Date: \_\_\_8/17/2023\_\_\_\_\_\_

Name \_\_\_Stanley Li\_(netid: zl4636)\_

Name \_\_\_Zeyu Chen\_(netid: zc2796)\_\_

**Project Part 4**

**Total in points** (100 points total): \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Professor’s Comments:**

**Affirmation of my Independent Effort:**  Zeyu Chen Stanley Li

**1. Business Use Cases and Data-Driven Workflow-Based Database Application:**

For an insurance company, the primary goal is to assess risk and provide policies tailored to individual needs. With the advent of big data, insurance companies can now leverage vast amounts of structured and unstructured data to make more informed decisions.

Module 1: Insurance Quote Generation Customers provide personal health information, especially concerning diabetes, given its significance in insurance risk assessment. The system, equipped with a machine learning model, processes this data to generate a preliminary insurance quote. This model can be trained on historical data, factoring in age, health conditions, and other relevant parameters.

Module 2: Negotiation and Creation of Insurance Contracts Post the preliminary quote, insurance agents negotiate with customers. They use the system to generate contract options tailored to the customer's profile. This module utilizes the prompt engineering to enable users to interact with a database using natural language.

Module 3: Company Registration and Contract Management Insurance companies often collaborate with other businesses, like healthcare providers or corporate entities seeking group insurance. The registration of these entities and the subsequent contracts are managed by this module. This module utilizes the prompt engineering to enable users to interact with a database using natural language.

**A diagram of a contract

Description automatically generated2. Documentation of Business Use Cases:**

* Insurance Quote Generation: A customer provides personal health details, especially concerning diabetes risk. The system then calculates potential insurance rates.
* Contract Negotiation: After receiving a quote, the customer can negotiate terms with an insurance agent. The system assists the agent by suggesting suitable contract options based on the customer's profile.
* Company Registration: Companies that partner with the insurance firm can register their details. The system manages these details and the associated contracts

**3. Data-Driven Program Module:**

A data-driven program module is designed to harness the capabilities of machine learning models developed for diabetes classification and premium prediction. Utilizing a blend of Python and the Scikit-learn library, this module incorporates the RandomForestClassifier for diabetes risk prediction and Ridge regression for premium estimation. The module is structured to align seamlessly with the business use cases discussed earlier, ensuring a consistent and efficient data flow. A pivotal aspect of this module is its data pipeline management system, which is specially crafted to manage unstructured data. This system preprocesses, normalizes, and transforms the incoming unstructured data, making sure it's in the ideal format for the models to process.

Contrary to incremental learning, our approach is more straightforward. Every time a new user's data is entered, the system predicts for that specific user and then stores their information in the database. When another new user is introduced, the system retrieves the previously stored data and retrains the model entirely from scratch. After this retraining, the model predicts for the new user and subsequently saves their details. This method ensures that our model is always trained on the most recent and comprehensive dataset, allowing for accurate and up-to-date predictions. This comprehensive strategy not only simplifies the process of data ingestion and analysis but also places the insurance company at the cutting edge of data-driven decision-making.

**4. End-to-End Workflow-Based Application:**

The application leverages the VUE frontend framework, the Python-based server framework - fastapi, and the AliCloud RDS MySQL Serverless hosted in the cloud as our database solution, achieving a decoupled front-end and back-end architecture to enhance the system's performance. For interactions between the backend server and the cloud-based MySQL database, we utilize the SQLAlchemy library, implementing an Object-Relational Mapping (ORM) framework. By adopting the ORM framework, developers can remain focused on business logic and application development without delving deep into the intricacies of the underlying database. This approach not only accelerates development but also minimizes errors, fostering code maintainability and scalability.

The system is segmented into three distinct modules. In the first module, when a user (e.g., Stanley) logs into the platform, they are prompted to complete a comprehensive form capturing vital health metrics. This data, spanning age, lifestyle habits, and specific health conditions, is subsequently stored within a relational database. Central to this module is a machine learning model. This model, trained on a vast repository of health records and corresponding insurance claims, evaluates Stanley's health metrics to forecast potential health risks. By doing so, it can generate a personalized insurance quotation, ensuring that Stanley is charged a fair premium reflective of his health status. Moreover, each time a new user's complete data set is incorporated into the system, it automatically triggers an update to generate a new machine learning model.

Modules two and three are anchored on the GPT-3.5-Turbo, a sophisticated model from OpenAI. Using prompt engineering, these modules manifest a complex insurance contract and associated data management system. They are architected to enable users to interact with various data tables through natural language commands. Whether it's initiating a new contract, updating details in the 'customeraccountmember' table, or querying the 'contractbenefit' table, operations are streamlined and intuitive. This is achieved through advanced natural language processing algorithms that decipher user inputs and transmute them into database operations. Architecturally, it's tailored to facilitate interactions with various data tables, such as 'contract', 'contractbenefit', 'customeraccountmember', and 'customerassociate'.

The user interface is user-centric, permitting seamless data input and management. For instance, when an insurance agent needs to add a new contract post-negotiation, they can effortlessly input the details. These particulars are then processed and safely stored in the SQL relational database system. This database not only ensures structured data storage but also offers efficient retrieval mechanisms, fortified with stringent security protocols.

**5. Documentation and Optimization:**

our insurance premium calculation application, tailored to the project's fourth specification, is a testament to modern software engineering principles combined with advanced natural language processing capabilities. When a user inputs health details, the system taps into the Online Transaction Processing (OLTP) database to fetch pertinent historical data. This data, amalgamated with the user's input, undergoes processing to deduce an insurance premium. The system's seamless integration with the Operational Data Store (ODS) ensures that all computations are based on the freshest data available.

To enhance the system's responsiveness, we've meticulously optimized our database interactions. Techniques such as query optimization and denormalization have been employed to bolster read performance. Furthermore, the application leverages SQLAlchemy, a robust Object Relational Mapping (ORM) framework, for streamlined database interactions.

However, the standout feature is the integration of the GPT API for prompt engineering. Recognizing that users might have diverse ways of phrasing their queries or commands, the GPT API is employed to interpret and process these natural language inputs. This ensures that even if a user phrases a request differently, the system comprehends the intent, making the application more user-centric and efficient.

**6. Reference Architecture Documentation:**

The reference architecture (RA) for our insurance system is a holistic blueprint that intertwines business operations, application functionalities, the DIKW (Data, Information, Knowledge, Wisdom) pyramid, and infrastructure, all while leveraging the capabilities of the GPT API.

The RA is grounded in principles that prioritize customer-centricity, data precision, and operational simplicity. It's meticulously layered, demarcating distinct functionalities and orchestrating data flow. At the business layer, the system is primed to offer insurance quotes, drawing from user data and historical records.

The application layer is where the GPT API shines. While it houses the user interface, a machine learning model for predictions, and database interactions, the GPT API ensures that user interactions are intuitive, allowing for natural language inputs and commands. The DIKW pyramid elucidates the transformation of raw data into actionable insights, such as insurance quotes.

From an infrastructure perspective, the system is built on robust technologies capable of managing vast data volumes while ensuring swift responses. Governance is a cornerstone of the RA, with stringent data governance protocols in place to uphold data quality and preempt potential data discrepancies. Measures to forestall data loss are integrated, and rigorous checks manage data throughout its lifecycle.

Bias mitigation is paramount. The machine learning model undergoes periodic evaluations and updates to counteract biases. While the system proffers insurance quotes, the final adjudications involve human agents, ensuring a balanced blend of automation and human judgment, thereby upholding fairness and transparency.

**Github:**

**https://github.com/marvinchen123/database2433\_final\_project.git**

Explanation: I have removed the links to the cloud database and the OpenAI API key from the code I uploaded to GitHub since GitHub is a public platform. However, the compressed file I uploaded on Brightspace contains the complete program, including all the mentioned keys.

**Demo of the projects:**

**Interface：**

A screenshot of a computer

Description automatically generated

**Module 1 Interface：**

A screenshot of a computer

Description automatically generated

**Enter customer information.**

A screenshot of a computer

Description automatically generated

**Customer info was successfully inserted.**

A screenshot of a computer

Description automatically generated

**Enter Diabetes information**

A screenshot of a computer

Description automatically generated

**Diabetes information is inserted successfully and returns the diabetes risk prediction value**A screenshot of a computer

Description automatically generated

**Enter Premium information**A screenshot of a computer

Description automatically generated

**Premium information is successfully inserted and returns the estimated premium forecast value**A screenshot of a computer

Description automatically generated

**Module 2 interface**

A screenshot of a computer

Description automatically generated

**Enter the following data using natural language commands:**

Contract Benefit:

• ContractID: 7001

• ContractBenefitID: 7001222

• Contract Details: Preventive Care

• ContractBenefitID: 7001223

• Contract Details: Dental

Please insert the above data into the 'Contract Benefit' table.

A screenshot of a computer

Description automatically generated

**Data inserted successfully**A screenshot of a computer

Description automatically generated

**Use natural language commands to query the data of the Contract Benefit: table:**

Please check the contents of the Contract Benefit tableA screenshot of a computer

Description automatically generated

**Query success**A screenshot of a computer

Description automatically generated

**Module 3 interface**

A screenshot of a computer

Description automatically generated

**Enter the following data with natural language commands：**

Account Information:

• AccountID: 1001

• AccountName: Apple

• Company: Apple Inc.

Insert the above data into the 'Account' table.

A screenshot of a computer

Description automatically generated

**Data inserted successfully**A screenshot of a computer

Description automatically generated

**Query Apple's data with natural language commands:**

Check to see if there is any Apple account information and return all of its information if there is.

A screenshot of a computer

Description automatically generated

**Query success** A screenshot of a computer

Description automatically generated

**Acknowledgements:**

I would like to acknowledge the following sources that contributed to the completion of this report: Fundamentals of Database Systems-Pearson (2015), Wikipedia, new Bing.