Project Design Phase-II

Technology Stack (Architecture & Stack)

Date	15 February 2026
Team ID	LTVIP2026TMIDS76348
Project Name	Prosperity Prognosticator – Machine Learning for Startup Success Prediction
Maximum Marks	4 Marks

Technical Architecture

The Prosperity Prognosticator system follows a layered architecture consisting of a User Interface layer, an Application Logic layer, a Machine Learning layer, and a Data Storage layer. The frontend collects startup metrics (funding details, milestones, relationships) via an HTML form. The Flask backend receives and preprocesses the input data, then invokes the trained Random Forest classifier to generate a startup success prediction (Acquired or Closed) which is rendered back to the user on the result page.

Technical Architecture – Prosperity Prognosticator

Layer	Component	Technology
User Interface	Web form and results display	HTML5, CSS3, JavaScript
Application Logic	Backend server and routing	Python (Flask Framework)
Machine Learning	Startup success classifier	Random Forest (Scikit-learn)
Data Storage	Dataset and saved model	CSV File / .pkl (joblib)

Table-1 : Components & Technologies

S.No	Component	Description	Technology
1	User Interface	Web interface for user input and displaying prediction results (Acquired / Closed)	HTML, CSS, JavaScript
2	Application Logic-1	Backend server logic handling HTTP requests, routing and response rendering	Python (Flask Framework)

S.No	Component	Description	Technology
3	Application Logic-2	Data preprocessing including feature ordering, numeric validation and array reshaping	NumPy, Pandas, Scikit-learn
4	Machine Learning Model	Startup success prediction using ensemble classification algorithm	Random Forest (Scikit-learn)
5	Database / Data Storage	Startup dataset storage used for model training (startup1.csv)	CSV File / Local Storage
6	Model Storage	Serialized trained model file saved after training for reuse in Flask app	Joblib (.pkl file)
7	Infrastructure (Server)	Local application deployment and development environment	Local System / PyCharm / Jupyter Notebook

Table-2 : Application Characteristics

S.No	Characteristics	Description	Technology
1	Open-Source Frameworks	Open-source tools used throughout the development lifecycle — from EDA to model training to deployment	Flask, Scikit-learn, Pandas, NumPy, Joblib
2	Security Implementations	Input validation on all form fields (numeric type enforced), error handling in Flask routes, and safe model deserialization	Flask validation, Python error handling
3	Scalable Architecture	Modular project structure (separate HTML templates, app.py, model file) allowing easy migration to cloud deployment (Heroku / AWS)	Flask + Microservice ready structure
4	Availability	Web application accessible via any browser on localhost; portable to cloud platforms without code changes	Localhost / Deployable to Cloud
5	Performance	Prediction generated in under 2 seconds using pre-trained and serialized Random Forest model (no retraining required at runtime)	Random Forest optimized model (Scikit-learn)

Table-3 : Complete Technology Stack Summary

Category	Tool / Library	Version / Notes	Purpose
Language	Python	3.8+	Core programming language
Web Framework	Flask	2.x	Backend server and routing
ML Library	Scikit-learn	1.x	Random Forest model training and prediction
Data Manipulation	Pandas	1.5+	Dataset loading, EDA, preprocessing
Numerical Computing	NumPy	1.23+	Array operations, feature reshaping
Visualisation	Matplotlib	3.5+	Graphs, plots during EDA
Visualisation	Seaborn	0.12+	Heatmaps, box plots, distplots
Statistics	SciPy	1.9+	Statistical analysis during EDA
Model Serialization	Joblib	1.x	Save and load trained model (.pkl)
Frontend	HTML5 / CSS3	Standard	User interface and form design
Frontend	JavaScript	ES6	Client-side interactions
IDE / Environment	Jupyter Notebook	6.x / VS Code	Model training and experimentation
IDE / Environment	PyCharm	Community / Pro	Full project development
Dataset	startup1.csv	Kaggle	Training data for the ML model

Table-4 : Machine Learning Model Details

Parameter	Details
Algorithm	Random Forest Classifier (Ensemble of Decision Trees)

Parameter	Details
Library	Scikit-learn – RandomForestClassifier
Task Type	Binary Classification (Acquired = 1 / Closed = 0)
Input Features	age_first_funding_year, age_last_funding_year, age_first_milestone_year, age_last_milestone_year, relationships, funding_rounds, funding_total_usd, milestones, avg_participants (9 features)
Target Variable	status (1 = Acquired, 0 = Closed)
Training Split	70% Training / 30% Testing (train_test_split, random_state=0)
Evaluation Metrics	Accuracy, Precision, Recall, F1-Score, Confusion Matrix, ROC-AUC, Precision-Recall AUC
Training Accuracy	~100% (Random Forest fits training data completely)
Testing Accuracy	~82–86% (on held-out test set)
Model Saved As	random_forest_model.pkl (serialized using joblib)
Prediction Time	Under 2 seconds per request

Table-5 : Project File Structure

File / Folder	Type	Description
app.py	Python	Flask web server — routes, model loading, prediction logic
random_forest_model.pkl	Binary (.pkl)	Serialized trained Random Forest model (joblib)
startup1.csv	CSV Dataset	Startup dataset used for model training (from Kaggle)
Startup_Success_Prediction .ipynb	Jupyter	Notebook containing full EDA, preprocessing, and model training
templates/home.html	HTML	Landing page — project introduction and navigation
templates/predict.html	HTML	Input form — 9 startup metric fields for prediction

File / Folder	Type	Description
templates/submit.html	HTML	Result page — displays Acquired or Closed prediction with Jinja2
