Foresight Documentation

OVERVIEW.md

Executive Summary

Foresight is an innovative enhancement for ServiceNow that leverages advanced predictive analytics to improve IT support efficiency. By analyzing historical ticket submission patterns, Foresight predicts the three most likely support ticket categories that users may submit next. This proactive approach not only streamlines the ticket resolution process but also empowers IT teams to allocate resources more effectively, ultimately enhancing user satisfaction.

For stakeholders and sales teams, Foresight represents a strategic investment in IT service management. By integrating seamlessly with ServiceNow and utilizing a robust analytics pipeline powered by Snowflake, Foresight transforms raw ticket data into actionable insights. This capability allows organizations to anticipate user needs, reduce response times, and improve overall service delivery.

By adopting Foresight, organizations can expect to see a significant reduction in ticket resolution times, leading to increased productivity and a more responsive IT support environment. This positions Foresight as a critical tool for organizations looking to enhance their IT service management capabilities.

Key Features and Capabilities

- **Predictive Analytics**: Forecasts the next three ticket categories based on historical data.
- **Sequence-Based Frequency Analysis**: Utilizes user ticket patterns to enhance prediction accuracy.
- **Snowflake-Backed Analytics Pipeline**: Efficiently stores and processes historical ticket data.
- **FastAPI Prediction Service**: Provides a robust and scalable API for predictions.
- **ServiceNow Integration**: Seamlessly integrates with existing ServiceNow environments.

Business Value Proposition

Foresight delivers substantial business value by enabling organizations to: - **Enhance IT Efficiency**: By predicting ticket categories, IT teams can prioritize and address issues more effectively. - **Improve User Satisfaction**: Faster response times and proactive support lead to higher user satisfaction rates. - **Optimize Resource Allocation**: Anticipating ticket submissions allows for better planning and resource management. - **Leverage Data-Driven Insights**: Organizations can make informed decisions based on historical ticket trends.

Target Users and Use Cases

Target Users

- IT Support Teams
- Service Desk Managers
- IT Operations Managers
- Data Analysts

Use Cases

- **Proactive Ticket Management**: Anticipate and prepare for incoming ticket types based on historical patterns.
- **Resource Planning**: Allocate IT resources effectively based on predicted ticket volumes.
- **Performance Monitoring**: Use predictive insights to monitor and improve IT support performance.

Quick Start Guide for Developers

To get started with Foresight, follow these steps:

1. **Set Up the Environment**:

   - Ensure you have Python and FastAPI installed.
   - Set up a Snowflake account for data storage.

2. **Clone the Repository**:

   ```
   git clone https://github.com/your-repo/foresight.git
   cd foresight
   ```

3. **Install Dependencies**:

   ```
   pip install -r requirements.txt
   ```

4. **Run the FastAPI Service**:

   ```
   uvicorn main:app --reload
   ```

5. **Test the API Endpoints**:

   - Use tools like Postman or curl to test the available endpoints:
     - **Predict Next Ticket**: POST to `/predict-next-ticket`
     - **Health Check**: GET `/health`
     - **Upload Tickets**: POST to `/upload-tickets`

6. **Upload Ticket Data**:

   - Prepare your ticket data in the required format and upload it using the `/upload-tickets` endpoint.

For detailed API documentation, refer to the API Endpoints section in the README.

Links to Other Wiki Pages

- API Documentation
- System Architecture
- Installation Guide
- User Guide

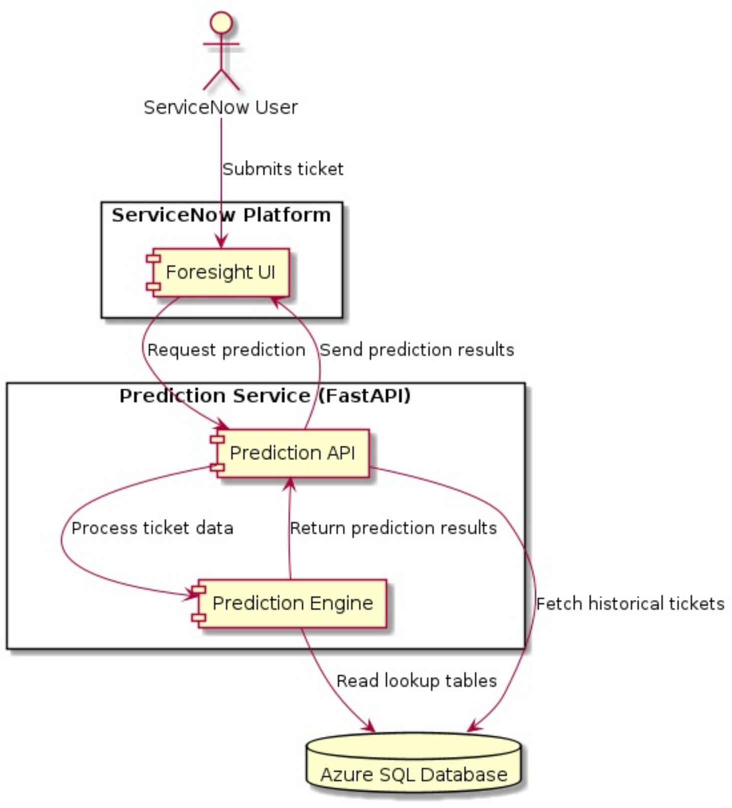
This comprehensive overview provides both technical and non-technical audiences with the essential information needed to understand and leverage the Foresight solution effectively.

ARCHITECTURE.md

Foresight - ServiceNow Ticket Prediction Architecture

1. System Architecture Overview

Foresight is designed to enhance the ServiceNow platform by predicting the next likely support ticket categories based on historical data. The architecture consists of several key components that work together to provide accurate predictions and seamless integration with ServiceNow.



Architecture Diagram

Main Components

Component	Description
ServiceNow	The source system generating support tickets.
Snowflake	A cloud-based data warehouse that stores the historical ticket dataset.
Prediction Engine	The core component that utilizes a frequency-based prediction model to forecast ticket categories.
FastAPI Service	A web service that exposes prediction APIs for external consumption.
Foresight UI	A user interface integrated within ServiceNow that displays predictions to users.

2. Data Flow Diagram Explanation

The data flow within the Foresight system can be summarized as follows:

1. **Ticket Submission**: Users submit support tickets through ServiceNow.
2. **Data Storage**: Submitted tickets are stored in Snowflake for historical analysis.
3. **Data Processing**: The Prediction Engine analyzes the historical ticket data to identify patterns and predict future ticket categories.
4. **API Interaction**: The FastAPI Service provides endpoints for users to request predictions based on their ticket submission history.
5. **User Interface**: The Foresight UI presents the predictions back to the users within the ServiceNow environment.

3. Technology Stack Breakdown

The technology stack for Foresight is carefully chosen to ensure performance, scalability, and ease of integration.

Layer	Technology	Justification
Backend	Python, FastAPI, SQLAlchemy	Python offers rich libraries for data analysis; FastAPI provides high-performance APIs; SQLAlchemy facilitates database interactions.
Data Processing	Jupyter, Python Data Analysis Libraries	Jupyter is ideal for exploratory data analysis and model development; Python libraries (e.g., Pandas, NumPy) are essential for data manipulation.
Infrastructure	Snowflake, ServiceNow	Snowflake is a scalable cloud data warehouse; ServiceNow is the existing platform for ticket management.

4. Integration Points with External Systems

Foresight integrates primarily with the following external systems:

- **ServiceNow**: Foresight enhances the ServiceNow platform by

predicting ticket categories based on user interactions.
- **Snowflake**: Acts as the data warehouse for storing historical ticket data, enabling the Prediction Engine to perform analysis and generate predictions.

5. Security Considerations

Security is a critical aspect of the Foresight architecture. Key considerations include:

- **Data Privacy**: Ensure that all ticket data is anonymized to protect user identities.
- **API Security**: Implement authentication and authorization mechanisms for API endpoints to prevent unauthorized access.
- **Data Encryption**: Use encryption for data at rest and in transit to safeguard sensitive information.

6. Scalability Aspects

Foresight is designed to scale efficiently:

- **Cloud Infrastructure**: Utilizing Snowflake allows for automatic scaling of storage and compute resources based on demand.
- **Microservices Architecture**: The FastAPI service can be deployed in a containerized environment, enabling horizontal scaling to handle increased API requests.
- **Load Balancing**: Implementing load balancers can distribute incoming requests across multiple instances of the FastAPI service.

7. Component Interaction Diagram Description

The component interaction diagram illustrates how different parts of the system communicate:

1. **User Interaction**: Users interact with the Foresight UI within ServiceNow.
2. **API Requests**: The UI sends requests to the FastAPI Service for predictions.
3. **Prediction Processing**: The FastAPI Service communicates with the Prediction Engine to fetch predictions based on the provided ticket sequences.
4. **Data Retrieval**: The Prediction Engine queries historical data from Snowflake to perform its analysis.
5. **Response Delivery**: The FastAPI Service returns the predictions to the Foresight UI for display to the user.

8. Database Schema Overview

While the specific database schema is not detailed in the README, the following entities can be inferred based on the functionality:

- **Tickets Table**:
  - `ticket_id` (string): Unique identifier for each ticket.
  - `category` (string): Category/type of the ticket.
  - `user_id` (string): Identifier for the user who submitted the ticket.
  - `timestamp` (datetime): Timestamp of when the ticket was created.

This schema supports the analysis of ticket submission patterns and facilitates the prediction of future ticket categories.

This document provides a comprehensive overview of the architecture of the Foresight project, detailing its components, data flow, technology stack, integration points, security considerations, scalability aspects, and interaction diagrams. Developers new to the project can use this information to understand how Foresight operates and how to contribute effectively.

Foresight API Documentation

Summary of Endpoints

HTTP Method	Endpoint Path	Description
POST	`/predict-next-ticket`	Predicts the next 3 ticket categories.
GET	`/health`	Checks the health status of the API.
POST	`/upload-tickets`	Uploads ticket data for analysis.

API Overview

The Foresight API is a FastAPI service that predicts the next likely support ticket categories based on historical ticket submission sequences. It integrates with ServiceNow and utilizes a Snowflake-backed analytics pipeline.

Base URL

The base URL for the API is not explicitly mentioned in the README. Please consult your deployment documentation for the specific URL.

Authentication

No authentication methods are specified in the README. Ensure to implement security measures as needed for your deployment.

Error Codes and Handling

The README does not specify error codes or handling mechanisms. It is recommended to implement standard HTTP status codes for error handling (e.g., 400 for bad requests, 500 for server errors).

Endpoints

1. Predict Next Ticket

- **HTTP Method:** POST
- **Endpoint Path:** `/predict-next-ticket`

Request Example:

```
{
    "ticket_sequence": ["incident", "problem", "change"]
}
```

Response Example:

```
{
    "predictions": [
        "incident",
        "problem",
        "change"
    ]
}
```

Parameters:

- `ticket_sequence` (array of strings): An array representing the historical sequence of ticket categories.

2. Health Check

- **HTTP Method:** GET
- **Endpoint Path:** `/health`

Response Example:

```
{
    "status": "healthy",
    "timestamp": "2023-10-01T12:00:00Z"
}
```

Description:

Returns the health status of the API.

3. Upload Tickets

- **HTTP Method:** POST
- **Endpoint Path:** `/upload-tickets`

Request Example:

```
{
    "tickets": [
        {
            "ticket_id": "INC001",
            "category": "incident",
            "user_id": "user123",
            "timestamp": "2023-10-01T10:00:00Z"
        },
        {
            "ticket_id": "PRB001",
            "category": "problem",
            "user_id": "user123",
            "timestamp": "2023-10-01T11:00:00Z"
        }
    ]
}
```

Response Example:

```
{
    "status": "success",
    "tickets_uploaded": 2,
    "message": "Ticket data uploaded successfully."
}
```

Parameters:

- `tickets` (array of objects): An array of ticket objects containing:
  - `ticket_id` (string): Unique identifier for the ticket.
  - `category` (string): Category/type of the ticket.
  - `user_id` (string): User who submitted the ticket.
  - `timestamp` (string): ISO 8601 timestamp of ticket creation.

Description:

Uploads historical ticket data for analysis and model training.

Code Examples

Python Example

```python
import requests

# Predict Next Ticket
url = "http://your-api-url/predict-next-ticket"
data = {
    "ticket_sequence": ["incident", "problem", "change"]
}
response = requests.post(url, json=data)
print(response.json())

# Health Check
health_response = requests.get("http://your-api-url/health")
print(health_response.json())

# Upload Tickets
upload_url = "http://your-api-url/upload-tickets"
upload_data = {
    "tickets": [
        {
            "ticket_id": "INC001",
            "category": "incident",
            "user_id": "user123",
            "timestamp": "2023-10-01T10:00:00Z"
        },
        {
            "ticket_id": "PRB001",
            "category": "problem",
            "user_id": "user123",
            "timestamp": "2023-10-01T11:00:00Z"
        }
    ]
}
upload_response = requests.post(upload_url, json=upload_data)
print(upload_response.json())
```

cURL Example

```
# Predict Next Ticket
curl -X POST http://your-api-url/predict-next-ticket \
-H "Content-Type: application/json" \
-d '{"ticket_sequence": ["incident", "problem", "change"]}'

# Health Check
curl -X GET http://your-api-url/health

# Upload Tickets
curl -X POST http://your-api-url/upload-tickets \
-H "Content-Type: application/json" \
-d '{
    "tickets": [
        {
            "ticket_id": "INC001",
            "category": "incident",
            "user_id": "user123",
            "timestamp": "2023-10-01T10:00:00Z"
        },
        {
            "ticket_id": "PRB001",
            "category": "problem",
            "user_id": "user123",
            "timestamp": "2023-10-01T11:00:00Z"
        }
    ]
}'
```

This documentation provides a comprehensive overview of the Foresight API, including endpoint details, request/response examples, and code snippets for integration. For further details, consult the specific deployment or additional documentation as required.

DEPLOYMENT.md

Table of Contents

Prerequisites

Before deploying the Foresight application, ensure you have the following:

Software

- Python 3.8 or higher
- FastAPI
- SQLAlchemy
- Docker (if applicable)

Credentials

- Access to ServiceNow API for ticket submission and retrieval.
- Snowflake account with necessary permissions to access and store data.

Access

- Access to the server or cloud environment where the application will be deployed.
- Permissions to deploy Docker containers (if using Docker).

Environment Setup

Development

- Clone the repository to your local machine.
- Set up a virtual environment and install dependencies using `pip install -r requirements.txt`.

Staging

- Create a staging environment that mirrors production.
- Configure the environment variables for staging.

Production

- Set up a production server with the necessary resources.
- Ensure that the Snowflake and ServiceNow integrations are properly configured.

Configuration Management

- Use environment variables to manage configuration settings for different environments (development, staging, production).
- Consider using a configuration management tool like Ansible or Terraform for infrastructure as code.

Step-by-Step Deployment Instructions

1. **Clone the Repository**

   ```
   git clone https://github.com/your-repo/foresight.git
   cd foresight
   ```

2. **Set Up Environment Variables** Create a `.env` file in the root directory and populate it with the necessary environment variables.

3. **Install Dependencies**

   ```
   pip install -r requirements.txt
   ```

4. **Run Database Migrations** If applicable, run any database migrations required for the application.

5. **Start the Application**

   ```
   uvicorn main:app --host 0.0.0.0 --port 8000
   ```

Docker Deployment

If deploying with Docker, follow these steps:

1. **Build the Docker Image**

   ```
   docker build -t foresight:latest .
   ```

2. **Run the Docker Container**

   ```
   docker run -d -p 8000:8000 --env-file .env foresight:latest
   ```

Cloud Deployment Specifics

- If deploying to a cloud provider (e.g., AWS, Azure), ensure that the necessary security groups, IAM roles, and permissions are set up.
- Use cloud-native services for database management (e.g., Amazon RDS for Snowflake).

Environment Variables Reference

Variable Name	Description
`SNOWFLAKE_ACCOUNT`	Snowflake account name
`SNOWFLAKE_USER`	Snowflake username
`SNOWFLAKE_PASSWORD`	Snowflake password
`SERVICENOW_URL`	URL for ServiceNow API
`SERVICENOW_USER`	ServiceNow API username
`SERVICENOW_PASSWORD`	ServiceNow API password

Health Checks and Monitoring

- Use the `/health` endpoint to check the health status of the API.
- Implement monitoring tools (e.g., Prometheus, Grafana) to track application performance and health.

Health Check Example

```
curl -X GET http://localhost:8000/health
```

Expected Response:

```
{
    "status": "healthy",
    "timestamp": "2023-10-01T12:00:00Z"
}
```

Troubleshooting Common Issues

- **Application not starting**: Check for missing environment variables or incorrect configurations.
- **API returns errors**: Review logs for stack traces or error messages.
- **Database connection issues**: Verify Snowflake credentials and network access.

Rollback Procedures

In case of deployment failure, follow these steps to rollback:

1. **Stop the Current Application**

   ```
   docker stop <container_id>
   ```

2. **Deploy the Previous Version** If using Docker, tag the previous version and run it:

   ```
   docker run -d -p 8000:8000 foresight:<previous_version>
   ```

3. **Verify Application Health** Use the `/health` endpoint to ensure the application is running correctly after rollback.

RELEASES.md

Version History

Version	Release Date	Feature Summary
v1.2.0	2026-02-25	Prediction model improvements
v1.1.0	2026-02-10	Lookup table generation
v1.0.0	2026-01-15	Initial ticket prediction release

Detailed Changelog

v1.2.0 - Prediction Model Enhancements (2026-02-25)

New Features

- Improved sequential frequency algorithm for better predictions.
- Optimized lookup table generation to enhance performance.
- Faster prediction API response time.

Bug Fixes

- Fixed minor bugs related to data handling in the prediction engine.

Breaking Changes

- None.

Migration Notes

- No special migration steps required for this version.

---

v1.1.0 - Lookup Table Generation (2026-02-10)

New Features

- Introduced lookup table generation for improved prediction accuracy.

Bug Fixes

- Resolved issues with ticket data upload validation.

Breaking Changes

- None.

Migration Notes

- Ensure to regenerate lookup tables after upgrading to this version for optimal performance.

---

v1.0.0 - Initial Ticket Prediction Release (2026-01-15)

New Features

- Initial release of the ticket prediction model.
- Basic API endpoints for ticket prediction and health check.

Bug Fixes

- Initial bug fixes related to API response formatting.

Breaking Changes

- None.

Migration Notes

- This is the first release; no migration is necessary.

Upcoming Features Roadmap

- Enhanced user interface for prediction results (Planned for v1.3.0).
- Integration with additional data sources for improved accuracy (Planned for v1.4.0).

Deprecation Notices

- No deprecations at this time.

Semantic Versioning Explanation

This project follows Semantic Versioning principles: - **MAJOR** version when making incompatible API changes, - **MINOR** version when adding functionality in a backward-compatible manner, and - **PATCH** version when making backward-compatible bug fixes.

How to Upgrade Between Versions

To upgrade between versions, follow these steps:

1. **Backup Your Data**: Always back up your existing data before upgrading.
2. **Review Release Notes**: Check the changelog for any breaking changes or migration notes.
3. **Update Your Codebase**:
   - Pull the latest version from the repository.
   - Install any new dependencies as specified in the documentation.
4. **Run Migration Scripts**: If applicable, run any migration scripts provided in the release notes.
5. **Test Your Application**: After upgrading, thoroughly test your application to ensure everything functions as expected.
6. **Monitor Logs**: Keep an eye on application logs for any unexpected behavior post-upgrade.

By following these steps, you can ensure a smooth transition between versions.