Intelligent Claims System

Team members

Bhavya Rema Devi
Elisa Tjhin
Low Kok Ann, Francis
Sindhuja Kumaran

CONTENTS

Page number

1. Executive Summary

Healthcare infrastructure in Singapore consists of a government-run universal healthcare system with a significant private healthcare sector. For its citizens, financing of healthcare costs is done through a mixture of direct government subsidies, compulsory savings, national healthcare insurance, and cost sharing. For the most part the government does not directly regulate the costs of private medical care. These costs are largely subject to market forces, and vary enormously within the private sector, depending on the medical specialty and service provided. Medical advances also increase cost, especially when Singapore is often where such newer solutions are introduced. To tackle the costs, many Singaporeans and Singapore Permanent Residents also have supplemental private health insurance for services not covered by the government's programmes. Part of having a health insurance is going through the claim process and waiting for the claims to get settled.

ABC Insurance Limited is a newly established insurance company in Singapore. Its range of products includes health insurance, among many things. It's health insurance plans cover hospitalization expenses, day care procedures, domiciliary expenses besides many others.

At present, the majority of hospitalisation claims are processed through e-filing, however the pre and post hospitalisation claims process remain as manual. Claims are submitted by customers, then these claims go through a round of verification. The claims approval process involves adhering to the business rules or policies set for the company and this can take many man hours. The company assigns the incoming claims to its staff for a first round of investigation. If the case needs further investigation, it will be assigned to a case manager based on a predefined set of rules.

The company finds that many of the claims are straightforward and the process can be improved by building a system to automate this, so as to boost productivity and achieve scalability.

Using intelligent reasoning system principles, this project aims to build a knowledge driven system powered by business rule engine to help streamline claims classification process, chatbot and document recognition as cognitive frameworks for better customer experience and a claim distributor system using constraint satisfaction tools to optimise staff productivity.

2. Business Problem Background

The company receives up to 6000 manual hospitalisation related claims per month. Each claim comes with a set of documents, such as hospital bills, clinic receipts, etc. This adds up to more than 10,000 documents monthly. Each claim is processed manually to verify if a bill can be claimed as agreed upon in the insurance policy.

If the claim is assessed as valid according to policy coverage and does not require further investigation, the claim is approved and does not need to be assigned to a case manager. If not, case supervisors will assign the case to a case manager.

When assigning cases, case supervisors have to keep in mind certain factors, such as ensuring a fair workload for everyone, and leave schedules.

For customers, the claim process is cumbersome. There is a lot of paperwork involved. Because of this, many turn to their insurance agents for help. If the process can be simplified, the workload on the agents can be reduced and thereby some cost savings can be achieved.

The company believes that by investing in a system that can automate the document verification process, and that can also assign cases to case managers, it will greatly enhance productivity and allow the company to scale more easily, while reducing human error.

It also plans on introducing a customer service chatbot to deal with the client as a platform for the submission of certain types of claims thereby increasing customer satisfaction and retention while achieving cost savings.

3. Market Research

We gathered information about the various policies and data concerning an Insurance company through our team member who has been working in the field for more than a decade. We went back and forth with details of policy for knowledge discovery and representation using business rules.

We received sample data for the purpose and learned the data and the processes associated with claim approval, rejection and settlement. We understood that for these types of problems where policies and rules are clear, a rules based system works very well.

We also gathered information about the way claims are handled internally in the company and the way staff are allocated these claims. We found we can improve on the current practices by automating the process of scheduling around the staff’s leave.

4. Project Objectives & Success Measurements

4.1 Project objectives

1. Ease the claim submission process for customers by allowing customers to submit claims through a chatbot user interface.

2. Reduce the number of agents / service staff needed to handle claim submissions.

3. Reduce the workload on case managers by automatically approving some claims where applicable. Constraints that define how to classify cases are set out in the later sections of this document.

4. Automate the assignment of cases to case managers based on their schedule.

4.2 Scope

4.2.1 Customer facing chatbot

- Accept claim submissions via a chatbot interface
- Accept open ended user input and extract entities from the input. For example, from this user input: “I want to submit a claim for my medical bill.” the chatbot should be able to determine the type of claim, and the date of the claim.
- Ask follow up questions until required information for a claim is complete.
- Handle spelling and grammatical errors.
- Automatically process documents that come with a claim. Documents shall be uploaded to the system in an image format. These documents include:

    - Bills
    - Receipts

- Information that needs to be extracted from documents are:

    - Medical entity such as hospital or specialist doctors
    - Bill document references
    - Date of document
    - Grand total of bill / receipt

4.2.2 Automated claim classification

- Automatically classify a claim to be “approved” or “needs intervention” after checking all of the given criteria:
    - Claim amount - claims with amounts within the stipulated limit need not go through human investigation
    - Claim validity - whether the claimed line items are valid for that policy and within limit
    - Claim history - whether the same customer has attempted many claims in the past
      - whether it’s the first claim for the policy
    - Insured’s history - whether the Insured had an illness before starting the policy.
      - whether the same customer has attempted many claims in Past
      - whether there are pending claims to be processed
      - whether there are previous rejected claims
    - Policy age - whether policy is inforce, if so how long
    - Premium - If there is any outstanding premium
    - Claim date - the date of occurrence in the claim is compared with the policy expiry date to make sure the claim is valid

4.2.3 Automated case distribution

- Automatically assign cases to case managers based on these rules:
    - Case managers who are not available shall not be assigned any cases
    - Cases shall be distributed to case managers to ensure fair and balanced workload among case managers.

4.3 Exclusions

In view of time constraints, the following items are not in scope for the project:

1. Claim rules implemented are part of the subset of rules from the real scenario and do not include components such as co-payment, subsidies, medisave, deductibles. The claim approval refers to the first level validation of the submitted claim.

2. Customers do not need to authenticate with the system in order to make a claim. He / she will however need to enter their ID in order for the system to identify which policy the claim is for

3. Only a set number of bill formats will be supported. Refer to the user guide on how to download these sample bills

4. The chatbot does not support file uploads from mobile devices

5. In an actual setting, the claim distributor should be set to run every x amount of time. However, for the scope of the project, it has to be manually triggered from the dashboard. This is to ease testing.

6. Everytime you restart the docker services, all new claims except pre-seeded ones will be deleted. This is intended, In order to ease testing.

4.4 Success measurements

Chatbot

- Chatbot are able to understand claim submission intent
- Chatbot are able to follow up conversation to collect claim information required
- Chatbot integrated with object repository system and able to validate policy based on information provided by user
- Claim submitted via chatbot and saved to claim repository
- Bill uploaded with recognized information saved to the claim repository

Classification

- Classifier able to listen to incoming claim submission event and process the claims
- Classifier able to handle data error.
- Classifier able to classify a list of claims without error
- Classifies makes changes for approved claims and gives reason for rejections

Distribution

- Distributer able to assign the new claims, which are classified for human intervention, to the available staffs.
- Distributer able to assign one claim to one staff.
- Distributer able to not assign claims to staffs, whenever they are not present.

5. Knowledge Specification

5.1 Knowledge Acquisition

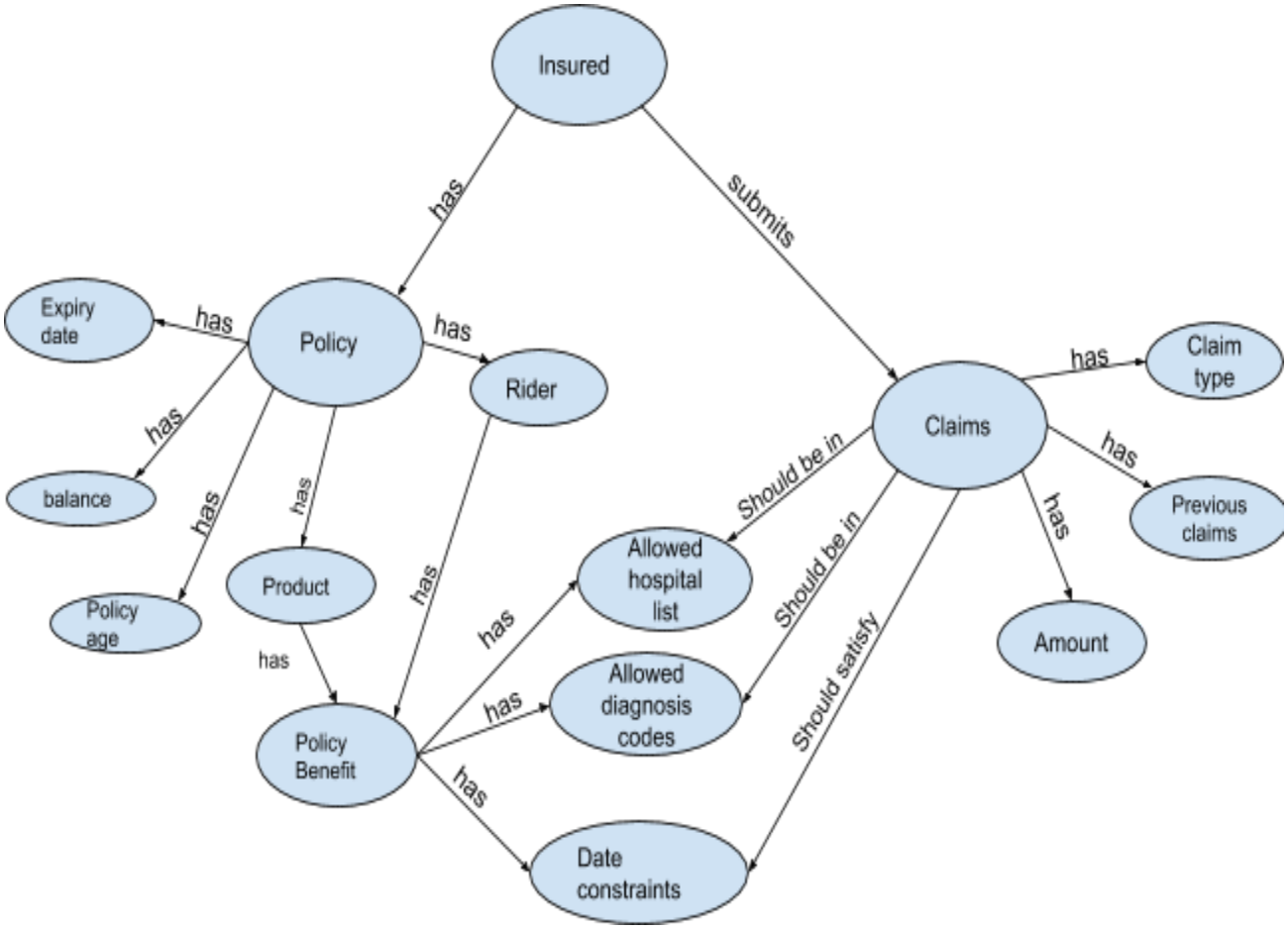
The knowledge for the system requirements is built based on following acquisition methods:

- Subject Matter Expert (Claim Staff)
- User Survey
- Online Media
- Case study

S.N	Source	Acquisition Methods	Information
1	SME/Claim Staff	Elicitation of tacit knowledge through interviews	- General business rule for a health claim process - Policy and claim forms
2	User Survey	Survey and conversation from general population	User experiences sharing on claim filing process
3	Online Media	Google to get facts and attributes of health claims	General information on insurance claim such as: - Hospitalisation policy coverages - Claims terms and conditions - FAQ
4	Case Study	Analysis of existing insurance claims system	- Design, architecture and business process - Claim cases with approval and rejection

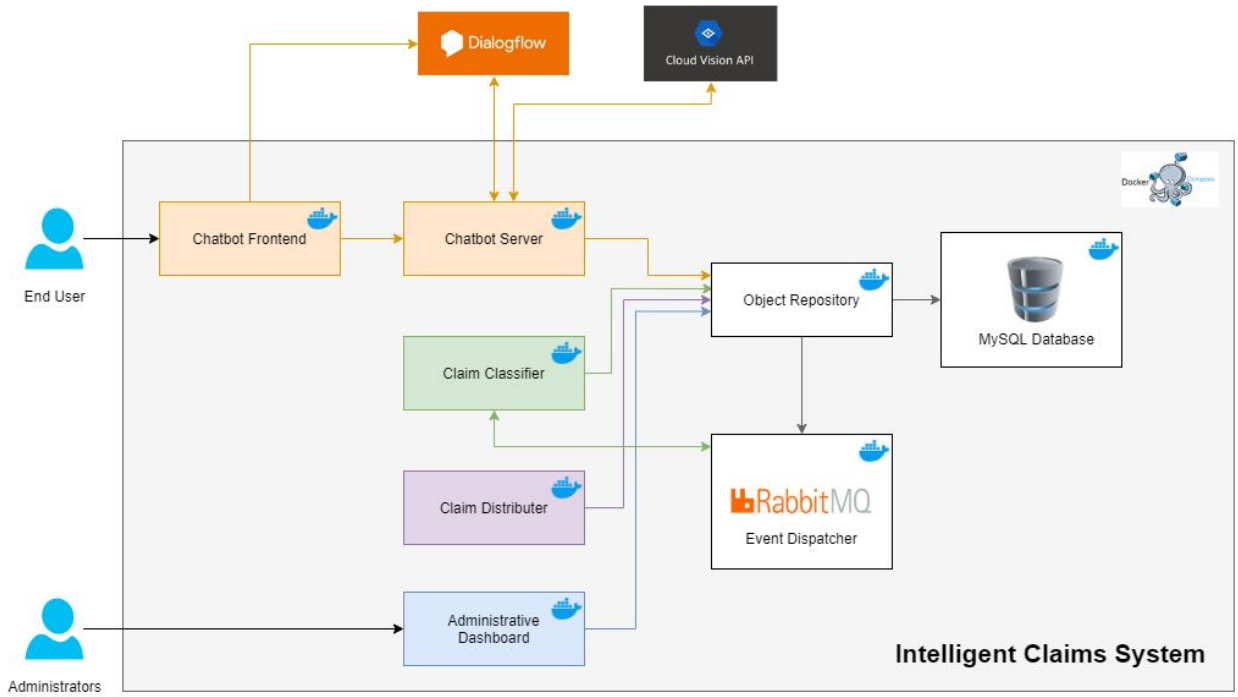
5.2 Knowledge Representation

The sample data that was cleaned and collected was used for the semantic representation of Knowledge. The data was used for facts to build an expert system in CLIPS which used the forward chaining mechanism. Upon receiving facts, the engine fires relevant rules which leads to the creation of new facts and firing of other relevant rules. The semantic representation of data is as follows



6. Project Solution

6.1 System Architecture



The diagram above shows the overall system architecture. The system is broken down into multiple microservices. Each service will be hosted in a separate Docker container. Docker Compose will then be used to manage these services.

Here are each of these services and their scope:

Chatbot Frontend

- Customer facing website that hosts the chatbot widget
- Technical stack: Dialogflow Messenger, ReactJS, Nginx

Chatbot Server

- Consolidates required information from customer and submits it to the object repository
- Handles fulfillment requests from Dialogflow

- Processes uploaded documents to retrieve information using Google Cloud Vision
- Technical stack: Python, Flask, Dialogflow, Google Cloud Vision
- Techniques involved: cognitive frameworks

Claim Classifier

- Automatically approves claims when all rules and constraints are satisfied
- Specifies reason when rules are not satisfied
- Updates data repository with classification results
- Technical stack: Python, Flask, Clips
- Techniques involved: rule based reasoning

Claim Distributer

- Automates assignment of claims to staff based on set constraints
- Updates data repository with assignment results
- Technical stack: Python, Flask, Google ORTools
- Techniques involved: constraint satisfaction

Administrative Dashboard

- Shows an overview of the system status. Including:
    - Number of claims that are automatically approved by the system vs number of claims that require human intervention
    - The schedule of each staff
- Shows each claim / policy's details
- Allow the administrator to artificially inject claims into the system
- Allow the administrator to update each staff's leave
- Technical stack: ReactJS, Nginx

Object Repository

- Exposes data of claims, policies, staff, etc through APIs
- Publishes event on the event dispatcher whenever a claim is submitted
- Technical stack: NodeJS, Express, Sequelize

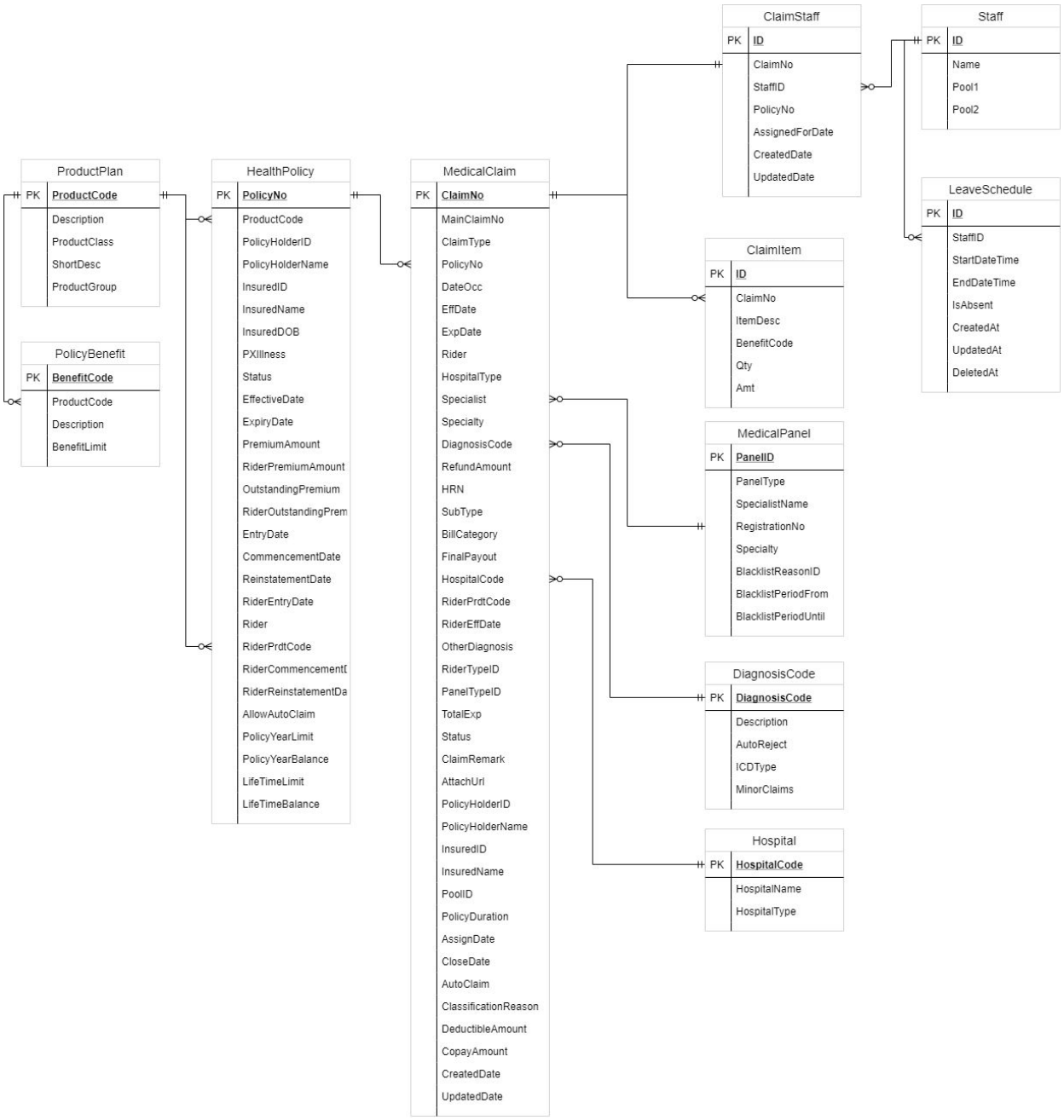
Database

- Persists data
- Technical stack: MySQL

Event Dispatcher

- Brokers messages between services
- Technical stack: RabbitMQ

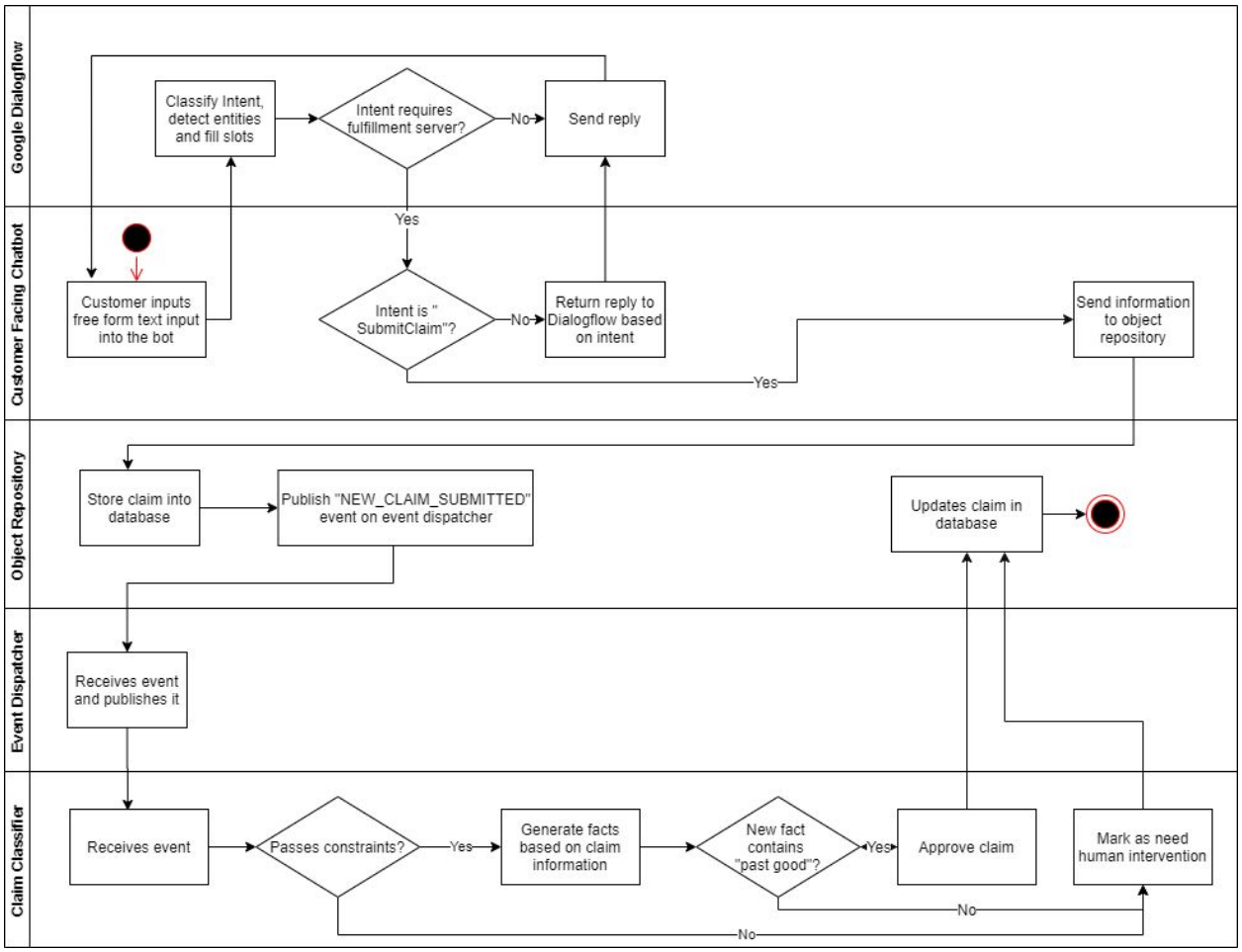
6.2 Data Model



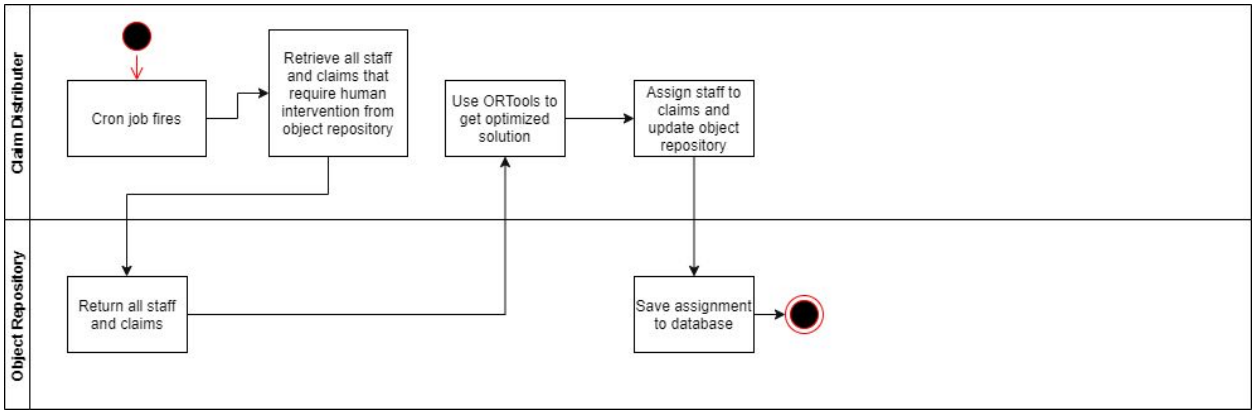
6.3 Process flowchart

The process can be broken into 2 separate parts. The claim submission / classification process, and the claim distribution process.

The claim submission / classification process is as such:

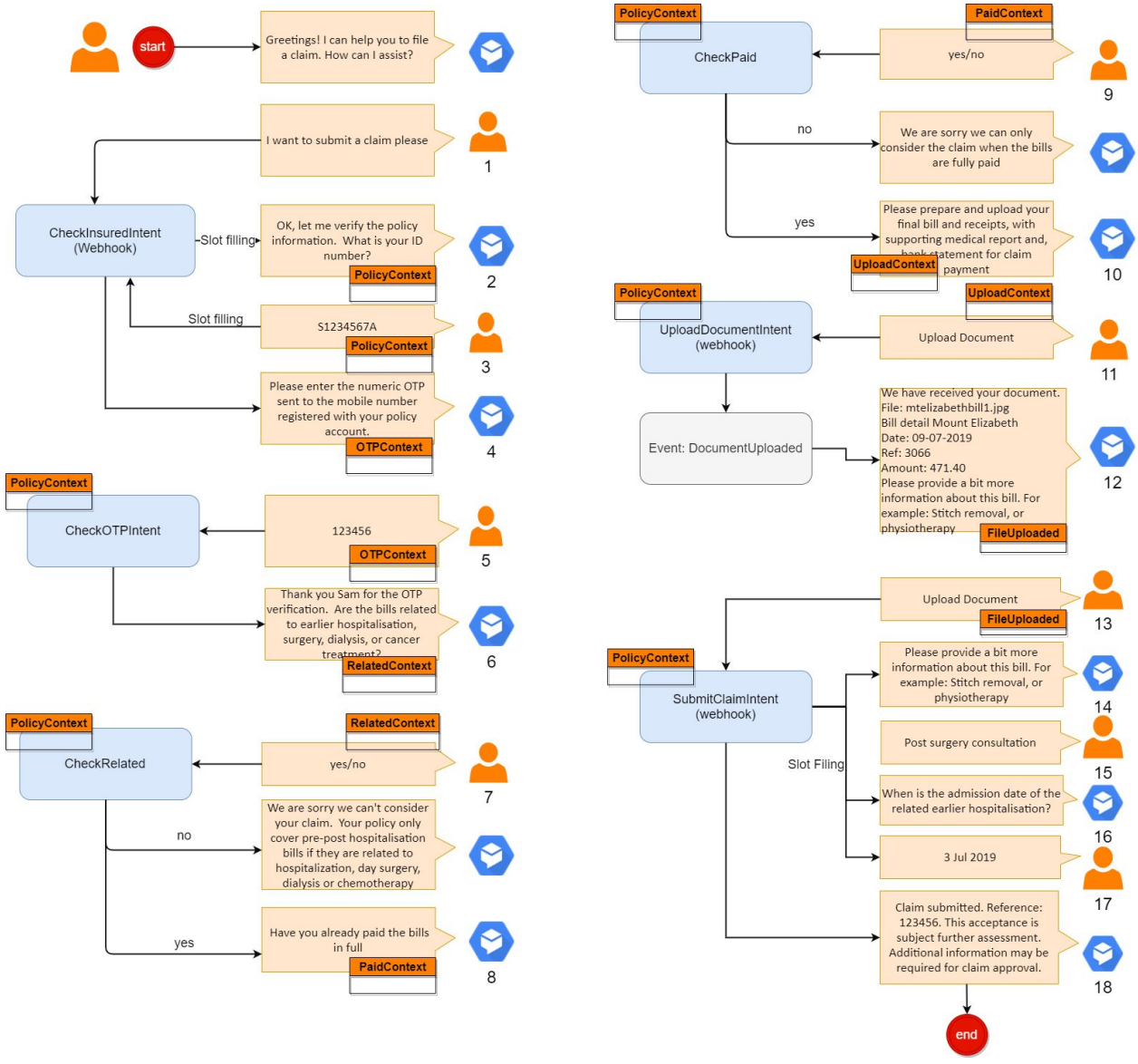


The claim distribution process is as such:


Claim Distributer
Object Repository
Cron job fires
Retrieve all staff and claims that require human intervention from object repository
Use ORTools to get optimized solution
Assign staff to claims and update object repository
Return all staff and claims
Save assignment to database

7. Project Implementation

7.1 Chatbot flow:



7.1.1 Chatbot Intents:

1. **INTENT: _Default Welcome Intent**

User phrases	Hi, Hello, Good morning
Description	Start message/welcome message
Prompt	Greetings! I can help you to file a claim. How can I assist? Hello there! Do you know that you can file pre and post hospitalisation claims here with me? You can say "I want to submit a claim" to start the claim process For example, you can ask "I want to submit my medical bill" to file your claim It's easy to start a claim, just tell me "Help me to claim my bill"
Webhook	No

2. INTENT: CheckInsuredIntent

User phrase	I want to file a claim I want to claim my medical bill Help me to submit a claim please
Input Context	N/A
OutputContext	PolicyContext
Description	This is to initiate claim process
Prompt	OK, let me verify the policy information. What is your ID number? Sure, What is your registered NRIC in the policy?
Fulfillment Webhook	**/claim/intenthandler** **checkInsuredIntentHandler (InsuredID)**
Response (Success)	Please enter the numeric OTP sent to the mobile number registered with your policy account.
Response(Fail)	The provided ID is no valid Please try again, I'm unable to communicate to claim system at the moment

3. INTENT: CheckOTP

INTENT	CheckOTP
User phrase	$OTP For example: 700111
Input Context	PolicyContext

OutputContext	CheckRelatedContext
Description	This is simulation to authenticate for the Insured login
Response (Success)	Thank you $InsuredName for the OTP verification. Are the bills related to earlier hospitalisation, surgery, dialysis, or cancer treatment?
Response(Fail)	Please enter the numeric OTP sent to the mobile number registered with your policy account.

4. INTENT: CheckRelated yes

User phrase	Yes, Can, ok, confirm
Input Context	CheckRelatedContext
OutputContext	CheckPaidContext
Description	TO check all answer must be Yes to proceed with claim: Are you bill related to hospitalization, day surgery, dialysis or chemotherapy Follow-up questions: - Have you already paid the bills in full
Webhook	None
Response (Success)	Have you already paid the bills in full?

5. INTENT: CheckRelated no

User phrase	no, nope, nah
Input Context	CheckRelatedContext
Output Context	None
Response (Success)	We are sorry we can't consider your claim. Your policy only cover pre-post hospitalisation bills if they are related to hospitalization, day surgery, dialysis or chemotherapy
Response(Fail)	None

6. INTENT: CheckPaid no

INTENT	CheckPaid no

User phrase	no, nope, nah
Input Context	CheckPaidContext
OutputContext	None
Description	To check all answer must be Yes to proceed with claim: - Have you already paid the bills in full
Response (no)	We are sorry we can only consider the claim when the bills are fully paid

7. INTENT: UploadDocumentIntent

User phrase	Yes, Can, ok, confirm
Input Context	CheckPaid Context
OutputContext	UploadContext
Description	To upload documents Document scanning Bill document text searching
Prompt	Please prepare and upload your final bill and receipts, with supporting medical report and, bank statement for claim payment
Webhook	**/claim/uploadfile (file)**
Response (Success)	Event(FileUploaded)
Response(Fail)	The file is not supported filesAnnotate error

8. INTENT: DocumentUploaded

Event	FileUploaded
Input Context	UploadContext
OutputContext	FileUploaded UploadContext
Description	Event triggered when FileUploaded successfully
Prompt	We have received your document. File: bill.jpg Bill detail Mount Elizabeth Date: 09-07-2019 Ref: 3066 Amount: 471.40

	Please provide a bit more information about this bill. For example: Stitch removal, or physiotherapy
Webhook	Triggered from webhook
Response (Success)	attachUrl,originalFileName,OCRResult,Company, InvoiceDate, InvoiceNo, InvoiceAmount
Response(Fail)	None

9. INTENT: SubmitClaimIntent

User phrase	Bill for chemotherapy treatment Specialist visit on NUH
Input Context	FileUploaded UploadContext
OutputContext	None
Description	Submission to the system
Prompt **(Slot filling)**	**Specialist** (Entity: specialist) slot prompt**:** What is the specialist, or doctor name of the clinic you visited? Can you provide the name of the specialist, or doctor name you have visited for this claim? **TotalExp** (Entity: sys.number) slot prompt**:** How much is the total amount of the bill What is the total expense incurred **DateOcc** (Entity: sys.date) slot prompt**:** When is this bill occurred What is the date of visit **HospitalDate** (Entity: sys.date) slot prompt**:** When is the admission date of the related earlier hospitalisation?
Webhook	/claim/intenthandler: **submitClaimIntentHandler** Parameters: InsuredID,ClaimRemark,PolicyNo, Specialist, TotalExp, DateOcc,HospitalDate, PolicyNo, attachUrl

Response (Success)	Claim submitted. Reference: %s. This acceptance is subject further assessment. We will send Additional information may be required for claim approval.
Response(Fail)	There is an error in the claim submission: %s. Please check and try again with valid information. Please try again, I'm unable to communicate to claim system at the moment

10. **INTENT: _Default Fallback Intent**

Description	Fallback intents are triggered if a user's input is not matched by any of the regular intents or if it matches the training phrases below. Read more in documentation
Prompt	I didn't get that. Can you say it again? I missed what you said. What was that? Sorry, could you say that again? Sorry, can you say that again? Can you say that again? Sorry, I didn't get that. Can you rephrase? Sorry, what was that? One more time? What was that? Say that one more time? I didn't get that. Can you repeat? I missed that, say that again?
Webhook	None

7.1.2 Chatbot Fulfillments in python

- /claim/uploadfile
- Handle file upload action from Dialogflow chatbot session (**INTENT: UploadDocumentIntent)**
- Save the uploaded image document
- Initiate receipt document recognition consist of integration to google cloud vision API to convert the document/image to text format
  Reference: https://cloud.google.com/vision
- Receipt recognition: The google vision result text format is not Perform text line segmentation from google vision result text
  Reference: https://github.com/rmrezarp/line-segmentation-gcp-vision

- Receipt recognition: Utilize fuzzy text search on the receipt lines based on keyword similarity with accuracy setting decrease from perfect match (1) until lowest accuracy (0.6) for each receipt lines
- regular expression defined in parser.config.yml configuration file as the knowledge model of the receipt document.
- difflib.get_close_matches to match keyword with list of phrases in the text line
- difflib.SequenceMatcher to match keyword with text line
  Logic is built on top of reference: https://github.com/mre/receipt-parser

- Trigger Dialogflow event: **fileUploaded** on the response and return event parameter such as receipt/bill company, date, bill ref, and amount resulted from the text search

- /claim/intenthandler: checkInsuredIntentHandler
- Checks valid policy by insured ID requested from Dialogflow chatbot session (**INTENT: CheckInsuredIntent)**
- Call policy repository [/HealthPolicy/byinsuredid/] to find valid policy based on provided insured ID
- On success, return policy dataset or
- On fail, return The provided ID is not valid

- /claim/intenthandler: submitClaimIntentHandler
- Handle claim submission intent from Dialogflow chatbot session (**INTENT: SubmitClaimIntent)**
- Perform validation towards claim data submitted
- Call claim repository [/MedicalClaim/policyno/] to find main claim number based on provided insured ID and return the claimno based on DateOcc=HospitalDate
- Call claim repository [/MedicalClaim/] to post claim data submitted
- On success, return created claimno
- On fail, return error message

7.2 Claim Classifier

Claim classifier classifies claims using business rules. Knowledge base for the classifier consists of sample data from the insurance company and expert knowledge on the business rules. The classifier is implemented using Clips expert system which uses

forward chaining. The expert knowledge was programmed into fact templates and rules using clips programming language. The classifier is notified of new claim submissions via the “NEW_CLAIM_SUBMITTED” event on the event dispatcher service.It fetches the facts of these new claims and sends these to the rule engine.
The facts send to Clips consists of

Claims details - Amount, claimtype, occurrence date, parent claim details, diagnosis Hospital type, doctor information
Policy details - Policy age, policy expiry date, balance, rider details, autoclaim allowed flag, past claims status list
Insured details - Past illness, outstanding premium, past claims count

The Rules implemented are

Rule	**Description**	**Clips Engine Verification**
outstanding_premium	Rule to check if there is any outstanding premium in main policy or rider	;checking outstanding premium (defrule premium (declare (salience 30)) (Insured(outstanding ~0)) =>(assert(reason outstanding_premium)))
New policy	Auto approval starts after the 1st year of policy	(defrule durationconstraint (Policy(policyduration ?pd)) (test (< ?pd 12)) => (assert(reason policy_new)))
Past pending or rejected	Rule to check if there are pending or rejected claims under the policy	(defrule status (Status(pending ?p)(rejected ?r)) (test (>= ?p 1)) (test (>= ?r 1)) => (assert(reason Pending_or_Rejected_claims)))
First claim	Rule to check if it’s Insured’s first claim with the policy	(defrule first_claim (declare (salience 30)) (Insured(claimsnum_total ?ct)) (test (= ?ct 0)) =>(assert(reason first_claim)))

Autoclaim allowed	Rule to determine if the policyholder has allowed claims to undergo Auto Claim verification	(defrule autoallowed (Policy(auto_allowed ~Y)) => (assert(reason not_autoallowed)))
Blacklisted doctor	Rule to determine if the doctor or specialist associated with the claim in blacklisted by the company	(defrule dconstraint (Doctors(black_list Y)) =>(assert(reason doctor_in_blist)))
Diagnosis code verification	Rule to determine if the claim has a diagnosis code which the company does not insure against.	(defrule diagnosisallowed (Diagnosis(diagnosis_code ~None) (autoreject Y)) => (assert(reason diagnosiscode_not_allowed)))
Verify occurrence date	Rule to check that the date of occurrence for the claim is before the expiry of the policy	(defrule dateconstraints (Claims(occurance_date ?odd ?omm ?oyy)) (Policy(policy_exp ?pdd ?pmm ?pyy)) => (if(eq ?pyy ?oyy) then(if(eq ?pmm ?omm) then(if(> ?odd ?pdd) then(assert(reason policy_expired)) (assert(autoclaim no))) else(if(> ?omm ?pmm) then(assert(reason policy_expired)) (assert(autoclaim no))))) else(if(> ?oyy ?pyy) then(assert(reason policy_expired)) (assert(autoclaim no))))

Policy balance check	Rule to check if the policy balance is less than the claim amount	(defrule amountconstraint (Policy(policy_balance ?pb)) (Claims(claimtotal ?ca)) (test(< ?pb ?ca)) => (assert(reason amount_exceeds_balance)))
Amount for hospital types	Rules to check the amount constraints for different hospital types.	(defrule pHospitalconstraint (Hospital(type V)) (Claims(claimtotal ?amt)) (test(> ?amt 6000)) => (assert(reason amount_exceeds_for_hospital_type)))
Claimitems limit check	Rule to verify that the claim amount for each line item or product is within its product limit	(defrule Plimitcheck (Claimproducts (pcode ?p)(amount?a)(limit ?l)) (test (> ?a ?l)) => (assert(reason amount_exceeds_limit)))
Past illness	Checking illness specified before policy started as part of assessing Insured's past	(defrule preillness (declare (salience 30)) (Insured(pre_illness ?)) =>(assert(reason pre-existing_illness)))
Autoclaim limit	Policy allows only 5 auto claims per policy year	(defrule autolimit (declare (salience 30)) (Insured(autonum ?pa)) (test (> ?pa 5)) =>(assert(reason autoclaim_limit_reached)))
Check for Parent Claim	For pre/post hospitalization claim parent hospitalization claim is required	(defrule mainclaimcheck (Claims(billcategory PP)(mainclaim ?m)) (test (= ?m 0)) => (assert(reason no_parentclaim_found_for_PPclaim))
Duration check for PP	Pre/post hospitalization claims should be within 180	(defrule claimduration (Claims(billcategory PP)

	days from the hospitalization occurrence	(mainclaim ~0)(duration ?md)) (test (> ?md 180)) => (assert(reason PPclaim_submitted_after_180days))

The classifier makes sure that all the constraints are met and the Insured's past is good before approving a claim. At the first encounter of verification failure, the clips engine returns with reason for rejection.

The classifier after receiving new facts about the claim which is whether it can be approved or not, update data by changing the Status to approved for approved claims and updating the failure reason for the ones that require intervention.

Reference: http://fumblog.um.ac.ir/gallery/668/big-2.pdf
https://kcir.pwr.edu.pl/~witold/ai/CLIPS_tutorial/CLIPS_tutorial_2.html
https://clipspy.readthedocs.io/en/stable/

7.3 Claim Distributor

Here we are considering the assignment problem of incoming new claims to the Staffs based on their availability, as a mixed integer programming (MIP) problem.

A mixed-integer programming (MIP) problem is one where some of the decision variables are constrained to be integer values at the optimal solution. We are using Google OR-Tools to create an assignment solver.

In our assignment problem, we are assuming the cost, for any staff to work on a claim, to be equal, as we do not have data to say how much longer or shorter any staff will take than others.

Whenever the ClaimDistributer is invoked, it will populate all the newly received claims that are classified for manual intervention in the database, and assign them to the Staffs, except for whenever they are absent.

Consider there are *s* staffs to execute *c* claims, that are new and classified for manual approval when the ClaimDistributer is invoked. Then each claim has to be assigned to

exactly one staff and each staff can be assigned to a maximum of 8 claims/day. The cost of a claim *j* to be approved by staff *i*, is $c_{i,j}$. Since the cost is assumed to be equal, $c_{i,j}$ = 1 for all *i, j*. The purpose is to find the assignment with minimum cost. The problem can be modeled by introducing binary variables $x_{i,j}$, *i=1:s*, *j=1:c*, with the meaning that $x_{i,j}$=1 if staff *i* is assigned to claim *j* and $x_{i,j}$=0 otherwise.

Objective:

$$\text{Min } \sum^{s}_{i=1}\sum^{c}_{j=1} c_{i,j}. X_{i,j}$$

Constraints:

$$l \leq \sum^{c}_{j=1} x_{i,j} \leq u \quad \text{(bound constraint)}$$

where,
l is the minimum number of claims that at least needs to be assigned to each staff. This helps in distributing the claims in a round robin manner.
u is the maximum number of claims that a staff can take based on the number of hours he will be active, excluding his meeting hours or absent hours.

8. Project Performance & Validation

1. Claim submission chatbot:
   The current MVP chatbot is able to submit hospitalisation related claims and documents with limited validations. More claim types and conversation flows would be value-add to the claim submission process, which will improve customer satisfaction.

2. Claim submission OCR: Bill document recognition accuracy:
   From our testing we noticed around 15 out of 20 bill documents detected accurately (75% accuracy level). The feedback and correction on the detection is possible in Chatbot can be the next thing to improve.

   The knowledge base for the bill document format is currently limited, and the improvement on the knowledge model such as keywords and regex database will help to improve accuracy level

3. Claim distributor:
   In the next iteration, we can consider optimizing the schedule based on the skill sets of the staff. Not all staff may be qualified to handle all claims. Finding an optimal solution

where schedules are fit based on availability and skill sets while maintaining fairness is something we can explore next.

9. Installation Guide

Refer to the installation guide in /UserGuide/01 - Installation Guide.pdf

10. User Guide

Refer to the installation guide in /UserGuide/02 - User Guide.pdf

11. Individual Contribution

Team Member: Bhavya Rema Devi

Contribution: Data preprocessing, Knowledge Representation, Classifier implementation with clips programming, testing, documentation

Choosing the right rule engine for knowledge representation was a challenge with many options around. After reviewing many, we decided on Clips as it is well documented and can be easily integrated in Python. The required data for implementing rules were organised into fact templates and the rules were written in clips programming language.

The project allowed me to understand the way an expert system works, improve coding skills and to learn clips programming. I now understand how a knowledge based system can represent data more efficiently with added flexibility. With an insurance company, there is a definite, but large set of business rules. My team members had been very supportive and with their help, I was able to narrow down the rules and implement them. I also learned about the importance and simplicity of developing services in containers especially when it comes to integrating different microservices to work together.

Team Member: Elisa Tjhin

Contribution: Chatbot, data model, knowledge/business rule, testing, documentation

The integration to rule engine (clips), optimization tool (or-tools), chatbot (Dialogflow), text recognition and text search, and adaptation to the existing business rule is quite challenging to deliver, but I learned from the decisiveness, commitment, and knowledge from the team members has overcome it. I feel grateful to all team members for the openness and the opportunity to learn through support and feedback to each other.

For the business rule engine implementation, I learned that sometime modification of the rules may need changes to existing codes as well, which may introduce some risks to existing implementation. This happens especially when adding new facts which are not captured previously. It is important to identify possible facts required in the initial rule implementation.

In the project implementation, using containers for development involves multiple systems minimized configuration effort and allowed us to focus on development of solutions.
In chatbot design, I learnt that information collection with bot conversational methods can be tricky, and strategy to combine the conversational UI with document recognition is beneficial, for information retrieval, verification and ranges of potential use scenarios. However, document recognition is required to be very accurate and it is another challenge to work on.

Team Member: Francis Low

Contribution: System architecture, systems integration, overall Docker setup, supporting systems (Object Repository, Administrative Dashboard, Chatbot Frontend), documentation

Through this project, I realised that technology is not enough to make a good chat bot. A big part has to do with proper UX design. A lot of thought needs to go into designing the user flow to make it seem "smart".

I also realised the benefits of using a rule engine. Coming from a programming background, I struggled to understand the power of a rule engine, since we can also write those rules in code. I have come to understand that a rule engine can allow you to separate business logic from code. That way, it is much more maintainable and scalable. It is also explainable. You can see which rules failed and which passed, which is a requirement in some scenarios (e.g in an insurance / medical setting).

We had many rounds of discussion involving how to approach the claim distribution problem, and I personally found that the same problem can be framed in many different ways, and thereby affecting the design of constraints and cost function, and ultimately, the results.

Having selfless team members greatly accelerated the learning process. Each of us tackled a separate problem and shared our learnings with each other which made things a lot easier.

Team Member: Sindhuja Kumaran

Contribution: Database modelling, Search Optimization, testing, documentation.

From this project, I learnt the major steps involved in building an Expert System like Knowledge Acquisition, Knowledge Modelling. I could see how the working memory with the rules and data submitted by the users in chatbot, is utilised to make a decision and how the decisions are explainable.

It is very interesting to see how intents and entities are recognised in dialogflow. It helps me to understand how Classification is being applied here to classify the intent and how the entities are identified to acquire data.

In this project, I have tried to solve the problem of assigning claims to staffs as Constraint Satisfaction Problem. I decided to use Google OR-Tools using python. I used mixed-integer programming (MIP) to make the decision to assign or not, satisfying the capacity constraints. I also used Flask and got to use other python libraries, like pandas and numpy. I can see how these libraries make it easier to manipulate with data.

12. Appendices

- Appendix A - System Architecture
- Appendix B - Data Model
- Appendix C - Claim Submission / Classification Process Flow
- Appendix D - Claim Distribution Process Flow