Znailla Claims System Solution Architecture

Version 1.0

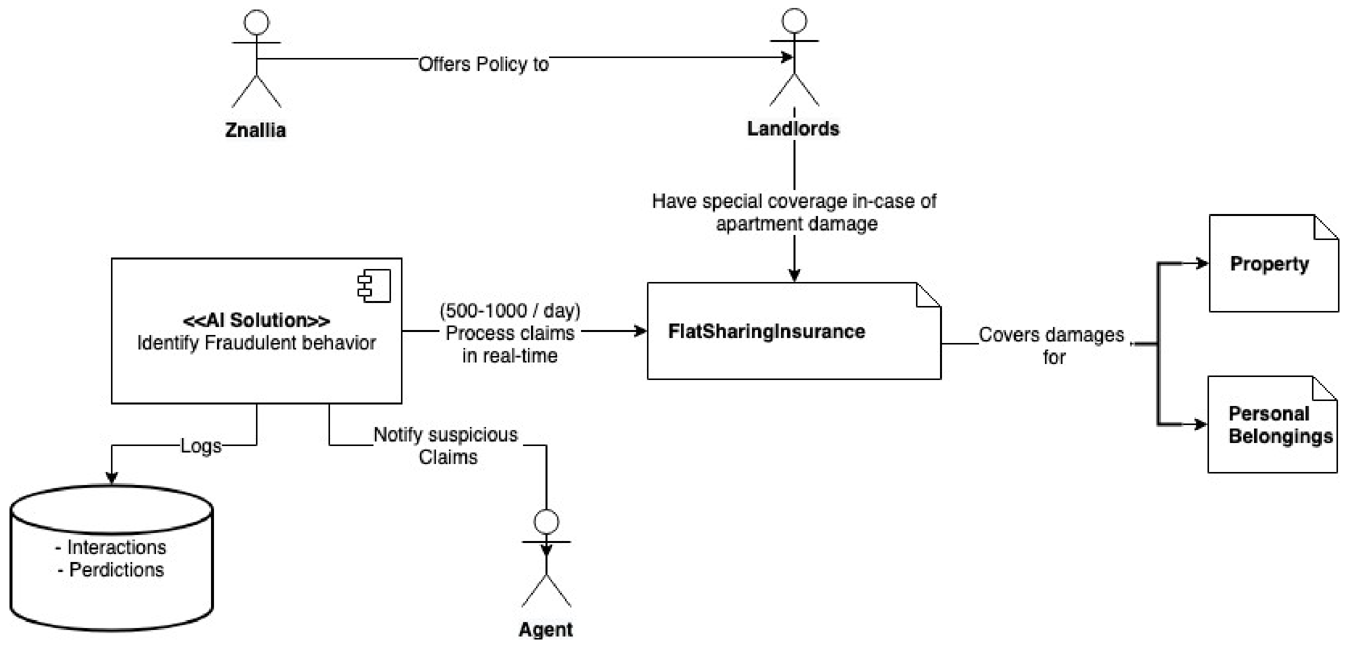
Document History

Version	**Date**	**Description**	**Submitted by**
1.0	09/09/19	Initial Draft	Sameh Amin

Table of Contents

Functional View

Context Diagram

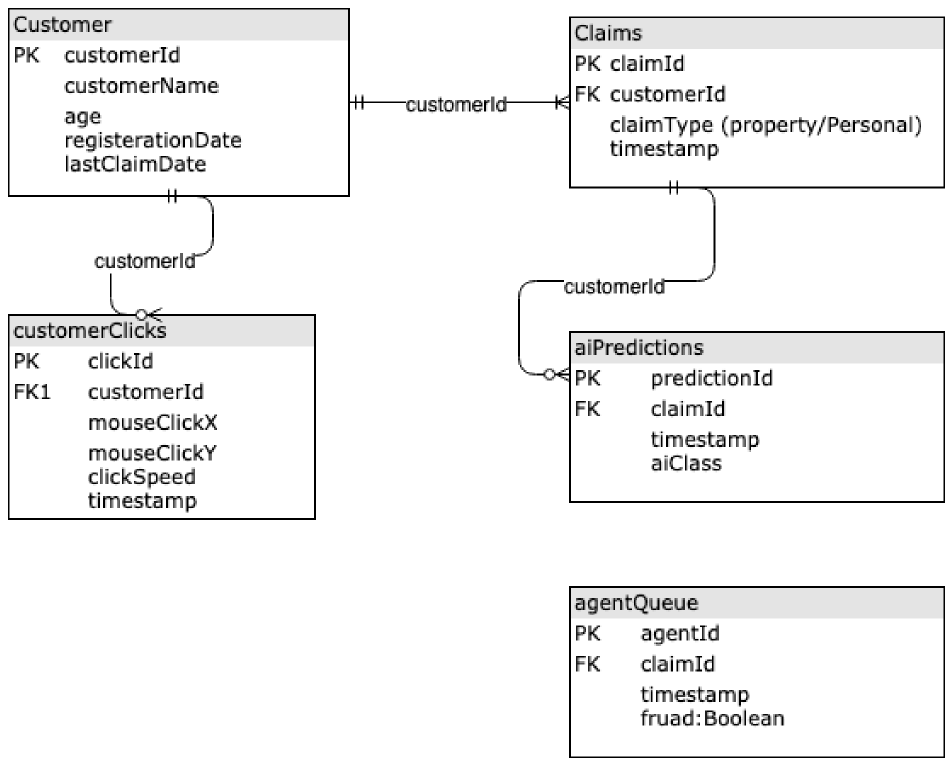


Functional Flow / Use Cases

ID	Use Case	Actor
1	Submit claims through Znailla website	Customer
2	AI Service to predict and classify fraud claims	Znailla AI System
3	AI system to Log all interactions and prediction results	Znailla AI system
4	AI system to notify Agents about fraud claims	Znailla AI system

Information View

ERD



Components description

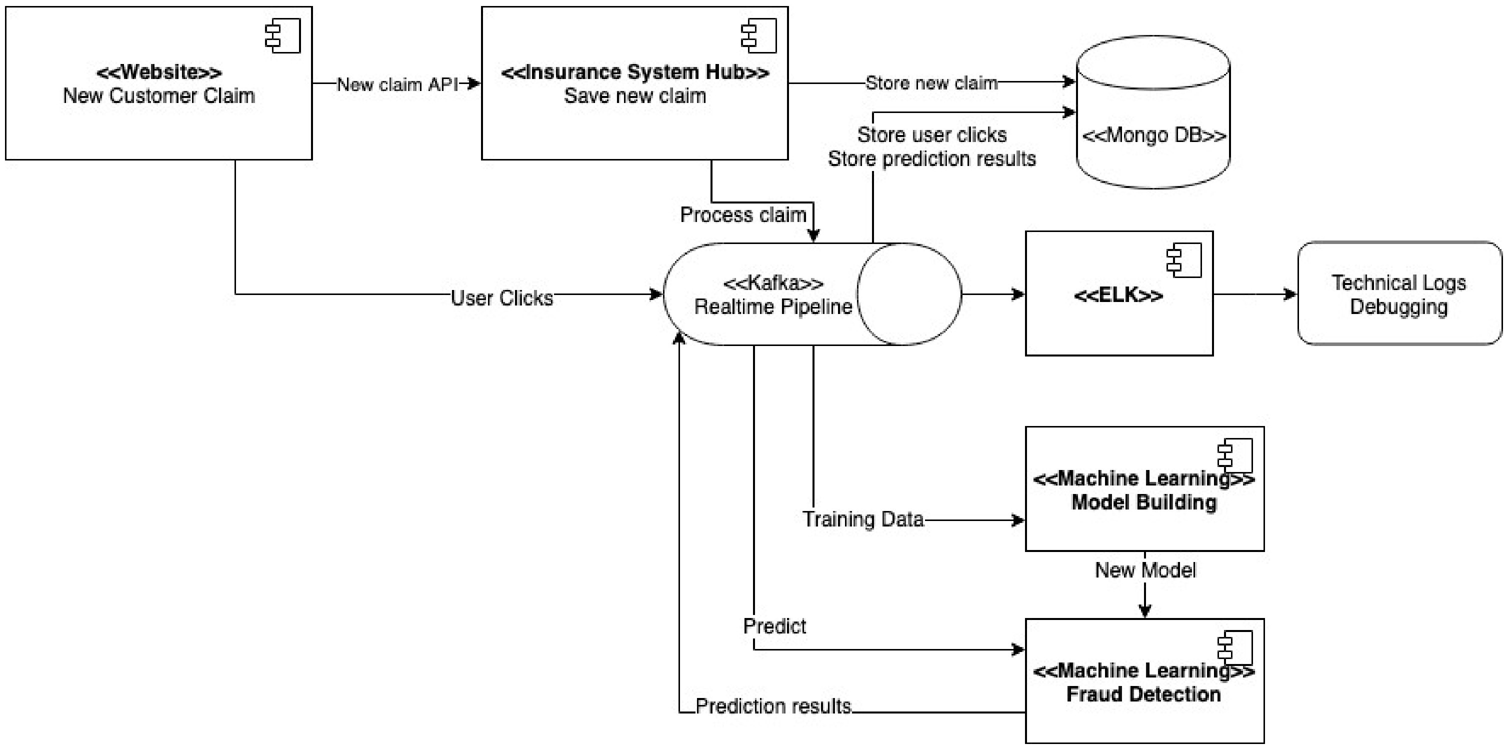
- This is an ERD to show all Entities suggested to store in the system.
- We're going to use MongoDB for flexible schema, indexing and fast queries.
- Store basic information about the system such as
    - Claims
    - Customers

Machine learning Features

- User clicks on the insurance website should be stored In order to use as an important feature by the anomaly detection machine learning algorithm.
- Customer information also could be important key features to use by the machine learning algorithm such as
    - Customer age

        - Registration date
        - Last claim date
    - Claims data also can be used as machine learning features, such as claimType.
- Also we need to store all AI Predictions related to all claims and the class matched with this claim, in this case it's binary classification (either Fraud or not). I name it aiClass so we're able to expand and add more classes in the future.

DataFlow



- The customer will register new claim on insurance website.
- Frontend will call the Insurance hub system API to save and process the new claim.
- The Hub should store the claim in MongoDB.
- Then the Hub should insert new message with the topic "claim_training" to the Kafka Queue.
- Then the Hub should also insert new message with the topic "claim_prediction" to the Kafka Queue.
- The Hub as-well should store the stream of user clicks on MongoDB through producing all click stream to Kafka Topic "click_stream".

- The Machine learning system as a Consumer, should consume the new message coming to the topic “claim_training”.
- The Machine learning system should train the model using the data comes from the two Topics “click_stream” and “claim_training”.
- If the new model accuracy on test, provided higher result than the existing model accuracy and higher than the threshold defined by business, then the machine learning system should deploy and replace the new model to be used on the upcoming predictions.
- The Machine learning system as a Consumer, should consume the new message coming to the topic “claim_prediction”.
- The machine learning system should do a binary classification to predict if it’s fraud/anomaly or normal claim.
- Then it should store the prediction results back to the MongoDB through Kafka message for the sake of Audit and logging.
- All transactions should be stored in the ELK stack (Elastic search, Logstash and Kibana) for the sake of technical debugging.

Deployment View

Operational View

Quality Attributes

Security perspective

TBD

Availability perspective

TBD

Performance perspective

TBD