



Building the Future of Agentic AI For IT Management

Team Name : Team Visionary

Team Leader Name : Sudarshanam Yessasvini

Problem Statement : Theme 3 - Growth / Financial Improvement

Project Title: Prophet - *The AI Co-Pilot for MSP Growth & Retention*



Brief about the Idea

Headline: From Reactive Reporting to Predictive Strategy.

Body: Prophet is an AI-powered dashboard that transforms raw MSP operational data into predictive intelligence. It doesn't just report on past performance; it forecasts future risks and opportunities.

Core Function:

Predicts Client Churn weeks in advance by analyzing support tickets, communication sentiment, and resolution times.

Calculates True Profitability per client by weighing Monthly Recurring Revenue (MRR) against real support costs.

Generates AI-Powered Recommendations to retain at-risk clients, improve service delivery, and unlock new revenue streams.



The Opportunity & How It's Different:

Headline: A \$700B+ Market Running on Outdated Tools.

How is it different?

Existing Tools: Most MSP tools are reactive (e.g., "Client X submitted 10 tickets last month").Our Solution (Prophet): Prophet is proactive and predictive (e.g., "Client Y has a 95% risk of churning next quarter. Here’s why and how to stop it.").

Fills a Critical Gap: No unified platform exists that uses AI specifically for MSP business intelligence and strategic forecasting. We move from operational data to strategic insight.

Existing Solutions	Prophet
<ul style="list-style-type: none"> ✓ Reporting ✓ Alerts ✓ 	<ul style="list-style-type: none"> ✓ Predictive Analytics ✓ AI Recommendations ✓ Profitability Analysis

How It Solves The Problem

The Problem:

Unexpected Client Churn: MSPs discover a client is unhappy only when they leave.

Hidden Unprofitability: A high-revenue client can be a net loss due to massive support overhead.

Lack of Strategic Foresight: MSPs struggle to allocate resources and budget effectively for future growth.

Our Solution:

Churn Prediction: Flags at-risk clients early, allowing for intervention.

True Profitability Engine: Reveals which clients are truly profitable ("VIPs") and which are loss-making ("Vampires").

Data-Driven Foresight: Provides actionable insights to strategically allocate technicians, negotiate contracts, and pitch new projects.



USP of the Proposed Solution

Headline: Predictive Intelligence, Tailored for MSPs.

Unique Selling Propositions:

- 1. Predictive, Not Just Descriptive:** We forecast business outcomes, not just describe past events.
- 2. Actionable AI Insights:** Amazon Bedrock generates plain-English recommendations, not just confusing data charts.
- 3. Holistic Profitability View:** The only tool that combines financial (MRR) and operational (time spent, tickets) data for a true profit/loss view.
- 4. Seamless Integration:** Built on the SuperOps API, making it a natural extension for existing users.

List of features offered by the solution



1. Predictive Client Health Score

Concept: A simple, at-a-glance scorecard for each client.

Visual: A series of client list items, each with:

Client Name: ABC Corp

Health Score Gauge: A semi-circle gauge (like a speedometer) pointing to a value (e.g., 92/100). Color-code it: **Green (81-100)**, **Yellow (61-80)**, **Red (0-60)**.

Trend Indicator: A small arrow next to the score pointing  (improving) or  (declining).

Why it works: It translates complex data into an intuitive, instantly understandable metric, similar to a credit score.

PREDICTIVE CLIENT HEALTH SCORE



2. Churn Risk Alert System

Concept: An alert panel that highlights critical risks.

Visual: A "High Priority Alerts" section on the dashboard.



Use a red exclamation mark (!) or a flaming fire icon 🔥 as the visual indicator.

Show a list: "ABC Corp: 92% Churn Risk | Key Drivers: Rising Ticket Sentiment Negativity, Slow Resolution Times"

Another entry: "XYZ Ltd.: 45% Churn Risk | Key Driver: Contract Expiring in 30 Days"

Why it works: It creates a sense of urgency and directs the MSP owner's attention to the most pressing issues immediately.

True Profit/Loss Dashboard

- **Concept:** A clear visualization of which clients make money and which lose money.
- **Visual:** A **horizontal bar chart** for a selected client.
 - The left side of the chart is "Costs" (technician time, software licenses, hardware).
 - The right side is "Revenue" (Monthly Recurring Revenue - MRR).
 - **The bar is color-coded:** Green if Revenue > Costs, Red if Costs > Revenue.
 - **The bar ends at a final number:** Net Profit: +\$1,200 or Net Loss: -\$450.
- **Why it works:** It makes the abstract concept of "true profitability" concrete and undeniable at a single glance.



AI-Generated Recommendations

Concept: The AI acting as a strategic consultant.

Visual: A text box or "sticky note" element on the client's profile.

Title: "AI Recommendations"

Content: Use bullet points with easy-to-follow icons.

 "Schedule a Quarterly Business Review to address concerns."

 "Propose a managed upgrade project for their outdated server. Estimated cost: \$5,000"

 "Upsell advanced cybersecurity monitoring package. 70% match based on their industry."

Why it works: It shows the solution is not just a passive tool but an active partner in problem-solving, providing clear next steps.



Resource Forecasting

Concept: Predicting future workload to optimize staffing.

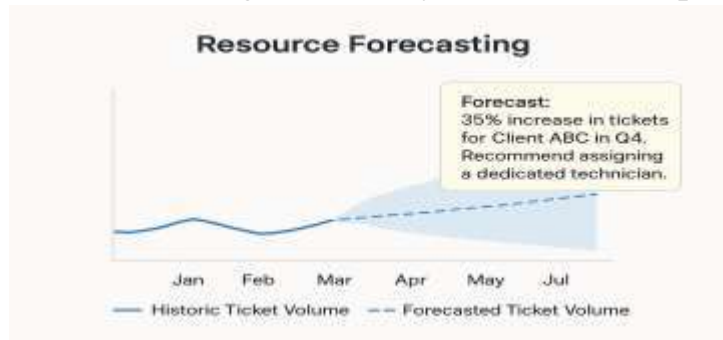
Visual: A line chart comparing "Historic Ticket Volume" vs. "Forecasted Ticket Volume" over the next 3-6 months.

The historic data is a solid line.

The forecast is a dashed line, perhaps with a shaded area showing a confidence interval.

A callout box highlights: "Forecast: 35% increase in tickets for Client ABC in Q4. Recommend assigning a dedicated technician."

Why it works: It demonstrates strategic value beyond finance, helping with operational planning and preventing team burnout.



Secure & Scalable (AWS Architecture)

Concept: Trust through technology.

Visual: A simple, high-level architecture diagram.

Draw a cylinder for "SuperOps Data" flowing into a cloud labeled "AWS".

Inside the cloud, draw:

A gear icon for "AWS Glue (ETL)"

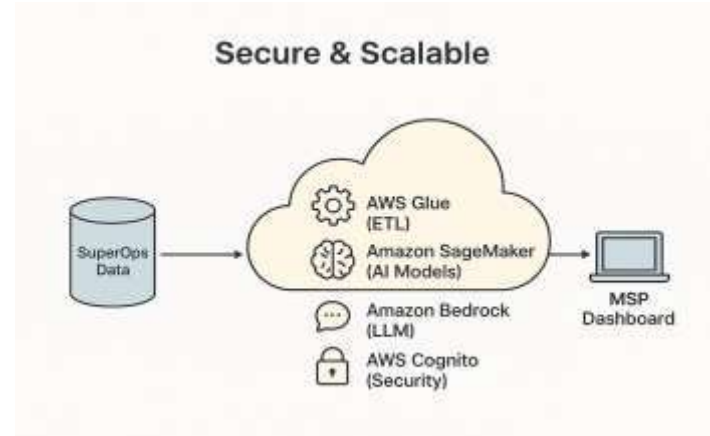
A brain icon for "Amazon SageMaker (AI Models)"

A chat bubble icon for "Amazon Bedrock (LLM)"

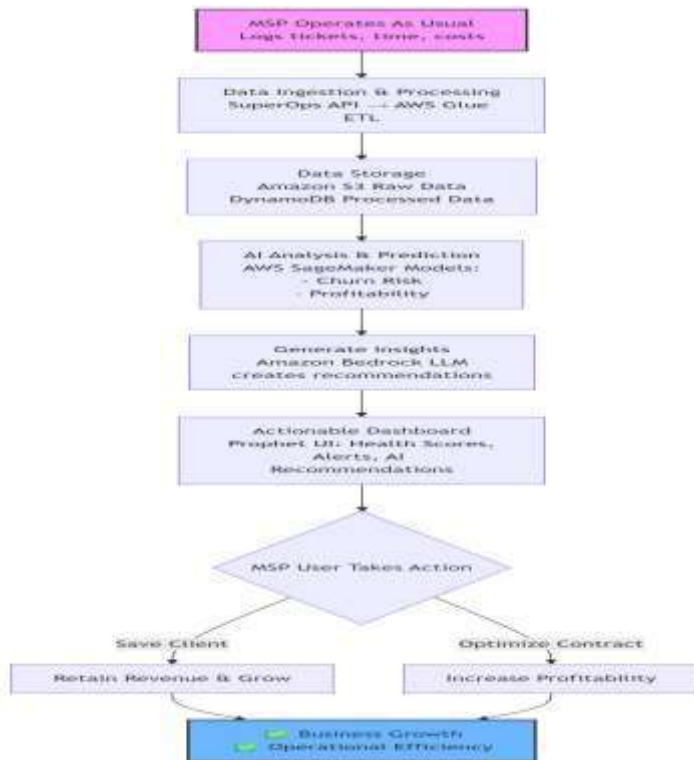
A lock icon for "AWS Cognito (Security)"

Arrows show data flowing from SuperOps, through these services, and out to a laptop/tablet icon labeled "MSP Dashboard".

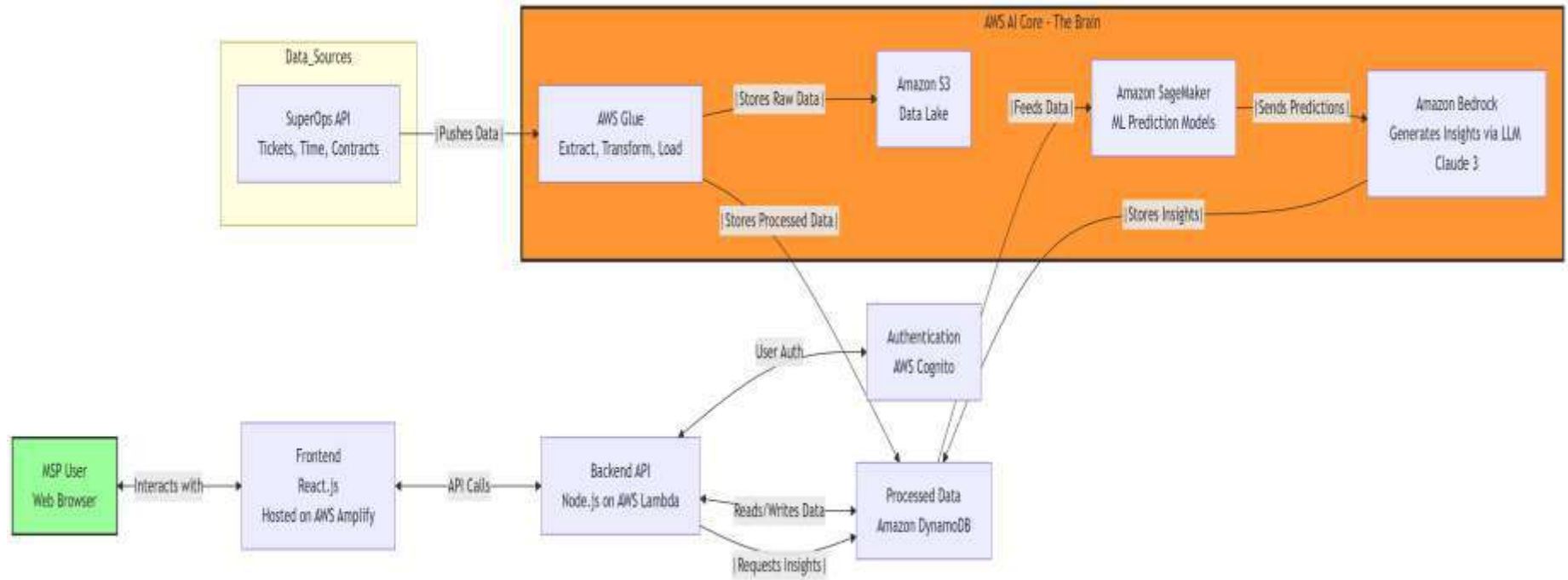
Why it works: It assures technical judges of the solution's robustness and scalability while being simple enough for non-technical audiences to grasp the flow of data.



Process Flow Diagram: How "Prophet" Works for an MSP

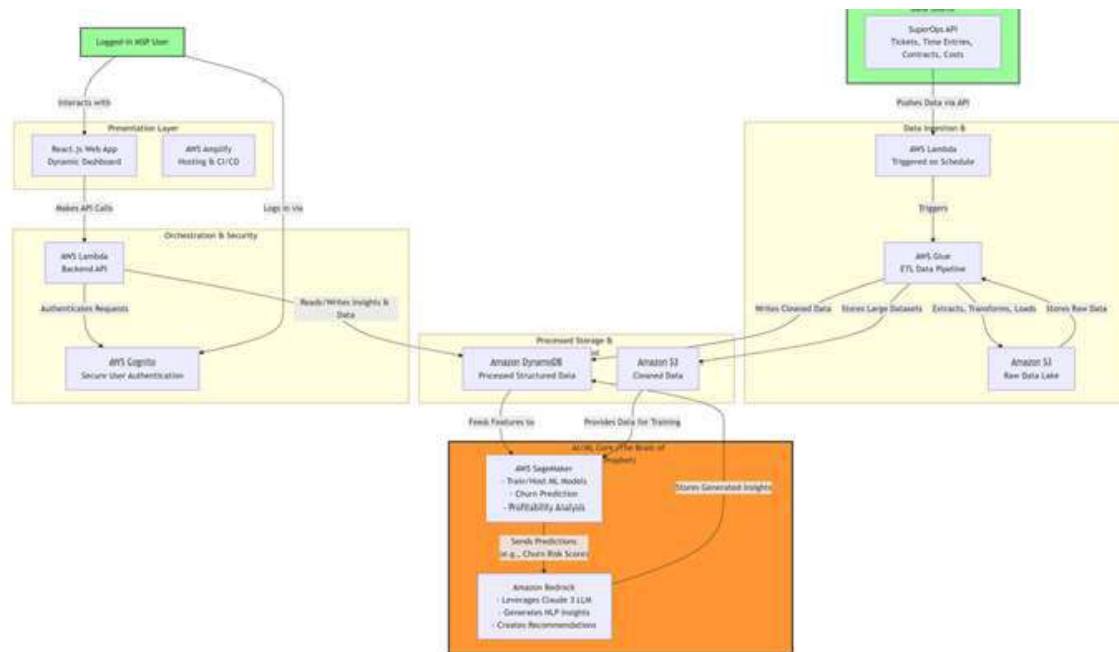


High-Level System Architecture Diagram:



Wireframes/Mock diagrams of the proposed solution (optional)

Architecture diagram of the proposed solution



Technologies to be used in the solution:

1. Artificial Intelligence & Machine Learning (The Core Intelligence)

AWS SageMaker: To build, train, and deploy our machine learning models for **predicting client churn** and **calculating true profitability**. This allows for scalable and managed ML workflows.

Amazon Bedrock: To leverage a powerful Large Language Model (LLM) like **Anthropic's Claude 3** for generating natural-language, actionable insights and recommendations from the model's predictions. This is the key to our "AI-Generated Recommendations" feature.

Python (with libraries like Scikit-learn, Pandas, NumPy): The primary language for developing and testing our machine learning models before deploying them on SageMaker.

2. Cloud Infrastructure & Backend (The Engine Room)

AWS Lambda: For serverless backend functions (API endpoints). This ensures our backend scales automatically and we only pay for the compute time we consume.

Amazon API Gateway: To create, publish, and secure the RESTful API that connects our frontend to the backend Lambda functions and other services.

AWS IAM (Identity and Access Management): To define secure roles and permissions for all AWS services interacting with each other, following the principle of least privilege.

AWS CloudWatch: For monitoring, logging, and gaining operational insights into the entire application stack.

3. Data Processing & Storage (The Foundation)

AWS Glue: For our fully managed **Extract, Transform, Load (ETL)** service. It will automatically catalog, clean, and prepare the data from SuperOps for analysis.

Amazon S3 (Simple Storage Service): To act as our secure, scalable, and cost-effective **data lake** for storing raw and intermediate data.

Amazon DynamoDB: A fast and flexible NoSQL database service for storing processed, structured data that our application needs for real-time access (e.g., client profiles, generated insights).

4. Frontend & User Interface (The Dashboard)

React.js: A modern and efficient JavaScript library for building a dynamic, component-based, and responsive user interface.

Recharts / D3.js: JavaScript libraries for building clean, interactive, and insightful data visualizations and charts for our dashboard.

CSS3 / Styled Components: For styling and creating a modern, professional, and user-friendly interface.

5. Security & Authentication

AWS Cognito: To handle secure user **authentication, authorization, and user management** for our application. It provides a built-in login/sign-up UI and integrates seamlessly with other AWS services.

6. Deployment & DevOps

AWS Amplify: For continuous deployment and hosting of our React.js frontend. It provides a seamless CI/CD pipeline, automatically deploying changes when we push code to our repository.

Docker: To containerize our application components, ensuring consistency across different development and deployment environments.

Git / GitHub: For version control and collaborative code management.

7. Integration

SuperOps API: The primary source of truth for all MSP operational data (tickets, time entries, contracts, assets). Our solution will ingest data from this API to power its predictions.

Estimated implementation cost (optional):

We have designed Prophet with cost-efficiency at its core. By leveraging AWS's extensive Free Tier offerings and a serverless architecture, we estimate the total monthly cost for running a functional proof-of-concept to be approximately \$30.

This low barrier to implementation is a key advantage. It proves that our solution is not only technically advanced but also commercially viable and accessible for startups and developers. Our architecture is designed to scale efficiently, meaning costs would only grow in direct correlation with user growth and revenue."

Business Mindset: It shows you're thinking like a founder, not just a developer. You understand that cost is a critical factor in a product's success.

AWS Proficiency: It demonstrates you've taken the time to understand the pricing models of various AWS services and how to use them efficiently.

Realism: It provides a realistic, defensible number rather than a wild guess or omitting the topic entirely.

Scalability Story: It hints at the scalable nature of serverless computing—costs remain low until usage grows.

Add as per the requirements for the hackathon:

Idea Submission Form completed on the SuperHack portal.

Idea Submission Deck (PPT) uploaded and finalized.

GitHub Repository created and made public with a detailed README.md.

Project Video recorded, edited, and uploaded to YouTube.

DevPost Submission Page completed with all project details, team info, and links.

Live Demo Environment deployed on AWS Amplify for judging.

Team Profile updated on all platforms.



Building the Future of Agentic AI For IT Management

THANK YOU