AI Driven Insurance Claims: Smart Claim Decision Assistant powered with OpenAI

Author: Anantkreshna Vedanarayanan
Email: anantkreshnav@gmail.com

GitHub Source code: https://github.com/anantkreshnav1/AutoAI-InsuranceClaim-OpenAI.git

#OpenAI #Azure #AutoClaimAI #SmartClaimDecisions
#InsureTechRevolution #AIDrivenClaims
#FutureOfInsurance #ClaimSimplified #AIInsuranceAssistant
#DigitalClaimDetective #InstantClaimAnalysis #NextGenInsureTech

Problem Description:

Accident Report Analysis and Automated Claim Decision

Insurance companies receive thousands of accident reports daily. Manually analysing these reports for claim decisions is time-consuming and can be prone to human error. An automated system can save time, reduce errors, and streamline the claim processing workflow.

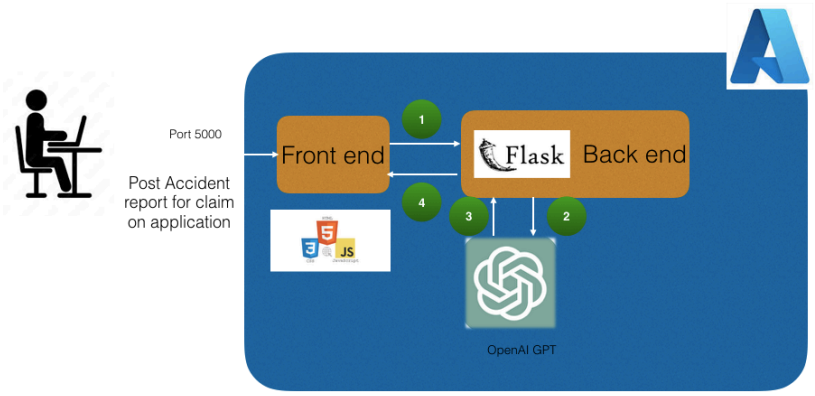
Solution:

Implement a web-based application where users can input accident reports. This application will use OpenAI's GPT model in the backend to analyse the report and decide whether a claim should be approved or not. The solution will be hosted on Azure cloud services to ensure scalability, security, and high availability.

Technical Architecture:

Architecture:

Auto Insurance Claim Assistant



1	Claim report from UI passed to Flask Python backend
2	Claim report to be shared as Prompt to to Open AI GPT API
3	Open AI response to be passed back to Flask
4	Response passed back to Front end UI

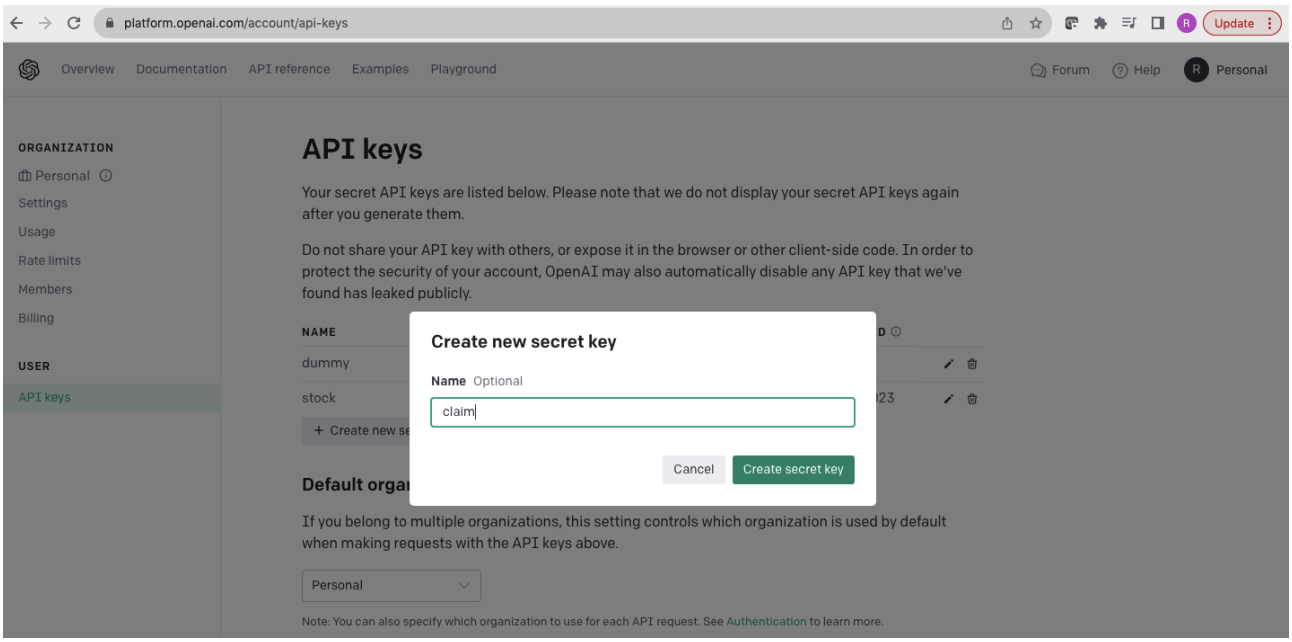
Frontend: Web-based application hosted on Azure VM.
Backend: Flask application integrated with OpenAI's API.
Model: GPT model from OpenAI, interfaced through its API.

Detailed Execution Plan:

1. Generate OpenAI API Key

Navigate to: https://openai.com/

By clicking the “Create secret Key” button as noted below, open ai api key can be generated. Save it in a safe place as it cant be accessed again once created.



2. Azure VM setup:

Set up an Azure subscription.

Configure the Azure VM (Ubuntu)

- Create Resource Group
- Create a Azure VM (Linux Ubuntu VM)
- Open port 5000 in network to allow traffic
- Install Python libraries by executing the below commands in sequence:

```
sudo apt update
sudo apt install python3-pip
pip3 install Flask openai
```

3. Application Development:

Develop a Flask backend to handle the report submissions and communicate with OpenAI's API.

```
import os
import openai
from flask import Flask, render_template, request, jsonify

app = Flask(__name__)
openai.api_key = "<OPEN AI KEY>"  // provide the generated open ai key here

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/submit', methods=['POST'])
def submit():
    accident_report = request.form['accident_report']

    # Send the accident report to GPT-3 for decision
    response = openai.Completion.create(
        engine="text-davinci-003",
        prompt=f"Based on the following accident report, should the insurance claim be approved or denied as Decision? Provide the reason for approval or denial as Reason\n\n{accident_report}\n\nDecision:",
        max_tokens=1000
    )

    decision = response.choices[0].text.strip()

    return render_template('result.html', decision=decision)

if __name__ == '__main__':
```

```
    app.run(debug=True, host="0.0.0.0")
```

Design the frontend using HTML and integrate with the backend.
We will create a basic HTML page (**index.html and result.html**) under a folder "**templates**" with a form to input the user's question and to display the assistant's response.

index.html:

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/
bootstrap.min.css">
  <style>
    /* Add custom CSS styles for vibrant colors and styling */
    body {
      background-color: #f5f6f7;
    }

    .header {
      background-color: #007BFF;  /* A vibrant blue color */
      color: white;
      padding: 20px 0;
      text-align: center;
      border-bottom: 5px solid #0056b3;  /* A slightly darker blue to create a border
effect */
      margin-bottom: 20px;
    }

    .card {
      border-radius: 15px;
      box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2);  /* Soft shadow for card */
    }

  </style>
  <title>AutoClaim AI: Your Smart Claim Decision Assistant</title>
</head>

<body>
  <div class="header">
```

```
    <h1>AutoClaim AI: Your Smart Claim Decision Assistant</h1>
  </div>

  <div class="container mt-5">
    <div class="row justify-content-center">
      <div class="col-md-8">
        <div class="card">
          <div class="card-header bg-primary text-white">Enter Accident Write-up</
div>
          <div class="card-body">
            <form action="/submit" method="post">
              <div class="form-group">
                <label for="accident_report">Accident Report:</label>
                <textarea class="form-control" name="accident_report" rows="5"
required></textarea>
              </div>
              <button type="submit" class="btn btn-primary">Submit</button>
            </form>
          </div>
        </div>
      </div>
    </div>
  </div>
</body>

</html>
```

result.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/
bootstrap.min.css">
  <title>Claim Decision</title>
</head>
<body class="bg-light">
  <div class="container mt-5">
    <div class="row justify-content-center">
      <div class="col-md-8">
        <div class="card">
```

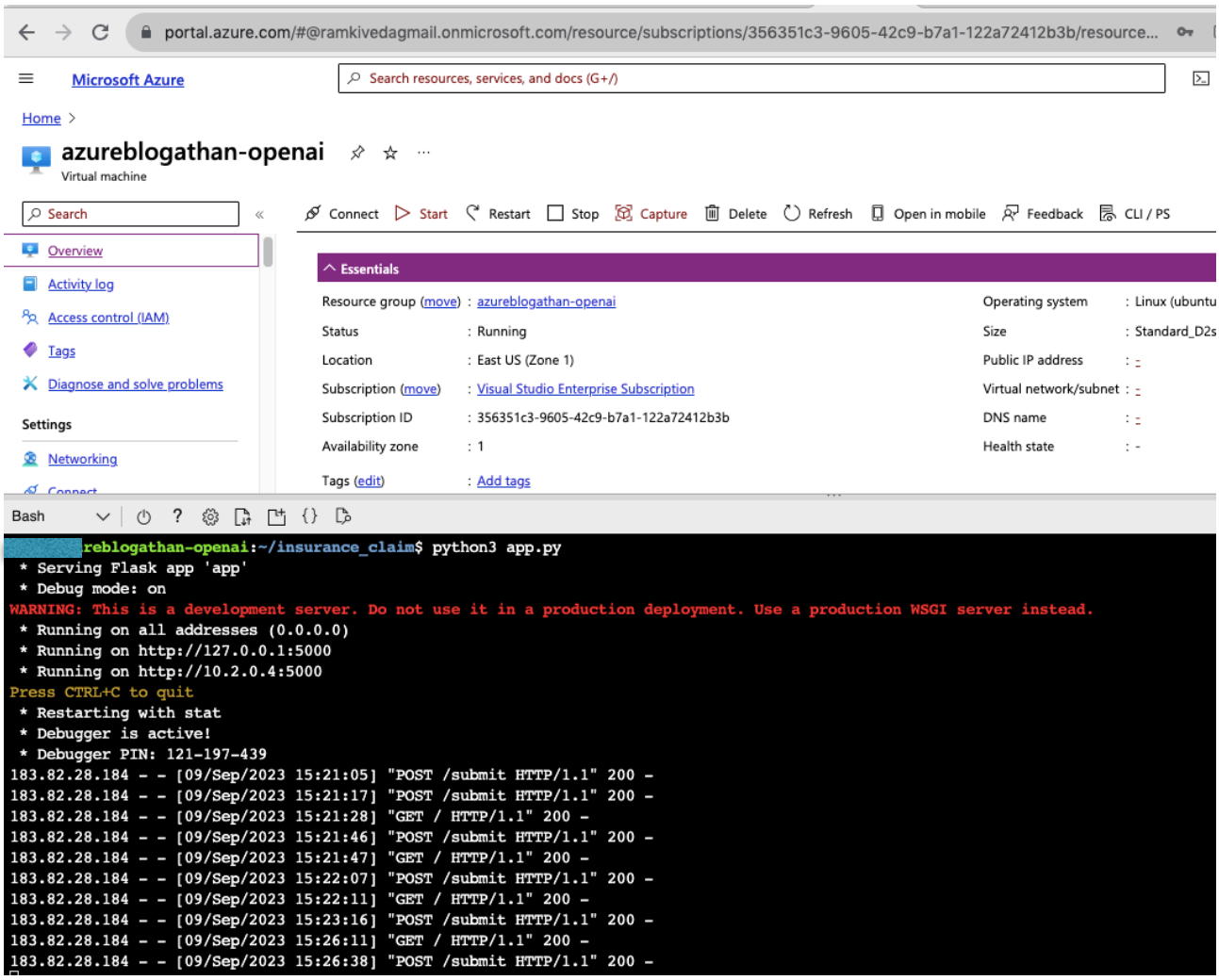
```
            <div class="card-header bg-primary text-white">Claim Decision</div>
            <div class="card-body">
              <p>{{ decision }}</p>
              <a href="/" class="btn btn-secondary">Go Back</a>
            </div>
          </div>
        </div>
      </div>
    </div>
</body>
</html>
```

4. Deployment:

Execute the application on Azure VM by running the below command:

```
python3 app.py
```

Ensure all connections to services (e.g., OpenAI) are secure and functional.

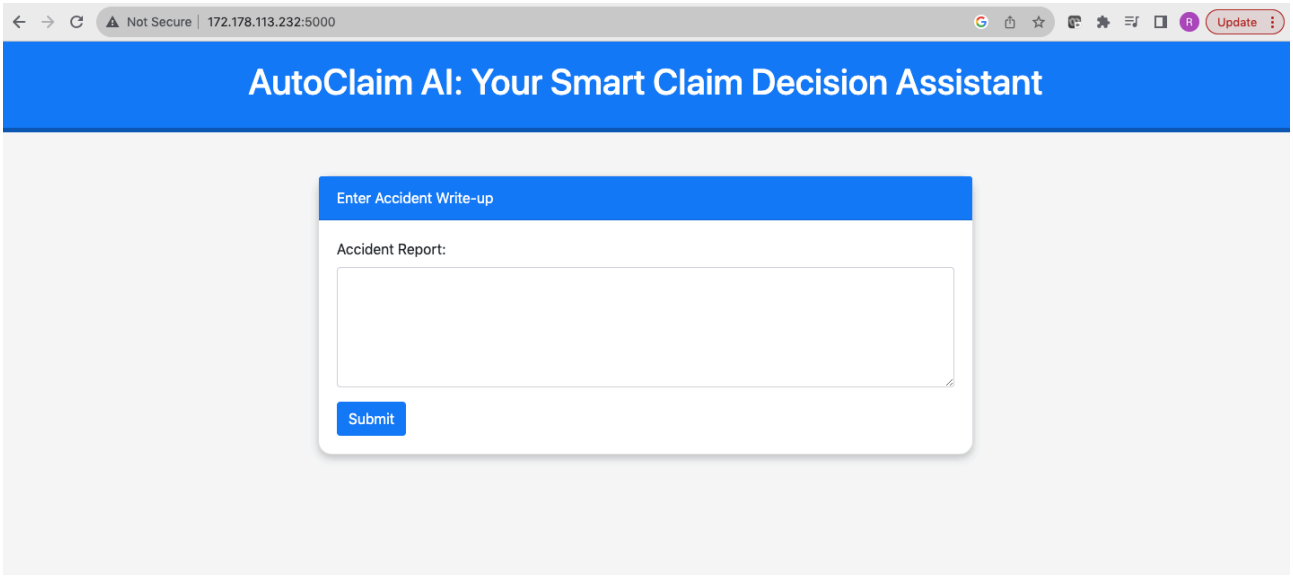


5. Testing:

Test the application for any bugs or vulnerabilities.

Validate the accuracy and reliability of the GPT model's decisions using a subset of accident reports. Regularly update the system and address any issues or bugs that arise.

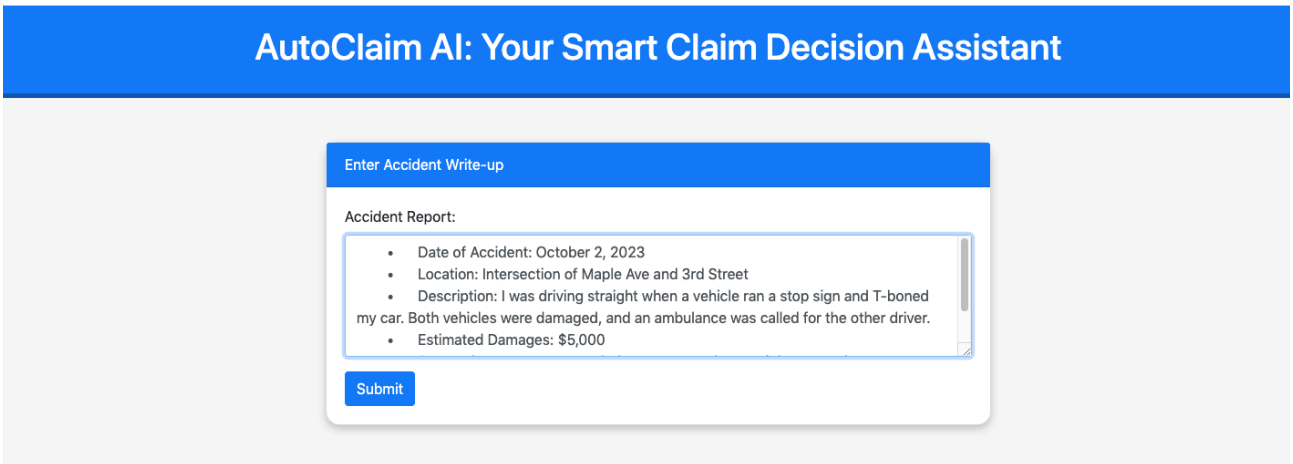
Launch the application on browser:



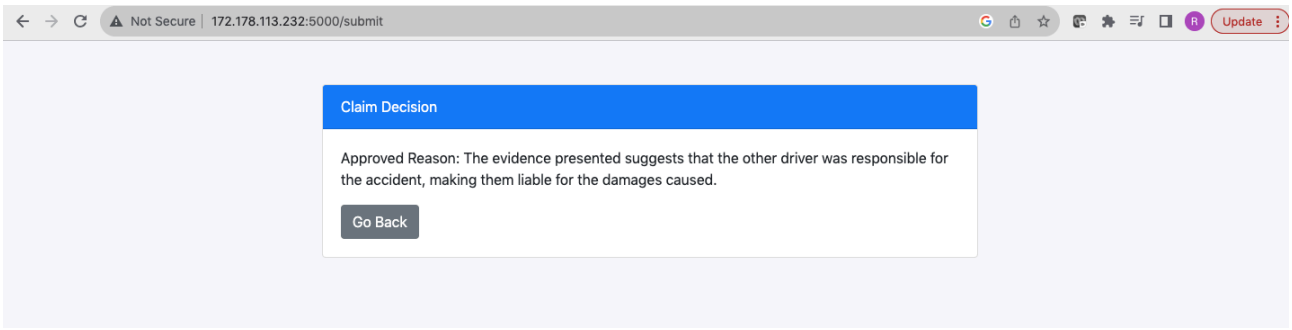
Claim Approval Sample:

Sample Claim Report for Approval Use case:

- Date of Accident: October 2, 2023
- **Location**: Intersection of Maple Ave and 3rd Street
- **Description**: I was driving straight when a vehicle ran a stop sign and T-boned my car. Both vehicles were damaged, and an ambulance was called for the other driver.
- Estimated Damages: $5,000
- **Supporting Documents**: Ambulance report, photos of damage, witness testimonies.

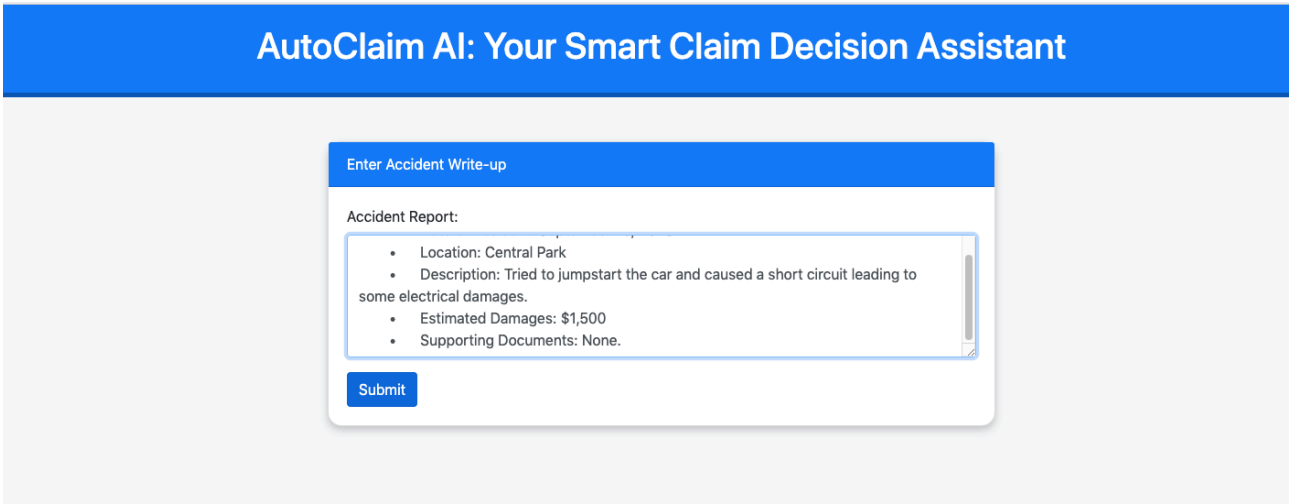


As expected, the predicted result is Approved with reason as noted below:

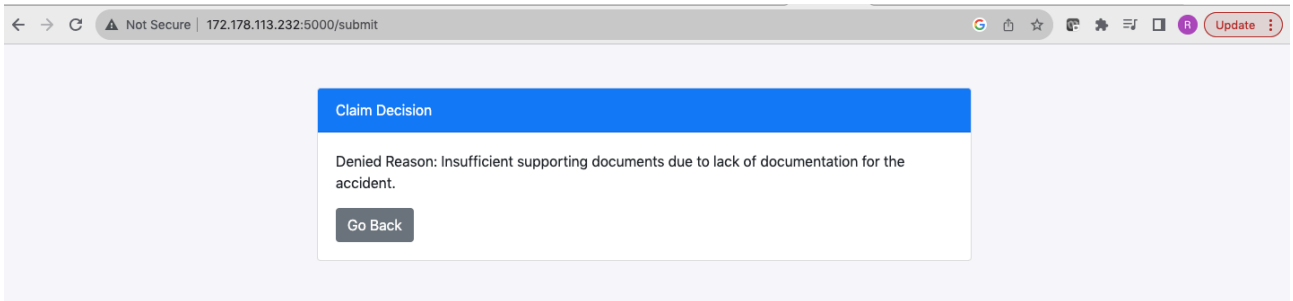


Sample Claim Report for Rejection Use case:

- **Date of Accident**: September 19, 2023
- **Location**: Central Park
- **Description**: Tried to jumpstart the car and caused a short circuit leading to some electrical damages.
- Estimated Damages: $1,500
- Supporting Documents: None.



As expected, the result is Denied with the reason as noted below:



Challenges in Implementation:

Accuracy of Decisions: The GPT model, while advanced, is not infallible. Some decisions might need human intervention.
Data Privacy: Handling sensitive accident reports requires strict data protection measures.
Scalability: Ensuring the system can handle a high number of concurrent submissions.
Integration Complexities: Integrating various Azure services and OpenAI's API seamlessly.

Benefits:

Efficiency: Automated decisions can drastically reduce the processing time for each claim.
Consistency: Minimise human error in claim decision-making.
Scalability: Azure services can scale as per the demand, ensuring consistent performance.
Security: Azure provides robust security measures, ensuring data safety.

Next Steps:

Feedback Loop: Implement a feedback mechanism for human agents to correct wrong decisions, improving the system over time.
Expand Use Cases: Look for other areas within the insurance domain where AI can assist, such as fraud detection.
Continuous Training: Periodically retrain the model using the latest accident reports and feedback.

Conclusion:

By leveraging the capabilities of OpenAI's GPT model and Azure's cloud services, we can develop a robust solution to automate accident report analysis and claim decisions. This will not only increase efficiency but also reduce costs for insurance companies. With the right implementation and continuous improvement, such a system can revolutionise the way insurance claims are processed.