ServiceNow AI Ticket Automation - Project Documentation

1. Project Overview

Objective:
- Automates incident management in ServiceNow using AI
- Predicts assignment groups, resolves known issues, optionally executes safe workarounds

Benefits:
- Faster incident routing and resolution
- Reduced manual effort for common issues
- Improved operational efficiency
- Audit logs and traceability of AI actions

2. Project Folder Structure

Create a folder named 'ticket_ai_project':

```
ticket_ai_project/
| - data.csv           # Training dataset for AI model
| - train_model.py      # Script to train the AI model
| - app.py              # Flask app to serve AI predictions via REST API
```

3. AI Dataset (data.csv)

Prepare ticket data for training the AI model. Include the following fields:

Column	Description
short_description	Short description of the incident
category	Category of the incident (Network, Desktop, Messaging, etc.)
team	Assignment group for the incident
known_issue	Whether it is a known issue (True/False)
resolution	Suggested resolution for known issues

Example CSV:

```
"short_description,category,team,known_issue,resolution"
"Email not syncing","Messaging","Messaging Team",True,"Restarted mail service"
"Cannot connect to network","Network","Network Team",False,"Manual investigation"
```

4. AI Model Training (train_model.py)

```
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
```

```
import joblib

data = pd.read_csv('data.csv')
vectorizer = CountVectorizer()
X = vectorizer.fit_transform(data['short_description'])

y_team = data['team']
X_train_team,  X_test_team,  y_train_team,  y_test_team  =  train_test_split(X,  y_team,
test_size=0.2, random_state=42)
team_model = RandomForestClassifier()
team_model.fit(X_train_team, y_train_team)

y_issue = data['known_issue']
X_train_issue, X_test_issue, y_train_issue, y_test_issue = train_test_split(X, y_issue,
test_size=0.2, random_state=42)
issue_model = RandomForestClassifier()
issue_model.fit(X_train_issue, y_train_issue)

joblib.dump(team_model, 'team_model.pkl')
joblib.dump(issue_model, 'issue_model.pkl')
joblib.dump(vectorizer, 'vectorizer.pkl')

print("Models trained and saved!")
```

5. AI REST API (app.py)

```
from flask import Flask, request, jsonify
import joblib

app = Flask(__name__)

team_model = joblib.load('team_model.pkl')
issue_model = joblib.load('issue_model.pkl')
vectorizer = joblib.load('vectorizer.pkl')

@app.route('/predict', methods=['POST'])
def predict():
    data = request.get_json()
    description = data.get('description', '')

    X = vectorizer.transform([description])
    team = team_model.predict(X)[0]
    known_issue = bool(issue_model.predict(X)[0])
    resolution = "Restarted service to resolve issue" if known_issue else ""

    return jsonify({
        "team": team,
        "known_issue": known_issue,
        "resolution": resolution
    })

if __name__ == '__main__':
    app.run(port=5000)
```

6. Flow Diagram

```
[Incident Created]
    |
    v
[Call AI REST API] --> Predict Team & Known Issue
    |
    |- If Assignment Group Empty -> Assign Default/Category Group
    |
    |- If Known Issue True -> Auto-Resolve
         |
         \-> Optional: Trigger MID Server Workflow (Restart Service)
    |
    v
[Incident Updated & Resolved]
```

7. References

- ServiceNow Business Rules: https://docs.servicenow.com/bundle/utah-platform-administration/page/administer/business-rules/concept/c_BusinessRules.html
- RESTMessageV2 API: https://developer.servicenow.com/dev.do#!/reference/api/rome/server_legacy/c_RESTMessageV2API
- ngrok Documentation: https://ngrok.com/docs

8. C

This documentation provides a complete end-to-end guide for AI-driven ServiceNow ticket automation.
- Local AI project setup
- Training and running AI models
- ServiceNow integration
- Auto-assignment, auto-resolution, and optional MID Server workarounds
- Testing, logging, and safety measures