DK OSRS

A website dedicated to learning and searching about the online game known as Old School RuneScape

By

Dylan Kelly

1. Introduction

1.1 Project Description and Overview

- The project consists of building and hosting a functional website that will allow the users to browse and learn more about the online game RuneScape. The site will offer in game item searching, in game player searching and in game monster searching. The user will have option to register for the site which will unlock more advance features such as, Item saving, player comparison highlighting, profile section and voting for their favorite RuneScape streamer and youtubers. The voting system will allow a user to vote only once to keep from spamming the system.

1.2 Project Deliverables

- The project will be maintained through GitHub and correspond to the gantt chart listed below. Using git timestamp to measure and record when the task is completed. The repository link can be found here : https://github.com/Dylandk10/cosc_412_solo
- Product can be expected by May 2021 and plan to keep iterating improvements.

1.3 Development Approach

- This project will be built using agile methodology. Starting first with the planning phase which will consist of completing the following diagrams: Gantt chart, HLA (High Level Architecture), WBS, Use-case diagrams. Following completion of the planning phase the next phase will be the implementation and maintenance phase, this is an ongoing life cycle phase. The implementation and maintaining phase will use a spiral approach to the development of the site starting with implementation -> testing -> fixing and when ready deployment -> repeat. This will allow myself to build test and deploy the website on an ongoing basis while maintaining bug handling and added features.

Justification

I selected the agile approach with a spiral iteration because this would allow me to continue updating and fixing new features and bugs as they are discovered through the site. I prefer to use a spiral approach during the development phase so I can keep iterating over a task to make sure I keep up with the demanding change from the website users.

2. Managing Changes

2.1 Management Objectives and Priorities

- To manage the on going changes the project will be using GitHub for version control.
- Each new iteration will check bugs and deploy patches.
- No money will be spent until the hosting service Heroku reaches the capacity for free use then a payment will be put down to scale the database and app.
- The Gantt chart will serve as the bases for time of completion on the project.

2.2 Assumptions, Dependencies and Constraints

- GitHub version control
- Unicorn for server configuration
- Python and Django for backend configuration
- SQLite Database
- Vanilla JavaScript for DOM manipulation
- jQuery for Ajax calls
- OSRS API for fetching player, monster and item data

2.3 Risk management

- OSRS API has been known to be slow and frequent experiences downtime to overcome downtime I will also use OSRS Buddy to prevent downtime on the site.
- Using Heroku can pose a risk for poor performance and lack of storage for database
- As risk emerge we will meet via virtual to discuss next steps to take in overcoming downtime and scaling

3. Technical Process

3.1 Tools

3.1.1 Back End

- Python – version 3.8.9
- Django – version 3.0.7
- SQLite
- OSRS API
- Django Migration
- Django Unicorn – version 20.1
- Django Whitenoise – Version – 5.2

3.1.2 Front End

- HTML

- CSS
- JavaScript
- jQuery

3.1.3 General Tools

- GitHub version control
- Git Bash
- Heroku
- Heroku-Git Bash

3.2 Requirements

3.2.1 General Requirements

- Landing Page
- Login page
- Register page
- Item search page
- Player search page
- NPC search page
- Voting page

3.2.2 Functional Requirements

- Creating user with (name, game name, email, password)
- OSRS API connection and calls
- Player searching for in game stats
- Item searching for in game prices and descriptions
- User sign in
- User sign out
- Register users
- Voting table
- User voting rights
- Users to add players to vote for
- Little game break for players to play
- Change password
- Delete account

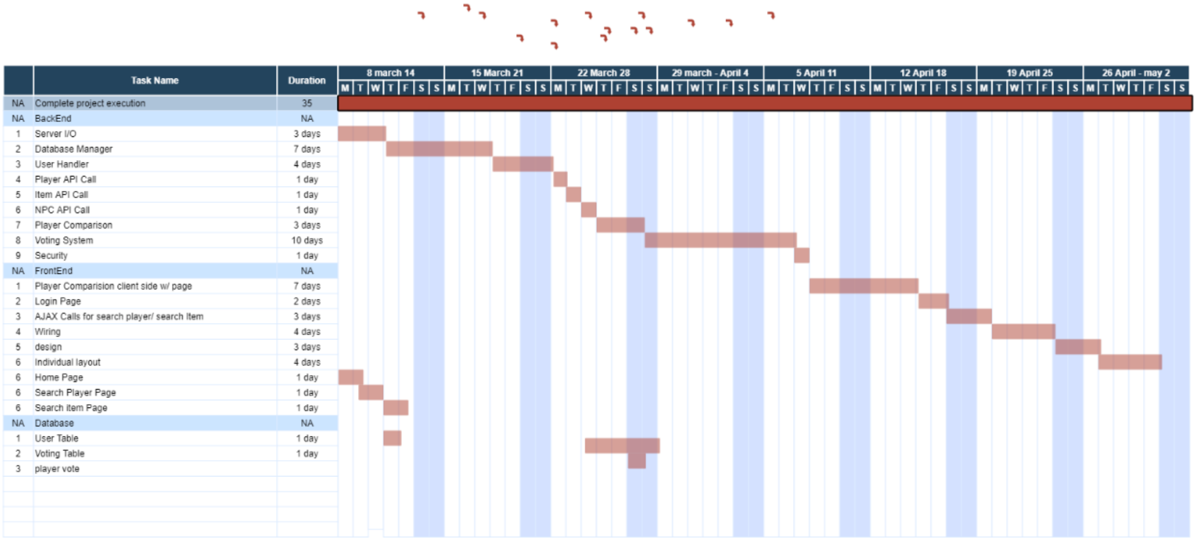
3.2.3 Non-Functional Requirements

- Database to scale with more users
- Data table to scale with more users
- Data voting table to scale with more players entered

- User information to be secure
- Passworded hashed and salted
- reCAPTCHA for detecting bots
- Admin SQL pulling

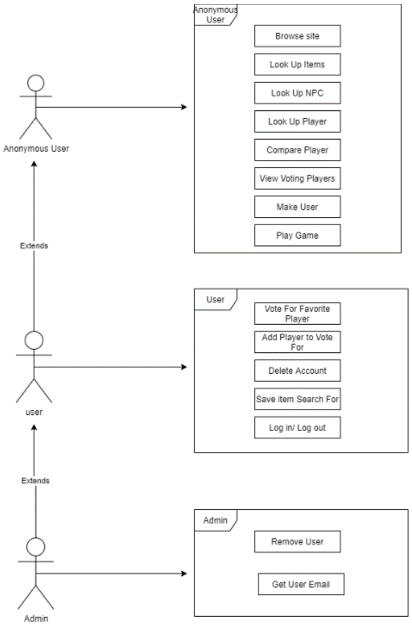
4. Figures

4.1 Gantt Chart

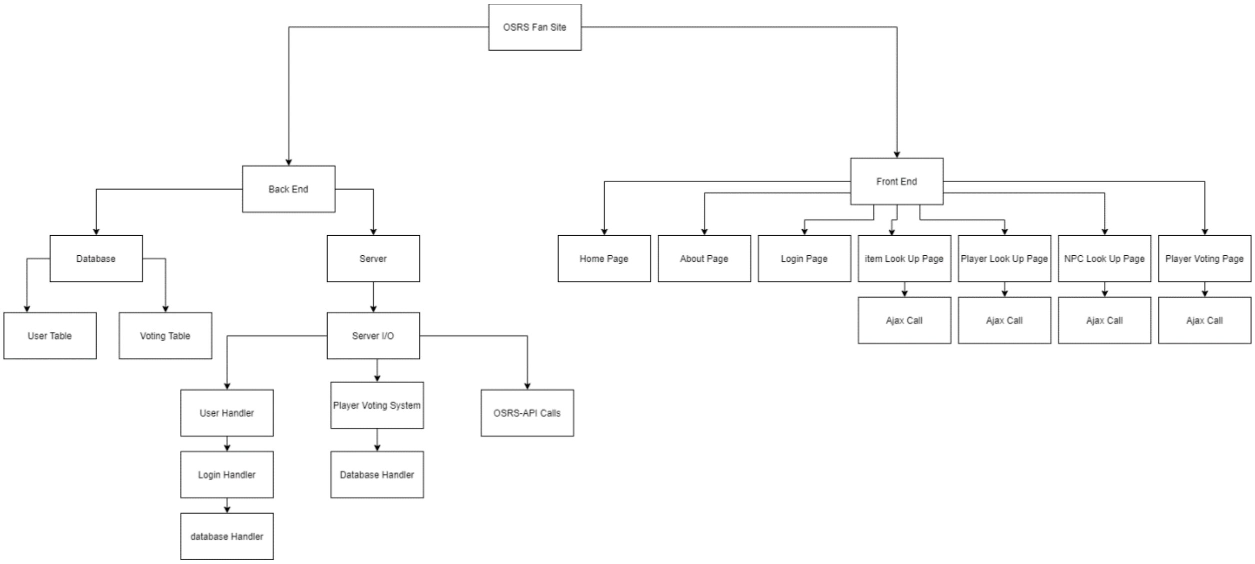


	Task Name	Duration
NA	Complete project execution	35
NA	BackEnd	NA
1	Server I/O	3 days
2	Database Manager	7 days
3	User Handler	4 days
4	Player API Call	1 day
5	Item API Call	1 day
6	NPC API Call	1 day
7	Player Comparison	3 days
8	Voting System	10 days
9	Security	1 day
NA	FrontEnd	NA
1	Player Comparision client side w/ page	7 days
2	Login Page	2 days
3	AJAX Calls for search player/ search Item	3 days
4	Wiring	4 days
5	design	3 days
6	Individual layout	4 days
6	Home Page	1 day
6	Search Player Page	1 day
6	Search item Page	1 day
NA	Database	NA
1	User Table	1 day
2	Voting Table	1 day
3	player vote	

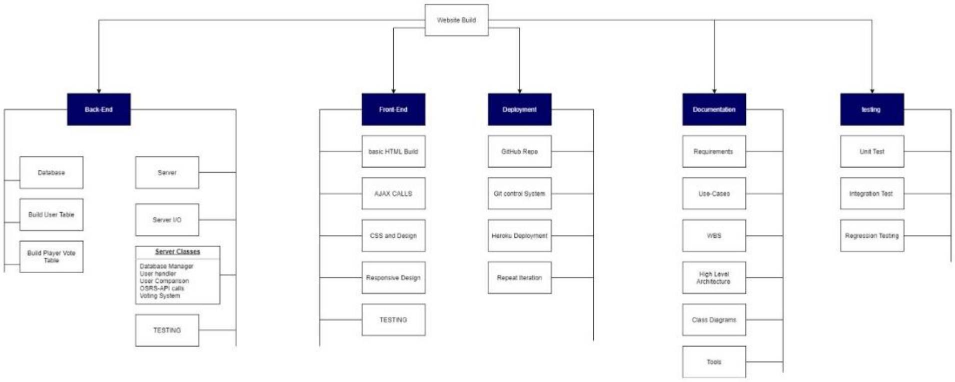
4.2 User-use case diagram



4.3 HLA



5.3 WBS



Inside the folder there are many files for specific use case, API tools, requirements, class-diagrams and other files related to the project.