**DIPLOMARBEIT**

DermaAI: Intelligente Hautanalyse

Gesamtprojekt

Entwicklung eines KI-Modells zur Detektion und Klassifikation pigmentierten Hautläsionen und dessen Integration in eine Mobile App

Diplomarbeitsnummer  
5AHINF-24/25-DA11

**Datenbank, Verwaltung der medizinischen Daten**

Jonas Maier 5AHINF Betreuer: Dipl.-Ing. Dr.

Gerhard Gaube

**Mobile Integration, Frontend**

Jonas Bogensberger 5AHINF Betreuer: Dipl.-Ing. Dr.

Michael Wimmer

**KI-Modell in Python, API und Appanbindung**

Daniel Jessner 5AHINF Betreuer: Dipl.-Ing. Dr.

Gerhard Gaube

Schuljahr 2024/25

Abgabevermerk:

Datum: TT.MM.JJJJ übernommen von:

Inhalt

1. **Allgemeines / Projektübersicht**
   1. **Projektbeschreibung**
   2. **Projekt und Schnittstellen**
2. **Funktionale Anforderungen**
   1. **Use Cases**
3. **Nicht Funktionale Anforderungen**
4. **Projektplanung**
   1. **Variantenbildung**
   2. **Machbarkeitsstudie**
      1. **Projektziele**
      2. **Benötigte Ressourcen**
      3. **Entwicklungsmethodik**
      4. **Kommunikations- und Berichterstattungsstrategie**
   3. **Softwarearchitektur**
      1. **Aktivitätsdiagramme**
      2. **Sequenzdiagramme**
      3. **Komponentendiagramme**
      4. **Verteilungsdiagramme**
      5. **Softwarediagramme / Komponenten**
      6. **SW Programme**
      7. **SW Komponenten**
5. **Projektdurchführung**
   1. **Webscraping & labeln der Daten**
      * 1. **Datenquelle und Beschaffung**

Das mit Abstand wichtigste Thema bei diesem Projekt ist die Bildbeschaffung und Bildbearbeitung, deshalb ist es auch das, was mich am längsten zum Implementieren gebraucht hat. Um ein zufriedenstellendes Ergebnis zu erhalten habe ich die bekanntesten und besten medizinischen Datenbanken durchsucht und bin letztendlich auf das Isic-Archive gekommen. Das Isic-Archive ist eine Kollaboration medizinischen Bildern von diversen Krankheiten und Erscheinungen und da es eine gute Dokumentation zum öffentlichen Endpunkt gibt, habe ich schnell meine Wahl getroffen. Meine Wahl für diesen Anwendungsfall fiel schnell auf die Programmiersprache **Python**, da diese viele Bibliotheken diese Situation bereitstellt und einfach und schnell erstellt ist. Die wichtigsten Packages, welche ich für das main script verwendet habe, sind die folgenden:

* **BeautifulSoup4 – 4.12.3 🡪** Bibliothek für das sammeln von Informationen aus dem Internet.
* **Re** 🡪 Prüft Text auf RegularExpressions um nach bestimmten Sachen zu filtern.
  + - 1. **Datenbereinigung und Vorverarbeitung**

Zum Webscrapen, damit wir die Bilddaten für unser Projekt verwenden können, habe ich mithilfe von verschiedenen Python Modulen gearbeitet. Der erste Schritt war die Links und die Diagnose mittels RegEx herauszufiltern und mittels Request an den Link des Bildes letztendlich das Bild selbst zu erhalten. Im nächsten Schritt war es vonnöten diese Bilder zuzuschneiden und für die KI vorzubereiten. Einfach gesagt habe ich mittels einer OpenCV-Funktion die Anomalien in einem Bild ausgeschnitten und zu Graustufen verändert. Das führte letztendlich zu inkonsistenten Bildern in Graustufen, welche jedoch aufgrund von fehlender Skalierung nicht für das Machine-Learning verwendbar sind. Eine weiter wichtige Information für diesen Abschnitt ist auch, dass ich für Performancezwecke auch die Scripte aufgeteilt habe und deshalb auf einen „shared Memory“, auch geteilten Speicher, zugreifen musste. Dieser geteilte Speicher ermöglichte es mir einfach große Datenmengen, wie die Bilddaten selbst, zwischen Scripten auszuteilen.

* + - 1. **Labeln der Daten**

Das Labeln der Daten habe ich mittels JSON-Objekts umgesetzt, was heißt, dass das es ein Attribut Picture gibt, welches logischerweise auch ein Bild als Numpyarray repräsentiert. Das zweite Attribut der JSON ist die Diagnose, welches die Diagnose des Arztes selbst ist. Wichtig hinzuzufügen ist auch, dass ich einen großen Teil meiner Zeit in das Zuschneiden der Bilder auf das nötigste gelegt habe.

* + 1. **Datenbank-Backend**
       1. **Datenmodell und Struktur**

Als Datenbank haben wir eine No-SQL Datenbank verwendet, da es für unser Vorhaben perfekt in unser Bild gepasst hat. Durch die Funktionsweise sind wir nicht an ein reguläres Schema gebunden, sondern können einfach mittels JSON-Objekte an die Datenbank senden. Das Backend meiner API verwendet jedoch doch ein Schema, welches ein Bild und die Diagnose in einem JSON-Dokument haben will. Mit dieser Konfiguration ist es uns möglich eine einheitliche persistente Speicherung der Daten zu erzielen, was uns später vieles vereinfachen wird. Eine Implementation von Usern ist nicht Teil unserer Muss-Ziele, jedoch wollen wir, wenn genug Zeit vorhanden ist ein solches System wie in unseren Soll-Zielen umsetzten und somit weitere Funktionalität zu unserer Applikation bringen.

* + - 1. **Technologien und Tools**

Um uns das bestmögliche Backend zu machen habe ich auf modernste Technologien wie das Typescript-Framework [**Adonisjs**](https://adonisjs.com) gesetzt. Um eine Verbindung zur Datenbank aufzubauen habe ich auf das Package Mongoose gesetzt, welches mich nun bereits durch das 3 Projekt im Schulischen Umfeld begleitet. Der Techstack von Adonisjs ist nicht gering, bietet jedoch für sehrviele Situationen wichtige Funktionalität, welche das Entwickeln um einiges einfacher machen. Mithilfe der gut formulierten und instandgehaltenen Dokumentation ist das Einleben in das Framework besonders leicht gemacht.

* + - 1. **Datenbankoperationen**
      2. **Verschlüsselung der gespeicherten Daten**

Mithilfe der Hashing-Funktionen SHA-256 und „salting“ werden Passwörter auf dem Backend gespeichert, was heißt, dass es nicht möglich ist Passwörter anzeigen zu lassen. Datentransfer zwischen Client und Server wird auch mittels eines gehashten Wertes ermöglicht.

* + - 1. **Test und Validierung**

AdonisJS stellt bereits ein Framework zum Schreiben und ausführen von Tests bereit, was heißt, dass mittels CLI-Befehlen einfach Tests erstellen und ausführen kann. Eine wichtige Unterscheidung zwischen Tests gibt es jedoch, welche sich von den herkömmlichen unit-tests unterscheiden.

Unit-Tests prüfen wie gewohnt einzelne Komponenten und Funkionen des Codes, während Funktionale-Tests einen größeren Abschnitt der Applikation abdeckt und diese aus der Sicht des Benutzers ausgeführt werden. Um die tests laufen zu lassen muss man im vorhinein die Packages mithilfe von **npm i** runterladen und anschließend mit ***node ace test*** aktivieren, dies startet die Tests und gibt eine formatierte Ausgabe für den Benutzer in der CLI wieder.

* + 1. **Vorbereitung**
    2. **Aktionsphase**
    3. **Ende**

1. **Installation / Software deployment**
2. **Projektabschluss**

# Literaturverzeichnis

# Abbildungsverzeichnis