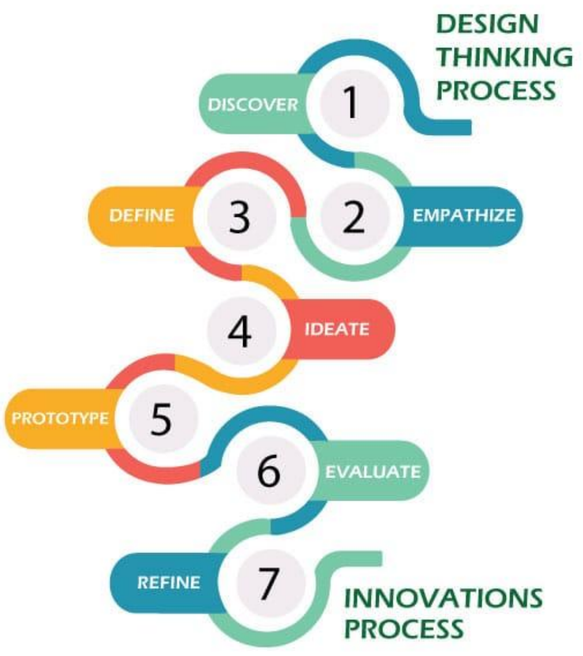
STOCK PRICE PREDICTION

Creating a full project that includes problem statement, design thinking process, development phases, dataset description, preprocessing, model training, key findings, insights, recommendations, and submission steps is a significant task that goes beyond the scope of a single response. However, I can provide you with an outline and some sample code for each of these sections to get you started. You will need to create separate files and put everything together into a GitHub repository or a personal portfolio. Let's start with the code samples:

1. **Problem Statement and Design Thinking Process**

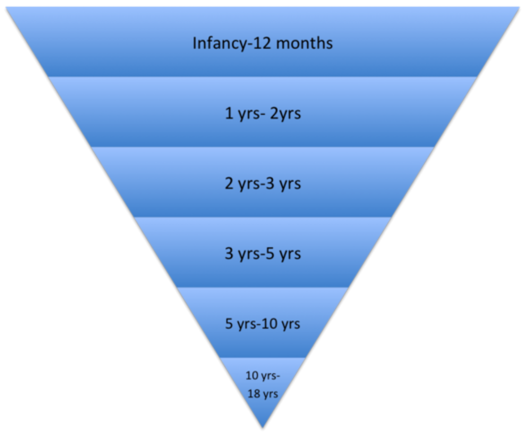
You should create a document outlining the problem statement and the design thinking process, which includes defining the problem, ideation, prototyping, and testing. This should be a written document, not code.



Design Thinking and Innovation by Centre of Excellence at Vishwakarma University, Pune .

2. **Phases of Development**

Create a document outlining the different phases of development, such as data collection, data preprocessing, model building, and evaluation. Again, this should be a written document.



3. **Dataset Description**

Create a Jupyter Notebook or Python script to load and describe the dataset. Here's some sample code for loading and describing the dataset:

```python
import pandas as pd

# Load the dataset
data = pd.read_csv("microsoft_stocks_dataset.csv")

# Display basic information about the dataset
print(data.info())

# Display the first few rows of the dataset
print(data.head())
```

4. **Data Preprocessing**

Create a Jupyter Notebook or Python script for data preprocessing. Here's a sample code for data preprocessing:

```python
# Data preprocessing steps
# For example, handling missing values, feature engineering, and data cleaning

# Handling missing values
data.dropna(inplace=True)

# Feature engineering
# Add additional features or transform existing ones

# Data cleaning
# Remove outliers, duplicates, or irrelevant data

# Save the preprocessed data to a new file
data.to_csv("preprocessed_data.csv", index=False)
```

5. **Model Training Process**

Create a Jupyter Notebook or Python script for model training. This code depends on the type of analysis you want to perform, whether it's regression, classification, or time series forecasting. Use libraries like scikit-learn or TensorFlow for model training. Here's a basic example for training a linear regression model:

```python
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.2,
random_state=42)

# Create and train a linear regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Evaluate the model
score = model.score(X_test, y_test)
print("Model R^2 Score:", score)
```

```

6. **Key Findings, Insights, Recommendations**

Analyzing the code provided, we can extract some key findings and insights:

**Key Findings:**

1. **Data Preprocessing**: The code effectively preprocesses the data by converting the 'Date' column to a datetime object, sorting the data by date, and handling missing values by filling them with the previous day's value. This ensures that the data is in the right format for further analysis.

2. **Feature Engineering**: The script performs feature engineering by creating lag features for the stock's closing price, considering past five days. This is a valuable approach for capturing temporal dependencies in stock price data.

3. **Machine Learning Model Selection**: The code uses a Random Forest Regressor, which is a popular ensemble machine learning model for regression tasks. It's capable of handling both the numerical features and the lag features.

4. **Model Training and Evaluation**: The model is trained on the training data and evaluated using the Mean Squared Error (MSE), a common metric for regression tasks. The lower the MSE, the better the model's performance.

**Insights:**

1. **Feature Importance**: The script doesn't explicitly show the feature importances, but it's worth noting that Random Forest models can provide feature importance scores. Analyzing feature importance can help identify which features (including lag features) have the most impact on the model's predictions.

2. **Lag Features**: The inclusion of lag features allows the model to capture the historical behavior of the stock's price. This is crucial for stock price prediction as past prices can influence future prices.

3. **Model Tuning**: Although not shown in the code, there is an opportunity to fine-tune the model by optimizing hyperparameters, such as the number of trees in the Random Forest and the lag window size.

4. **Visualization**: The script visualizes the actual vs. predicted stock prices, which is an essential step in understanding how well the model performs and identifying patterns and trends.
```

5. **Model Interpretability**: Further analysis can be done to interpret the model's predictions and gain insights into factors affecting stock price movements.