

Maintenance Document

By:

Pei Wei | Andre Christian | Xinzhuo Yu | Hanwen Wang | Charlotte Katwa



21st October, 2019

Final Version

# 

# Table of Content

[**System Application**](#_8ykd67p6nxhm) **3**

[**Technology**](#_ghv4tl3eecjh) **3**

[**Database Characteristics & Data Sources**](#_3lkm3e7zz2s) **4**

[**Authentication**](#_u9aqs1j9ym8v) **5**

[**System Organisation**](#_8uxbdvllwgn1) **5**

[Frontend/Client](#_moa5lrmg4zld) 5

[JQuery](#_cj8pyt3knc0m) 7

[Authentication](#_u29nllla9vlb) 7

[Game Initialisation](#_hlg7c8y9o56x) 8

[Water/Fertiliser/Pesticide/Scissor](#_5cgb37su95b6) 8

[Backend/Server](#_zecqdl6stpbg) 8

[Server.js](#_4mf1tloitjq5) 9

[Models](#_bejhoh9kh9fj) 9

[Db.config.json](#_pwqf935gzm8) 9

[Routes](#_sbrwyfpe82an) 10

# 

# System Application

Bunny's Garden is a web-based simulator game aimed to help teach children of age group 5-9 years to grow plants in a virtual environment. It generates interest among young children to know and learn about the process of growing plants. The web application allows users to take care of their own virtual plants by using the 4 tools that the user must use and see the progress by which the plant is growing.

The functionalities that we have developed in Bunny’s Garden are:

* Allows persisting of user data in the server without the need of a user login/register
* Viewing a brief user instructions in the beginning of the game along with audio that have an option to be skipped
* Planting a new plant or visiting an existing plant that was previously created
* The watering of the plant to keep it alive
* Putting fertiliser to stimulate the growth of the plant
* Putting pesticide to remove pests around the plant
* Trimming wild grass around the plant
* Change of background depending on the time of day (day and night background)
* Giving random tips about the plant with text bubble and recorded audio when the plant inside the plant is clicked
* Showing all the tips in a page that we have stored on a plant when the user clicks on the name tag of the plant
* Over time, whether the user is on or off the game, the user can observe the growth of the plant through the progress increase in the progress bar or the plant growth stages displayed through different pictures for different stages of growth.
* There is a health value that is a metric of how well the user is taking care of the plant; if the user regularly uses the tools when it is available, the plant’s health will be maintained at 100.

The data that is kept in the database, in general, stores the following information:

* Real information about how to cultivate plants (time to grow, daily watering, etc.). This data is obtained from (<https://www.kaggle.com/rameshgupta74/plant-nurturing-guidelines>)
* Unique ID generated by the server for each user
* The plants that is owned by users
* The actions performed by users

# Technology

Frontend: HTML, CSS, Javascript with Tumult Hype template (<https://tumult.com/hype/>)

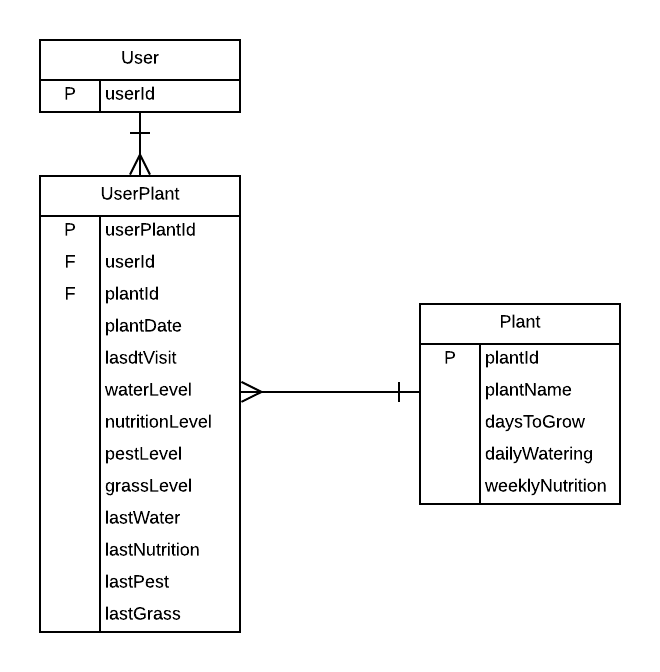
Backend: Nodejs/Express

Database: MySQL Database

One important technology to highlight in this situation is Tumult Hype as a templating tool for our game. The use of Tumult Hype in particular is to produce all the animations and resource interactions with ease.

# Database Characteristics & Data Sources

Below is the ERD model for Bunny’s Garden system:



**User:** Stores user unique ID for authentication to the server. A new row in the table will be added every time a first time user (client) clicks the “Play” button at the start of the game. This table is also used to check whether a user is a returning user.

**Plant:** From a high level perspective, this table acts as a knowledge base for the game. It stores all the information needed to cultivate a plant. The data in this table is curated from an open dataset originating from <https://www.kaggle.com/rameshgupta74/plant-nurturing-guidelines>. This source of data came in the form of a csv download, and went through a small data wrangling process to properly format the names of the plants. Since the game currently only offers 3 types of plant selection, currently there are only 3 rows in this table, but if in the future there is a plan to expand the plant selection, refer back to the dataset if there is already data on the plant.

**UserPlant:** Stores the users plant data including its progress and current condition. This database is updated in real time whenever a first time user clicks a plant type in the plant selection page that has not been created by the system before. The rows in this database are also updated in real time whenever a user performs an action on their plants such as watering or putting fertiliser.

**Action:** Stores all the possible actions the user can perform.

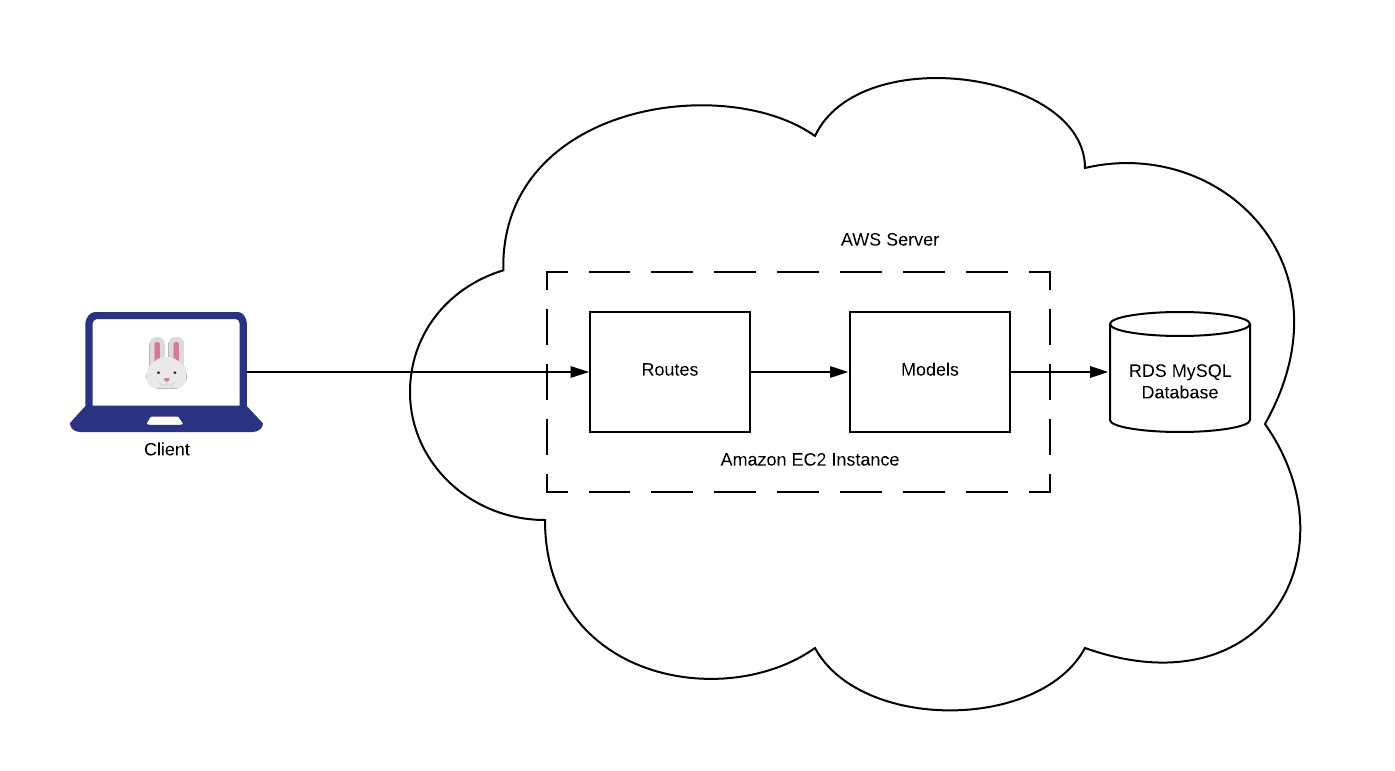
**UserAction:** Stores the actions that a user performs for each of their plants (watering, putting fertiliser, using pesticide, trimming grass). This is required to calculate the health of UserPlant rows.

# Authentication

The authentication of a user to the server utilises a local storage function of browsers to store a key pair element for further use when the user revisit the page. The item stored in the user is a unique key that acts as an ID for the server to authenticate and identify a returning user, to retrieve the user’s progress. This is done to avoid any form of registration/login to the server because we assume that the target audience (children aged 5-9) are incapable of performing a login. The authentication process are as follows:

1. The client presents a unique ID assigned by the server that is stored in the local storage of the browser. If it is a first time user, an empty object will be sent instead.
2. The server verifies the user and retrieves the appropriate user progress and data according to the ID.

# System Organisation

The image above is the system architecture employed by our team. The system is divided into two components: The frontend/client code and the backend/server-side code. This section will hence be divided into 2 components.

## Frontend/Client

Tumult Hype is the main workspace that we will use for the frontend development work. To work on the HTML file, we need to work on the .hype file extension in the app: Open Hype -> Open… -> Go to the project directory and open “BunnysGarden.hype”. The structure of the workspace is as shown below:



In the screenshot above, scenes corresponds to multiple display pages that is served as one HTML page as an output.

The animations in the application are done through Hype and GIF animations. To edit and add new animations into specific pages in the application, open the desired scene to edit and add new animations or change object properties through timelines.



## JQuery

The JQuery library is required in the application to make API calls into the server. Import was done in “Resources” -> “add file” -> and importing the JQuery library script. For more information and reinstallation: visit [https://jquery.com](https://jquery.com/).

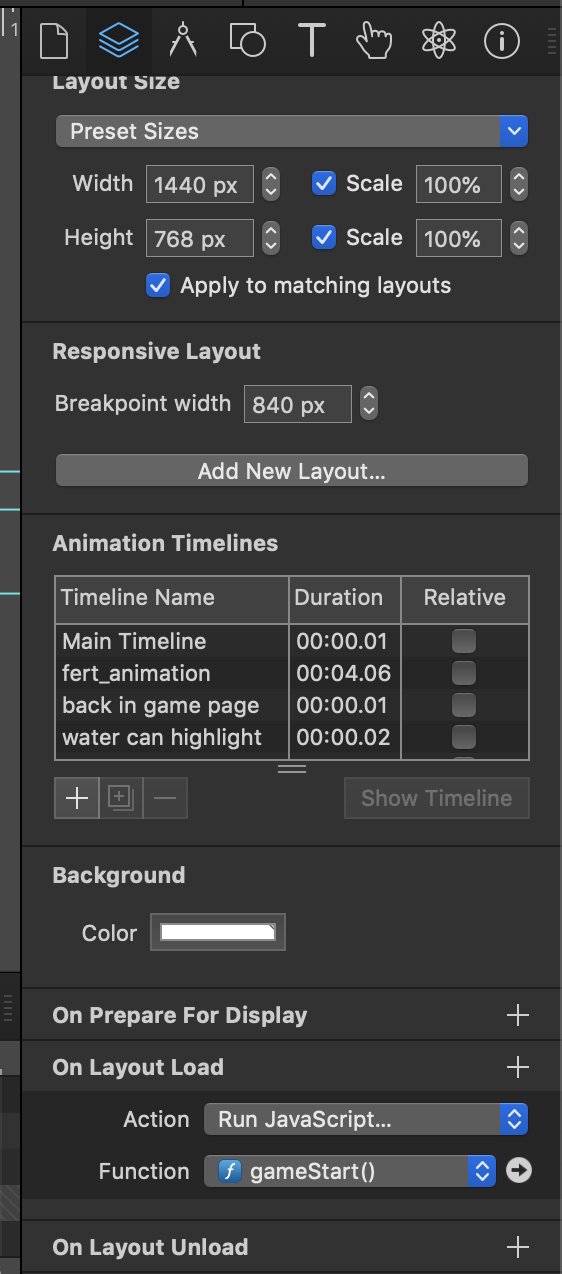
## Authentication

The code snippet for authentication on the client’s side can be found in:



The code snippet to authenticate can be accessed in the onclick function of the “PLAY”. The “authenticate()” function send a saved key in the client’s browser localstorage to the server. If the user is a first time user, the request will have an empty body and will save the key that is replied by the server.

### Game Initialisation



The function that controls the initialisation the game scene are initialised on layout load of the “Game” scene. The function makes a GET request to the server, sending the userId and the plantId that is requested by the user. This code will control the HTML elements according to the user plant’s data.

## Water/Fertiliser/Pesticide/Scissor

The function that is called respective of the actions that the user performs are:

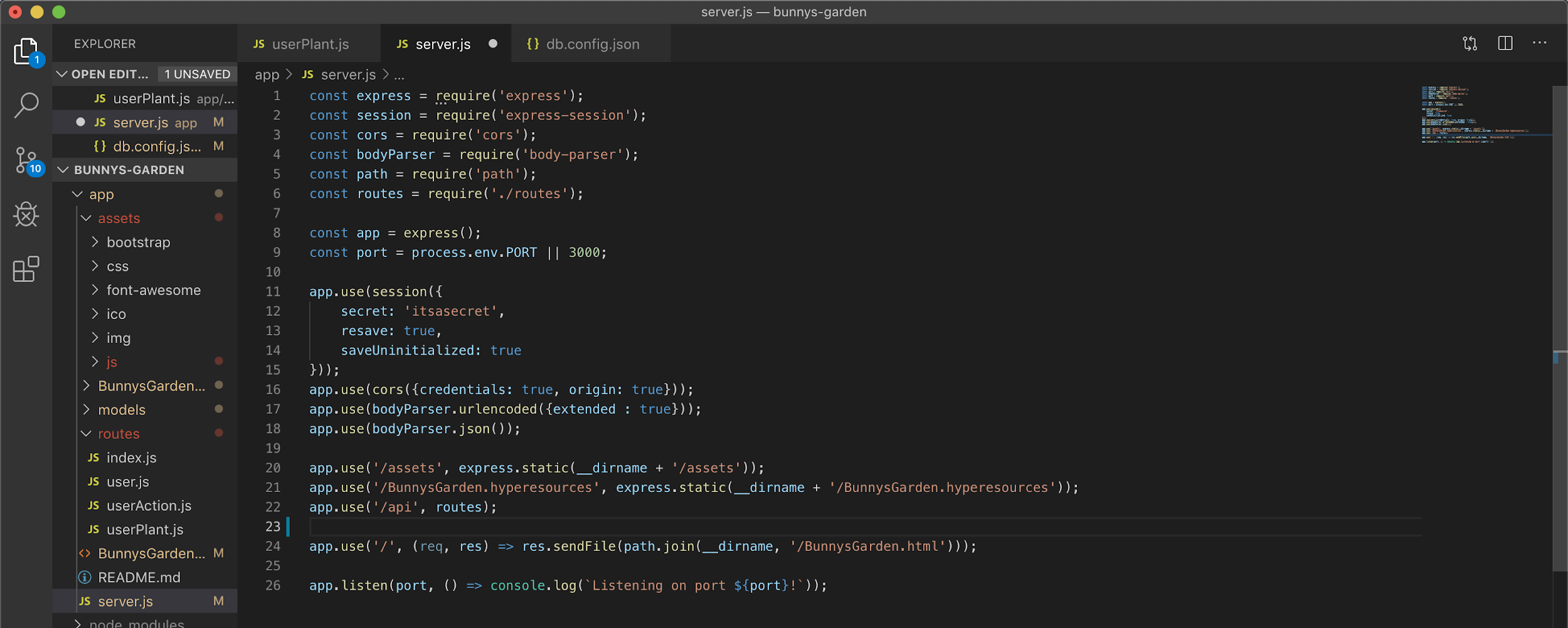
1. Watering can: water()
2. Fertiliser: fertilizer()
3. Pesticide: pesticide()
4. Scissor: trim()

The functions triggers a POST request to the server to persist the actions done by the user, and also provide the appropriate animations and sound. These functions can be found in Hype resources tab.

## 

## Backend/Server

### Server.js



The main server file can be accessed in /app/server.js. As observed, the server only serves one file that is produced by the Hype export to the client. The server bundles all the routes that is created and then serve this route as an API.

### Models

The files that runs SQL queries to the database are located in the folder “/app/models”. These scripts are filled with callback functions that will be utilised by the routes scripts to interact with the database.

The **Plant** table are populated with the data obtained from the dataset: <https://www.kaggle.com/rameshgupta74/plant-nurturing-guidelines>. The following SQL scripts are run to create the database:

INSERT INTO plant.Plant (plantName, daysToGrow, dailyWatering, weeklyNutrition)

VALUES ("Chive", 47, 3, 5);

INSERT INTO plant.Plant (plantName, daysToGrow, dailyWatering, weeklyNutrition)

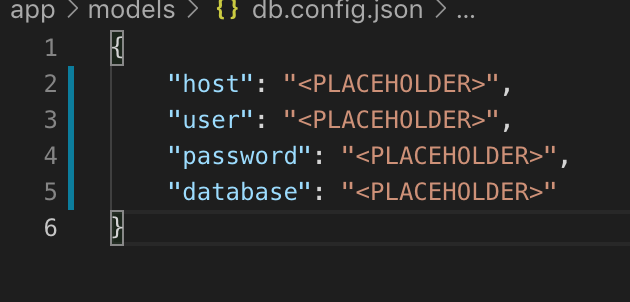
VALUES ("Tomato", 50, 2, 1);

INSERT INTO plant.Plant (plantName, daysToGrow, dailyWatering, weeklyNutrition)

VALUES ("Sunflower", 59, 3, 1);

### Db.config.json

To connect to your database, edit the file db.config.json and fill in the hostname, username, password, and database name of the DB hosted on the AWS server. To setup the database, refer to the support document: [insert link]



## Routes

All the API routes can be found in the /app/routes folder. These files are structured to handle requests from clients and import one or more scripts in the models folder (primarily of the same script name) to interact with the database and serve client requests. For example: “app/routes/user.js” imports the file “app/models/user.js” to run SQL queries to the database and then provide the result to the client. The routes also controls the flow of the game by updating the values inside userPlant rows.