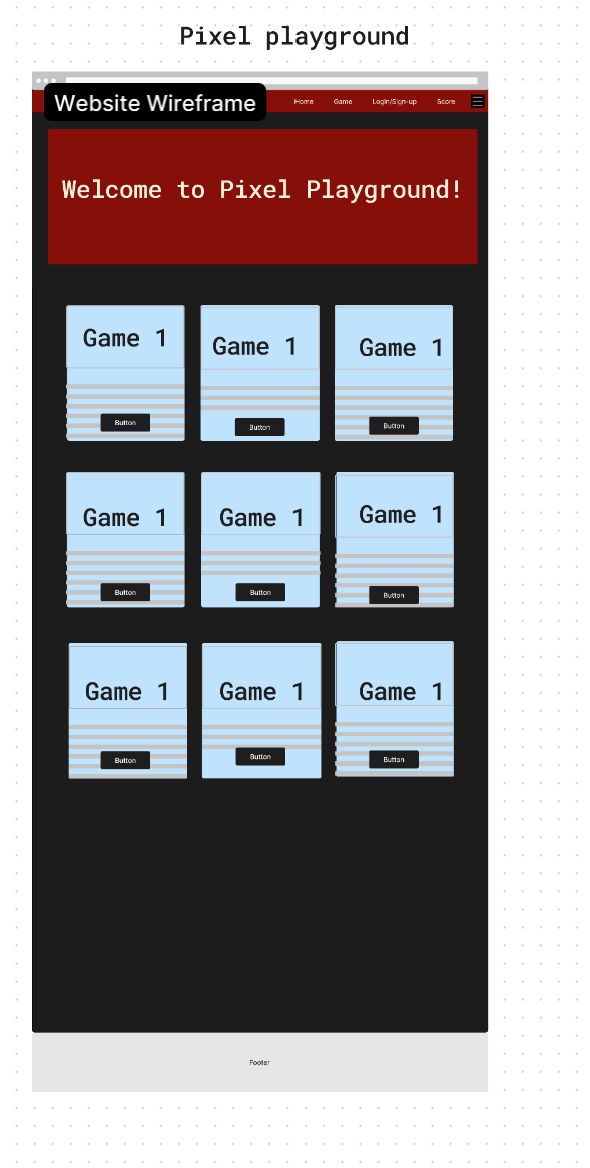
The user will be able to create an account and/or log in from the login page.

After logging in, all scores that the user sets while playing minigames will be recorded, and the user can view their own scores whenever they want.

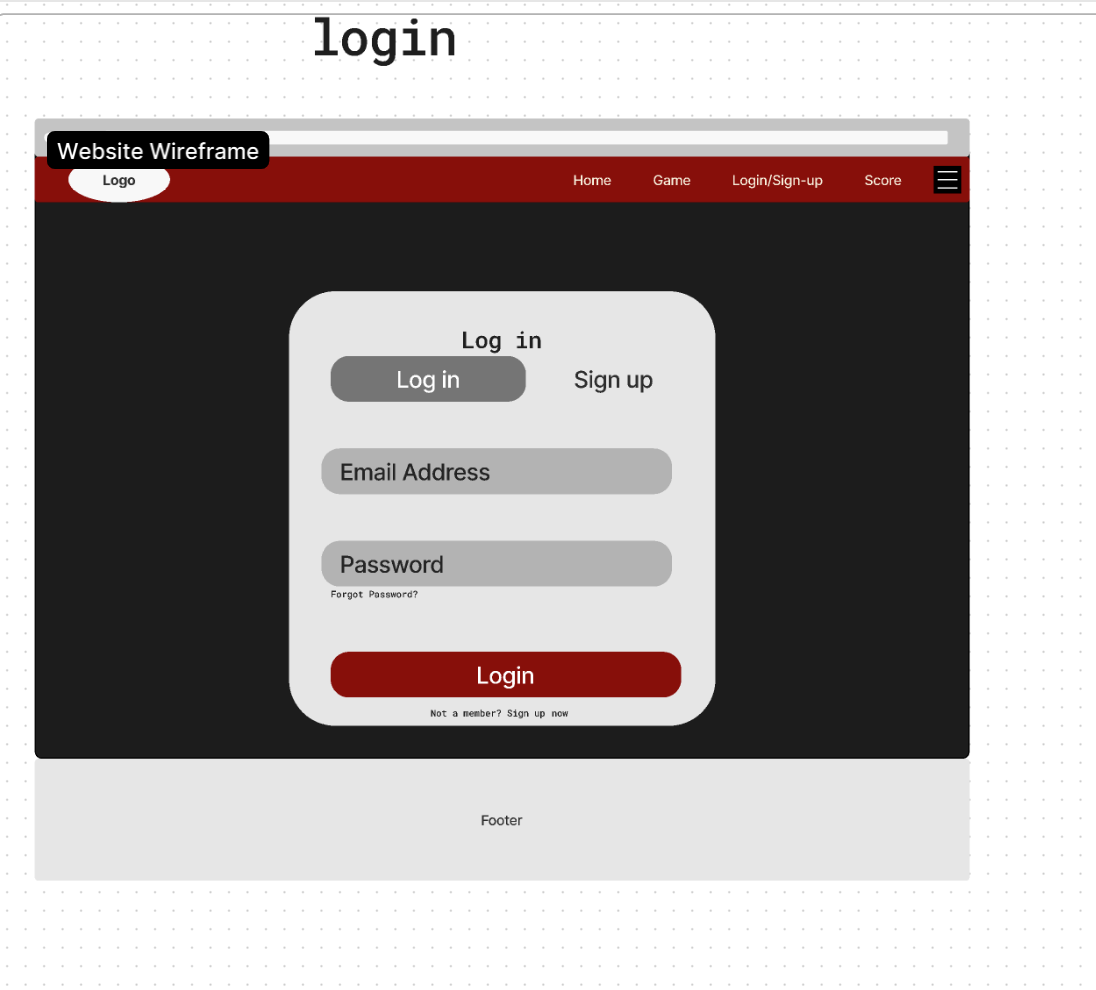
The user can also view where they rank on a global leaderboard, compared to other users.

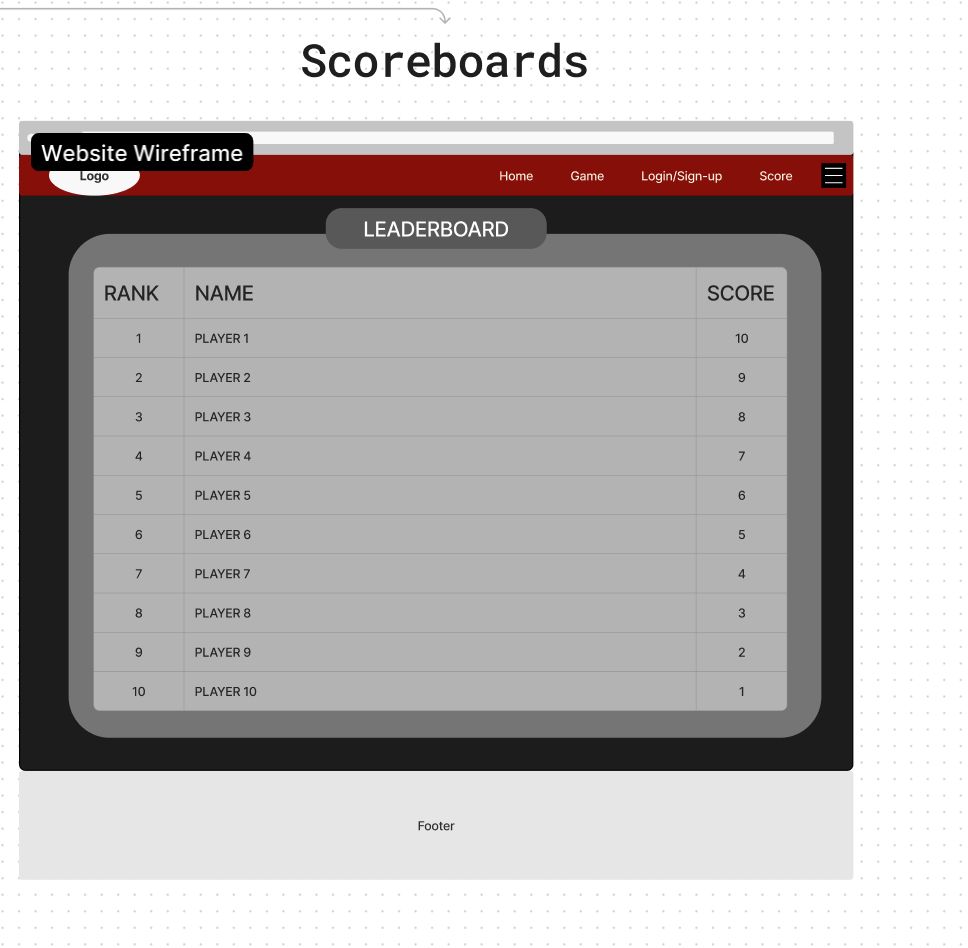
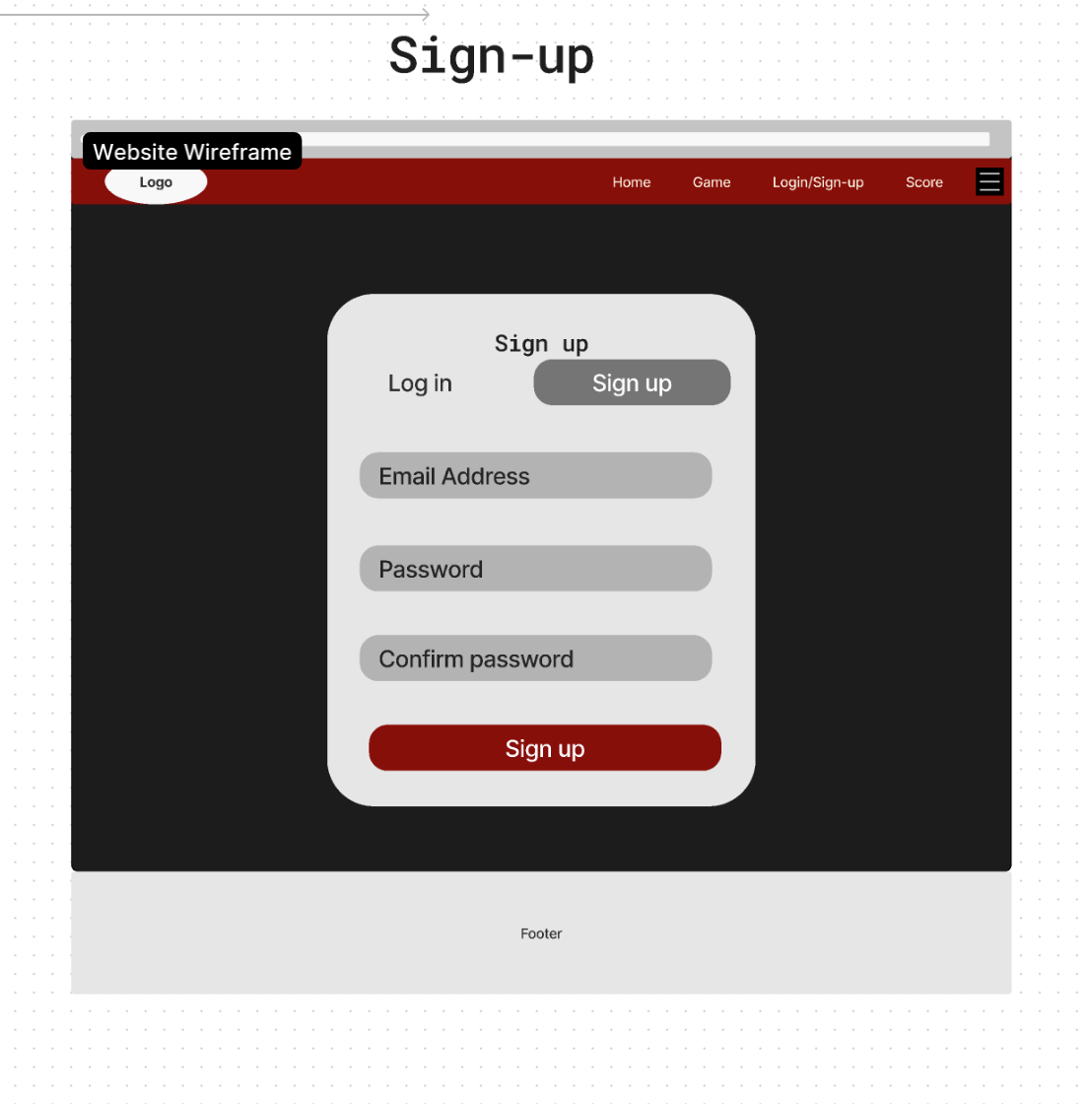


* 1. Screen Images









1. REQUIREMENTS TRACEABILITY MATRIX

| Requirement-ID | Requirement Description | Design  Component | Data Design Component | Interface  Design  Component |
| --- | --- | --- | --- | --- |
| 1 | Lightweight Frontend | 3.1 - HTML/CSS | N/A | 6.2 - HTML Layout |
| 2 | Minigames | 3.1 - Godot | N/A | N/A |
| 3 | User Account System | 3.1 - JavaScript and PHP | 3.1 - SQLite3 Database | 6.2 - Login/Signup |
| 4 | Scoreboard | 3.1 - JavaScript and PHP | 3.1 - SQLite3 database | 6.2 - HTML Frontend |

Revision Blurb:

Reformatted whole document so it’s easier to read

3.1 - Changed framework used for game dev that we decided on after research. Changed cloud hosting to our finalized decision.

Changes were made to reflect finalized decisions from research and also to reflect changes made to requirements specification document.