**Enterprise Data Architecture Part 4: End-to-End Solution Integration**

*Rahul Patel (rbp320), Sophia Tsilerides (smt570)*

*New York University CSCI-GA.2433-001 - Summer 2020*

*Github link:* [*https://github.com/sophia-tsilerides/CSCI2433-termproject*](https://github.com/sophia-tsilerides/CSCI2433-termproject)

1. **Abstract**

The following paper describes the final process of engineering a front-end platform for our Insurance database system and the finalized reference architecture and data governance measures it takes into consideration.

The Introduction discusses the previous parts of the project this paper is built upon and how they together create an end-to-end platform. Section III describes use cases of our Physical Model and their implementation in the database system. Section IV finalizes our previous Data Lake Reference Architecture and Section V documents our data governance principles. We conclude the paper with a discussion of the system and the role it plays in the Insurance company.

1. **Introduction**

Our project began with the reverse engineering of a given logical model for an insurance company which was converted into a conceptual model for a business-use database system. The Entity - Relationship Model is a critical component of the Enterprise Data Architecture. Good E-R models lead to effective database solutions for business problems, ease of application development for better user experience and practical governance implementation guidelines. Specific support cases that the E-R model upheld were also discussed in this stage. At a high level, customers have accounts maintained by agents who provide the customers with contracts for products in exchange for payment.

Conceptual models can not be used to directly implement a database. A Logical Model allows us to convert the conceptual representation of our database into a model that can be directly implemented. Our Logical Model contains the relations Accounts, Agents, Customers, Agreements, Payments, Rates and Claims. Together, these constructs make up the entities of our conceptual model, and the relations of the Logical Model given, to represent an insurance company. In addition to these relations, we have implemented a Data Lake which will better serve the customers by allowing forecasts for new products. Forecasting with a Data Lake and using machine learning can improve and refine the underwriting process for the insurance industry. We detailed the creation of the machine learning model applied to the data that performs analytics and make predictions.

This paper considers an optimized physical database model that allows the database system to scale and support the business by anticipating growth and failure of the system. This allows for a durable and secure set of processes to digest and analyze structured and unstructured data. It also details the front end implementation of the database that gives product predictions to complete the end-to-end system. These product predictions for new and existing customers help drive business decisions and improve user experience and organizational excellence by anticipating the needs of customers before they know it themselves.

1. **Business Use Cases**

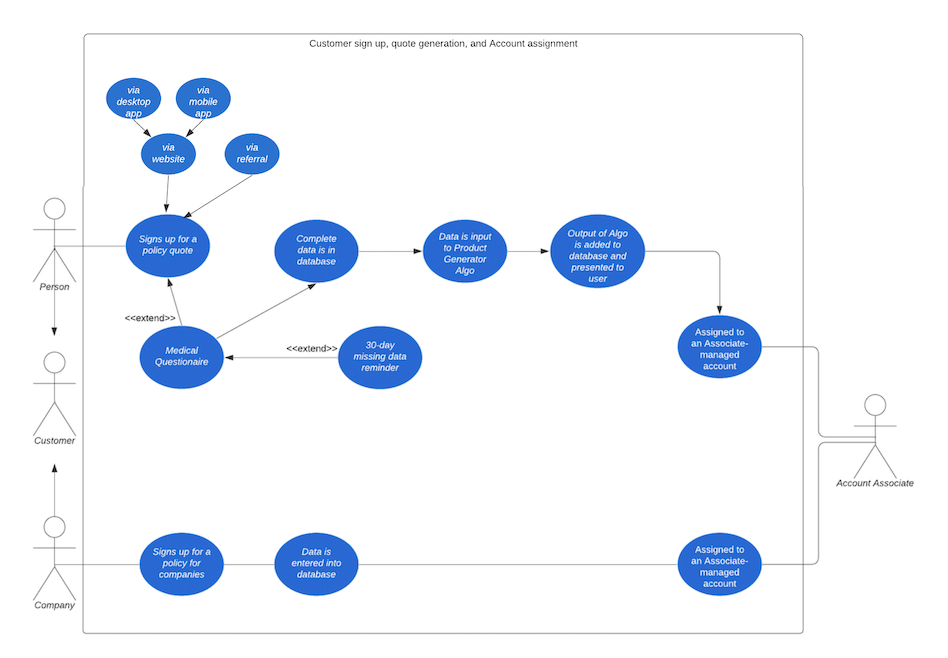
The objective of our system is to allow a Customer to obtain an insurance quote and policy automatically. Beneath the documented Use Cases are the complete processes illustrated by a Use Case Diagram. The formal rules of the Use Case diagrams were derived from Warren Lynch’s article *All You Need to Know About Use Case Modeling*[[1]](#footnote-1).

Customers can come from the website directly or through a referral. They can also inquire through the mobile app or a desktop version of the website. A Customer signs up for an account online or by phone. The computer or operator takes down their information and inputs it into the user interface of the database system which processes the information and inputs it into the CUSTOMER table. If the Customer has their medical records, they can fill out a questionnaire online to input the data needed to run the Machine Learning algorithm that predicts their quote and policy. Otherwise, these records are kept NULL and updated later. If records are NULL for more than 30 days, a reminder is sent to the customer to the email address provided. If a Customer is a Company, this survey does not apply, they are automatically issued a policy for companies.

Once the algorithm processes a Customer’s information, the SUGGEST\_PRODUCT table is populated with one of the products offered by the company, also in the PRODUCT table and is presented to the Customer. Additionally, the customer is assigned to an Account based on their region and product offering so that an Associate can manage them properly.

This back-end database is linked to the front-end by a “Get Quote” button present on either of the platforms - mobile app, website, or internal UI for the agent. Once clicked or requested for, the customer receives a quote for the suggested policy. In case they wish to amend or completely change the policy, a new quote will be generated by inputting the details provided by the customer.

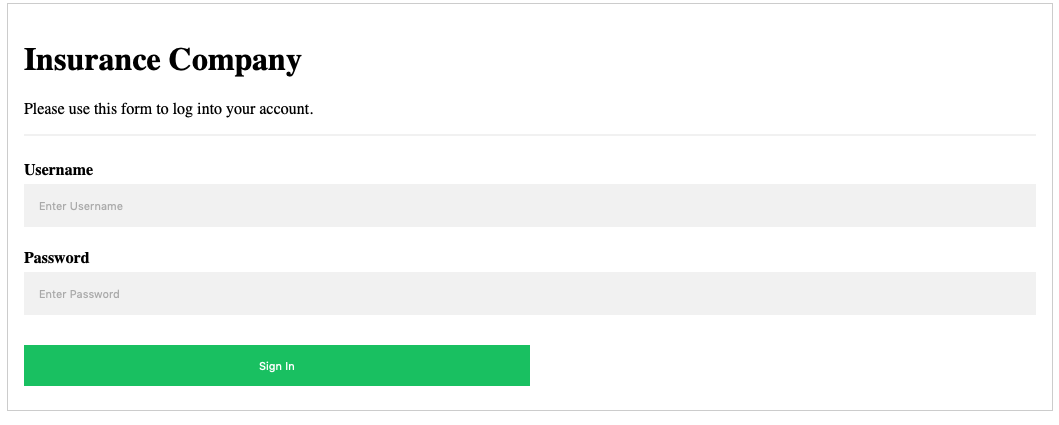
*Figure 1: Use Case model of Customer sign up, automatic quote generation and assignment of Customer to an account to be managed by an Account Associate*



To implement this business use case on the front-end side of our system we first created three prototype HTML web pages to serve the clients needs: generate a quote, select a product or update existing customer information. The HTML and CSS files can be found in our [GitHub repo](https://github.com/sophia-tsilerides/CSCI2433-termproject/tree/master/website).

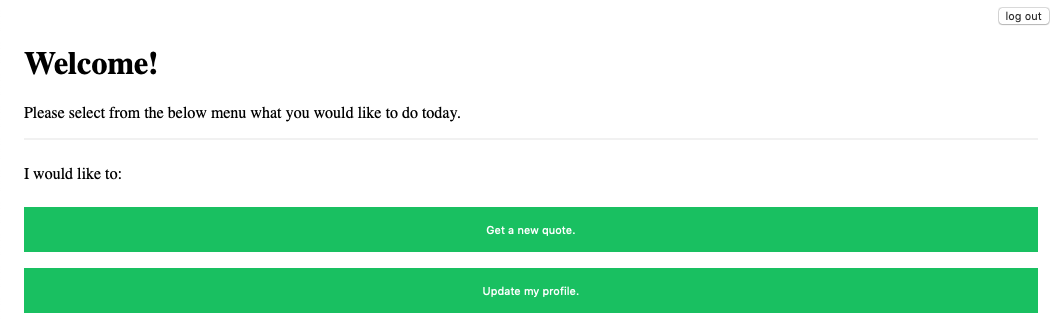
The Log In page is a user’s landing page for access to their account information and plan information.

*Figure 2. Log In landing webpage*

**

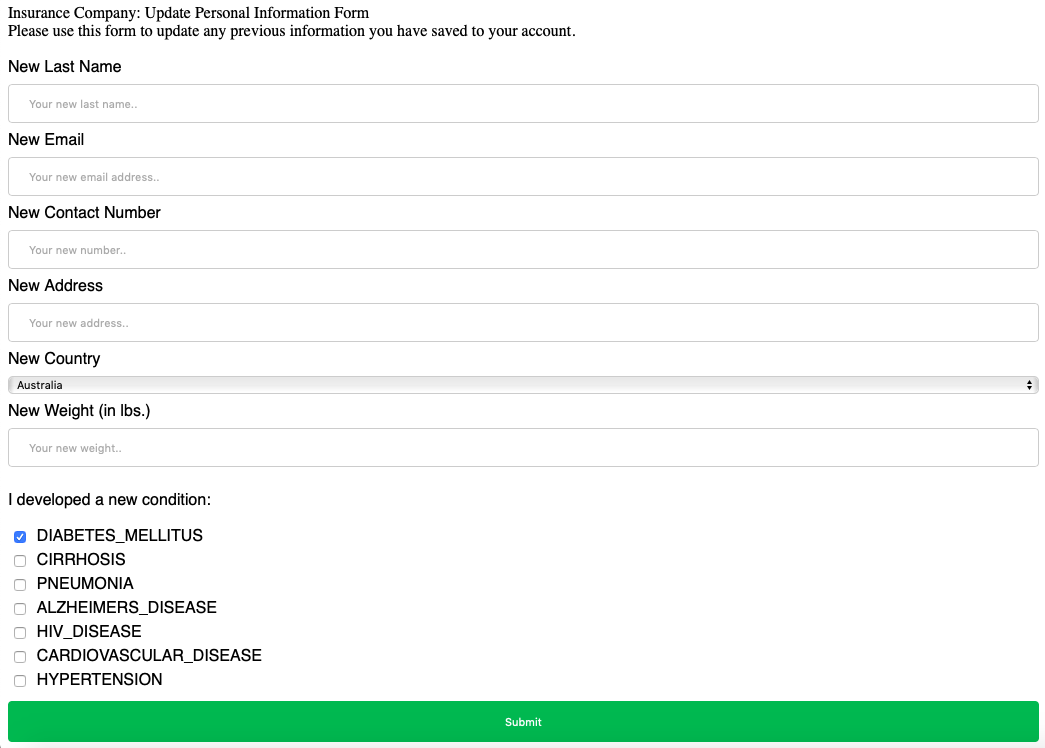
Users are then greeted with a Welcome Page where they can select from a menu of options if they would like a new quote or would like to update their profile.

*Figure 3. Customer Welcome Page*

**

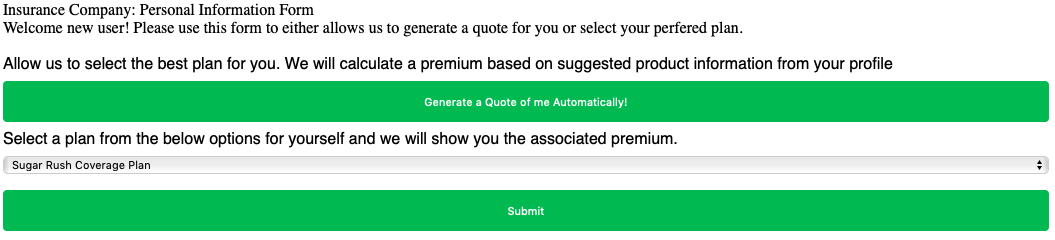
The Update Form page allows a customer to make any changes they would like to their account to reflect their current needs or life changes that they would like to tell the company about.

*Figure 4. Customer Update Form webpage*

**

Lastly, the Quote Generation form gives a customer two options. They can either have their premium automatically calculated based on suggested product information in their profile, or they can select a product from a list to view an associated premium.

*Figure 5. Customer Quote Generation Form webpage*

**

With the client having entered their request, a Java script page with PHP connectivity to the RDS host server receives the client’s query and responds to the customer with an insurance quote after processing their information through our Machine Learning algorithm.

*Implementation of user interface*

* MAMP was installed on our local machine to host the web files on Apache’s web service. If and when the application is scaled up, we can register for a DNS and access the server from anywhere. This would mean that employees and admins of the insurance company wouldn’t have to host separate files on their local machines or virtual machines such as the ones on the cloud (Amazon’s EC2 instance). This will make data and file management centralized.
* Next, the .html, the .css, and the .php files were hosted in the parent directory, htdocs and can be accessed via port :8888.
* The user interface as governed by the .html and the .css files is simplistic in nature and is easy to understand. We chose this design as a prototype to showcase the overall end-to-end nature of the application.
* The .php files hold the key to session management. For every user login, the credentials are used to authenticate and establish the current session. This is done by executing an sql query that makes use of the user’s input of username and password.
* Each .php page that executes a database query must include config.php file at the beginning. This file holds the required parameters to connect to our primary database that sets on AWS’ RDS instance.
* Subsequent .php pages such as quote\_form and update\_form get the logged in user’s username.
* This username along with the corresponding form input variables on that page are used to query the database to ensure data retrieval and/or modification of only that particular customer. Login, Customers, Product, and Suggested\_Product tables are joined depending on the appropriate primary and foreign key relationships to get the specific task performed.
* Lastly, the user can terminate their current session by clicking on the logout button provided on the top right corner of the customer’s home page.

A demo of this implementation is provided in Appendix A.

1. **Reference Architecture**

A visual representation of our Data Lake Reference Architecture is shown below.

*Figure 6: Conceptual Model of our Data Lake Reference Architecture.*

A screenshot of a map

Description automatically generated

Our data sources begin in a raw data store without any transformations and are stored in a Data Lake. The structured data is directly imported to the database, however the unstructured data goes through an analytical transformation process to make the data consumable. Both data is then inserted to the database systems. Data from the host server is imported into R for use in a data mining process that performs multinomial regression to make predictions for Customer Products. The output prediction for each row is then stored as a new entry into the SUGGESTED\_PRODUCT table using a SQL query executed in R. The data is then stored on the host server where the database resides. This provides a secure and durable way to ingest and store structured and unstructured data.

The Customers table in the schema is the heart of the database. The primary business use cases revolve around this table. Initially, we had 10K records in the Customers table which have now been increased to 100K records. From a business perspective, our customer base has increased 10 times of what it was in the last release. We are expecting that in the future, the database will have to be scaled up further to accommodate business growth. As a result, the number of times the Customers table is being queries has also increased. Typically, the table is queried to fetch the customer record of a particular customer using the primary key Customers.CUSTID. Furthermore, the customer or the agent may want to inquire about the policy rate for which the Customers, Suggested\_Product, and the Products table need to be joined. Another case could be for analytical purposes. The business might query the database from time to time to extract insights about all the Customers who have a ‘Basic Coverage Plan’ for instance. For such types of operations the most frequently accessed columns are CUSTID and PLANNAME. As and when the database is scaled up, it becomes more and more challenging to retrieve records while keeping I/O operations and cost of retrieval in check. Thus the need to optimize the schema especially for frequently accessed indices.

1. *Indexing*

There are a number of indexes created keeping in mind specific business use cases.

The most frequently accessed column is Customers.CUSTID. This was defined to be the primary key earlier, however due to partitioning constraints to include all columns in the unique key, the primary key constraint on CUSTID was dropped. Instead a new UNIQUE KEY (CUSTID, PREVIOUS\_PLAN) is created. Out of these, the database by default makes the first one i.e. (CUSTID, PREVIOUS\_PLAN) to be the clustered index.

Next, a couple of composite indexes are manually created to aid data analytics. Index (GENDER, PREVIOUS\_PLAN, CUSTID) is created to speed up extraction of those customers that belong to a particular gender and have the same plan.

Lastly, Index (STATE, PREVIOUS\_PLAN, CUSTID) optimizes the retrieval of those customers that have the same plan and live in the same state.

1. *Partitioning*

We have used horizontal partitioning of the table Customers. One hundred thousand records are partitioned for a specific use case - to group together customers that have the same plan. This use case seems most relevant and most accessed. The Customers table is partitioned by a list of values each corresponding to a distinct plan name. Thus there are 7 partitions made to the table.

1. *Results of Optimization*

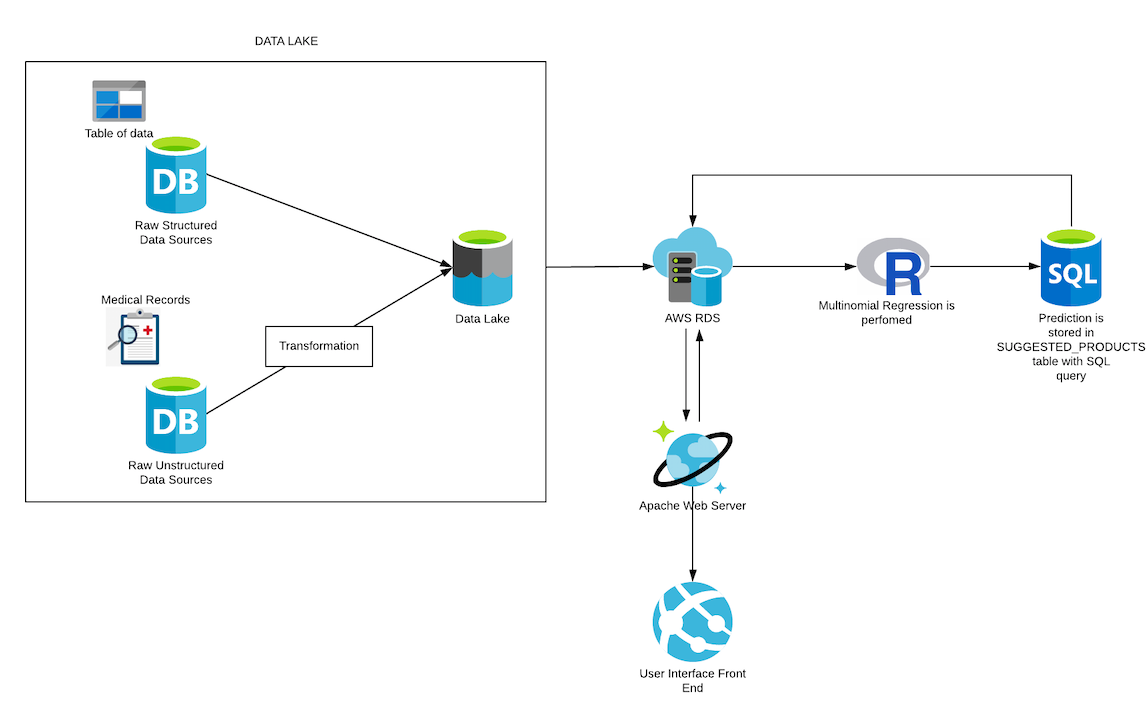
|  |  |  |  |
| --- | --- | --- | --- |
| Query | Avg. fetch time before optimization (in ms) | Avg. fetch time after optimization (in ms) | Improvement |
| SELECT \* FROM Customers; | ~ 2622 | ~ 2291 | ~ 12.6% |
| SELECT \* FROM Customers where GENDER = 'L' AND PREVIOUS\_PLAN = 'Peace of Mind Coverage Plan'; | ~ 145 | ~ 103 | ~ 28.9% |

1. **Modifications from Originally Proposed Model**

To accommodate the reference architecture and use cases described in this paper, a few changes from our original logical and physical model have been made. Namely:

* *Addition of LOGIN table to schema.* After a Customer signs up for an account online or by phone, Log In credentials are stored in the LOGIN table of the database along with a “type,” a field signifies whether the user is a customer or an admin. “Type” is automatically assumed depending on whether the welcome page is customer.php or admin.php.
* *Updating the Reference Architecture to include UI components*. A front end user interface is developed using technologies such as HTML, CSS, and PHP. The code files for these rest on Apache web server and hence the need for an inclusion of these components in the reference architecture.

*Figure 7: Finalized Conceptual Model of our Data Lake Reference Architecture.*



1. **Data Governance**

For database systems with significant amounts of data, data governance plays a key role in the maintenance and longevity of the system. Governance requires data storage to be persistent, inexpensive, reliable and shareable storage methods with relatively rapid access time[[2]](#footnote-2). We adhere to these principles for the following reasons:

* Our data is persistence because it persists after power is removed in Azure.
* Our data is inexpensive to store when measured on a dollar per Gigabyte or dollar per Terabytes basis.
* Our data is reliable because it does not have to be replaced due to excessive errors.
* Our data is sharable by facilitating Azure accounts among many users.
* Our data is accessible as it can be accessed within a short period of time.

These points and others are further addressed in the individual topics of data governance below.

1. *Data Quality Management*

Data Quality was managed at every part of the project planning stages, from modeling data types in Entity Relationship Modeling to type checking and RegEx processes in our web scraper. This process is known as data profiling, where we provided systematic documentation on the design and content of the data in our database[[3]](#footnote-3). It is an important part of the database management system process to understand the quality and nature of the data before undertaking the machine learning part of the project. Some specific aspects we looked at include: the overall size of the data, missing values, relationships amongst columns (functional dependencies) and relationships between tables (foreign key constraints).

The results of our data profiling stages were used to develop rules to check data quality and rules to correct data problems during the first stages of our ETL pipeline.

1. *Managing Data Loss and Data Leakage*

While most security measures of our database were configured with the help of the vendor of our cloud platform, we took other measures for having security within the database system itself. In general, employees of the company should not have access to personal information like social security numbers, credit card numbers and other financials, and health records of customers. To combat this, we have made the key of our CUSTOMERS table a synthetic Customer ID instead of SSN. Other sensitive information could be hashed and translated by a function *f* for record keeping and viewing by company employees.

The inputs to our machine learning model were also carefully selected so that there is no data leakage. A leak is a situation where a variable collected in historical data gives information on the target variable—information that appears in historical data but is not actually available when the decision has to be made.[[4]](#footnote-4) To do this we had to ask ourselves: “What information would we have only known after the target results became clear?” and consequently removed those attributes from the model. To do this, we had to understand where our data was coming from, how to get it, what it looks like, and it’s limits. This left us with the following input attributes for target product prediction:

* Attributes relating directly to the customer’s medical vitals and metrics: HEIGHT\_IN, WEIGHT\_LBS, PULSERATE\_PER\_MIN, BODY\_TEMP\_DEG\_F, BP\_SYSTOLIC\_MM\_HG, BP\_DIASTOLIC\_MM\_HG, CHOLESTEROL\_MG\_PER\_DL, BLOODSUGARFASTING\_MG\_PER\_DL, and BLOODSUGARAFTERMEALS\_MG\_PER\_DL
* Binary attributes indicating known health issues of a customer: DIABETES\_MELLITUS, CIRRHOSIS, PNEUMONIA, ALZHEIMERS\_DISEASE, HIV\_DISEASE, CARDIOVASCULAR\_DISEASE, and HYPERTENSION
* Previous Plan of a customer (this is the prediction we try to mine, described in Section VI): PREVIOUS\_PLAN

1. *Addressing Bias in Dataset*

Our data was carefully selected to avoid unknowingly training our machine learning model to give better products or quotes to people of certain demographics. To avoid this, race and location were intentionally left out of the modeling inputs as they contribute to ethical implications the model might indirectly learn while recommending products to customers.This was so that we don’t have more data on certain regions, people, etc. than others. It would not be a true random sample if our model learned from a bias data set. With an unbiased dataset, we could learn some kind of truth and be able to make predictions for the whole population.

Addressing bias and ethical practices in datasets and models has become an important part of data governance and should be constantly monitored. Thus, further work on unbiasing the data could be done. A bias correction method based on estimating the probability that an example is selected into the sample can be applied by using rejection sampling to obtain unbiased samples of the correct distribution.

1. *Managing the Data Lifecycle*

Today, data is growing at an exponential rate. While we have included a minimal amount of data in our database for exemplary purposes, the data is actionable which is where time critical decisions, like product suggestion generation, take place. It relies on streaming data rather than historical data and therefore the data is an asset and goes through the data lifecycle. It is through the data lifecycle that the data can be made scalable and flexible for our machine learning algorithm.

The steps of the data lifecycle are cyclically stated as storage, processing, analytics/machine learning, archive and create according to AWS[[5]](#footnote-5). The below sections document how we plan to manage these stages.

1. Storage

The physical implementation of the conceptual model can be described in the following sequential phases: Hosting the database on Amazon AWS Cloud platform, Connecting to AWS RDS using a client machine,Data Lake Storage, and Extracting and Loading of data into the database. Each phase describes individual processes to store, load, extract, transform, and mine the data. The process is created from our Logical Model and has been expanded to optimize data storage with the use of indexing.

1. Processing

Our modeling process supports structured *and* unstructured data. The Customer table has many attributes that are populated from the Data Lake described in the section below to the EDA. The predictions for products for customers are the output of the Data Lake and stored in the SUGGESTED\_PRODUCT relation. Data from our cloud hosted repository is extracted from the structured and unstructured data sources and loaded into the primary database that the Logical Model exemplifies. The structured Comma Separated Values file with structured data can be directly consumed by the target database, but the unstructured data will go through transformation first from HTML to CSV, and then from CSV to the database via ETL.

To complete our Data Lake, we utilized sample patient medical records from the Agency for Healthcare and Quality[[6]](#footnote-6). Electronic health records (EHRs) contain detailed clinical data stored in proprietary formats with non-standard codes and structures. Taking advantage of variable names and values as key-value pairs in doctor’s notes, we were able to extract useful information from this unstructured data source.

The Agency for Healthcare and Quality can not realistically give up real EHRs as they are private and can not be used without a license. Instead, we web scraped their six sample patient medical records to use for exemplary purposes in this paper. Our web scraper was built using python’s BeautifulSoup module, a package for parsing HTML documents. It can be found in our [GitHub repository](https://github.com/sophia-tsilerides/CSCI2433-termproject/blob/master/Patient_Data_Scraping_and_ETL.ipynb). Then, python code was written to transform the dirty data into usable structured data within the script. The complete dataset is loaded into the cloud platform hosting our target database.

1. Analytics / Machine Learning

We have implemented a Data Lake which will better serve the customers by allowing forecasts for new products. Forecasting with a Data Lake and using machine learning can improve and refine the underwriting process for the insurance industry. Old processes could take weeks to come up with a premium for a potential customer, who might churn during the process of waiting for a response from the company. Instead, creating and using models learned from past data on an individual’s accessible history could offer the potential customer a quote in minutes rather than weeks. This process takes away the invasive and time consuming medical testing aspect of the insurance industry while also giving a more reliable and comprehensive view of the insured. It can help better classify and determine risks to help an underwriter deal with the complexity of categorizing thousands of applicants with different backgrounds and needs. It will also help view risk factors more confidently and standardize the process of scoring an applicant companywide to avoid misunderstandings and contradictory results. More accurate risk assessments not only can better customer experiences, but also have significant cost benefits to insurers.

The Data Lake stores attributes of customers and non customers in its purest form. The Logical Model supports the Data Lake by storing the cleaned and useful version of the data as information. The data is transformed and analytics are applied to make it meaningful for the EDA. In the EDA, one CUSTOMER may have one or more SUGGESTED\_PRODUCTs and each SUGGESTED\_PRODUCT belongs to one and only one CUSTOMER. Additionally, one PRODUCT may be a prediction for one or more SUGGEST\_PRODUCTs, but each SUGGEST\_PRODUCT predictions one and only one PRODUCT at a time.

1. Archive

Our data sources begin in a raw data store without any transformations and are stored in a Data Lake. The structured data is directly imported to the database, however the unstructured data goes through an analytical transformation process to make the data consumable. Both data is then inserted to the database systems. Data from the host server is imported into R for use in a data mining process that performs multinomial regression to make predictions for Customer Products. The output prediction for each row is then stored as a new entry into the SUGGESTED\_PRODUCT table using a SQL query executed in R. The data is then stored on the host server where the database resides. This provides a secure and durable way to ingest and store structured and unstructured data.

1. Creation

Business critical applications and how they have been integrated within our system has been described in detail in Section III so will not be repeated here.

1. **Conclusion**

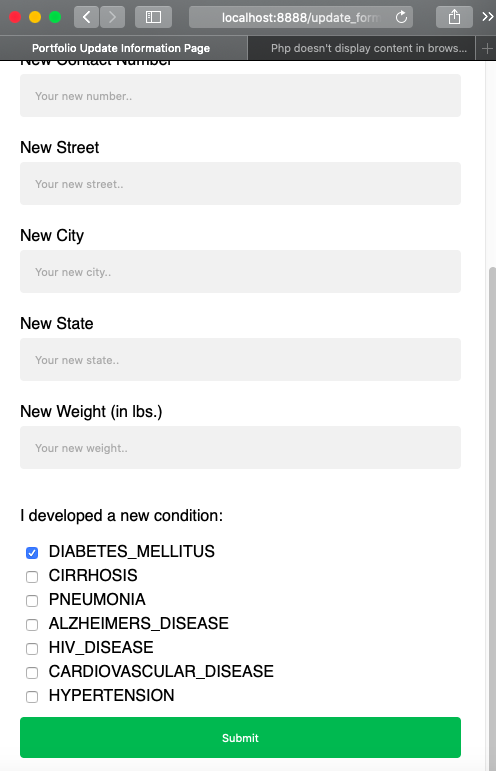
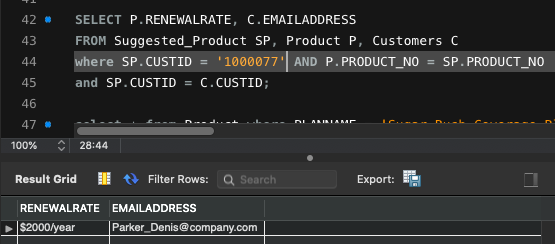
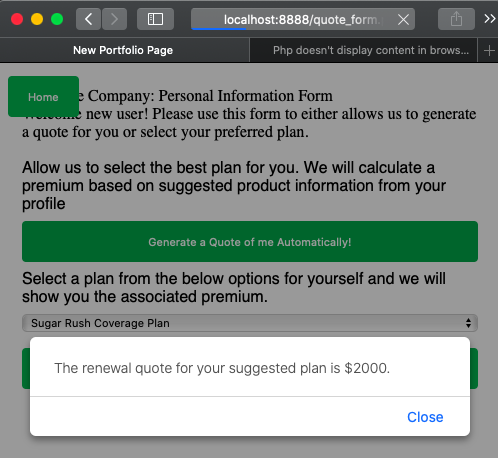
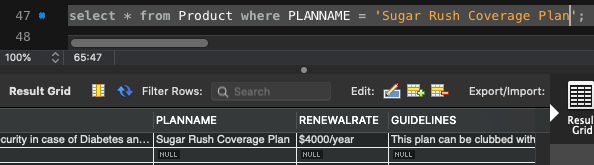
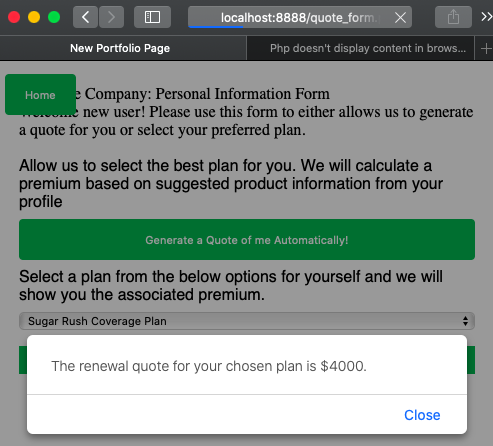
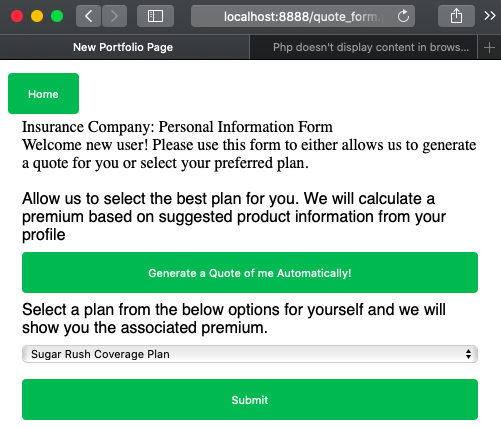
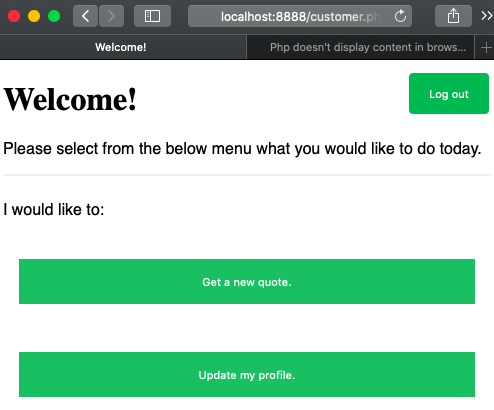
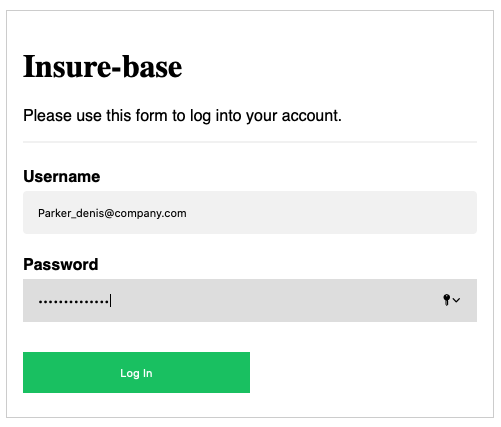
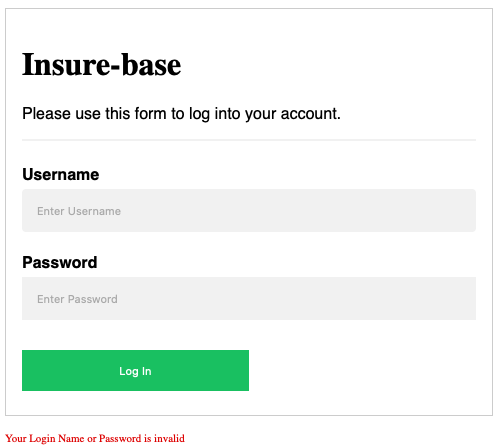
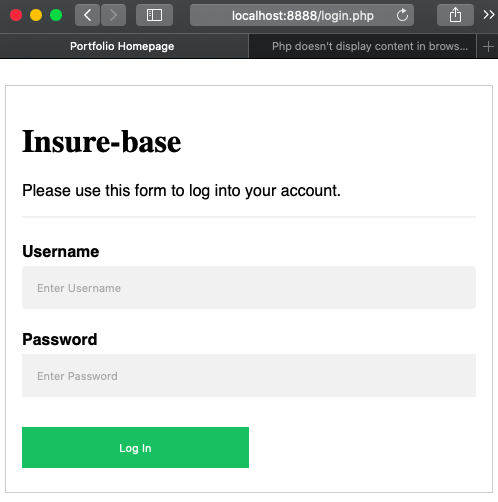
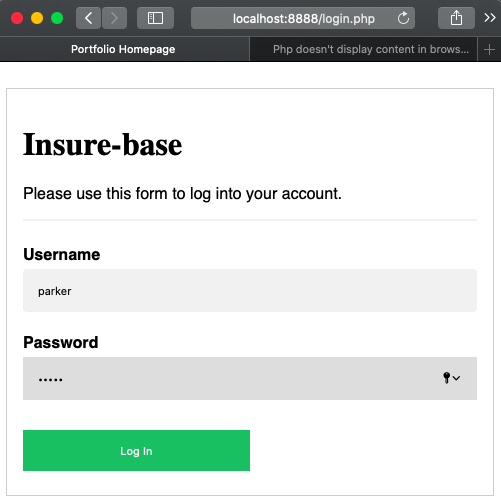
By building an end-to-end solution, we have provided the insurance company an integrated and concise way to run their operations. This kind of system allows data to flow smoothly between customers and the company and within the company itself. Policy, claims, accounting, reinsurance and analytics can all be aggregated to be informative in a fluid nature, allowing management to make better business decisions. However, with single integrated solutions, data governance measures must be unambiguous to prevent security breaches and unexpected partitions which can cause employees to lose their data or customer data leaks. It is thanks to cloud hosted services like Azure that allow secure, flexible, access-anywhere solutions like this one to help companies focus on their business instead of their data.

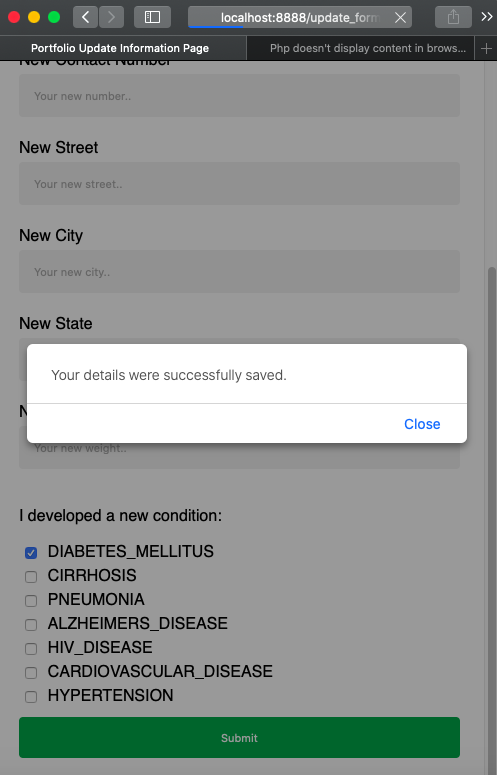
1. **Appendix A**

Demo of our application, Insure-base.

A screenshot of a computer

Description automatically generated





1. Lynch, Warren. “All You Need to Know About Use Case Modeling.” *Medium*, Medium, 22 Mar. 2019, medium.com/@warren2lynch/all-you-need-to-know-about-use-case-modeling-828756da3215. [↑](#footnote-ref-1)
2. Holowczak, Richard. “File Organization and Indexing.” *Database and Programming Tutorials*, 11 Mar. 2013, holowczak.com/file-organization-indexing/. [↑](#footnote-ref-2)
3. Holowczak, Richard. “Data Profiling with DataCleaner and Pentaho Data Integration.” *Database and Programming Tutorials*, 7 July 2017, holowczak.com/data-profiling-with-datacleaner-and-pentaho-data-integration/. [↑](#footnote-ref-3)
4. “Chapter 1.” *Data Science for Business.*, by Forest Provost, O'reilly Editions, 2013. [↑](#footnote-ref-4)
5. 2019, Amazon Web Services, Inc. or its Affiliates. Storage Processing Analytics / Machine learning Archive / Retire Creation / Ingest At every stage of your data’s lifecycle, you need scalable services that can flex with your data growth. [↑](#footnote-ref-5)
6. Sample Medical Record: Adam Pie. Content last reviewed May 2013. Agency for Healthcare Research and Quality, Rockville, MD. ahrq.gov/ncepcr/tools/pf-handbook/mod8-app-b-adam-pie.html [↑](#footnote-ref-6)