LOW LEVEL DOCUMENT

Project Title	**FlipKart Scrapper**
Created by	**iNeuron Private Limited**
Author	Boktiar Ahmed Bappy
Document Version	**0.1**
Date:	**12/4/2022**

Document Control:

VERSION	DATE	AUTHOR	COMMENTS
0.1	12/4/2022	Boktiar Ahmed Bappy	Introduction & Architecture defined

Reviews:

VERSION	DATE	REVIEWER	COMMENTS
0.1	12/4/2022	Boktiar Ahmed Bappy	

Approval Status:

VERSION	DATE	APPROVER	COMMENTS
0.1	12/4/2022	Boktiar Ahmed Bappy	

Contents

1. Introduction

1.1 Why this Low Level Design Document?

1.2 Scope?

2. Architectural Description

2.1 Design

2.2 Detailed Description

2.2.1 Data Access Layer

2.2.2 Entity Layer

2.2.3 Business Logic Layer

2.2.4 Exception Layer

2.2.5 Logging Layer

2.2.6 Testing

2.2.7 Utils

2.2.8 Initial Setup

2.2.9 Presentation

2.2.10 Docker File

2.2.11 Requirements

3. Test Case

3.1 Test Case Table

4. Instruction to Run

4.1 Detailed commands to run project

1. Introduction

1.1 Why this Low-Level Design Document ?

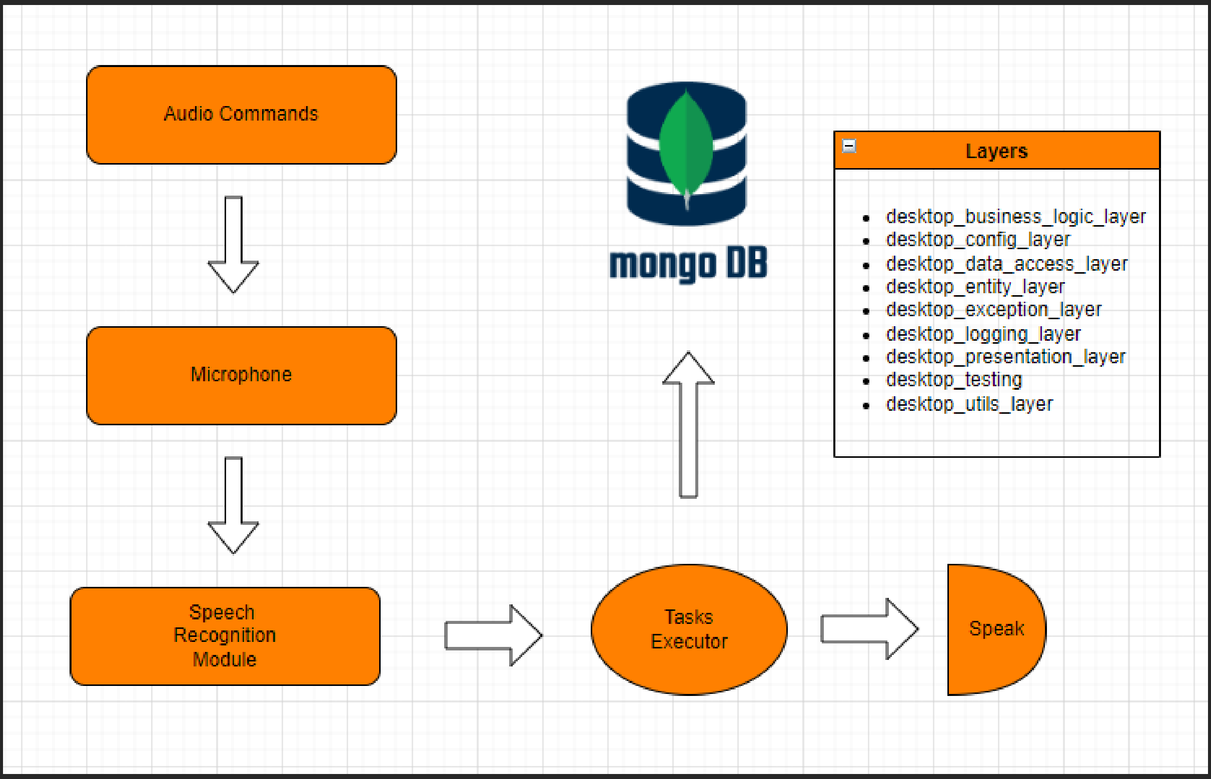
The goal of LLD or a low-level design document (LLD) is to give the internal logical design of the actual program Systems. LLD describes the class diagrams with the methods and relations between classes and program specs. It describes the modules so that the programmer can directly understand the program from the document.

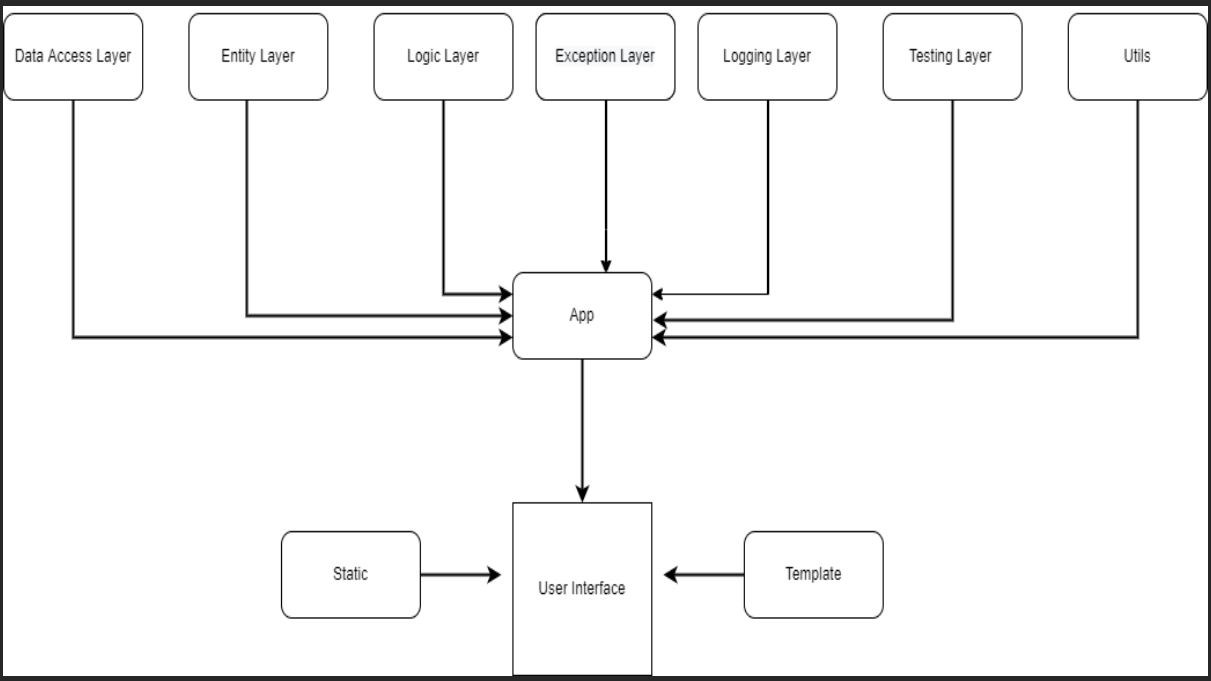
1.2 Scope ?

Low-level design (LLD) is a component-level design process that follows a step-by-step refinement process. This process can be used for designing structures, required software architecture, source code and ultimately, complex systems. Overall, the organisation may use it during requirement analysis.

2. Architectural Description

2.1 Design Diagram





2.1 Detailed Description

While designing this project we followed all industry standards coding is done in the object oriented fashion. The whole architecture is divided into layers as shown in the above diagrams.

2.2.1 Data Access Layer

This layer is responsible for establishing connectivity between client and server. where the client is our platform and the server is the Mongodb database. This layer contains **Create collection, Get collection, Drop Collection Methods**.

2.2.2 Entity Layer

This layer is responsible for holding attributes of the product. All entity classes required are defined here.

2.2.3 Logic Layer

This layer is responsible for the logic which will execute the query.

2.2.4 Exception Layer

This layer is responsible for handling all the exceptions that occur in the system. We have written an exception class to catch file name, line number and error message of the entire system to ease the error lookup if needed.

2.2.5 Logging Layer

This layer is responsible for logging information about the system, We also log errors caught by the Exception layer. This is where error and information look will be done in case if needed.

2.2.6 Testing layer

This layer is responsible for testing the entire application interface before deployment of the application. Developers have to run this class to test if everything is working properly before deployment, also if any new functionality is being added in the future test case has to be written for it in the testing layer.

2.2.7 Utils

This layer is responsible for utility functions. In this layer we have written read configuration.

2.2.8 Presentation Layer

This layer is responsible for releasing application programming interfaces through which we are interfacing FrontEnd.

2.2.9 Docker File

This file is responsible for creating docker images in the system. Prerequisite is that docker should be installed in the system.

2.2.10 Requirements

This file contains all the dependencies required to run this application.

3. Test Case

3.1 Test Case Table

TEST CASE	PREREQUISITE	RESULT
Verify whether the Application is accessible to the user	Application Should be running	App is accessible or not. with response code
Verify if start button is working	Application Should be running	Response starting engine
Verify stop button is working	Application Should be terminated	Response ok bye