Phyton/Django folder structure

Release 2021/10/25

App
conf
dendama
Nginx
requirements

.dockerignore
.env
.flake8
.gitignore
Docker-compose.yml
Dockerfile
Grous.json
Manage.py
README.md

Apps
conf
fixtures
templates

Main app folder:
This is the place where all the flutter files are and templates/views

Main folder for statics and settings.

Json to create the base of the first games as models

I18n_swiftcher

Mobile: all the files in this folder are from the flutter

Room: models/templates for games and different views

Users: models/templates for contact, certification, etc and users.

Utils

Vendors

locale

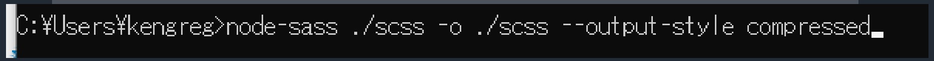
settings → Base.py → Local.py → prod.py General settings for each dev or production environment

Static

Asgi.py
Storage_backends.py
Urls.py this file manage all the urls for the app
Wsgi.py

CSS Font img js pages scss

Don’t use the img folder for images, instead use the bucket.
Only use the scss. You will need to use a console command with a plugin to compile the scss into CSS.



```
C:¥Users¥kengreg>node-sass ./scss -o ./scss --output-style compressed
```

Phyton and environment coding

Release 2021/10/25

Environment variables and settings for the docker container

.env → This file is used to give special variables to each environment. This file is different for the dev/prd environment and **cant not be added to git repository.**

Docker-compose
Dockerfile
Dockerignore

All are files necessary for docket to work and give the parameters/settings for each environment

VIEWS FOR GAMES and general pages

Release 2021/10/25

Dendama/apps/room/api

Serializers.py → This file is used to transform the data from the endpoints(apis) to a Json object which can be handle by python language. It also arrange all the data to be used in the views.py and others.

Urls.py → This file is used to create all the urls for the games. These urls are made using a dynamic id which is created when the user click in a item in the list of games in the flutter.

Views.py → This file is used to create all the views of games. Here you send information from the database to the templates to be used later with javascript.
Also the restriction to enter by user agent in each view. Set the templates for each view here.

Dendama/apps/room/api

Views.py

```
NewsPagination(PageNumberPagination) used to create navigation for the news in the top page.
NewsListAPIView(APIView, NewsPagination) used to create the news
GamesListAPIView(APIView, StandardResultsPagination) this is used to create the different list of
games depending on the tag to be showed in the flutter game page.
RoomAPIView(APIView, StandardResultsPagination) this is used get the correct room for each game for
each player.
RoomDetailAPIView(APIView) this is used to create the details for each game
GamePlay(APIView) used to create the games urls, pass data from database to the template, and
create restrictions to pages looking the useragent.
RankingAPIView(APIView) to get the 30 best rating players
ThreeDApiView(APIView) to create the 3d check dendama view
OnlineLobbyView(APIView) to create the online lobby
ContactAPIView(APIView) to create the contact page
CertificationAPIView(APIView) used to create the certification issue page
HistorialKenteiView(APIView) used to create the history for kentei
TopPageAPIView(APIView) used for domain top page
GamePlayOnline(APIView) used to make the online game
```

In order to create a restriction and don't allow users to see the webapp, we added a condition in the native part to add to the user agent some strings. If the user browser/mobile doesn't send this variables the browser will redirect the user to the axel.Tokyo main page.

```
str("DDOS/Android;") in str(request.META["HTTP_USER_AGENT"]) and str("DENAPP") in str(
            request.META["HTTP_USER_AGENT"]
        )
```

Using the context, you can send information from the backend to the HTML

```
context = {
                "room": room,
                "analytics_code": settings.ANALYTICSCODE,
                "default_locale": room.room_creator.language,
                "locale": room.room_creator.language,
```

Dendama/apps/room/templates

example.html

```
For each template its necessary to use some variables/functions in order to make work correctly
each game.

LANGUAGE:
The I8n plugin use the lang propriety to know which language show to the player.
<html lang="{{ room_creator.language }}">

BASE STYLING:
The base and common styles such as modals are handled using the common.scss
<link rel="stylesheet" href="{% sass_src 'scss/common.scss' %}" />

ANALYTICS BY PAGE:
The analytics code is saved in the .env of each environment.
gtag("config", "{{ analytics_code }}");


Body Wrappers:
gameCover is a hide the content of the page. Usually used when entering to the page. It disappear
as soon as the game is reloaded
Frame. Is a wrapper for the game
Main#after_dark_index inside of this div start the page.
<body onload="" id="horizontally_long" class="w-100 h-100">
    <article class="gameCover w-100 h-100"></article>
 <section class="frame w-100 h-100">
      <main
        id="after_dark_index"
        class="content game after_dark_common w-100 h-100"
      >
```

Dendama/apps/room/templates

example.html

```
JAVASCRIPT NECESSARY VARIABLES AND FILES

In order to show an alert to those who use old version of any OS we need this part with a function.
// 下記はネイティブアプリがバージョンアップする度に変更
      var latest_app_version = { Android: 2.7, iOS: 2.6 }; // 現在のnative app の最新バージョン
      var lowest_app_version = { Android: 2.7, iOS: 2.6 };

defaultLocale = this is used in the games to know the default language used by the user.
Locale = same  as the defaultLocale
Request_from = important variable to know the device OS information
User_agent = user agent information
Request_os_version = os version of the device (it’s a number in string)
Is_ratingGame = this is important to know if the game need to be rated or not (rating system)
Room_creator.id = id of the user in the database
room_creator.name = username of player
Room_creator.language = same as language
room_creator.country = user country
room_creator.classRank = information about the ranking, rating, past ranking, level, and past ppr.
ratingGames = group of games which the player has played which are rated. Each game will have a
history of ppr values, it can only be 10 values each.
Rating = current rating of the player
Kendama_kentei_grade = an abbreviation of the class and level for the kentei exam.
<script src="{% static 'js/jquery-2.2.4.min.js' %}"></script> jquery framework
<script src="{% static 'js/i18n.js' %}"></script> plugin for language
<script src="{% static 'js/i18n/en.js' %}"></script> translation for english
<script src="{% static 'js/i18n/ja.js' %}"></script> translation for japanese
<script src="{% static 'js/ble.js' %}"></script> essential function/variables for ble and dendama
<script src="{% static 'js/common.js' %}"></script> common functions/variables used in many places
<script src="{% static 'js/native_webview.js' %}"></script> functions to comunicate with the native
<script src="{% static 'js/static_data.js' %}"></script> data used for tricks and different games
<script src="{% static 'js/trick_judge.js' %}"></script> data and tricks used in different games
```

VIEWS FOR GAMES: ble javascript

Release 2021/10/25

Dendama/conf/static/js/

Ble.js

```
Variable to know if the app has enabled the bluetooth. Its saved in the SessionStorage.
var is_ble_on = false;

Variable which is saved in the sessionStorage, get the information fromt he native to know which
dendama is connected, the keys are always Main for the 1th dendama and Guest for the 2nd.
var dendama_uuid = {
  Main: null,
  Guest: null,
};

Saved in the SessionStorage. It gets from the native the information about the battery. This is
updated periodically by the native. Also use Main and Guest as keys.
var battery_value = {
  Main: null,
  Guest: null,
};

Saved in the SessionStorage. It gets from the native the information about the hardware (which version
of dendama).
var dendama_hardware_revision = {
  Main: null,
  Guest: null,
};

Saved in the SessionStorage. It gets from the native the information about the firmware (which version
of firmware).
var dendama_firmware_revision = {
  Main: null,
  Guest: null,
};

Default key to search in any of the objects for a Main/guest
var dendama_key = "Main";
```

Dendama/conf/static/js/

Ble.js

```
Variable for random LED actions (actually this need to be confirmed if it is necessary or not to be
deleted).
var random_led_timeout1,
  random_led_timeout2,
  random_led_timeout3,
  random_led_timeout4,
  random_led_timeout5; // ランダムLED発光用のタイムアウトインスタンス
var random_led_timeout_num = 5; // ランダムLEDのタイムアウトインスタンス数
var random_led_mode = false; // ランダムLEDモード中かどうか

Variable used for manage when request battery update information while in the games or in the flutter.
Very important to fix a bug when going from flutter to games and viceversa.

var batteryConnectedMain = false;

Variables used for communication with the native.
var native_trick; // function to start the judge of tricks
var native_sensor; // function to check the sensors
var native_connect; // function to connect the dendama (set the dendama_uuid and other necessary
variables)
var native_disconnect; // function to disconnect the dendama (remove the values in dendama_uuid and
others variables)

To fix a bug and try to control when start/stop the searching for dendama function, we use this
function to stop or allow the function to search work. (this is mostly required for the flutter when
the user click search dendama)
var discoverLoop = false;

Check if the user disconnected the dendama inside a game or other way. Basically this is for special
games such as dendama01
var disconnectedByUser = false;

Get the current version of the app in google play or apple store (currently is not in use but need to
be fixed to request the user to update the app or send notifications to the native)
var currentVersion = null;
```

Dendama/conf/static/js/

Ble.js

This javascript function will take a variable which could be: solo, guest, online
Depending on the variable it will get first if the phone has or not Bluetooth, then to the correct condition. Depending on the condition results, It will show up immediately a modal message with buttons to connect/disconnect dendama.

```
modalConnectionDendama()
```

This function give a template HTML for all modal messages. The structure of the content will depend on the action taken by the user, game and conditions of the game.
Parameters are:
Label = a css class name for the whole HTML section for styling later (height mostly)
Title = title of the modal
Content = this can have any kind of HTML content.
Footer = this can have any kind of HTML content.

```
modalContentTemplate(
      "bluetooth_off",
      `${I18n.t("view.ble.on_message")}`,
      `<article class="desc"><p>${I18n.t("view.ble.on_message")}</p></article>`,
      ""
    );
```

This function takes all the information from the modal Template function to be inserted in the modal container and open it with an animation and size. The first parameter is to style the modal height, etc.

```
modalOpen("main_guest_connected", content);
```

Function to close the modal window

```
actionModalClose()
```

Dendama/conf/static/js/

Ble.js

Function that works when the dendama is already connected. This will check which mode of games is, if it is a ratingame , which rating level was choosed by the player, and if the content of the modal is not the normal (to change height, style or content).
If the condition of "contentDifferent" is false it will close the modal as soon as the player press the button, however if its true it will show another conditions such as "choose level of difficulty" which is used in "Shibuya After dark"

```
//show up the modal to start, true = rating game
// modes could be: guest, solo, online
//options: mode, ratingGame, difficulty/chooseRatingLevel, contentDifferent
modalConnectedDendama("solo", false, true, true);
```

Function used from the native to communicate with the frontend. It tells if the application went to the background.
When it this happened we need to set the "is_background" variable to true.

```
app_pause() // inside we need to use the function game_pause() to execute the required process to stop the game (each game is different).
```

This function is used to let the games or current page know that the app is not longer in the background but in the front.

```
app_resume() // game_resume() is necessary to be used here to let the games execute the required process to start the game again.
```

This function change the variable and storage it in the session. For Bluetooth connection.

```
ble_on()
```

This function change the variable and remove it from the storage. It removes all values of dendama.

```
ble_off()
```

This function is used to tell to the native to disconnect a dendama. It needs a parameter which could be Main or Guest.

```
disconnect_dendama(user)
```

This function is used by the native to tell the front to disconnect the dendama

```
Disconnect()
```

Dendama/conf/static/js/

Ble.js

This function activate the functions inside of the flutter to disconnect a Dendama.
The communication between the flutter and the vanilla/jquery pages is made using an object stringified.

```
js_disconnectedNative(user)
```

A template for send data from the vanilla/jquery pages to the flutter

```
// first value: key of the object, name of the action.
// second value: key of affected item.
// third value: value/action/status
templateJsonSingle("disconnected", "user", user);
```

Function used when the user or app want to start a search and it found a dendama to connect. It will try to show a modal message with action buttons and stop the searching function in the native.
It takes as parameters the string 1, 2
If its different than 1, it will show a message that multiple dendamas are turned on.

```
discover_dendama(count)
```

Function used to show a modal message with a failed status. This happens when the native couldn’t connect to a dendama

```
fail_connect(device_id)
```

Functions used when the firmware or hardware couldn’t be obtained by the native. Currently not used.

```
fail_firmware_revision(device_id)
fail_hardware_revision(device_id)
```

use this when the phone uuid cant be get by the native

```
fail_regist()
```

Tells to the navite to stop the scan for dendama (searching)

```
stopScan()
```

The native tells to the frontend that it couldn’t find any dendama or failed

```
fail_scan()
```

Dendama/conf/static/js/

Ble.js

```
This function is for do something when couldn’t update the battery.
fail_update_battery_level(device_id)

This function used to notify when the update for notify failed
fail_update_notify(device_id, type)

This checks which firmware has been used in the dendama
（形式：type.revision、例："2.0"）
// 1: V1
// 2: V2 通常タイプ
// 3: V2 SAOタイプ
firmware_revision(device_id, revision)


This is used to find the uuid of the phone
This was used to identify the phone and disconnect the application if it was in another phone/device.
This could be used in the renew of the app version 3
found_uuid(phoneUuid)

Function used when the native couldn’t get the phone Uuid
not_found_uuid()

This is used to activate the native gesture (put the ball in a cup longer to be pressed)
gesture(device_id, value)
native_gesture(device_id, value);

This checks which hardware has been used in the dendama
hardware_revision(device_id, revision)

This is used to get different values from the sensors in the dendama such as distance, speed, etc
sensor(device_id, prox, accel, gyro, quat, tama)
It activates the native communication function
native_sensor(device_id, prox, accel, gyro, quat, tama);
```

Dendama/conf/static/js/

Ble.js

This function is for do something when couldn’t update the battery.

```
fail_update_battery_level(device_id)
```

Function used by the native to tell the front that a dendama was successfully connected. The id of the dendama will be added to the dendama_uuid object and activate the hardware revision. Lastly the value will update the sessionStorage dendama_uuid.

```
success_connect(device_id)

native_connect();
```

Register correctly the phone uuid (after it has being saved in the database)

```
success_regist()
```

Function activated when the dendama is successfully connected or disconnected. It starts the evaluation of the sensors and battery update information.

```
 success_update_notify(device_id, type)
```

Function used to control the LED lights. It takes the following parameters:

```
Device_id = the dendama id
Which part of dendama = spike, cups
Color = purple
Others parameters
```

Example:

```
light_led(device_id + "/spike/purple/3/0/20/0");
```

Function to gather the information about the battery

```
read_battery(device_id);
```

Function to change dendamas dynamically to get status, etc. This is used in the checkdendama page.

```
change_dynamic_div(true);
```

Dendama/conf/static/js/

Ble.js

Main function to start judging a trick. Has many conditions and parameters which are going to be used to determinate if the trick is series, second, final, etc and all the different cases.
`trick(device_id, param)`

Function to update the values of battery, it is used by the native to tell to the frontend. It changes the values in the session storage, also communicate this to the flutter.
`update_battery_level(device_id, battery_int)`

Function to activate random LED
`random_led(dendama)`
`random_led_sub(dendama, num)`

Function to read the status of the battery. Communicate with the native.
`read_battery(uuid)`

Start the judge of a trick
`init_judge(dendama)`

Remove notification
`remove_notify(dendama_uuid, type)`

Set moshikame mode. Values of on_off are on or off.
`set_moshikame_mode(dendama_uuid, on_off)`

Function to let the app open an url in a different app browser.
`to_outer_link(link)`

VIEWS FOR GAMES: common javascript

Release 2021/10/25

Dendama/conf/static/js/

common.js

```
Variable used for get from the translation files all the text for each page. Python needs to add this
attribute using the context to the html template.
I18n.defaultLocale = $("html").attr("lang");
I18n.locale = I18n.defaultLocale;

Path of the browsers. This is used to get the status of the webview
const current_path = window.location.pathname;

Variable to see if using online sockets, stop them. Actually this could change with the new online
feature.
var online_socket = null;

Default volume for bgms
var volume = 0.1;

variable to know if the modal has to show single or double select box to choose Rating
var gameSingleChooseRating;

this variable is necessary to show a different content in modal. This show a content such as
difficulty used in the After dark game.
var gameDifficulty;

We need to get the full path of the url to check the domain. Depending on the domain the app will take
the dev or production server.
var apiURL = window.sessionStorage.getItem("apiURL");

This function is used to communicate with the native and let it know which orientation should have the
app
js_nativeOrientation("landscape");

This variable is used only in vertical orientation to tell the function that is vertical orientation
even do it’s a jquery/vanilla page.
var orientationVertical = true;
```

Dendama/conf/static/js/

common.js

Function used to get the size of the screen and update the height/width of the css before showing it to the user.
```
update_responsiveDesign()
```

Function used in the games when the player choose a mode of game and immediately need to start the game
```
startGame()
```

Function to show buttons of retire or going to top of the game
```
actionToTop()
```

Function that activate the audio for the counting in some games
```
countingAudio()
```

Get the youtube tutorial video , used mostly in the kentei
```
get_youtube_id(trick_name, key)
```

Trick which takes the trick that was done by the player and translate it to the current user language either English or Japanese.
```
locale_trick_name(trick_name, name_type)
```

Get the Class Rank abbreviation using the new PPR from a game. The parameter needs to be a number
```
get_rating_info(newRating)
```

Get the current Class Rank information
```
//example: {
      classRank: "C-1-1",
      TLvl: 1,
      classLabel: "beginner",
      rating_class: "C-1-1",
      rating_class_num: "1",
    }
get_current_rating()
```

Dendama/conf/static/js/

common.js

```
Function used to delete tricks which cant be handle by the version2.
delete_v2_tricks(arr)

Function that shuffle an array of items
shuffle_ary(ary)

Function used for HP bars in games
cal_progress(progress_value)

Function to show to the user that the version 1 of dendama cant be used
v1_limit_modal(message)

Function to count the letters in Japanese
charcount(str)

Function to fix Japanese text
truncate(str, size, suffix)

Function to remove the fix of Japanese text
reverse_truncate(str, size, suffix)

Function to stop all bgm and sound
stopAllAudio()

Function to set up the difficulty of a game
gameDifficultyInteger(mode)

Function to go back to the flutter app
toFlutter()

Function to get the information of the arcade card
getArcadeData(id)

Function for effect of ripple in a butoon
ripple(e, element)
```

Dendama/conf/static/js/

common.js

Function to add link to twitter.

```
linkToTwitter(time, difficulty, name)
```

Function to process and get the new data to be saved as rating and classrank for the player

```
processRating(params)
```

Function that makes the calculation of the rating taking in consideration the previous data and the new ppr from the game

```
//1. calculate the new PPR for all 10 times (or less) games
  /* ex:
      calculating_rank({
      "TLvl": 4,
      "pastTG": 11,
      "pastAVGR": 40.82,
      "pastRank": "B-6-4",
      "classRank": "B-6-4",
      "classLabel": "amateur",
      "classLevel": "48,48",
      "currentPPR": "41.92",
      "currentRating": "41.92"
    }, 30.00)
  */
calculating_rank(currentclassRank, params.results.rating_point);
```

Function that works after the rating was saved in the database, it need to show a message telling the user if he get a new level or not.

```
showPostGameModals
```

VIEWS FOR GAMES: static Data javascript

Release 2021/10/25

Dendama/conf/static/js/

Static_data.js

This constant object has all the necessary tricks for the kentei and levels. Also it has all the arcade cards ids.
```
const STATIC_MODEL
```

This constant has all the ids for the tricks videos
```
const YOUTUBE_ID_LIST
```

This constant has all the values for each rating and limits of range
```
const RATING_DATA_2020
```

This constant has all the rating class abbreviation in order as an array to be used in loops
```
const RATING_DATA_2020_ARR
```

This constant is used only when the game has “levels” and we need to loop and get tricks according to levels instead of abbreviation. The game time attack use this.
```
const FILTER_BY_LEVEL
```

This is an array of games which are rating games (need to use an array from database instead of this)
```
const RATING_GAMES_LIST
```

This is the default data for a class rank
```
const DEFAULT_CLASS
```

VIEWS FOR GAMES: trick judge javascript

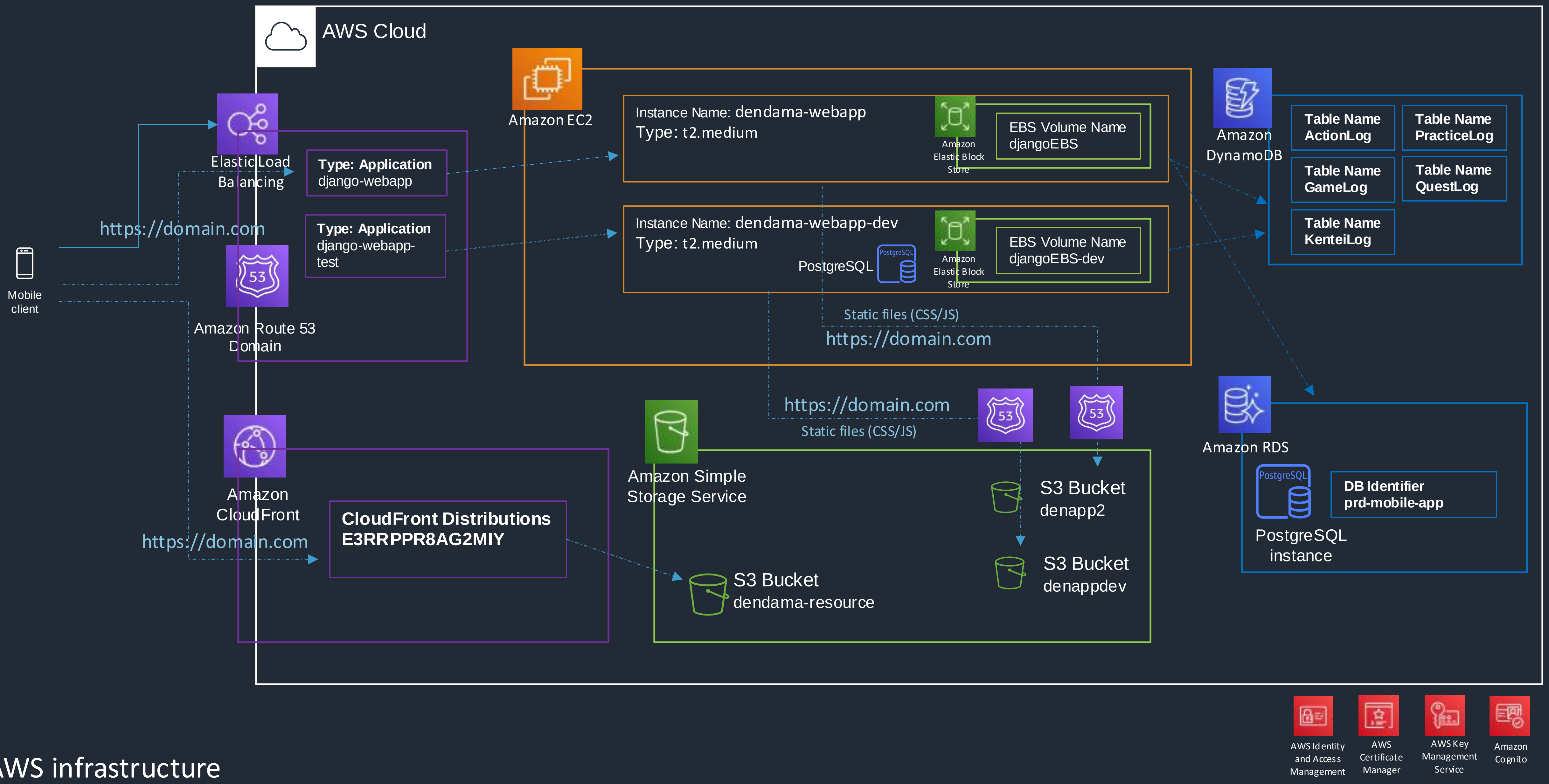
Release 2021/10/25

Dendama/conf/static/js/

trick_jugde.js

AWS infrastructure

Release 2021/10/25



AWS infrastructure

PRD – DEV environment (2021)