Low Level Design

Project name :Calculator App

Written By	Sachin
Document Version	0.3
Last Revised Date	15 july 2024

Document Control

Change Record:

Version	Date	Author	Comments
0.1	15 july 2024	Sachin	Introduction & Architecture defined
0.2	16 july 2024	Sachin	Architecture & Architecture Description appended and updated
0.3	17 july 2024	Sachin	Unit Test Cases defined and appended

Reviews:

Version	Date	Reviewer	Comments
0.2	Date	Date	Document Content , Version Control and Unit Test Cases to be added

Approval Status:

Version	Review Date	Reviewed By	Approved By	Comments

111

Contents

LOW LEVEL DESIGN (LLD)

Neuron

1. Introduction

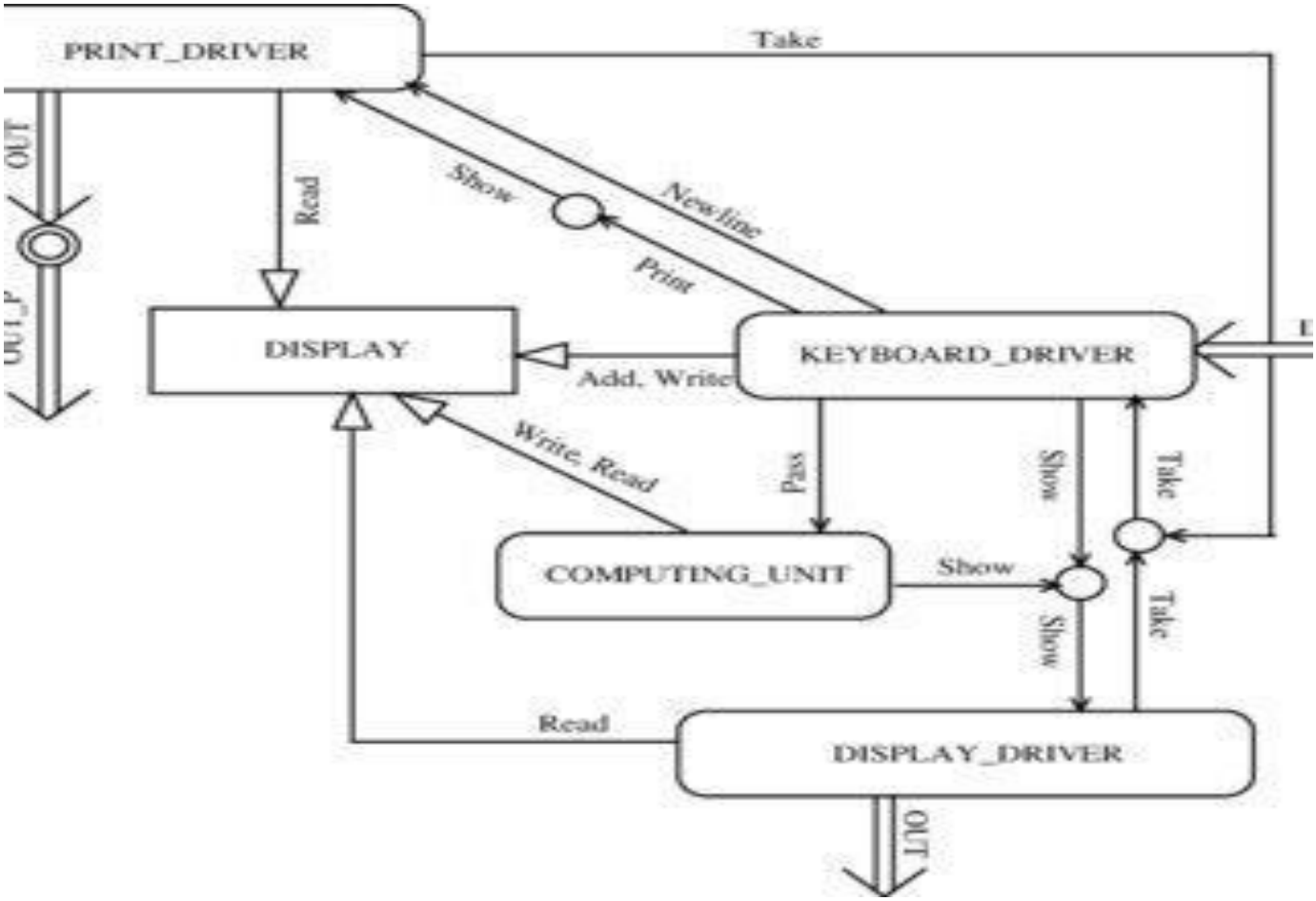
1.1. What is Low-Level design document?

The goal of LLD or a low-level design document (LLDD) is to give the internal logical design of the actual program code for Food Recommendation System. 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 code the program from the document.

1.2. Scope

Low-level design (LLD) is a component-level design process that follows a step-bystep refinement process. This process can be used for designing data structures, required software architecture, source code and ultimately, performance algorithms. Overall, the data organization may be defined during requirement analysis and then refined during data design work

2. Architecture



3. Architecture Description

3.1. Data Description

When describing the data of a calculator, we're generally referring to the different types of data it processes, stores, and the functionalities it supports. Here's a comprehensive breakdown:

1. Input Data

- **Numbers:** Calculators handle numerical inputs, including integers, floating-point numbers (decimals), and sometimes fractions.
- **Operators:** Operators include basic arithmetic symbols (+, -, *, /), and sometimes more advanced ones (%, √, ^ for exponentiation).
- **Functions:** Some calculators support functions like sin, cos, tan, log, exp, and various mathematical or statistical functions.

- **Parentheses:** Used for grouping operations to define the order of operations.

2. Output Data

- **Results:** The primary output is the result of calculations, which can be integers, floating-point numbers, or complex numbers in advanced calculators.
- **Error Messages:** Calculators may output errors (e.g., division by zero, syntax errors).

3. Internal Data Storage

- **Memory Storage:** Many calculators have memory functions (e.g., M+, M-, MR, MC) to store and recall values temporarily.
- **History:** Some advanced calculators keep a history of calculations for reference.

3.2. Web Scrapping

Web scraping a calculator typically involves extracting data from a webpage where a calculator is embedded or described. The data you might want to scrape can include information about the calculator's features, specifications, or even its functional outputs if the calculator performs live calculations.

3.3. Data Transformation

Data transformation involves converting data from one format or structure into another, making it more suitable for analysis or use in different contexts. When dealing with calculators, data transformation can cover a range of activities depending on the type of data and the intended use. Here's how you might approach data transformation for a calculator:

3.4. Data Insertion into Database

inserting data into a database for a calculator involves storing user inputs, results, and possibly other relevant information in a structured way. This can be useful for tracking calculations, analyzing usage patterns, or creating historical records.

3.5. Export Data from Database

Exporting data from a database involves extracting the data and saving it in a format that is suitable for analysis, sharing, or reporting. For a calculator application, this data might include historical calculations, user inputs, and results

3.6. Data Pre-processing

Data pre-processing is a crucial step in preparing data for analysis or modeling. For a calculator application, pre-processing may involve cleaning, transforming, and organizing data to ensure it's accurate and usable.

3.7. Data Clustering

Data clustering is a technique used to group similar data points into clusters so that data within each cluster is more similar to each other than to those in other clusters. For a calculator application, clustering can be used to analyze user behavior, categorize types of calculations, or identify patterns in usage.

3.10. Model Building

Model building for a calculator application can involve various types of models, depending on what you want to achieve. Here are some common scenarios and approaches:

1. Predictive Modeling

If you want to predict certain aspects of user behavior or calculation outcomes, you might build predictive models. Here are some examples:

- **Predicting Calculation Results:** Based on input patterns, predict the likely result or flag potential errors.
- **User Behavior Prediction:** Predict user actions or preferences based on past behavior. **Example Scenario: Predicting Calculation Errors**

3.11. Data from User

In a calculator application, capturing and processing user data is essential for understanding user behavior, improving functionality, and personalizing the user experience. The data collected from users can include inputs, results, and additional context that can be analyzed to provide insights or enhance the application

3.12. Data Validation

Data validation is an essential part of managing any application, including a calculator. It ensures that the data entered by users is accurate, meaningful, and

usable. For a calculator application, data validation typically involves checking the inputs, outputs, and any intermediate data for correctness and reliability.

3.13. User Data Inserting into Database

Inserting user data into a database for a calculator application involves several steps, including setting up the database schema, connecting to the database, and executing SQL commands to insert data.

3.14 Data Clustering

Data clustering is a technique used in data analysis to group a set of objects or data points so that objects in the same group (or cluster) are more similar to each other than to those in other groups. This technique is useful for identifying patterns, structures, or categories within data.

When it comes to clustering data for a calculator, there are a few scenarios you might be considering:

1. **Clustering for Data Analysis:** If you have a dataset of numerical values (e.g., financial data, measurements, user behavior statistics), you might want to cluster these values to find patterns or group similar data points.
2. **Clustering for Calculation Tasks:** If you're building a calculator or a tool that performs complex calculations, you might use clustering to optimize performance or to group similar calculation tasks for better efficiency.
3. **Clustering in Calculator Applications:** If you have a calculator application and you want to group similar calculation functions or user inputs for better usability, clustering could be used to organize these functions.

Here’s a high-level overview of how you might approach clustering in these contexts:

1. Clustering for Data Analysis

- **Choose a Clustering Algorithm:** Common algorithms include K-means, hierarchical clustering, and DBSCAN.

- **Prepare the Data:** Ensure your data is cleaned and normalized. Each data point should be represented in a feature space (e.g., using numerical values).

- **Apply the Algorithm:** Run the clustering algorithm on your data. For K-means, you'll need to specify the number of clusters.

- **Analyze the Results:** Interpret the clusters to understand the patterns or groupings in your data.

Example: K-means Clustering

1. **Data Preparation:** Suppose you have a set of numbers representing user spending habits.

2. **Apply K-means:** Decide on the number of clusters (e.g., 3 clusters for low, medium, and high spenders).

3. **Cluster Assignment:** The algorithm will assign each user to one of the three clusters based on their spending behavior.

3.15. Model Call for Specific Cluster

If you want to call or apply a specific model based on the cluster a data point belongs to, you can follow a structured approach. This is useful in scenarios where you have different models or algorithms for different groups of data. Here's a step-by-step guide on how to implement this:

Scenario Overview

1. **Cluster Data:** Use clustering techniques to group your data into different clusters.
2. **Assign Models:** Define or train specific models for each cluster.
3. **Model Call Based on Cluster:** Use the cluster assignment of a data point to determine which model to apply for that point.

3.16. Recipe Recommendation & Saving Output in Database

o create a recipe recommendation system with a calculator that also saves the output in a database, you can follow these steps:

1. Define the System Requirements

- **Recipe Recommendation:** The system should suggest recipes based on input criteria such as available ingredients or dietary preferences.

- **Database Storage:** The system should store recommended recipes and related data in a database for future reference.

2. System Design

1. **Input Data Handling:** Collect user input such as available ingredients, dietary restrictions, or preferred cuisines.
2. **Recipe Recommendation Logic:** Implement the logic to recommend recipes based on the input criteria.
3. **Database Integration:** Save the recommended recipes and user preferences to a database.

3. Implementation Steps

A. Recipe Recommendation Logic

For simplicity, let's assume we have a dataset of recipes and their ingredients. We'll use a basic approach for recommending recipes based on ingredient matches.

3.18. Deployment

Deploying a recipe recommendation system integrated with a calculator involves several steps to ensure that the system is operational and accessible to users. Here's a guide on how to deploy such a system, including considerations for web or mobile deployment:

1. System Architecture

Components:

1. **Backend:** Handles recommendation logic and database operations.
2. **Frontend:** User interface for interacting with the calculator and viewing recommendations.
3. **Database:** Stores recipes, user data, and recommendations.

4. Unit Test Cases

Unit testing is crucial to ensure that each component of your application (in this case, a calculator or recommendation system) works as expected. For a recipe recommendation system integrated with a calculator, you'll want to create unit tests for various functionalities, such as the recommendation logic, database interactions, and any other components.