1. Google Forms Matrix System

* System architecture description:

Google Forms is an existing form system, therefore its architecture primarily relies on Google's existing services. This system does not require custom development, but can still be analyzed through front-end form design and back-end data management.

* Architecture hierarchy:

1. User interaction layer (front-end):

Google Forms frontend: Users access Google Forms through a browser, select items, and rate them. The front-end of the form is a static HTML page, with users directly interacting with it.

1. Business logic layer:

Google Forms built-in logic: Google Forms executes simple business logic internally through form controls and rules such as forced selection, selection quantity limits, etc. This layer is mainly responsible for verifying the legitimacy of user input and managing the process of submitting data.

1. Data storage layer:

Google Sheets database: All selection and rating data submitted by users are directly stored in Google Sheets. Google Sheets, as a tabular database, is responsible for recording user selection behavior and scores.

1. Backend management team:

Google Sheets Export: Administrators can view and analyze user selection data through Google Sheets, and export the tables as Excel files or for subsequent analysis. Without complex backend management functions, data analysis relies on the native functionality of Google Sheets.

1. Pulley free distribution system

* System architecture description:

The pulley free allocation system is a customized web system where users select items on the interface and freely adjust scores through the slider. The system requires front-end and back-end interaction, as well as a database to store user selections and scores.

* Architecture hierarchy:

1. User interaction layer (front-end):

Front end technology stack: dynamic web pages built on HTML, CSS, and JavaScript. The slider control is implemented through JavaScript or front-end frameworks such as Vue.js or React.

Function implementation: Users can freely assign scores from 1 to 7 to the selected items through the slider, and the score display is dynamically updated in the front-end.

1. Business logic layer (backend):

Application logic processing: The backend is supported by web frameworks such as Node.js, Django, Flask, etc. It processes user score submission requests, verifies whether the score is valid (such as whether it meets the range), and records the score in the database.

API service: The front-end and back-end communicate through the API. When users adjust their scores, the front-end sends the data to the back-end through AJAX or API, and the back-end receives and processes these requests.

1. Data storage layer:

Database: User choices and scores are stored in a database, commonly used databases may be MySQL, PostgreSQL, or NoSQL databases (such as MongoDB). Each user's selection and score are saved as a record.

1. Backend management team:

Data management and export: Administrators can access data in the database through a management interface, or export user selections and scores as Excel spreadsheets through customized export functions. The backend system can provide data query and simple analysis functions.

1. Fixed Total Distribution System

* System architecture description:

The fixed total allocation system is similar to the pulley free allocation system and is also a customized web system, but it has a limit function for the total score in the front-end. Users must allocate scores within the specified total score, and the back-end needs to handle these limit logics.

* Architecture hierarchy:

1. User interaction layer (front-end):

Front end technology stack: Similar to the pulley free allocation system, the front-end is based on HTML, CSS, and JavaScript. The pulley control allows users to adjust scores, but each time they adjust, the system automatically updates the remaining assignable total scores, providing dynamic score feedback.

Restriction mechanism: JavaScript scripts will detect the scores assigned by users in real time, ensuring that the total score does not exceed the preset total score (such as 10 points). If exceeded, the pulley cannot be further adjusted.

1. Business logic layer (backend):

Complex application logic: The backend needs to verify whether the user score allocation meets the total score limit and ensure that each item score adjustment is reasonable. The backend logic needs to handle user requests and interact with the database.

Dynamic feedback: When users submit scores, the backend also needs to verify whether all conditions (such as total score constraints) are met. If the conditions are not met, the system should prompt the user to make modifications.

1. Data storage layer:

Database: Similar to the pulley free allocation system, user selection and score records are stored in the database, but the data structure also includes the total score verification logic for each user selection.

1. Backend management team:

Backend management and export function: Administrators can access the backend data management system, export user selected data, and analyze score allocation. The backend system may also have visualization tools for total score allocation to help analyze users' decision-making patterns within limited scores.