# Projektdokumentation: JumpBot AI



[Projektdokumentation: JumpBot AI 1](#_Toc159597512)

[Analyse 3](#_Toc159597513)

[Projektinformation 3](#_Toc159597514)

[Beschreibung der Idee 3](#_Toc159597515)

[Skizze / Layout der Idee 3](#_Toc159597516)

[Technische Spezifikationen 4](#_Toc159597517)

[Programmier-Tools: 4](#_Toc159597518)

[Programmiersprachen: 4](#_Toc159597519)

[Weitere Tools und Bibliotheken: 4](#_Toc159597520)

[Anforderungen 5](#_Toc159597521)

[Planung 6](#_Toc159597522)

[Aufwandschätzung 6](#_Toc159597523)

[Kostenabschätzung 7](#_Toc159597524)

[Zeitabschätzung 7](#_Toc159597525)

[Meilensteine 7](#_Toc159597526)

[Design 9](#_Toc159597527)

[Wireframes und Prototyping 9](#_Toc159597528)

[Tests 9](#_Toc159597529)

[Getestet 9](#_Toc159597530)

[Unittests 9](#_Toc159597531)

[Spielinitialisierung: 10](#_Toc159597532)

[Spieleraktionen: 10](#_Toc159597533)

[Kollisionsdetektion: 11](#_Toc159597534)

[Spielneustart: 11](#_Toc159597535)

[Black-Box-Testing: 11](#_Toc159597536)

[Installationsanleitung 13](#_Toc159597537)

[Voraussetzungen 13](#_Toc159597538)

[Installationsschritte 13](#_Toc159597539)

# Analyse

## Projektinformation

|  |  |
| --- | --- |
| Schüler Name(n): | Max Foll |
| Betreuer Name: | Martin Hager |

## Beschreibung der Idee

Ein Webbasiertes Mini-Game programmieren. Der Spieler steuert eine Spielfigur, die sich automatisch nach vorne bewegt. Die Hauptaufgabe des Spielers ist es, Hindernissen auszuweichen, indem er die Spielfigur springen lässt. Das Spiel endet, wenn die Spielfigur ein Hindernis berührt. Dazu wird eine AI entwickelt, die das Spiel lernt und optimiert spielt.

## Skizze / Layout der Idee

Ein Bild, das Screenshot, Text enthält.

Automatisch generierte Beschreibung

## Technische Spezifikationen

### Programmier-Tools:

* Visual Studio Code: Eine leistungsstarke und vielseitige Quellcode-Editor von Microsoft, die für die Entwicklung in verschiedenen Programmiersprachen geeignet ist, mit umfangreicher Unterstützung für Python und JavaScript durch Erweiterungen.
* TensorFlow: Eine umfangreiche Open-Source-Plattform für maschinelles Lernen, die es ermöglicht, komplexe neuronale Netzwerke zu erstellen und zu trainieren. In diesem Projekt wird TensorFlow für die Entwicklung der künstlichen Intelligenz genutzt, die das Mini-Game spielen lernt.

### Programmiersprachen:

* JavaScript: Eine dynamische Programmiersprache, die aufgrund ihrer Vielseitigkeit und weit verbreiteten Unterstützung für Webentwicklungsaufgaben eingesetzt wird. In diesem Projekt wird JavaScript vermutlich für die Frontend-Entwicklung und die Spiellogik verwendet.
* Python: Eine interpretierte Hochsprache, die für ihre klare Syntax und Lesbarkeit bekannt ist. Python wird in diesem Projekt verwendet, um die Backend-Logik zu implementieren, insbesondere die KI-Aspekte mit TensorFlow.

### Weitere Tools und Bibliotheken:

* asyncio: Eine Python-Bibliothek, die zur Schaffung von ereignisgesteuerten, asynchronen Programmen verwendet wird, was besonders nützlich für skalierbare Netzwerkanwendungen ist.
* websockets: In diesem Projekt wird die websockets-Bibliothek verwendet, um eine Echtzeit-Kommunikationsbrücke zwischen der in Python entwickelten künstlichen Intelligenz und dem in JavaScript implementierten Webgame zu schaffen. Diese Verbindung ermöglicht es der KI, Spielzustände in Echtzeit zu empfangen und darauf basierend Entscheidungen zu treffen, wie z.B. wann die Spielfigur springen soll, um Hindernissen auszuweichen. Die Nutzung von WebSockets ist entscheidend für die Interaktivität des Spiels und ermöglicht eine nahtlose Integration der KI in das Spielgeschehen.
* json: Das JSON-Format wird in diesem Projekt intensiv genutzt, um Daten zwischen dem Webgame und der KI effizient zu formatieren und zu übertragen.
* tensorflow und tf.keras: Kernkomponenten von TensorFlow, die für das Erstellen und Trainieren von maschinellen Lernmodellen genutzt werden. In diesem Projekt werden sie für die Entwicklung der KI verwendet, die das Spiel spielt.
* numpy: Eine Bibliothek für Python, die Unterstützung für große, mehrdimensionale Arrays und Matrizen bietet, zusammen mit einer Sammlung von mathematischen Funktionen, um diese zu bearbeiten.
* random und deque aus der collections-Bibliothek: Diese Python-Module werden für Zufallsoperationen und für die Implementierung von Datenstrukturen wie Warteschlangen verwendet, was für Algorithmen im Bereich des maschinellen Lernens nützlich ist.
* concurrent.futures: Ein High-Level-Interface für asynchrone Ausführung von Aufrufen mit Threads oder Prozessen, was die Performance von I/O-bound und High-Level strukturierten Netzwerkanwendungen verbessert.
* logging: Ein Modul in Python, das ein flexibles Framework für das Ausgeben von Log-Nachrichten aus Anwendungen bietet.

## Anforderungen

Must have's:

* - Funktionierendes Spiel - funktioniert  
  - Scores - funktioniert  
  - AI kann das Spiel spiele - funktioniert

Nice to have's:

* - AI ist besser im Spielen als der Entwickler – funktioniert nicht  
  - AI kann auch andere Spiele lernen/spielen – funktioniert nicht

# Planung

## Aufwandschätzung

* Analyse und Konzepterstellung - 3 Stunden
* Diese Phase beinhaltet das Sammeln von Anforderungen, die Definition der Projektziele und die Erstellung des Konzepts, was in der Regel einen signifikanten Teil des anfänglichen Aufwands ausmacht.
* Einrichtung der Entwicklungsumgebung und Erstellung der Spielumgebung - 10 Stunden
* Die Einrichtung der Entwicklungsumgebung und die Erstellung einer grundlegenden Spielumgebung erfordern sowohl technisches Setup als auch erste Entwurfsarbeiten.
* Implementierung des KI-Agenten - 15 Stunden
* Die Implementierung des KI-Agenten, einschließlich der Auswahl der Algorithmen und der Programmierung der Lernmechanismen, ist eine der aufwändigsten Aufgaben im Projekt.
* Erster vollständiger Trainingsdurchlauf des KI-Agenten - 10 Stunden
* Die Durchführung des ersten Trainingslaufes umfasst das Setup, die Überwachung und erste Anpassungen, um die Leistung der KI zu bewerten.
* Überwachung und Analyse des Trainingsfortschritts - 5 Stunden
* Diese Phase beinhaltet die detaillierte Überwachung des Trainingsfortschritts und die Analyse der Ergebnisse, um Bereiche für Verbesserungen zu identifizieren.
* Feinabstimmung des KI-Agenten und Optimierung - 8 Stunden
* Die Feinabstimmung und Optimierung des KI-Agenten erfordert iterative Anpassungen und Tests, um die bestmögliche Leistung zu erreichen.
* Fertigstellung der Dokumentation - 12 Stunden
* Die abschließende Dokumentation des Projekts, einschließlich der Beschreibung der Architektur, des Designs und der Implementierungsdetails, nimmt vergleichsweise weniger Zeit in Anspruch.
* Abgabe des Abschlussberichts und Projektabschluss - 2 Stunden
* Die Vorbereitung und Abgabe des Abschlussberichts umfasst die letzte Überprüfung der Dokumentation und die Zusammenstellung aller notwendigen Unterlagen.

## Kostenabschätzung

Die Kosten dieses Projekts setzen sich aus zwei Hauptkomponenten zusammen: dem Arbeitsaufwand und dem Energieverbrauch für das Training der Künstlichen Intelligenz. Der Gesamtaufwand für das Projekt wurde auf 65 Stunden geschätzt. Mit einem angenommenen Stundensatz von 50 Franken ergibt sich daraus ein Personalkostenanteil von 3'250 Franken. Zusätzlich wird der Energieverbrauch für das Training der KI berücksichtigt, der bei etwa 0.1 Franken pro Stunde für insgesamt 60 Trainingsstunden angesetzt wird. Dies führt zu Energiekosten von insgesamt 6 Franken. Somit belaufen sich die geschätzten Gesamtkosten des Projekts auf 3'256 Franken.

## Zeitabschätzung

Die initiale Aufwandsschätzung für das Projekt betrug 20 Stunden. Unter Berücksichtigung der Empfehlung, den geschätzten Aufwand mit 3.14 zu multiplizieren, um Unsicherheiten und potenzielle Verzögerungen einzuplanen, ergibt sich eine Schätzung von 65 Stunden.

## Meilensteine

Jeder wichtige Projektschritt endet mit einem Meilenstein, der den Abschluss eines spezifischen Projektteils markiert. Meilensteine sind feste Zeitpunkte und haben keine eigene Dauer. Die Meilensteine für dieses Projekt sind:

Analyse und Konzepterstellung - 5. November

Einrichtung der Entwicklungsumgebung und Erstellung der Spielumgebung - 17. November

Implementierung des KI-Agenten - 6. Dezember

Erster vollständiger Trainingsdurchlauf des KI-Agenten - 19. Dezember

Überwachung und Analyse des Trainingsfortschritts - 9. Januar

Feinabstimmung des KI-Agenten und Optimierung - 16. Januar

Fertigstellung der Dokumentation - 16. Februar

Abgabe des Abschlussberichts und Projektabschluss - 23. Februar

# Design

## Wireframes und Prototyping

Ein Bild, das Screenshot, Text enthält.

Automatisch generierte Beschreibung

In diesem Projekt habe ich mich bewusst gegen die Erstellung von Diagrammen entschieden. Das mag zunächst unkonventionell klingen, aber diese Entscheidung spiegelt die speziellen Anforderungen und die Simplizität des Vorhabens wider.

Im Mittelpunkt steht die Künstliche Intelligenz (KI), die sich in einem äußerst simplen Spiel bewähren soll. Das Spiel selbst ist auf das Allernötigste reduziert: Es startet unmittelbar beim Laden, ohne jegliche Verzögerung oder komplizierte Menüführung. Ein Neustart? Nichts einfacher als dass, ein Sprung, ausgelöst durch die Leertaste oder den Pfeil nach oben, reicht vollkommen aus. Diese bewusste Vereinfachung ist kein Zufall, sondern dient dem klaren Ziel, die Herausforderungen für die KI so überschaubar wie möglich zu halten und dadurch das Lernen und die Anpassung an das Spielgeschehen zu erleichtern.

# Tests

## Getestet

Getestet von: Max Foll

Getestete Version: 4.2.0

Test-Datum: 21.02.2024

## Unittests

### Spielinitialisierung:

* Ziel: Korrekte Initialisierung aller Spielelemente überprüfen.
* Testdurchführung: Der Benutzer startet das Spiel und achtet darauf, ob Spielfigur und Hindernisse sofort und korrekt dargestellt werden.
* Bestanden: Ja/Nein (Ja)
* Anmerkungen: (-)

### Spieleraktionen:

* Ziel: Reaktion der Spielfigur auf Sprungbefehle sicherstellen.
* Testdurchführung: Der Benutzer drückt die Leertaste oder die Pfeil nach oben Taste oder die W taste und prüft, ob die Spielfigur wie erwartet springt.
* Bestanden: Ja/Nein (Ja)
* Anmerkungen: (-)
* Ziel: Reaktion der Spielfigur auf Duckbefehle sicherstellen.
* Testdurchführung: Der Benutzer drückt die Pfeil nach unten oder S Taste und prüft, ob die Spielfigur wie erwartet duckt.
* Bestanden: Ja/Nein (Ja)
* Anmerkungen: (-)
* Ziel: Reaktion der Spielfigur auf Duckbefehle während dem Springen sicherstellen.
* Testdurchführung: Der Benutzer drückt die Pfeil nach unten oder S Taste und prüft, ob die Spielfigur wie erwartet, während dem Springen duckt.
* Bestanden: Ja/Nein (Ja)
* Anmerkungen: (Dieses Feature habe ich bewusst eingebaut, um es zu ermöglichen, für Spieler unmögliche Geschwindigkeiten zu erreichen.)
* Ziel: Bei wiederholten Sprungbefehlen soll der Spieler landen müssen, bevor er wieder springen kann.
* Testdurchführung: Der Benutzer drückt wiederholt die Leertaste oder die Pfeil nach oben Taste oder die W taste und prüft, ob die Spielfigur wie erwartet springt erst nachdem sie gelandet ist.
* Bestanden: Ja/Nein (Ja)
* Anmerkungen: (-)

### Kollisionsdetektion:

* Ziel: Korrekte Erkennung von Kollisionen testen.
* Testdurchführung: Der Benutzer führt die Spielfigur in ein Hindernis, um zu sehen, ob das Spielende korrekt ausgelöst wird.
* Bestanden: Ja/Nein (Ja)
* Anmerkungen: (-)

### Spielneustart:

* Ziel: Funktionalität des Spielneustarts nach einem Spielende überprüfen.
* Testdurchführung: Nach einem Spielende drückt der Benutzer die Sprungtaste oder die Pfeil nach oben oder die S Taste, um den Neustart zu initiieren, und beobachtet, ob das Spiel wie erwartet neu startet.
* Bestanden: Ja/Nein (Ja)
* Anmerkungen: (-)

## Black-Box-Testing:

* Benutzererfahrung und Spielverhalten:
* Ziel: Konsistente und zufriedenstellende Benutzererfahrung gewährleisten.
* Testdurchführung: Der Benutzer spielt mehrere Durchgänge, um die Spielbarkeit und die Reaktion des Spiels auf Benutzereingaben zu beurteilen.
* Bestanden: Ja/Nein (Ja)
* Anmerkungen: (-)
* Grenzfall- und Stress-Tests:
* Ziel: Reaktion des Spiels auf extreme Bedingungen oder unerwartete Benutzereingaben testen.
* Testdurchführung: Erzeugung von Szenarien mit hoher Spielfigur- und Hindernisdichte sowie Testung ungewöhnlicher Eingabemuster, um die Robustheit des Spiels zu prüfen.
* Bestanden: Ja/Nein (Ja)
* Anmerkungen: (-)

# Installationsanleitung

## Voraussetzungen

* Stellen Sie sicher, dass Sie folgende Software installiert haben:
* Node.js und npm: Für die Ausführung des Backend-Servers und die Verwaltung der Node.js-Pakete.
* Python: Für Python-basierte Skripte oder Anwendungen.
* Java: Für Java-basierte Anwendungen. Stellen Sie sicher, dass Maven oder Gradle für den Build-Prozess installiert ist, abhängig von dem von Ihnen verwendeten Tool.

## Installationsschritte

* Klonen des Projekts: Klonen Sie das Repository mit folgenden befehlen:

• git clone < https://github.com/StrollinShadow/SA\_JumpBot\_AI.git>

* cd <project-directory>
* Installieren der Node.js-Abhängigkeiten: Führen Sie npm install im root des Projekts aus.
* Starten der Anwendung: Um Ihre Web-Applikation zu starten, öffnen Sie die Datei index.html in einem Browser. Um die KI zu starten, geben Sie npm start in die Konsole ein.

# Schlusswort

Ganz am Ende, als ich dachte, ich könnte die KI noch ein bisschen schneller machen, hab ich ein bisschen herumexperimentiert, um sie zu optimieren. Ich wollte eigentlich nur die unnötigen Teile rausnehmen, damit sie schneller lernt. Dabei hab ich aber aus Versehen die Reset-Befehle aktiv gelassen, die eigentlich dafür sorgen, dass die KI bei jedem Neustart wieder bei Null anfängt. Kurz gesagt, meine KI ist jetzt wie ein unbeschriebenes Blatt, komplett untrainiert.

Das ist natürlich ärgerlich, aber ich bin optimistisch, dass, wenn ich die Reset-Befehle rausnehme und die KI erneut trainiere, sie nach etwa 10-20 Stunden Training viel besser sein wird als vorher. Vor diesem kleinen Malheur hat sie schon ganz gute Ergebnisse gezeigt, und ich denke, sie kann ziemlich konsistent bis zu 1000-2000 Punkte erreichen, wenn alles glatt läuft. War definitiv eine Lektion für mich, in Zukunft doppelt zu checken, was ich im Code ändere!