Hope Care

Project Overview

A full-stack predictive analytics platform that helps healthcare providers and insurers.

Uses a two-stage ML pipeline:

- **Stage 1:** Risk stratification model predicts probability of chronic conditions.
- **Stage 2:** ROI model estimates proactive vs. reactive treatment costs.

Technology stack:
React (frontend), FastAPI (backend), Scikit-learn/XGBoost (ML models).

Provides an interactive interface for patient data upload, analysis, and visualization.

Designed as a proof-of-concept for real-world adoption by healthcare providers and insurers.

Objectives

Predictive Risk Assessment

- Develop a machine learning model that stratifies patients' risk for multiple chronic conditions using demographic and clinical data.

ROI Calculation

- Implement a cost-prediction model that translates patient risk into financial terms, comparing proactive (preventive) care with reactive (emergency) treatment.

Data Visualization

- Design an intuitive web interface that clearly presents patient risks, costs, and potential savings using tables, summary cards, and charts.

Dashboard

- Deliver a functional prototype that healthcare providers and insurers can use to understand the value of preventive care.

Scope

Risk Stratification

Risk stratification for conditions like heart failure, COPD, kidney disease.

ROI Analysis

ROI analysis showing cost savings from preventive treatment.

Data Upload

Data upload via CSV and real-time predictions.

Result Visualization

Visualization of risk scores, cost comparisons, and savings in a user-friendly interface.

Stakeholders

Primary Users

Healthcare providers - for patient management and cost control.

Secondary Users

Policy makers → for population health management.

Development Team

Team Lead & Backend Developer - Pradeep Kumar A

Frontend Developers - Kishore R,Jeyvanti S

Backend Developer - Suriya SK

Cloud Engineer - Bala Mohanan M

Hackathon Stakeholders

Mentors - Rakesh
- SrinivasaRao Pinnaka
- Deepa R

HopeCare: Key Deliverables

Core Backend (FastAPI) →

- /predict_risk: ML model for patient risk tiers (1-5).
- /estimate_roi: ML model for intervention cost savings.

Machine Learning Models ←

- Risk Stratification: XGBoost model for multi-condition risk prediction.
- ROI Estimation: Regression model to project financial outcomes.

User Interface (React SPA) →

- Dashboard: High-level overview of patient risk and ROI.
- Patient Profiles: Detailed view with risk factors and intervention history.

Timeline

Friday, August 29 – Project Kickoff

- Selected the problem statement.
- Finalized the technology stack for the project.

Saturday, August 30 – Initial Design

- Designed the initial project architecture.
- Collected and cleaned datasets.
- Front-end team began conceptualizing the web application's design.

Sunday, August 31 – Version 1 Development

- Started building and training the risk stratification model.
- Front-end team implemented Version 1 of the web application.

Monday, September 1 – Refinement

- Refined the risk stratification model for improved accuracy.
- Front-end team reviewed and updated the application's design.

Tuesday, September 2 – First Mentor Meeting & Backend Integration

First Mentor Meeting: Presented use case and architecture, received approval, implemented ROI model, integrated FastAPI, and refined front-end design.

Wednesday, September 3 – Version 2 & Backend Connection

- Front-end team completed Version 2 web design with improved domain relevance.
- ROI model completed and integrated with the backend.
- Successfully connected and tested the backend with Version 1 of the web design.

Thursday, September 4 – Second Mentor Meeting

- Conducted the second mentor meeting:
- Demonstrated Version 1 web application with stratification data processing demo.
- Received corrections and suggestions for additional features.
- Successfully connected stratification and ROI models into Version 2 web design.

Friday, September 5 – Final Demo & Deployment

- Conducted the final mentor meeting:
- Showcased corrected implementations.
- Delivered a full demo using Version 2 web design.
- Received final approval for project deployment.
- Initiated deployment process on AWS.

Functional Requirements

User Interaction

- Users must be able to upload CSV files through the front-end.
- System should validate uploaded files (schema, missing values, duplicates).
- Users must be able to generate reports with one click.

Risk & Condition Stratification

- Assign patients to a Risk Tier (1–5) using ML models.
- Predict future medical conditions using a multi-label classifier.
- **<u>Output must include:</u>**

  Name
  Age
  Present Medical Condition
  Future Predicted Medical Condition
  Risk Tier (1–5)

Reporting & Visualization

- Generate reports showing:
- Patient details
- Risk tier
- Cost predictions
- Display interactive dashboards (charts/graphs) for trends and outcomes.
- Export reports in CSV/PDF formats.

Model Management

- Store trained models in AWS S3 .
- Support versioning of models (.pkl files).
- Ensure retraining pipelines are available for model updates.

System Deployment

- Backend must expose APIs to serve predictions.
- Frontend must consume backend APIs to display insights.
- System must support scalable deployment on cloud infrastructure (AWS EC2).

Technical Specification

Frontend

React.js → User interface for file uploads, dashboards, reports.

Backend

- FastAPI → API layer for ML models and data processing.
- Python (Pandas, Scikit-learn, XGBoost) → Data cleaning, feature engineering, ML models.

Models

- Risk Stratification Model: XGBoost for tiering.
- Condition Classifier: Multi-label classification.
- Cost Regression Model: Predicts intervention vs. non-intervention costs.
- Explainability: SHAP for interpretability.

Deployment

AWS EC2 with Uvicorn → Model serving.

AWS S3 → Model artifacts, CSV files.

Cloud Infrastructure

AWS EC2 → Scalable compute.

AWS S3 → Object storage.

Constraints and Assumptions

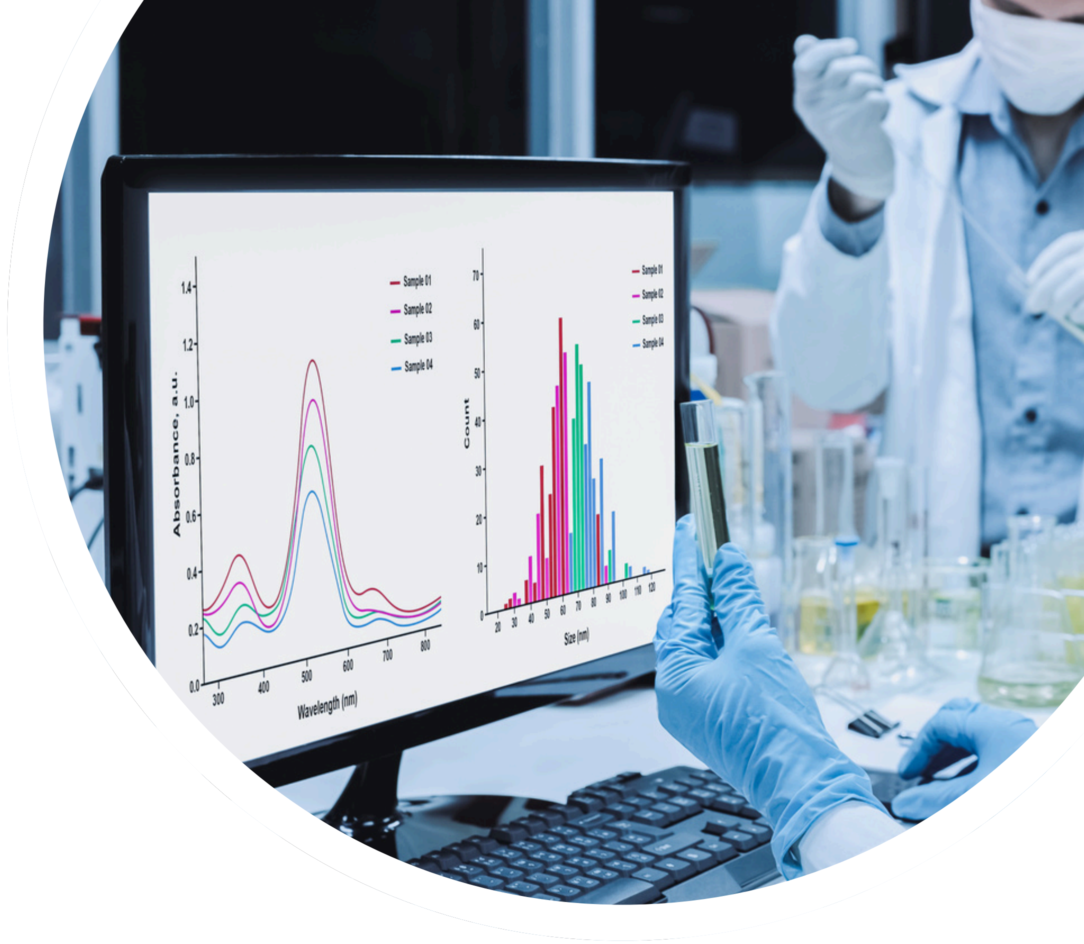
Constraints:

- Limited hackathon time → no deep hyperparameter tuning or advanced optimizations.
- AWS credits and computing resource limits may restrict execution.
- Limited hardware availability for training ML models.

Assumptions:

- The care manager will provide patient records for analysis.
- The patient records will contain the required attributes (e.g., AgeSP_CHF,SP_CHRNKIDN, and other clinical features).
- The care manager will use the dashboard to view results (risk scores, ROI analysis, and savings).

Risk Assesement



Data Availability Risk

- **Description**: Access to large, high-quality patient and cost datasets was limited.
- **Mitigation**: For the hackathon, we designed the models to work with smaller, structured datasets and ensured the pipeline can easily scale when richer data becomes available.

Scalability Risk

- **Description:** The prototype was initially designed for limited patient records.
- **Mitigation**: We built the system with modular components (frontend–backend separation, API-based communication) so it can be extended to handle batch processing in future iterations.

Performance Risk

- **Description**: Limited hackathon time, hardware, and cloud credits restricted heavy training and tuning.
- Mitigation: We used lightweight but effective model like XGBoost to balance accuracy and efficiency, ensuring predictions run smoothly on available resources.

Deployment Risk

- **Description:** Cloud deployment posed challenges due to time and credit limitations.
- **Mitigation:** We focused on ensuring smooth local deployment first and structured the backend for easy containerization (Docker) so it can be moved to the cloud with minimal effort later.

User Adoption Risk

- **Description**: Non-technical care managers may find machine learning outputs difficult to interpret.
- **Mitigation:** We implemented clear tables, summary cards, and highlighted key factors behind predictions to make the dashboard intuitive and actionable.

Success Criteria

Functional Success

- The platform allows care managers to input patient records and view risk stratification and ROI results seamlessly through the dashboard.
- The system correctly generates risk scores, cost comparisons, and potential savings for patient cases.

Technical Success

- Backend (FastAPI) and frontend (React) integrate smoothly, with secure and reliable API communication.
- Machine learning models run within acceptable time limits, even on limited hardware.
- The dashboard presents results in an intuitive, clear, and visually appealing format.

Usability Success

- Non-technical stakeholders (e.g., care managers) can easily navigate the dashboard without needing technical expertise.
- Visualizations (tables, summary cards, graphs) are easy to interpret and actionable.

Scalability & Maintainability Success

- The system design (frontend–backend separation, modular ML pipeline) supports future scaling and integration with real-world datasets.
- Documentation for setup, usage, and future improvements is complete and accessible.

Impact Success

- The platform demonstrates clear cost savings by comparing proactive vs. reactive healthcare scenarios.
- The project builds confidence among healthcare providers/insurers to explore predictive analytics for preventative care.

Appendix - References

Datasets

CMS Synthetic Beneficiary Data –CMS Public Use Files

Frameworks & Libraries

FastAPI Documentation – Fastapi
React Documentation – React
Scikit-learn Documentation – Scikit-learn

Research / Background Reading

Wagner J. et al., “Implementing Risk Stratification in Primary Care: Challenges and Strategies,” Journal of the American Board of Family Medicine 32(4): 585–595 (2019).
Link: Risk Stratification

“Projected return on investment of a corporate global health programme,” BMC Public Health (2019).
Link: Roi