**Documentatie project Steam**

**Groepsnaam: pay2win**

**Groepsdeelnemers: Thijn, Dieu, Daan, Yverdon, Glenn**

**Inleverdatum: 31 januari 2024**

**Klas: 1vo**

**Inhoudsopgave**

[**Inleiding** 3](#_Toc157469088)

[**Verplichte onderdelen** 4](#_Toc157469089)

[**Basis applicatie met data:** 4](#_Toc157469090)

[**Customer Journey Canvas:** 4](#_Toc157469091)

[**Statestiekalgoritme:** 4](#_Toc157469092)

[**Toepassingen hardware:** 4](#_Toc157469093)

[**Aanvullende onderdelen** 5](#_Toc157469094)

[**Artificial intelligence:** 5](#_Toc157469095)

[**Steam API:** 5](#_Toc157469096)

[**Sorteren:** 5](#_Toc157469097)

[**Zoeken:** 5](#_Toc157469098)

[**Grafieken en diagrammen:** 5](#_Toc157469099)

[**Normaalverdeling:** 5](#_Toc157469100)

[**Technische Informatica:** 5](#_Toc157469101)

[**Extra onderdelen** 6](#_Toc157469102)

[**Library page:** 6](#_Toc157469103)

[**Detail pagina:** 6](#_Toc157469104)

[**Aanmaken van database:** 6](#_Toc157469105)

[**Scripts schrijven voor importeren van data uit API:** 6](#_Toc157469106)

[**Sales pagina:** 6](#_Toc157469107)

[**Use case diagram:** 7](#_Toc157469108)

# **Inleiding**

Het doel van ons project is om Steam-gebruikers beter van dienst te zijn door in onze applicatie het gaming gedrag van vrienden op het Steam-platform te laten zien. Ons doel van dit project is om ervoor te zorgen dat de gebruiker met behulp van onze applicatie zijn game ervaring een nieuwe boost kan geven. Gebruikers kunnen via onze applicatie onder andere ontdekken welke games op dit moment gespeeld worden door hun vrienden. Daarnaast bieden we een overzicht van de meest populaire spellen tussen je vrienden op Steam waardoor jij en je vrienden meer inzicht krijgen in wat jullie leuk zouden vinden om samen te spelen. Daarnaast bieden wij ook inzicht in wat in de toekomst populaire spellen zouden kunnen gaan worden en vele andere functionaliteiten die hieronder aan bod komen.

Het document is opgedeeld in drie onderdelen: Verplichte onderdelen, Aanvullende onderdelen en extra onderdelen. Bij deze onderdelen zijn weer verschillende functionaliteiten en deze worden toegelicht in het desbetreffende kopje.

# **Verplichte onderdelen**

## **Basis applicatie met data:**

De basis applicatie is gemaakt doormiddel van Flask. Dit is een framework waarmee je html kan runnen met Python. De data wordt binnen gehaald doormiddel van requests naar de Steam API. Dit komt terug in een Json formaat en dit wordt gebruikt om de data weer te geven.

## **Customer Journey Canvas:**

Wij hebben dit voor twee momenten gedaan de Ist(het heden), dit is het moment voordat onze applicatie in is gezet en de Soll(de toekomst), dit is het moment nadat onze applicatie in is gezet. Je kan de verbetering zien in de Soll situatie door middel van de verbeterde gedachten van de persona’s. Wij hebben verschillende mensen geïnterviewd en daar mee twee persona’s gecreëerd.

## **Statestiekalgoritme:**

Voor de voorspellende statistieken werd er voorspeld welke games populair gaan worden en welke games er gaan dalen in populariteit. Dit is gedaan met lineaire regressie en wordt gevisualiseerd door middel van een grafiek. Ook de grafiek is een aanvullend onderdeel dus dat wordt later besproken.

De kwalitatieve variabel was de datum in de grafiek voor het voorspellen van de populariteit van games. Deze kan je onder aan de grafiek vinden zodat je weet welk datapunt bij welke datum en tijd hoort.

Voor de kwantitatieve variabel hebben wij de speeltijd van de vrienden gebruikt om de meest populaire games onder je vrienden over de laatste twee weken te bepalen. Dit werd gedaan met behulp van het merge sort algoritme, dit is tevens ook een aanvullend onderwerp, dus hier over meer bij het volgende onderdeel. De data komt binnen via de getPlayerGames functie.

## **Toepassingen hardware:**

De gekozen toepassing is het laten zien van personastate van jouw vrienden op de neopixel. Als het lampje bijvoorbeeld groen is, is de persoon online en als het rood is, is de persoon offline. Het ophalen van de gegevens gebeurt via de getplayersummaries api.

# **Aanvullende onderdelen**

## **Artificial intelligence:**

### **Steam API:**

De data die in de gui te zien is komt uit verschillende Steam API’s. Als het heel veel data was is de data opgeslagen in de database. Dit hebben wij bijvoorbeeld gedaan met alle games op Steam. Op deze manier hoeven wij niet voor elke game een API-call te make en kunnen wij dit dus uit de database ophalen.

### **Sorteren:**

Zoals eerder benoemd, is er gebruik gemaakt van het merge sort algoritme op de games die populair zijn tussen vrienden. Dit was eerst het insertion sort algoritme, maar dit was niet snel genoeg. Merge sort heeft een voorkeur en aan gezien het ligt aan hoeveel vrienden de gebruiker heeft en hoeveel games zij de afgelopen twee weken hebben gespeeld, is het goed om ervan uit te gaan dat de dataset groot gaat worden.

### **Zoeken:**

Er is een simpel zoek systeem geïmplementeerd die alle games uit de database zoekt dit gebeurt via ajax en queries voor postgres. Ajax zorgt ervoor dat een live search mogelijk is waardoor als er een character toegevoegd aan de search bar dat er dan een script wordt uitgevoerd die data ophaald en live geupdated wordt op de website applicatie.

**Grafieken en diagrammen:**

Zoals eerder benoemd is er een grafiek gebruikt voor het visualiseren van de voorspelling voor de populariteit van de games. Dit gebeurd met Chart.js. Wij hebben dit gebruikt omdat de documentatie heel duidelijk is. Het opzetten van een grafiek was makkelijk. Door de vele configuraties die Chart.js te bieden heeft, hebben we een duidelijke grafiek gekregen.

## **Technische Informatica:**

Als ondersteuning voor de Neopixel is er een lcd-scherm. Hierop krijg je te zien welke spelers op welk lampje zitten.

# **Extra onderdelen**

## **Library page:**

Op deze pagina kan je al de games zien die jij in je bezit hebt. De lijst voor de game Ids wordt opgehaald vanuit de api en de games worden opgehaald uit de database. De games komen dan tevoorschijn met het plaatje en de naam van de game. Het is de bedoeling dat je dan op de game kan klikken zodat je doorverwezen kan worden naar de detail pagina.

<https://api.steampowered.com/IPlayerService/GetOwnedGames/v0001/?key=KEY&steamid=STEAMID&format=json>

Gemaakt door: Glenn

## **Detail pagina:**

De detail pagina laadt met de id die mee is gegeven via de routing van de pagina. Als je namelijk op het plaatje van de game op de library pagina klikt dan staat de id van die game meegegeven in de url. Met deze id wordt de game opgehaald uit de database en dingen als de naam, beschrijving, de requirments voor je pc, screenshots, ect.

Gemaakt door: Daan

**Server**

Onze webapp haalt data vanuit een server, die meerdere essentiële taken uitvoert om de soepele werking van onze applicatie te ondersteunen. Naast het draaien van een database, speelt de server een cruciale rol bij het actueel houden van gamegegevens.

Een van de belangrijkste functies van de server is het continu bijwerken van gamegegevens. Dit omvat het toevoegen van nieuwe games aan onze database en het regelmatig updaten van de details van bestaande games. Hierdoor blijft onze informatie altijd accuraat en actueel, wat van essentieel belang is voor een betrouwbare gebruikerservaring.

Bovendien hebben we een geautomatiseerd script geïmplementeerd dat periodiek wordt uitgevoerd. Dit script voert een uurlijkse controle uit om het aantal spelers in de top 100 games te zien.

## **Aanmaken van database:**

Met het oog op het voorkomen van trage laadtijden op de site en het minimaliseren van het aantal API-requests, hebben we besloten een database te creëren. Dit begon met het opstellen van een Entity-Relationship Diagram (ERD), waarna deze conceptuele database werd omgezet naar een fysieke database.

Gemaakt door: Dieu

## **Script voor importeren van game data uit API:**

Om de data in de database te krijgen zijn er scripts gemaakt die een API request maakt voor alle game id’s op de steam store, en deze id’s een voor een in een request zet die de details van deze games ophaalt. Dit gaat dan door een functie die filtert of het ook echt van het type “game” is en geen DLC of iets anders, en dan worden alle gegevens verdeeld over de desbetreffende tabellen en kolommen.

* **Alle Apps API:**

[**http://api.steampowered.com/ISteamApps/GetAppList/v0002/?key=JOUW\_STEAMKEY&format=json**](http://api.steampowered.com/ISteamApps/GetAppList/v0002/?key=JOUW_STEAMKEY&format=json)

* **App Details API:**

[**http://store.steampowered.com/api/appdetails?appids={APP\_ID**](http://store.steampowered.com/api/appdetails?appids={APP_ID)**}**

Gemaakt door: Thijn

**Sales Pagina**

Op de sales pagina worden niet alleen de games van jouw Steam Wishlist weergegeven die momenteel in de aanbieding zijn, maar ook populaire games die afgeprijsd zijn. Voor games op je wishlist wordt een API aangeroepen om te kijken welke games zich in jouw wishlist zitten. Deze API kijkt meteen welke spellen in jouw wishlist een discount hebben.

Voor de populaire games wordt een iets andere aanpak gehanteerd. We slaan periodiek, elk uur, het aantal spelers op in onze database. Op basis van deze gegevens wordt een berekening uitgevoerd om de top 100 populaire spellen te bepalen. Vervolgens wordt in de database gekeken welke van deze 100 spellen momenteel een korting hebben. De lijst van populaire games met kortingen wordt weergegeven op de sales pagina.

<https://store.steampowered.com/wishlist/profiles/userid/wishlistdata/?p=0>

Gemaakt door: Thijn

**Voorspellingsfunctionaliteit**

Onze voorspellingsfunctionaliteit maakt gebruik van Gradient Descent om een grafiek te genereren die aangeeft of er in de nabije toekomst een stijging of daling wordt verwacht in het aantal spelers. De benodigde gegevens worden verkregen via de Steam API: <https://api.steampowered.com/ISteamChartsService/GetGamesByConcurrentPlayers/v1/?>.

Deze API wordt eens per uur uitgevoerd op een server. Het bijbehorende script, concurentPlayersToDatabase verzamelt het aantal spelers voor de top 100 games en slaat deze gegevens op in de database. Tijdens de uitvoering van dit script worden direct de nodige berekeningen uitgevoerd en de resultaten toegevoegd aan de database. Deze aanpak zorgt ervoor dat de benodigde berekeningen niet worden uitgevoerd bij het opstarten van de website, waardoor de reactiesnelheid van de website wordt geoptimaliseerd.

## **Use case diagram:**

Om in kaart te brengen wat wij allemaal gebruiken en waar dit vandaan komt is er een Use Case diagram gemaakt. (misschien meer toelichting hierbij)

Gemaakt door: Dieu

Bronnen  
https://www.chartjs.org/docs/latest/