# Introduction

### What is a High-Level Design Document?

The goal of this HLD or a high-level design document is to add the necessary detail to the current project description to represent a suitable model for coding. This document is also intended to help detect contradictions before coding and can be used as a reference manual for how the modules interact at a high level.

The HLD will:

⦁ Present all of the design aspects and define them in detail

⦁ Describe all user interfaces being implemented

⦁ Describe the hardware and software interfaces

⦁ Describe the performance requirements

⦁ Include design features and architecture of the project

⦁ List and describe the non-functional attributes such as security, reliability, maintainability, portability, reusability, and application compatibility. resource utilization, serviceability

### Scope

The HLD documentation presents the structure of the system, such as database architecture, application architecture (layers), application flow (Navigation), and technology architecture. The HLD uses non-technical to mildly technical terms which should be understandable to the administrators of the system.

## Tools used

Python programming language and frameworks such as NumPy, Pandas, Scikit-learn, Flask, and a few other libraries were used to build the whole model.

### 

⦁ For visualization tasks, matplotlib, seaborn and plotly were used

⦁ Flask was used for building the web application and server to run the code

⦁ NumPy and Pandas were used to clean and interpret data

⦁ Scikit-learn was used to cross-validate and compare different models

⦁ GitHub is used as a version control system

## Hardware Requirements

⦁ Windows Server, Linux, or any operating system that can run as a web server, capable of delivering HTML5 content.

⦁ Minimum 1.10 GHz processor or equivalent.

⦁ Between 1-2 GB of free storage

⦁ Minimum 512 MB of RAM

⦁ 3 GB of hard disk space

## Constraints

The front end must be user-friendly and should not need anyone to have any prior knowledge in order to use it.

## Project Objective

The Credit Card Default Prediction project aims to predict whether a credit card holder is likely to default on their payment based on various features. This predictive modelling task helps financial institutions assess and manage credit risk effectively.

### Scope

* Analyze credit card data to identify patterns associated with default risk.
* Develop and train a machine learning model (XGBoost) to predict defaults.
* Implement a user-friendly interface for predictions.

### Key Features

* Data preprocessing and feature engineering.
* Model training using the XGBoost algorithm.
* Web-based user interface for input and prediction.

## System Components

1. **Data Ingestion**: Collect and preprocess the data.
2. **Model Training**: Train the XGBoost classification model.
3. **Prediction Service**: Create an endpoint for predictions.
4. **User Interface**: Develop a web-based interface for user inputs and displaying predictions.

## Deliverables

* Trained XGBoost model with 80% accuracy.
* Flask web application for predictions.
* Documentation including HLD, LLD, Architecture, and Wireframe.

## Reusability

The code written and the components used should have the ability to be reused with no problems.

## Error Handling

Errors should be encountered, an explanation will be displayed as to what went wrong. An error will be defined as anything that falls outside the normal intended usage.