Candy Sandbox :: Yaying Liang Li (Blob), Joshua Kloepfer (Pillow), Thomas Yu (Perry), Mark Zhu (Bob the 3rd Jr.)

Software Development

P02 – Design Doc

03/07/2022

Livin' Ducky
(based off Duck Life)

Tasks

- Ducky Sprites: our duckies / original Duck Life sprites [Yaying]
- Animating Duck doing actions (eating animation, running animation, etc) [Yaying/Thomas]
- Abilities of Duck (ie. jump, crouch, move up, down, left, right for certain races) [Mark]
- Stats / Levels [Joshua]
  - Running, Swimming: stats correlate to speed
  - Flying: stats correlate to distance
- Games associated with each stat [Joshua]
- Different races (map-generation /obstacle course) [Thomas]
  - Background: infinitely scroll through background until new background spawns ("you've reached a new place yay!!")
- Controls (Arrow keys) [Mark]
- Duck NPCs for races [Yaying]
- Multiple players can start a new game on computer
  - Can have multiple ducks per game
- Login button (save skins, stats) [Mark]
  - user does not need to be logged in to play game, but if they want their progress to be saved, they do

- HTML pages (home, register, login) and navigation bar [Thomas]
- Coin system (collected in races & training) [Mark]
- Food system (outside races; buy with coins) and stamina system (for training; determines when the training ends) [Joshua]
    - Feed duck more, stamina stat increases
- Cosmetic Shop: Different skins/hats, inspired by our duckies [Yaying/Thomas]
- ~~**Organic Polymer :)**~~
- ~~Suffering of high school students :,D~~ (Executive Decision)

Optional (if we have time)

- Sound
- Pause/Continue Button
- Climbing Race – future Duck Life games (past ver. 1) have it
- Kill the ducks option – continuum of Duck Life → Duck Death lawl

Program Components and How They Relate

- **Python:**
    - Set up database tables and update them (Users, Ducks, Cosmetics, Images)
    - Randomizer obstacle course
        - Randomly put coins and obstacles into training games
- **Flask: to run website**
    - Facilitates connection between backend and frontend
- **SQLite3:**
    - Database to store information (allows the user to save and continue)
    - **Users**:
        - Temporary row for user if user is not logged in → if not-logged in user quits game without logging in, pop temporary row out of db;

if not-logged in user logs in after playing to save their progress, update temporary row to permanent row w/ proper info

Username (TEXT)	Password (TEXT)	Races Completed (INT)	Coins (INT)

- **Ducks**:

Username of Account Duck is Linked to	Duck Name	Running Level	Swimming Level	Flying Level	Stamina	Cosmetics on Duck

    - May separate the cosmetics column if there are too many that can be on at once [idt it'll be a problem; orig. games had hat / skin only]

- Cosmetics:

Username	Cosmetic Item

- Images (in general, ie. backgrounds, duck images) to draw the game:

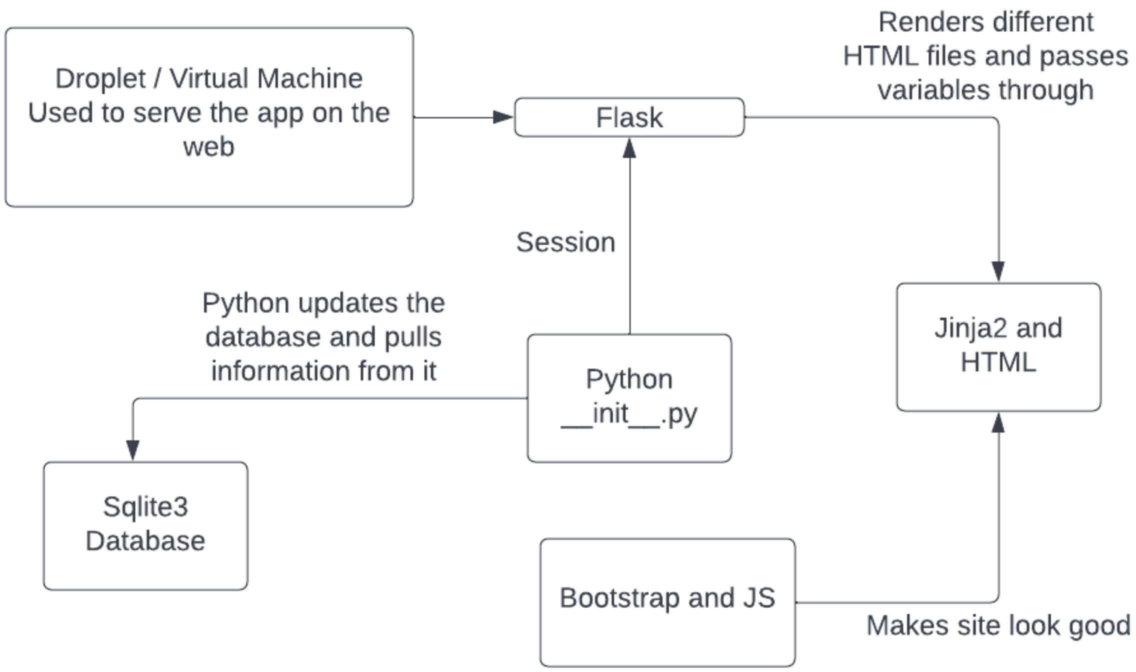
Image	Link to Image

    - Or images will be stored in a directory within the project to be accessed

- **DigitalOcean and Apache(Server):**
    - Allows the site to be run from the web rather than only local host
- **HTML: to set up the canvas and menu**
    - Home Page
        - Path to login, register, and game
        - New game and continue options
    - Login Page
    - Register Page
    - Game Page

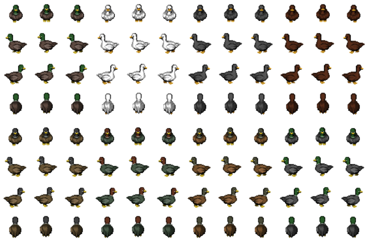
- **JS: graphics for game**
    - Creates the animations for the game
        - Duck movement (hopping, swimming, and flying)
        - Background scrolling to the side (training games and races)
        - Picking up and moving the duck with the cursor
        - Duck eating food
- **Jinja2:**
    - Passes data from Python to the HTML file to be displayed
        - Duck levels/stats
        - Cosmetics
        - Navigation bar? (Register, Login, Play)
- **Bootstrap/CSS: to make the menu look nice**
    - Formatting for the login page and the game page

Component Map


Droplet / Virtual Machine
Used to serve the app on the web
Flask
Renders different HTML files and passes variables through
Session
Python updates the database and pulls information from it
Python
__init__.py
Jinja2 and HTML
Sqlite3 Database
Bootstrap and JS
Makes site look good

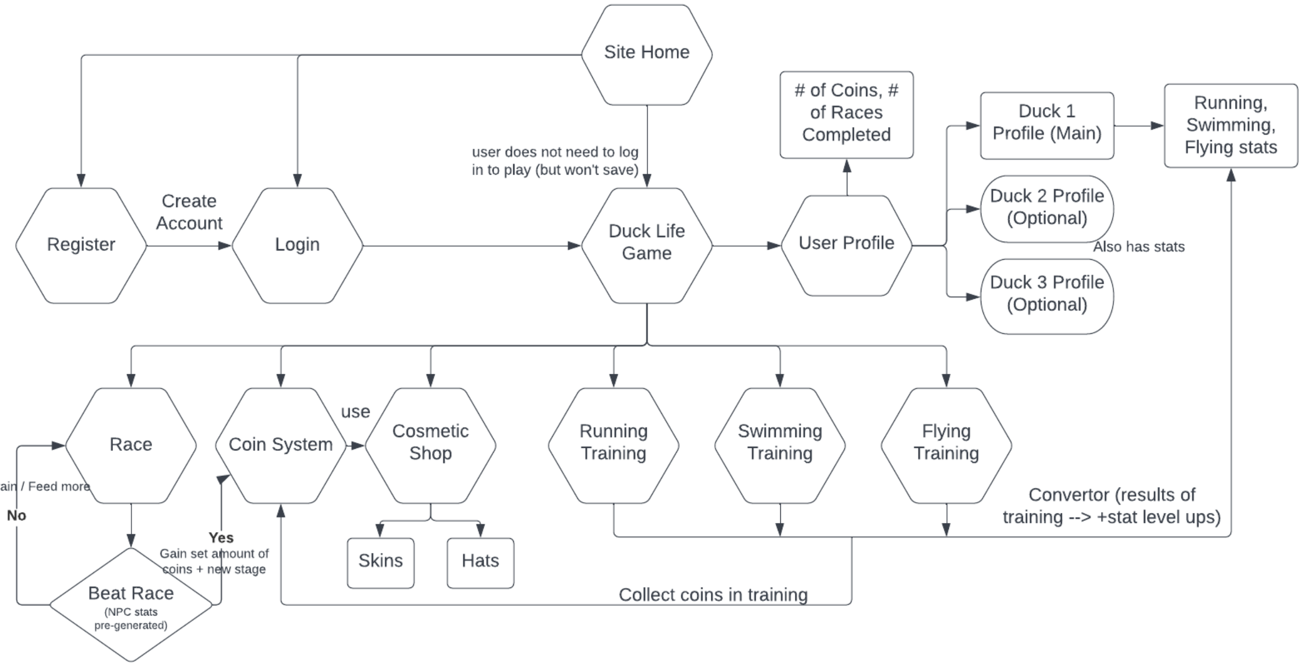
*JavaScript for animations as well (redraw canvas for background, crop through sprite sheets to represent frames in animation)

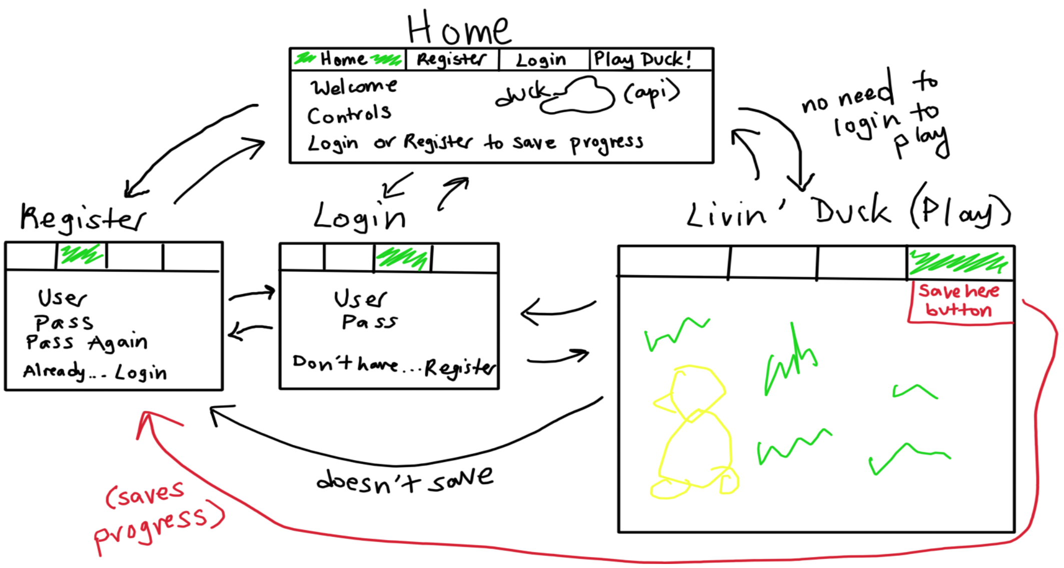
Possible Sprite Sheets to Use



https://rpgtileset.com/?s=duck

Sitemap


Site Home
user does not need to log in to play (but won't save)
Register
Create Account
Login
Duck Life Game
User Profile
# of Coins, # of Races Completed
Duck 1 Profile (Main)
Running, Swimming, Flying stats
Duck 2 Profile (Optional)
Also has stats
Duck 3 Profile (Optional)
Race
Train / Feed more
No
Beat Race
(NPC stats pre-generated)
Yes
Gain set amount of coins + new stage
Coin System
use
Cosmetic Shop
Skins
Hats
Running Training
Swimming Training
Flying Training
Convertor (results of training --> +stat level ups)
Collect coins in training



APIs

- Random Duck API - https://random-d.uk/api

FrontEnd Framework

- **Bootstrap** – navigation bar with containers and buttons easier to implement (in our opinions)
- All 4 members previously used bootstrap (++familiarity)

Breakdown of Tasks

- Frontend (HTML, CSS, Javascript)
  - Thomas
  - Yaying
- Backend (Databases, Python, Flask)
  - Joshua
  - Mark

Target Ship Date

***April 4th –** possible area of difficulty: getting familiar with sprite animations