**Heart Disease Predictor System Report:**

**1. Executive Summary**

This report presents a complete pipeline for building a heart attack risk prediction system. It covers:

* Data discovery and cleaning using pandas.
* Exploratory data analysis (EDA) with visualizations.
* Feature engineering, label encoding, and handling class imbalance with SMOTE.
* Training, evaluating, and comparing multiple classification models.
* Packaging the best-performing model with its encoders.
* Deploying the model as a Flask web application with a Bootstrap-based UI and Chart.js visualization.
* Persist every user submission (with patient name) into a local SQLite database (litsql.db).
* Offer a “Show Previous Results” history page with per-column search filters.

**2. Dataset Description**

* Source: heart\_disease\_uci.csv (14 feature columns plus the target).
* Key features:
  + Demographics: age, sex
  + Clinical measures: resting blood pressure, cholesterol, maximum heart rate achieved, ST depression (oldpeak)
  + Categorical indicators: chest pain type, fasting blood sugar (>120 mg/dl), resting ECG results, exercise-induced angina
  + Target: num column indicating presence and stage of heart disease (0–4)

**3. Data Preprocessing**

1. **Column Renaming**: Renamed cryptic abbreviations to descriptive names (e.g., trestbps → resting\_blood\_pressure).
2. **Missing Value Imputation**:
   * Numeric features: imputed with mean (or mode for fasting blood sugar’s categorical case).
   * Zero-value correction: replaced zeros in cholesterol and resting\_blood\_pressure with respective means.
3. **Dataset Reduction**: Selected a subset of relevant columns and dropped duplicates or redundant dataset field.
4. **Type Conversion**:
   * Converted fasting\_blood\_sugar to boolean.
   * Encoded all categorical features (sex, chest\_pain\_type, etc.) with scikit-learn’s LabelEncoder.
5. **Resampling**:
   * Addressed class imbalance using SMOTE to oversample minority classes in the target.

**4. Exploratory Data Analysis (EDA)**

* **Descriptive Statistics**: Computed means, medians, ranges, and distribution shapes via .describe() and histograms.
* **Boxplots**: Visualized numeric feature distributions across target classes to detect outliers.
* **Countplots & Pie Charts**: Explored categorical feature distributions and their relationships to the target.
* **Correlation Analysis**: Generated a Pearson correlation heatmap to identify highly correlated predictors.

**5. Model Training and Evaluation**

* **Model Candidates**:
  + Decision Tree (entropy)
  + Gaussian Naive Bayes
  + Logistic Regression
  + k-Nearest Neighbors
  + Support Vector Classifier (RBF kernel)
* **Cross-Validation**: 5-fold CV to compare average accuracy across models.
* **Final Choice**: k-Nearest Neighbors demonstrated strong performance (accuracy 64% in CV).
* **Train/Test Split**: 80/20 split to assess generalization on unseen data.
* **Performance Metrics**: Generated classification report with precision, recall, F1-score for each heart disease stage.

**6. Model Packaging**

* Saved the chosen classifier using joblib under model\_le/model.pkl.
* Persisted each LabelEncoder for categorical features to model\_le/le\_<feature>.pkl.

**7. Web Application Deployment**

* **Backend**: Flask app
  + Loads model & encoders at startup.
  + " /" route renders form.html (now with Patient Name field).
  + "/predict" route:
    - Reads & encodes inputs.
    - Predicts probabilities, maps to labels.
    - Inserts a record into SQLite table predictions (see schema below).
    - Renders results.html (with “Show Previous Results” button).
  + " /history" route:
  + Queries all rows, deserializes probabilities JSON, passes to history.html.
* **Frontend**:
  + form.html: Bootstrap-based responsive form for user input (demographics and clinical measures).
  + results.html: Displays prediction result with a Chart.js pie chart illustrating probabilities for each class and a list of percentages.
  + History.html: show the user inputs history for all patients with filters to filter data based on the columns data
  + Ensures an intuitive UX with navigation back to the form.

**8. Usage Workflow**

1. User visits the root URL and inputs their data in the form.
2. Upon submission, Flask encodes inputs and predicts risk stage.
3. Results page visualizes probabilities and highlights the most likely outcome.
4. All data will be saved into litsql.db dataset.
5. Users may repeat with different inputs by clicking on back to form or by clicking on show previous results user can show the history of all results.

**9. Conclusion**

The Heart Disease Predictor system demonstrates a full data science lifecycle—from raw data to a user-friendly web service. It combines robust preprocessing, exploratory analysis, model evaluation, and practical deployment, providing a template for similar healthcare analytics projects.