

Golden Royal Casino – V 1.0

The game concept is simple. One can bet on any given number and can win 10x the bet. Once can bet on multiple numbers to increase the chance of winning.

How to play?

Select any chip that available i.e. 5, 10, 20, 25, 50 chips. They are multiples of 1000 coins.

Click on the number on which you want to place a bet.

Maximum 5 places you can bet.

Maximum 150 chip value or 150000 coins value you can bet.

Then click the spin button.

After payout. You can re-bet on same place by clicking rebet button.

If you want to change you can clear the bets and change your bet.

You can turn on/off the music if you want by clicking music button.

What about Back-End development?

PHP with Composer package manager is implemented for easy future updates for the packages.

Backend is handling the payout logic for the game after validating the request on every spin. It validates data, JWT token and then generates a number which is mapped to another random array for security.

Why this game?

This game is developed as a template to create as many games as possible in the same lines or on other categories of games which allow restricted access to game play. This template has following features and improvements are on the go. This game is built on PWA concept.

Features covered:

1. Security at high priority for real money game play
2. PHP development with composer module loader
3. JSON validator for PHP, validates request object against JSON schema
4. Restricted XSS attacks
5. Restricted SQL injection using PHP PDOs
6. Phaser 3 game
7. PWA with dialog to install app in local
8. JWT for authentication for APIs which restricts CSRF
9. Login

10. Register
11. Restricted access
12. Webpack build setup to create minified chunks

Why PWA?

Progressive web apps are the future to distribute updates as and when there is an updated version of the game. The updates are pushed to client as they connect to the game or app.

What about Security?

Security is being taken as the top most priority for the application, keeping in mind the real money game play. This app has PHP as backend and MySQL as its Database in v 1.0. In future the support of MongoDB and NodeJs will be developed.

In PHP we have PDOs to query the database which makes the database integrity and secure. The request is sent to server through POST and JSON format, the request object is encrypted. On backend we have JSON Validator which validates every request. Stops XSS, CSRF, SQL Injection, etc. through multiple levels of security checks.

JWT is being implemented for user authentication through token systems.

APIs are generated for user registration, login and game access.

Frontend security is also being taken care by using Webpack bundling system, base64 encoded requests to APIs, game access only after user registers and logs in.

What about the game framework?

The game is developed by using HTML5, JavaScript and Phaser 3 framework. Graphic elements are being created as sprite sheet and being used. Animations are in Phaser code.

Where is the payout logic of game implemented?

Payout logic of the game is implemented at backend. This game has a bank balance which is the JACKPOT amount. Every time a user is winning, the payout is being made from bank. Every time user loses the amount is being added to the bank. This way the JACKPOT amount is being updated.

How is the payouts being calculated?

This is the secret. At backend, we have many parameters which is being considered to generate the random number on which the payout is made. It checks for bank balance for the game, checks for user consistency, bet amount, coins values, and much more based on which the payouts are made.

Can I integrate this game for real money game play?

Yes. If you have all permissions from required authorities, you can launch the game as a real money game play. Payment gateway integration is still pending in v 1.0. Very soon will integrate PayPal payment gateway. Later on demand can be added more providers.

How about logs?

Every transaction from frontend to backend has been logged into a database. So if any user is reporting any issue, the admin of the game can go and check the logs where is the issue being aroused.

What can you expect from us?

We are there to support you through 24x7. For any kind of feature requirements, you can drop a message or a comment. We are ready to help you with in your budget. We are ready to setup multiple games with this framework for your next big online presence.

What assets I will get?

The layered PSD, the game source, the Database SQL queries, PHP code for the backend, end to end help document for the developers. All code well commented.

Developer Guide

IDE used for the development

1. Visual Studio Code
2. Sprite sheet of images are being created using free TexturePacker app using <https://www.codeandweb.com/tp-online/index.html> or <http://free-tex-packer.com/app/>
3. Photoshop for the graphics
4. WAMP server for PHP development with MySQL
5. Used Postman to test the APIs functionality

Local setup

To setup the development environment locally, please follow the below steps

1. Front-end
For front end of this game, we use Phaser 3 framework. The game logic is written in game.js file.
2. Backend
Backend is written in PHP. Used composer for package manager. In local I am using [WAMP server](#) to test all functionality.
3. Database
Used MySQL as the database and its local setup is easy with WAMP

Running the project locally

- Unzip the files and from the location of server folder, right click and open integrated terminal in Visual Studio Code or in window address bar, delete the file path and type cmd which opens command line terminal, and run composer command to install all PHP dependency using ``composer install``.
- From the location of package.json, right click and open integrated terminal in Visual Studio Code or in window address bar, delete the file path and type cmd which opens command line terminal, install all packages for front end by running command ``npm i``.
- Then run ``npm start`` in command line, You are ready to see the app in localhost:8080.
- To enable the API calls for register, login, spin, you must run WAMP server in local. Run the given ``db.sql`` file in MySQL database import tab by following the steps here (<https://www.digitalocean.com/community/tutorials/how-to-import-and-export-databases-in-mysql-or-mariadb>).
- Check the database access credentials that you setup for the DB. Restart WAMP server if needed. Same database credentials must be updated in the server/api/config/database.php file.

Now in the src/scripts/game.js, you must update the API paths where it says `http://localhost/Phaser3/Complete-Project/server/api/***.php`. Locate the API path where it is being hosted in WAMP server. If you are running it in localhost, follow the steps given below in **Locating to localhost**.

- Now from the game in browser, enter 't' as email and 't' as password as it is included in the db.sql file or it is in the db users table then click login. If you need to use another login please register.

Modifying the theme

To modify the theme or design, I have added the individual images which can be recreated and generated the sprite sheet again by using any of the given links above.

Deploying on server

Once your development is ready, you then need to take it to online by hosting your application online in a server. To do this please follow the following steps,

1. Run `npm run build` command from the package.json file location from your terminal.
2. The dist folder is being generated
3. This folder need to be copied to your server.
4. No need to include any other source code on prod server.

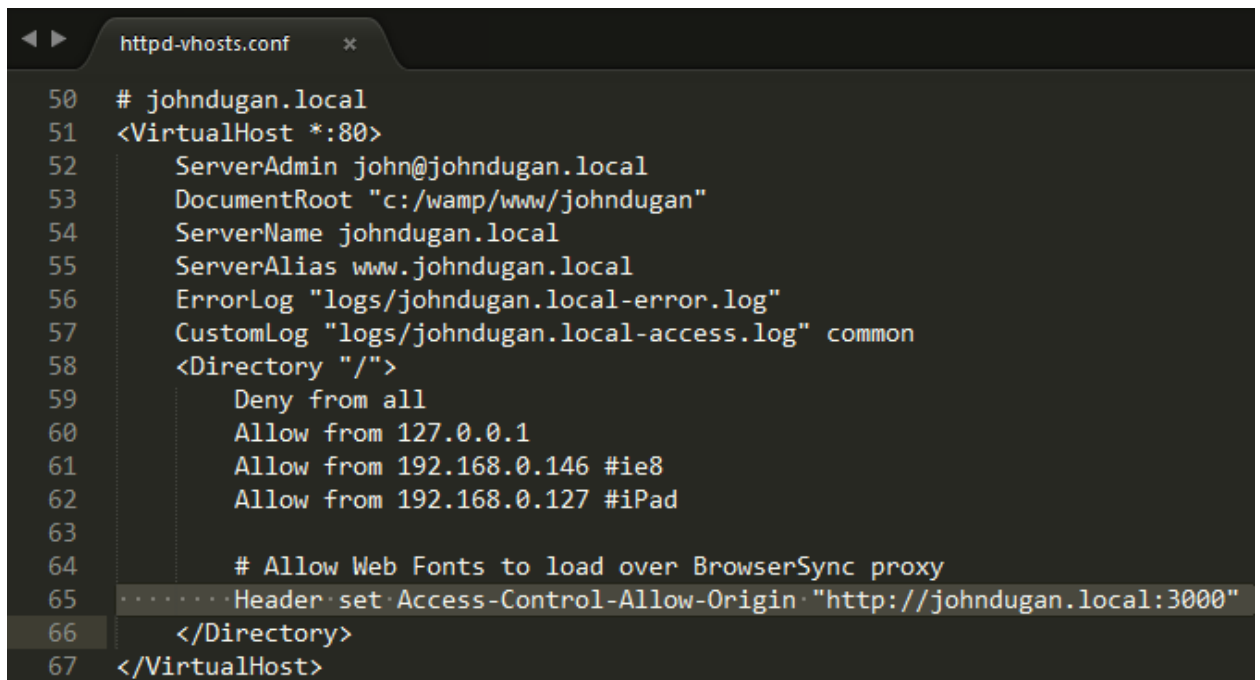
Locating to localhost

As we are using localhost:8080 for the game while developing and the database is in localhost, a different port for your APIs, then we must allow CORS in localhost/on server for cross domain requests in WAMP.

1. Enable the **headers module** in Apache.
 - If you are running WAMP, enable the headers module by selecting the proper option in the Apache Modules section of the WAMP menu.



Add the `Access-Control-Allow-Origin` header directive to all HTTP responses for your virtual host(s).

A screenshot of a code editor showing the configuration file httpd-vhosts.conf. The file contains a VirtualHost block for johndugan.local. The configuration includes settings for ServerAdmin, DocumentRoot, ServerName, ServerAlias, ErrorLog, CustomLog, and a Directory section with Deny and Allow directives. A comment indicates that Web Fonts should be allowed to load over BrowserSync proxy, and a corresponding Header directive is present. The file is displayed with line numbers from 50 to 67.

```
50 # johndugan.local
51 <VirtualHost *:80>
52     ServerAdmin john@johndugan.local
53     DocumentRoot "c:/wamp/www/johndugan"
54     ServerName johndugan.local
55     ServerAlias www.johndugan.local
56     ErrorLog "logs/johndugan.local-error.log"
57     CustomLog "logs/johndugan.local-access.log" common
58     <Directory "/">
59         Deny from all
60         Allow from 127.0.0.1
61         Allow from 192.168.0.146 #ie8
62         Allow from 192.168.0.127 #iPad
63
64         # Allow Web Fonts to load over BrowserSync proxy
65         .....Header set Access-Control-Allow-Origin "http://johndugan.local:3000"
66     </Directory>
67 </VirtualHost>
```

Access-Control-Allow-Origin

2. Restart Apache.