Project Report

Designing the database

The steps we went through designing our database:

1. We first drew the ER diagram by hand on papers multiple of times until we got close to something concrete.
2. We then used an online tool to help us draw the final ER diagram containing most of the data we needed to implement the database.
3. And yet it wasn't the final design due to some implementation difficulties.
4. As soon as we reached our final design, we implemented the database using MySQL and it was ready to use.
5. Along the way designing the web app, we had to add some features to our final design but it was minimal changes.

Our assumptions

1. Individual (customer/dependent) can only have one contract and a single contract is linked to only one Individual.
2. Contract holds the plan type and the beneficiary of that plan.
3. Hospital can support one or more plan.
4. Many claims can be filed under the same contract, but the single claim can't be filed by more than one contract.
5. The dependent is weak entity and depends mainly on the customer
6. Claim must be filed regarding to only one hospital.

7. Customers can purchase a new plan and replace the old one with it for themselves and their dependents one at a time.

Main Entities in our database

- Customer
- Dependent
- Plan
- Claim
- Hospital
- Contract

Three Entities are independent which are:

1. Customer
2. Hospital
3. Plan

Three Entities are dependent:

1. Dependent: depends mainly on customer entity
2. Contract: depends on (customer or dependent) and plan entities.
3. Claim: depends on contract, and hospital entities

Entities discerption:

1- Customer entity: The customer of the medical insurance company.
2- Dependent entity: The dependent of a certain customer.
3- Plan entity: The insurance plans offered by the company.
4- Hospital entity: The hospitals contracted with the company.
5- Contract entity: The contract between the customer or any of his/her dependents and the medical insurance company.
6- Claim entity: The insurance claim filed by the customer and can be reviewed by the admin to get accepted or rejected.

- The three independent entities hold basic info without the need for foreign keys.

- **Contract entity:**
  - It's the most complex entity in out design
  - It can only reference the dependent or the customer but not both at the same time. We managed to implement this using check constraints in MySQL.
  - We did it this way so that the individual (customer or dependent) can only have one contract and hence benefits from only one plan.

- **Claim Entity:**
  - Claim entity holds the id of the contract regarding the individual this claim is filed for.
  - From the contract id we can have all the data about the individual (customer/dependent)
  - The customer can file claims for his dependents using the dependent contract.

- **Dependent Entity:**
  - Depends mainly on the customer entity and has partial key of (name and kinship)

Web application report:

- We have designed the frontend of the web application using HTML, CSS, and Bootstrap5.
- We first designed the base template of the web application which contains the stamp that exists in every webpage of the web app I.e. (header, footer, title, scripts, and CSS includes)
- Then we designed only the body of each page with placeholders to show our data in it.
- We also used jinja2 to manipulate these html pages using the data passed from the backend.

- We used HTML input validations to ensure the data entered by normal users are correct and ready to be input in the database.
- Basic customer login with no authentication is implemented and logs in as the first customer in the database.
- Admin dashboard is available and contains the options that are available to the admin
- Customers can file claims, add dependents, purchase plans, view available hospitals for themselves and each of their dependents, view their claims.
- Used library 'mysql-connector-python' to run pure MySQL queries
    - Create the database if it doesn't exist (with some demo data)
    - Get data from the database.
    - Add records to the database.
    - Alter data inside the database.
- Received data from users sent by 'POST' and 'GET' requests and checked if all the data are correct in format and then use it to generate new pages of manipulating the database.
- If the data entered by the user are incorrect, we show a flash message to the user containing instructions about what might be wrong.

- We hosted the web application online and can be accessed at:

  https://medical--insurance.herokuapp.com/

- A GitHub repo of this project with some docs can be found at:

  https://github.com/m-agour/Medical_Insurance

s

Team Members:

1- Mariam Khalid Ragheb Yassien Noureldin
2- Mohamed Nagy Ibrahim Abouagour
3- Mohamed Abdelfatah Abdelfatah Elfeky
4- Menna Allah Moataz Medhat Zaki Mansour
5- Nada Yasser Mahmoud Mohammed Ali
6- Zaynab Sabry Abo Elmakarem Ali
7- Yara Nasser El-den Mohammed