DATABASES

USERS DATABASE: SQLite

Table Name		
users		
Column Name	**Data Type**	**Notes**
user_id	varchar	PRIMARY KEY, NOT NULL
username	varchar	NOT NULL, UNIQUE
password	varchar	NOT NULL

GAMES DATABASE: SQLite

Table Name		
guesses		
Column Name	**Data Type**	**Notes**
guess_id	int	PRIMARY KEY, NOT NULL
game_id	varchar	NOT NULL
guess	varchar	NOT NULL
guess_num	int	NOT NULL
Notes Foreign Key (game_id) References games(game_id)		

Table Name		
games		
Column Name	**Data Type**	**Notes**
game_id	varchar	PRIMARY KEY, NOT NULL
user_id	varchar	NOT NULL
guesses_rem	int	DEFAULT 6, NOT NULL
word	varchar	NOT NULL
finished	int	DEFAULT 0, NOT NULL
username	varchar	NOT NULL

Leaderboard: Redis

Table Name	
leaderboard	
key	**value**
score:<username>	List: [<int, score>]
leaderboard:avg	Sorted set: {member <str, username>: score<int, avg>}

POST	**Path: /register**
Description	Allows the user to self-register with web application; all user-supplied information must be checked for correctness. After registered, user will get stored in database. If user exists already or the username or password is too short, returns a 400 status code.
Consumes: application/JSON	**Produces:** application/JSON

Request Model
 	
USERNAME (string, required): *must be a string > len 5.*
 	
PASSWORD (string, required): *sent as plaintext and must be greater than len 5*
	Example JSON:
	
	{
	"username": "user1",
	"password": “password”
	}

Response Model
 	
USER_ID (string, required)
 	
USERNAME (string, required)
	Example JSON:
	
	{
	“User_id”: “asdflkj13f12fas…”,
	“Username”: “user1”
	}

Http Status Code	**Http Response Body**
401 Bad Request	Empty
409 Resource Already Exists	Empty
406 Invalid Param Length	Empty
201 OK	Contains Response Model as JSON

POST	**Path: /checkPassword**
Description	Allows the user to login with their email and password via Basic Authentication. NOTE: Shouldn’t be called as all /game endpoints will automatically call it and perform basic authentication.
Consumes: application/JSON	**Produces:** application/JSON

Request Model
 	
USERNAME(string, required): *must be a string greater than length 5*
 	
PASSWORD (string, required): *sent as plaintext. Must be greater than len 5*
	Example JSON:
	
	{
	"Username”: “user1”,
	"password": “password”
	}

Response Headers **WWW-Authenticate** (optional): *check httpbin for an example. Field filled if user registration fails.*	**Example Headers:** “Www-authenticate”: “Basic realm = ‘Login Required’”

Response Model
 	
AUTHENTICATED(bool, required)
	Example JSON:
	
	{
	“authenticated”: true
	}

Http Status Code	**Http Response Body**
401 Bad Request	Empty
200 OK	Contains Response Model as JSON

POST	**Path: /game/create**
Description	Allows a user to create a new game and add it to the database. It will generate a randomly chosen word and return to the user the unique game id.
Consumes: application/JSON	**Produces:** application/JSON
Request Headers **Username** **User-id**	**Example Headers:** "username": "user1" "User-id": "7f832c2e054ba732f7d4b7e26…"
Request Model **NONE**	**Example JSON:** NONE

Response Model
 	
Game_ID (string, required): *returns unique game id of the newly created game to the user.*
	Example JSON:
	
	{
	"Game_id": "fj4klfklsdlkf23…"
	}

Http Status Code	**Http Response Body**
401 Bad Request	Empty
201 OK	Contains Response Model as JSON

GET	**Path: /game/list**
Description	Allows a user to retrieve their in progress games and lists the game ids back to the user. If the user is not found or there are no games for a given user, returns an empty list.
Consumes: application/JSON	**Produces:** application/JSON
Request Headers **username** **User-id**	**Example Headers:** "username": "user1" "User-id": "7f832c2e054ba732f7d4b7e26…"

Response Model
 	
Games (array[string], required): *returns a list of unique game ids for in progress games for the user.*
	Example JSON:
	
	{
	"Games": ["fj3l3kfj87hgbff", "fjsaswei3235rf123r", "2flkjdsf3f48fjs"]
	}

Http Status Code	**Http Response Body**
401 Bad Request	Empty
200 OK	Contains Response Model as JSON

POST	**Path: /game/guess**
Description	Allows a user to make a guess on an ongoing game. It returns data about the game including correct letters, misplaced letters, and number of guesses left. If the guess is valid, it decrements the guesses remaining.
Consumes: application/JSON	**Produces:** application/JSON
Request Headers **Username** **user-id**	**Example Headers:** “username”: “user1” “user-id”: “7f832c2e054ba732f7d4b7e26…”

Request Model
 	
Game_ID (string, required): *must be a valid, ongoing game id.*
 	
GUESS (string, required): *must be a 5 letter word. API will check for validity.*
	Example JSON:
	
	{
	“Game_id”: “fj243f823fn2f8…”,
	“Guess”: “crane”
	}

Response Model
 	
Game_ID (string, required): *returns the game that is being played.*
 	
VALID (boolean, required): *returns whether or not the guess is a valid 5 letter word.*
 	
GUESSES_REMAINING (int, required): *returns how many guesses are remaining and decrements if they made a valid guess. (6 max)*
 	
CORRECT_GUESS (boolean, optional): *returns whether or not the guess was correct. Only used if the guess is valid.*
 	
CORRECT (array[int], optional): *returns the letter indices that are in a correct position. Only used if they guess incorrectly.*
 	
MISPLACED (array[int], optional): *returns the letters indices that are in the word but in the wrong position. Only used if they guess incorrectly.*
	Example JSON:
	
	{
	“Game_id”: “fj243f823fn2f8…”,
	“valid” : true,
	“Correct_guess”: false,
	“Guesses_remaining”: 4,
	“Correct_guess”: false,
	“corect”: [0,1]],
	“misplaced”: [3]]
	}

Http Status Code	**Http Response Body**
401 Bad Request	Empty
200 OK	Contains Response Model as JSON

GET	**Path: /game/{game_id}**
Description	Allows a user to retrieve the state of a requested game they are currently playing. It returns information of the number of guesses, correct letters and misplaced letters for the guesses made.
Consumes: application/JSON	**Produces:** application/JSON
Request Headers **username** **User-id**	**Example Headers:** “username”: “user1” “User-id”: “7f832c2e054ba732f7d4b7e26…”

Response Model
 	
GUESSES(int, optional): *returns the guesses the user has made with the indexes of the correct letters and misplaced letters. Not used if the game is over.*
 	
NUM_GUESSES (int, required): *returns how many guesses the user has used.*
 	
STATUS (string, optional): *Only used if the game is over. Returns win or lost depending on the game state.*
	Example JSON:
	
	{
	“Num_guesses”: 1,
	“Guesses”:[
	{
	“Guess”: “crane”,
	“Correct”: [0,1],
	“Misplaced”: [3]
	}
	]
	}

Http Status Code	**Http Response Body**
401 Bad Request	Empty
200 OK	Contains Response Model as JSON

GET	**Path: /leaderboard**
Description	Allows any user to get the leaderboard of the top 10 players sorted by their average score.
Consumes: application/JSON	**Produces:** application/JSON

Response Model
 	
Top10(list, required): *Returns a sorted list of max size 10 where each element is a list containing the username and the average score of the user.*
	Example JSON:
	
	{
	
	“Top10”: [
	[“user1”, 4.5],
	[“user2”, 4],
	[“user3”, 2.75]
	]
	}

Http Status Code	**Http Response Body**
401 Bad Request	Empty
200 OK	Contains Response Model as JSON

POST	**Path: /leaderboard/update**
Description	A private endpoint only accessible internally via localhost and not tuffix-vm. Allows the games service to add a new score for a given user and recompute their average to add back into the leaderboard.
Consumes: application/JSON	**Produces:** application/JSON

Request Model
 	
USERNAME (string, required): *username of the player.*
 	
SCORE (int, required): *score of the game the be added for the given user.*
	Example JSON:
	
	{
	"username": "user1",
	"Score": 4
	}

Response Model
 	
USERNAME (str, required): *username of the player requested.*
 	
AVERAGE (int, required): *the newly computed average of the given user.*
	Example JSON:
	
	{
	"Username": "user1",
	"average": 4.3
	}

Http Status Code	**Http Response Body**
401 Bad Request	Empty
200 OK	Contains Response Model as JSON