“Il semble que la perfection soit atteinte non quand il n’y a plus rien à ajouter, mais quand il n’y a plus rien à retrancher”

- Antoine de Saint-Exupéry

AAOS Project for ACME Insurance Solutions

420-436-VA: Deliverable 6

Team AAOS:

Alexander Cécile (1655919@edu.vaniercollege.qc.ca, tel. 514-984-8350)

Amer Al Jawabra (2205182@edu.vaniercollege.qc.ca, tel. 514-799-9024)

Olivier Leone (2045231@edu.vaniercollege.qc.ca, tel. 450-601-8296)

Saâd Salmane (2256755@edu.vaniercollege.qc.ca, tel. 438-540-5970)

Client contact: John Smith

“Il semble que la perfection soit atteinte non quand il n’y a plus rien à ajouter, mais quand il n’y a plus rien à retrancher”

- Antoine de Saint-Exupéry

Table of Contents

Executive Overview 3

Business problem 3

Narrative description of the database design 3

Appendix 1 – Data dictionary with entities, relations and attribute types 4

Entities 4

Relationships 6

Attribute Types 7

Appendix 2 9

Deliverable 3 diagram 10

Explanation of Similarities 10

Appendix 3 12

Appendix 4 12

“Il semble que la perfection soit atteinte non quand il n’y a plus rien à ajouter, mais quand il n’y a plus rien à retrancher”

- Antoine de Saint-Exupéry

Executive Overview

Team AAOS is developing a web application for the health insurance department of ACME Insurance Solutions, aimed at improving the claim submission and review process. The application will streamline operations, reduce processing time, and enhance customer satisfaction by leveraging digital solutions.

Business problem

The current manual process of claim submission and review at ACME Insurance Solutions is time-consuming, error-prone, and leads to customer dissatisfaction due to delays in processing. ACME needs a solution to enhance operational efficiency while ensuring accuracy in claim processing to meet customer expectations.

Narrative description of the database design

In the database, users interact with the system as one of the primary actors. Users utilize the web application to submit claims, providing essential information such as claim details, supporting documents, and contact information. They can track the status of their claims and receive updates through the application, ensuring transparency and ease of use in the claim submission process. Also, users can enable two-factor authentication (2FA) for enhanced security, adding an extra layer of protection to their accounts. Furthermore, users have access to a profile page where they can manage their personal information, preferences, and security settings, providing them with greater control and customization options within the system.

On the other hand, admins play a crucial role in overseeing and managing the claim review and approval process. With access to the system, admins are responsible for reviewing and processing claims submitted by users. They have the authority to view claim details, review supporting documents, and make decisions on claim approval or denial. Also, admins can update claim statuses and communicate with users about claim decisions or inquiries, ensuring efficient and effective management of the claim processing workflow.

Appendix 1 – Data dictionary with entities, relations and attribute types

Entities

User

Attributes: UserID (Primary Key, int), PolicyID (Foreign Key, int), Username (varchar), Password (varchar), Email (varchar), RoleID (Foreign Key to Role, int), FullName (varchar), Phone (varchar), Address (varchar)

Admin

Attributes: AdminID (Primary Key), Username, Password, Email

Claim

Attributes: ClaimID (Primary Key, int), UserID (Foreign Key to User, int), ClaimType (varchar), SubmissionDate (datetime), Status (varchar), ClaimDetails (text)

ClaimDocument

Attributes: DocumentID (Primary Key), ClaimID (Foreign Key), DocumentName, FileType, FileSize, UploadDate

ClaimReview

Attributes: ReviewID (Primary Key), ClaimID (Foreign Key), AdminID (Foreign Key), ReviewDate, ReviewOutcome, Notes

Payment

Attributes: PaymentID (Primary Key), ClaimID (Foreign Key), Amount, PaymentDate, PaymentMethod

Notification

Attributes: NotificationID (Primary Key, int), UserID (Foreign Key to User, int), Timestamp (datetime), NotificationType (varchar), NotificationContent (text)

Feedback

Attributes: FeedbackID (Primary Key, int), UserID (Foreign Key to User, int), Timestamp (datetime), FeedbackText (text)

ClaimCategory

Attributes: CategoryID (Primary Key), ClaimID (Foreign Key), CategoryName, Description

ClaimAssignment

Attributes: AssignmentID (Primary Key), ClaimID (Foreign Key), RoleID (Foreign Key), AssignmentDate

ClaimInteraction

Attributes: InteractionID (Primary Key, int), ClaimID (Foreign Key to Claim, int), UserID (Foreign Key to User or Admin, int), InteractionType (varchar), Timestamp (datetime), InteractionDetails (text)

AuditResult

Attributes: AuditID (Primary Key), ClaimID (Foreign Key), AuditDate, AuditorName, Findings

Ticket

Attributes: TicketID (Primary Key, int), UserID (Foreign Key to User, int), Timestamp (datetime), TicketText (text), Status (varchar), AssignedTo (Foreign Key to Admin or Support Staff, int)

Role

Attributes: RoleID (Primary Key, int), RoleName (varchar), Description (text)

Profile

Attributes: ProfileID (Primary Key, int), UserID (Foreign Key), Birthdate (Date/Time), ProfilePicture (String), DateCreated (Date/Time)

Policy

Attributes: PolicyID (Primary Key, int), UserID: Integer (Foreign Key referencing User.UserID), PolicyNumber: String, PolicyType: String, CoverageAmount: Numeric, PremiumAmount: Numeric, PolicyStartDate: Date/Time, PolicyEndDate: Date/Time, PaymentMethod: String, RenewalDate: Date/Time, PolicyStatus: Enum or String (Active, Expired, Cancelled), DateCreated: Date/Time

Relationships

User-Claim (One-to-Many)

A User can submit multiple Claims.

A Claim is submitted by one User.

Admin-ClaimReview (One-to-Many)

An Admin can perform multiple ClaimReviews.

A ClaimReview is for one Claim.

Claim-ClaimDocument (One-to-Many)

A Claim can have multiple ClaimDocuments.

A ClaimDocument belongs to one Claim.

Claim-ClaimReview (One-to-Many)

A Claim can have multiple ClaimReviews.

A ClaimReview is for one Claim.

Claim-Payment (One-to-One)

A Claim can have one Payment.

A Payment is for one Claim.

User-Notification (One-to-Many)

A User can receive multiple Notifications.

A Notification is sent to one User.

User-Feedback (One-to-Many)

A User can submit multiple Feedback entries.

A Feedback entry is submitted by one User.

Claim-ClaimCategory (Many-to-One)

A Claim can belong to one ClaimCategory.

A ClaimCategory can have multiple Claims.

Claim-ClaimAssignment (One-to-One)

A Claim can be assigned to one Assignee.

An Assignee can be assigned multiple Claims.

Claim-ClaimInteraction (One-to-Many)

A Claim can have multiple ClaimInteractions.
A ClaimInteraction is associated with one Claim.

Claim-AuditResult (One-to-One)

A Claim can have one AuditResult.
An AuditResult is for one Claim.

User-Ticket (One-to-Many)

A User can create multiple Tickets for customer support.
A Ticket is created by one User.

User-Role (Many-to-Many)

Users can have multiple Roles.
Each Role can be assigned to multiple Users.

User-ClaimInteraction (One-to-Many)

A User can have multiple interactions related to Claims.
Each ClaimInteraction is associated with one User.

User-Profile (One-to-One)

A User has one Profile.
A Profile belongs to one User.

Policy-User (One-to-Many)

A Policy can belong to one User.
A User can have multiple Policies.

Attribute Types

- UserID, AdminID, ClaimID, DocumentID, ReviewID, PaymentID, NotificationID, FeedbackID, CategoryID, AssignmentID, InteractionID, AuditID, TicketID, RoleID, ProfileID, PolicyID: Integer (Primary Key)

- UserID, AdminID, ClaimID, CategoryID: Integer (Foreign Key)

- Username, Password, Email, UserType, DocumentName, FileType, PaymentMethod, NotificationType, FeedbackType, ReviewOutcome, Notes, Description, CategoryName, InteractionType, AuditorName, Findings, TicketText, RoleName, PolicyNumber, PolicyStatus, PolicyType, ProfilePicture: String

- PolicyNumber: String or Integer (depends on the policy numbering system)

- DateSubmitted, UploadDate, ReviewDate, PaymentDate, NotificationDate, FeedbackDate, AuditDate, InteractionDate, AssignmentDate, Timestamp, Birthdate, DateCreated, PolicyStartDate, PolicyEndDate, RenewalDate: Date/Time

- Status: Enum or String (e.g., "Pending", "Approved", "Denied")

- UrgencyLevel: Enum or String (e.g., "High", "Medium", "Low")

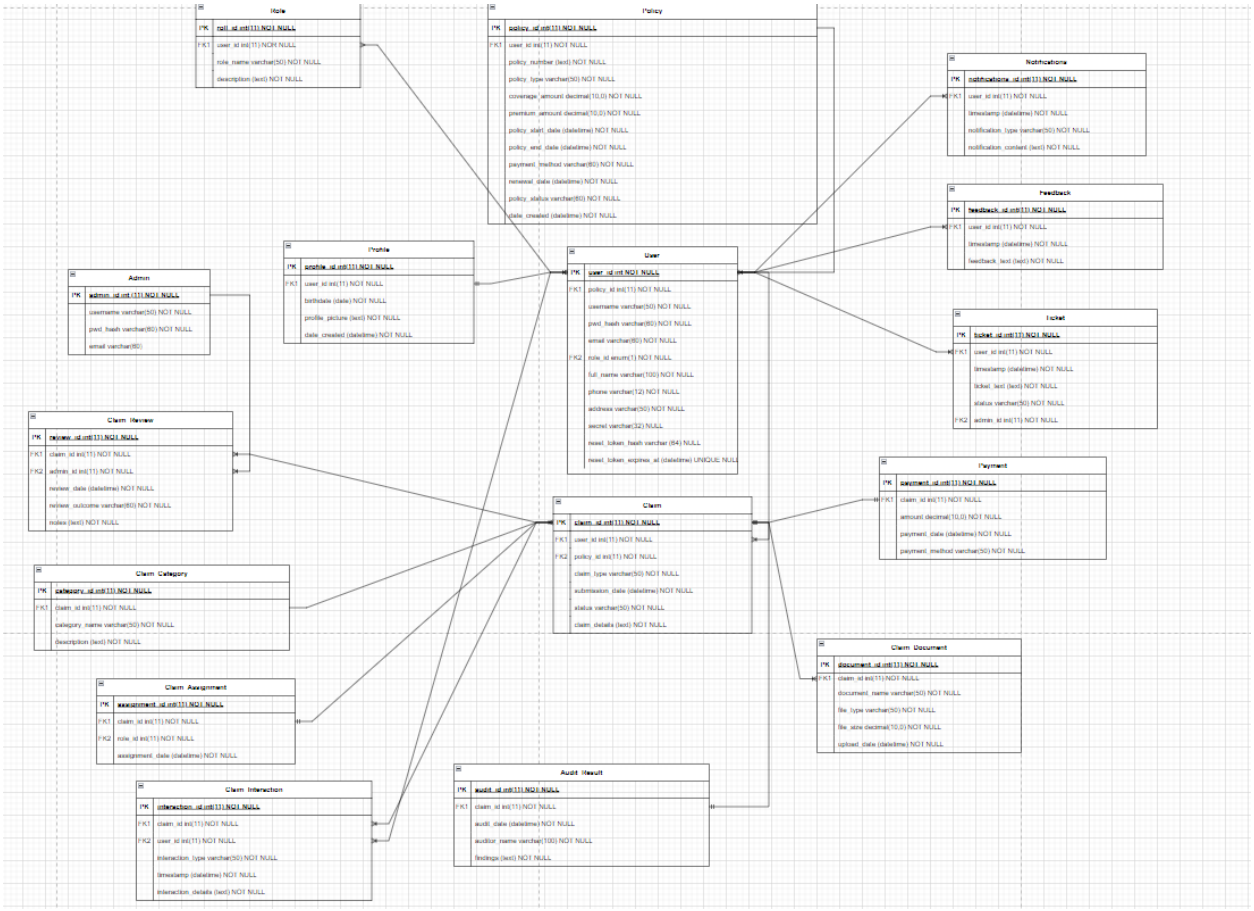
- FileSize, Amount, CoverageAmount, PremiumAmount: Numeric (e.g., in KB or MB for FileSize, in currency format for Amount)

- IsRead: Boolean (for Notification read status)

“Il semble que la perfection soit atteinte non quand il n’y a plus rien à ajouter, mais quand il n’y a plus rien à retrancher”

- Antoine de Saint-Exupéry

Appendix 2



Deliverable 3 diagram

Explanation of Similarities

Entities Representation:

Both the ER diagram and the class diagram depict entities/entities of the system, such as User, Claim, Admin, etc.

Both representations illustrate the attributes of these entities, albeit with different levels of detail. For example, both diagrams likely include attributes like UserID, Username, etc.

Relationships Representation:

Both diagrams show relationships between entities, such as one-to-many and many-to-many relationships.

The cardinality of relationships (e.g., one-to-many) is likely represented similarly in both diagrams.

Primary and Foreign Keys:

Both diagrams likely indicate primary keys and foreign keys, although they may use different notations.

Explanation of Differences:

Level of Abstraction:

The ER diagram may focus more on the logical structure of the database, emphasizing tables, keys, and relationships directly related to data storage.

The class diagram in UML may have a broader scope, including not only data storage but also methods, behaviors, and relationships beyond just database entities.

Additional Details in UML:

The UML class diagram may include methods and behaviors of classes, which the ER diagram typically does not capture.

Associations in UML may include more details, such as multiplicity, navigability, and association classes, which are not always represented in ER diagrams.

Notation Differences:

The ER diagram may use a specific notation for entities, attributes, and relationships that differs from the UML notation.

UML class diagrams may use more standardized symbols and notations for representing classes, associations, and other elements.

The ER diagram focuses more on the logical structure of the database, whereas the UML class diagram provides a broader view of the system, including both data structure and behavior.

Appendix 3

To optimize queries in our website and make information retrieval much more efficient and simple, we first made sure that our tables are in the 3NF to avoid any data redundancy. We also used indexes on columns that we are using frequently to allow for much faster retrieval. While we could alse use stored procedure to save some queries that we want to reuse, we figured that it wouldn't be necessery since these queries are going to be inside our code anyway and that the size of our project isn't large enough to justify their usage. Finally, we will avoid using subqueries as they generally return large amounts of rows , making them slower to execute.

Appendix 4

The access speed required for our database will have to be relatively fast since the database will be accessed frequently by the users and the admins to perform multiple tasks such as viewing policies, submitting claims, checking claim statuses ,making payments etc. To allow all these operations to happen, the database records need to be updated properly and regurlarly to provide the user with good experience. We will make this possible through our database design (as explained in Appendix 3), which will allow for optimized interations on the database. Relatively fast access speed is expected from an insurance website where sensitive information will need to be accessed and where transactions (claim payments) are going to be recorded.