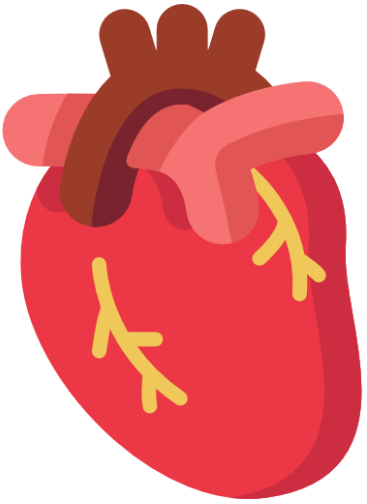


Software Development – Databases

Database Exam – Heart Disease Application

Due date: 03-06-2024

Video:
https://youtu.be/PUrCUQmycaU

GitHub Project:
https://github.com/christofferPerch/Software_1st_Semester
https://github.com/christofferPerch/AI-ML_Exam

Christoffer Perch Nielsen	Jonathan Braad	Mark Lundgaard Jensen	Oscar Tuff
cph-cn332@cphbusiness.dk	cph-jb442@cphbusiness.dk	cph-mj924@cphbusiness.dk	cph-ot58@cphbusiness.dk

Business Case

Objective:

- Develop an AI-driven application using machine learning to predict the likelihood of heart disease in individuals based on certain parameters, while also establishing an e-commerce webshop, allowing patients to purchase their medications online with ease. Finally a digital heart doctor that can consult on heart disease specific questions.

Market Analysis:

- Target Audience:
    - Individuals at risk of heart disease.
    - Consumers requiring regular prescriptions, elderly people, and those with mobility difficulties.
- Need Identification:
    - Heart diseases are the leading cause of death globally. Early prediction can help in preventive measures and reduce healthcare costs significantly.
    - Increasing demand for convenient access to medications, especially post COVID-19, with more consumers preferring online shopping.

Technology:

- AI and Machine Learning:
    - Utilize supervised learning algorithms (e.g. Logistic Regression, Random Forest and Neural Networks and OpenAI as our LLM for our generative AI model.
- Database:
    - Pinecone is used as a knowledge base for our digital heart doctor, the admin is able to train it on various sources of data which are then embedded and stored in the vector database Pinecone.
    - Microsoft SQL Stores all the data related to the webshop, user data, chatbot settings, chatbot history and some stored procedure for defragmentation.
    - MongoDB stores data that might be unstructured such as pdf documents for the knowledge base of our generative AI. Also the predictive machine learning models are stored in MongoDB with relevant metadata.

Challenges:

- Data privacy and security concerns.

- Securely storing patient data, ensuring compliance with data protection laws like GDPR.
- Need for continuous model updates and validation against new medical research.

FURPS (Functional and non-functional requirements)

Functionality:

- The system must allow users to input a wide range of health-related data including age, gender, lifestyle factors and medical history.
- The application must be able to process the input data using machine learning algorithms to predict heart disease risk.
- Capabilities for user registration, login and management must be included.
- The system must allow the admin to upload various sources of data such as pdf, csv, txt and doc files for training the chatbot.
- The system must allow the admin to train the digital heart doctor based on the sources that are uploaded.
- The application must allow the admin to change settings related to the digital heart doctor such as the initial message, base prompt, model and temperature.
- The admin must be able to view the chat history with the digital heart doctor from all users and search for messages.
- The system must allow the admin to check the fragmentation percentage of the database, and allow the admin to rebuild or reorganize specific indexes.
- The admin must be able to retrain prediction models based on data from the database.
- The platform must list medications with descriptions, prices and availability status.
- The users must be able to add items to a shopping cart and proceed through a checkout process.
- The admin must be able to view past orders, track and manage current orders.

Usability

- The user must be logged in to access the functionality of the application.
- The user interface should be simple and intuitive to interact with.
- The webshop should be designed for easy navigation, with a responsive design.

Reliability:

- The system should be available 24/7, with a downtime of less than 0.1% annually.
- The e-commerce platform should ensure data integrity and transaction consistency with minimal failures.

Performance:

- Responses from the predictive model should be returned within seconds after the data submission.

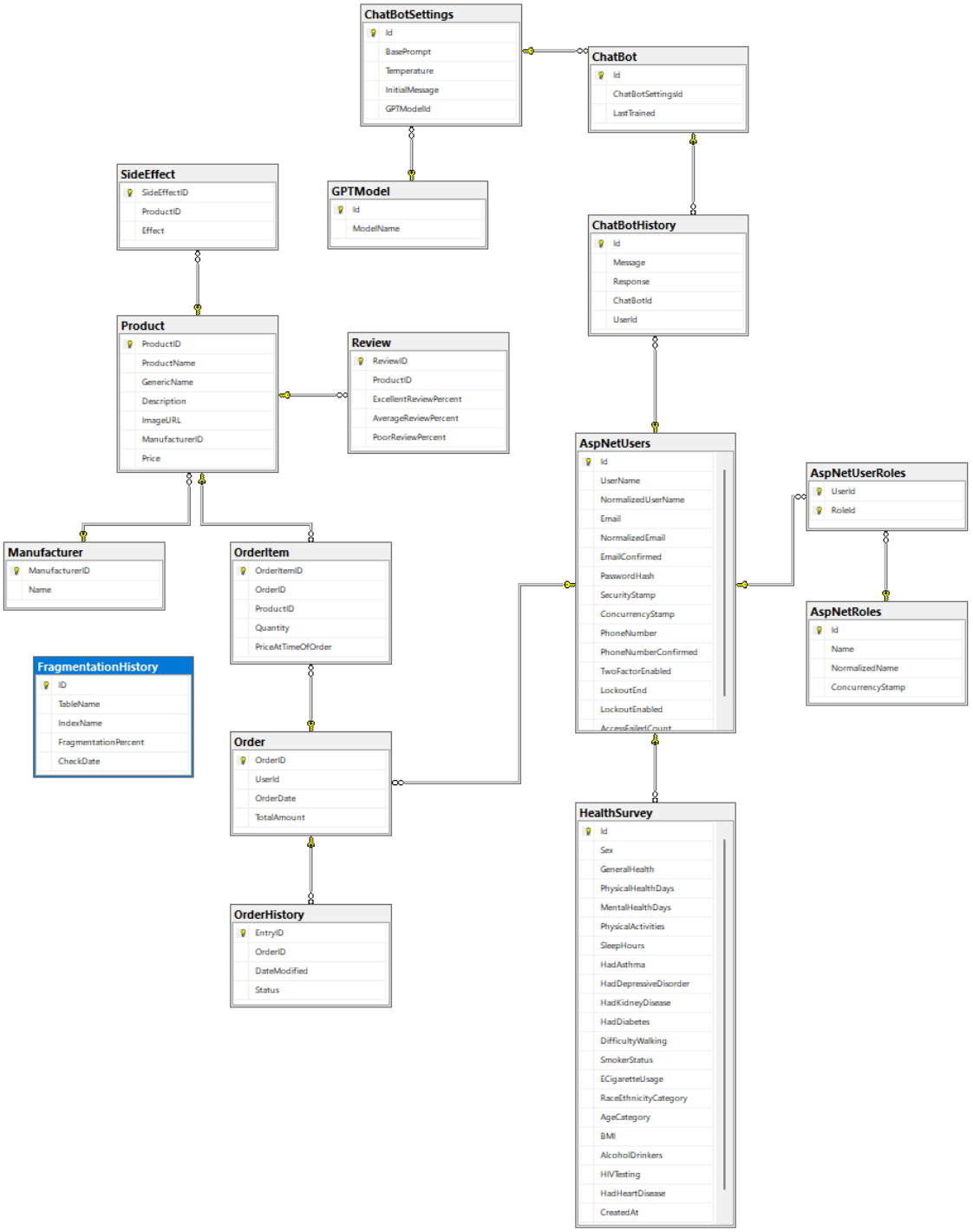
- The system should handle high traffic volumes, without degradation in response times.
- The system must use operations like searching, filtering and aggregation without affecting the response time by using indexing.

Supportability:

- The system should be easily maintainable, with capabilities for updates and upgrades. Documentation should be thorough and user-friendly.
- The platform should support easy integration with APIs.

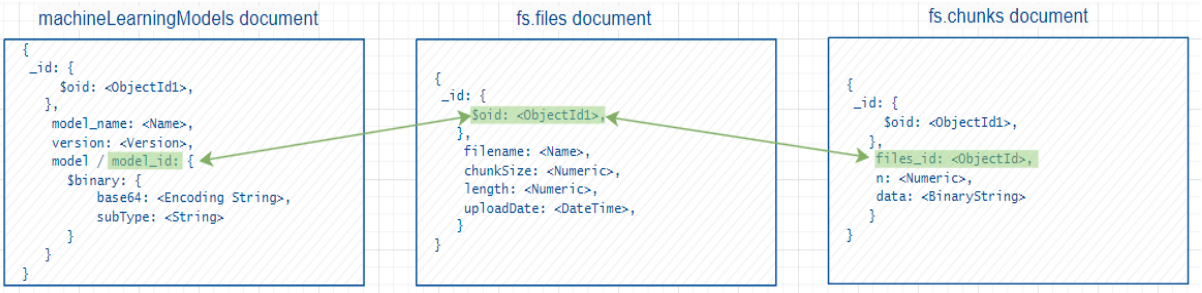
Diagrams

EER Diagram



Microsoft SQL is used as the main database for the application. It stores userdata, chatbot configurations and data related to the workshop. This is structured data which makes an SQL database the clear choice for data storage.

MongoDB Schema Design



MongoDB is used to store both machine learning models and files to use with the vector database for knowledge base before they are vectorized. The models have a model_name, version number and a binary base64 encoded model. The random forest model is saved with GridFS since it is too large to fit directly in the machineLearningModels document so it has model_id instead of model, which points to the corresponding file. Using MongoDB for this unstructured data gives flexibility to store the data with different metadata which might change in the future.

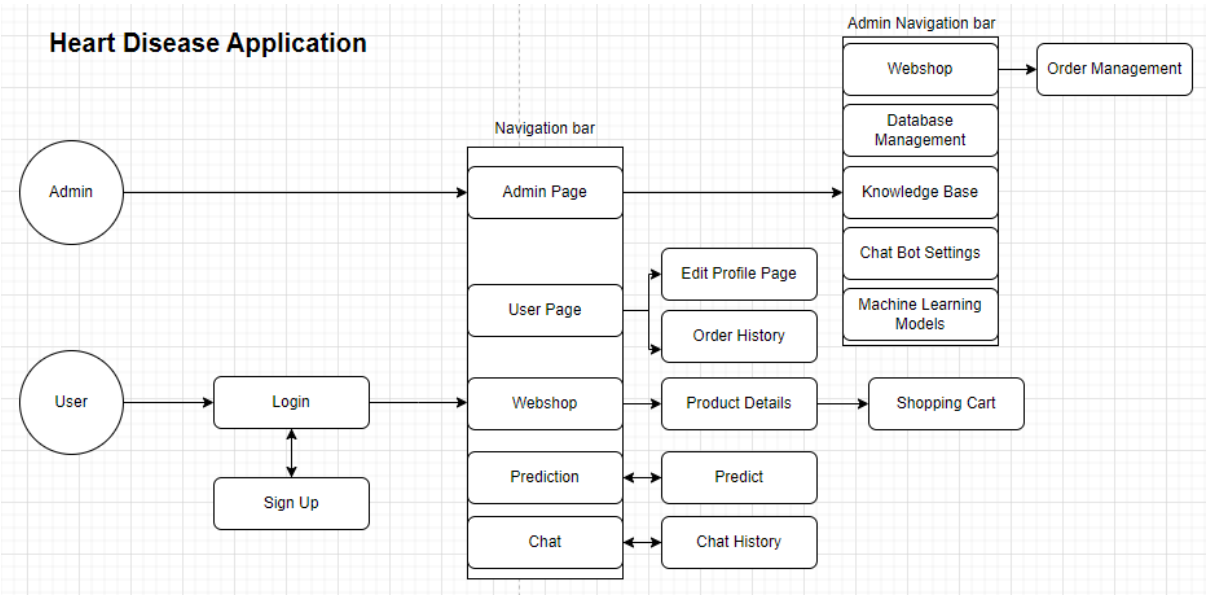
Vector Database

id	text_chunk	embeddings	cosine	filename
f2186482-8394-4696-a386-c4fc6f017c3d	“increase in cAMP (cyclic adenosine monophosphate)”	[-0.028436994, -0.009769647, -0.00426427601,-0.00724290498, -0.021486859,0.0274695754, 0.0135820378, -0.026502158, -0.022097861…]	[0.0027]	The_Heart_Anatomy_Physiology_and_Exercise_Physiolo.pdf
8f4c702c-ee06-4159-a4fc-fc26661b457a	“reticulum and the cisternae contain high concentrations of ionised Ca..”	[-0.00274657155, 0.000153152287, 0.00277097826, -0.0209312197, -0.015294889, 0.00269450387, -0.00727646239, -0.0239771791, 0.00475768698…]	[0.0019]	The_Heart_Anatomy_Physiology_and_Exercise_Physiolo.pdf
6cf89040-d956-40f0-8691-748b1c591c7b	“Oxygen extraction increases\nBlood flow to heart, muscle \nand skin increases\nBlood flow to brain increases (slightly)Blood flow to other viscera decreases18 Syed Shah, Gopinath Gnanaseg..”	[-0.0152014541, -0.00337409321, 0.0152801499, -0.0477947369, -0.0149784824, 0.0306914598, -0.01600153, -0.0279895626, -0.038928315, -0.013…]	[0.0012]	The_Heart_Anatomy_Physiology_and_Exercise_Physiolo.pdf

Above is a representation of how the data in the Pinecone vector database is stored. A vector database is purpose built for storing vector embeddings to use with Generative AI

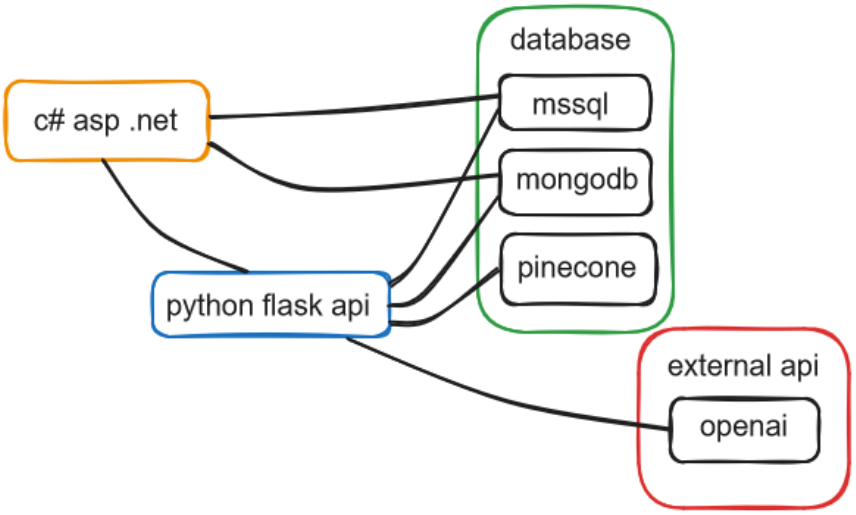
models. This is used to enhance the Generative AI doctor in the application with relevant data.

User Flow Diagram



The diagram represents the user flow and architectural structure of our Heart Disease Application. It details the interactions for two primary types of user roles: Admin and User.

System Architecture Diagram



The C# ASP .NET project connects directly to MSSQL and MongoDB to access and modify data while also using the python flask api for interacting with machine learning models. The

Python flask api uses all of the databases and also the external OpenAI api for generative AI models.

DataAccess Project in C#
In our solution, we have developed a DataAccess project, which the web application references. Within the DataAccess project, we have defined two interfaces: one for MongoDB and another for Microsoft SQL. These interfaces are implemented by two classes that provide the necessary CRUD operations for each database system. In the HeartDisease project we have used dapper for Microsoft SQL which is an ORM.
https://github.com/christofferPerch/Software_1st_Semester/tree/master/HeartDisease.DataAccess

Database Operations in our Project
Our project utilizes various database operations, including CRUD, as well as joins. The implementation of these operations is located in our Service folder, which can be accessed here:
https://github.com/christofferPerch/Software_1st_Semester/tree/master/HeartDisease/Services

In the Python project we have also worked with Microsoft SQL, MongoDB, and Pinecone. The script load_data_from_mssql.py extracts data from the HealthSurvey table. In chatbot_chat.py, we retrieve chatbot settings from Microsoft SQL and establish a connection to Pinecone. Additionally, chatbot_training.py utilizes both MongoDB and Pinecone. The relevant code can be found here:
https://github.com/christofferPerch/AI-ML_Exam

Stored Procedures
The stored procedures utilized in this project are available in the documentation folder of our GitHub repository:
https://github.com/christofferPerch/Software_1st_Semester/tree/master/HeartDisease/Documentation/MSSQL
These stored procedures include:

- A procedure for checking the fragmentation percentage and inserting the data into a table.
- A procedure for retrieving the latest data.
- Two procedures for rebuilding and reorganizing an index.

These procedures facilitate the effective management and optimization of the database.

Data Sources

In our project we have used two datasets from Kaggle. We used the dataset about heart diseases for our machine learning models and we used the medicine dataset for our webshop.

https://www.kaggle.com/datasets/kamilpytlak/personal-key-indicators-of-heart-disease
https://www.kaggle.com/datasets/singhnavjot2062001/11000-medicine-details

We used Python to insert the data into our Microsoft SQL database. We created the HealthSurvey table for the heart disease data. For the medicine data we had to change it a bit to achieve 2-3NF where it made sense and then we also used Python to insert it.