High-Level Design (HLD)

System Architecture:

The income prediction system follows a modular and scalable architecture:

1. Web Application Tier:

Flask App: User interactions occur through a Flask web application. It receives user inputs, sends them to the prediction module, and displays the results.

2. Prediction Module:

Machine Learning Model: Utilizes the trained machine learning model to predict income based on input features.

MLflow Integration: The MLflow component helps manage experiments, metrics, and model selection.

3. Model Training Pipeline:

DVC Pipeline: Uses Data Version Control (DVC) for tracking and managing data, ensuring reproducibility.

Machine Learning Models: Includes a variety of models tested during the model training stage.

MLflow Logging: Captures metrics, parameters, and artifacts during training for later comparison.

4. Database Integration:

MongoDB: Stores and retrieves data, providing seamless integration with the data retrieval component.

5. Docker Image:

Dockerized Flask App: Encapsulates the web application, its dependencies, and MLflow artifacts for portability.

CI/CD Integration: Used in Continuous Integration/Continuous Deployment (CI/CD) workflows for AWS deployment.

Flow of Operations:

1. *User Interaction:*

Users access the web application and input data.

Flask routes user data to the prediction module.

2. *Prediction Process:*

The prediction module utilizes the trained machine learning model to predict income.

MLflow logs the prediction results, metrics, and parameters.

3. *Model Training Pipeline:*

The DVC pipeline orchestrates data retrieval, transformation, and model training.

DVC ensures traceability, reproducibility, and versioning.

4. *Data Retrieval and Transformation:*

Data is retrieved from MongoDB using the retrieve_data.py script.

Data preprocessing, feature engineering, and encoding are performed with data_transformation.py.

5. *Model Training:*

Various models and hyperparameters are tested using model_training.py.

MLflow logs metrics and model artifacts.

6. *CI/CD Deployment:*

The Docker image is built, tagged, and pushed to Amazon Elastic Container Registry (ECR).

AWS EC2 deploys the application using the latest Docker image.