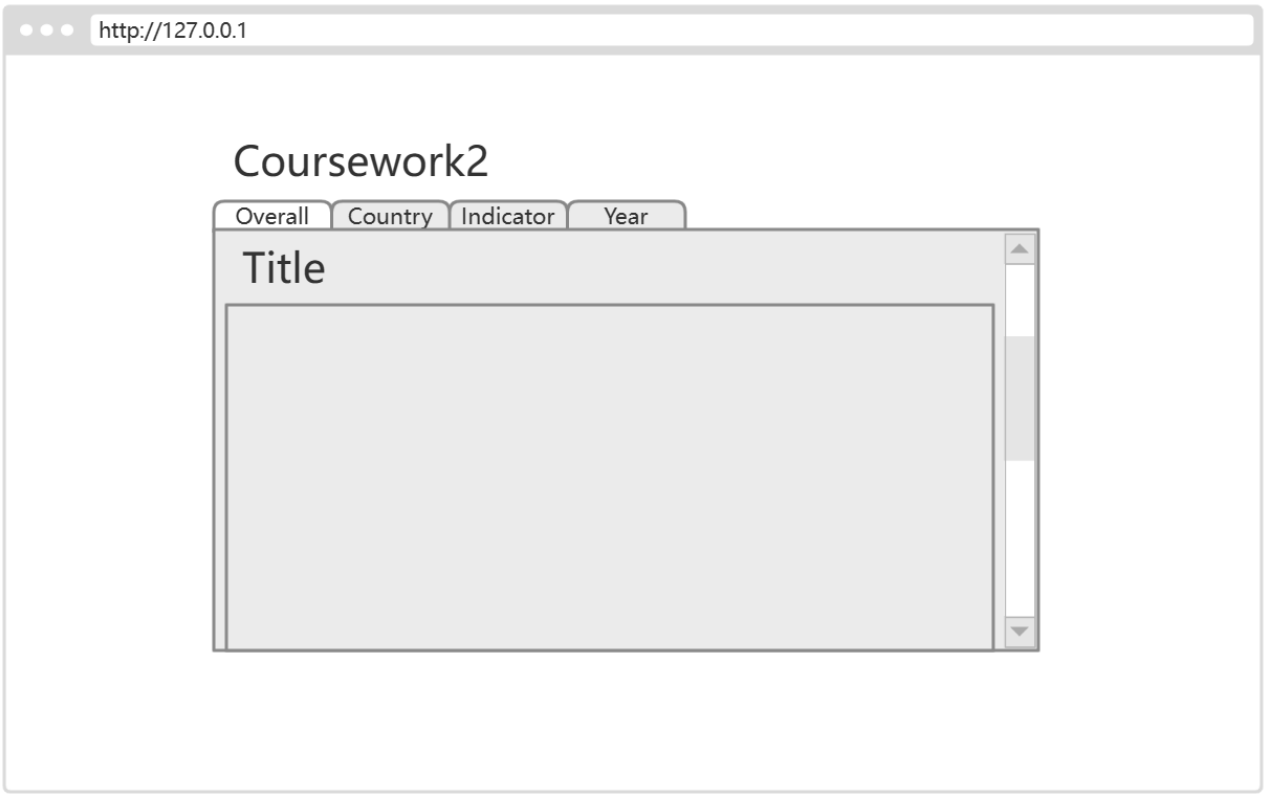
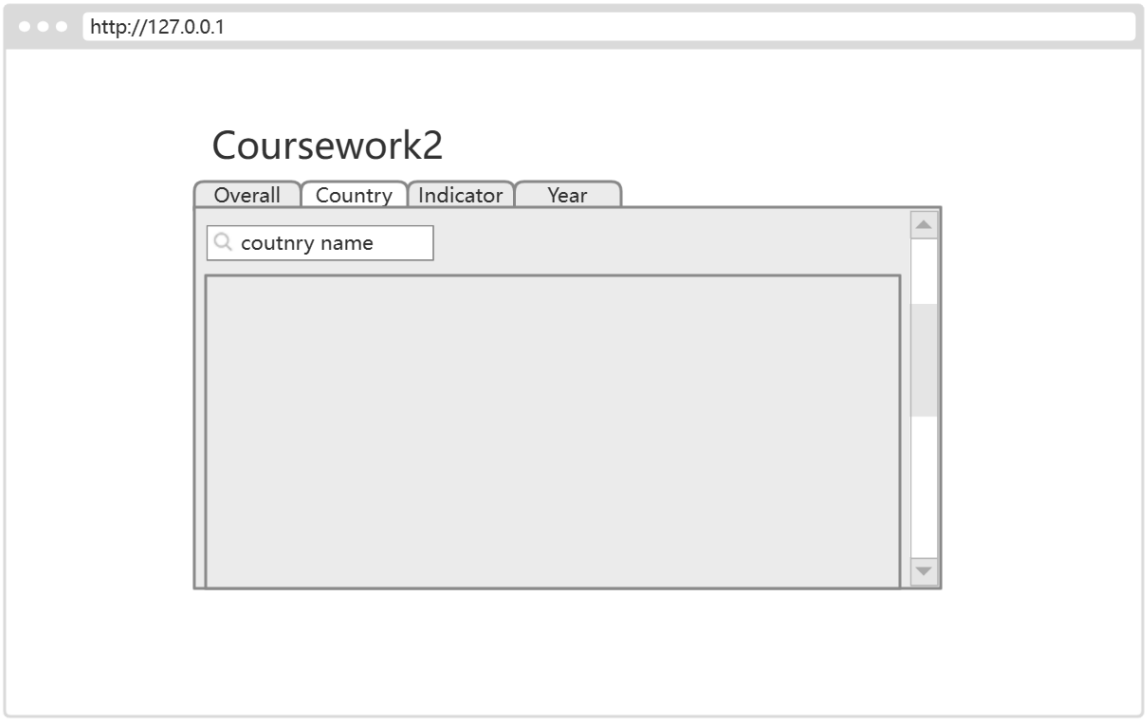
Design

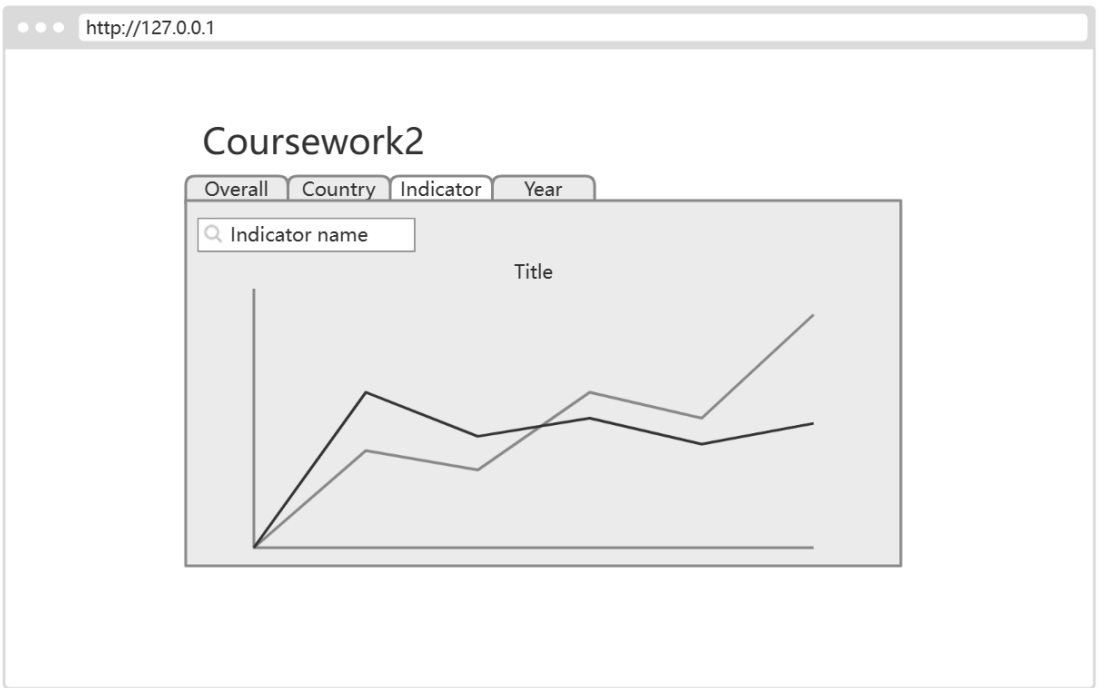
Interface design

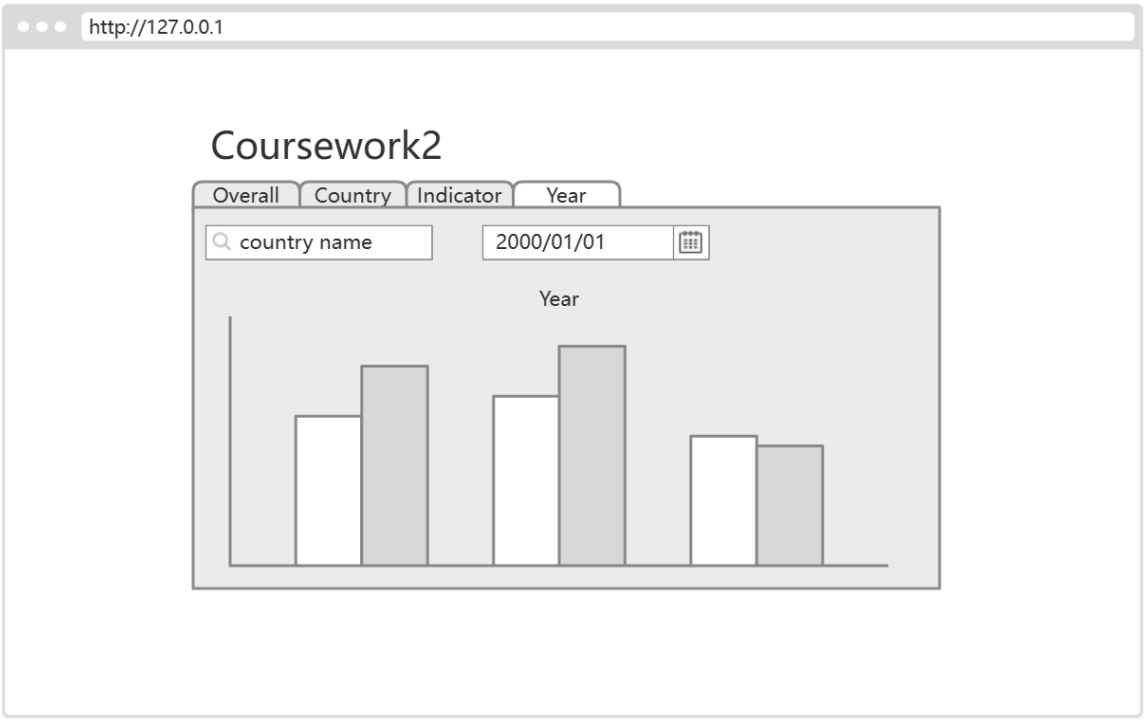
Since I am using streamlit technology, I use a single interface design pattern, through the tab tab to switch a simple sub-interface, there are a total of 4 such tab tags, corresponding to 4 requirements.

My interface looks like this:




http://127.0.0.1
Coursework2
Overall
Country
Indicator
Year
coutnry name

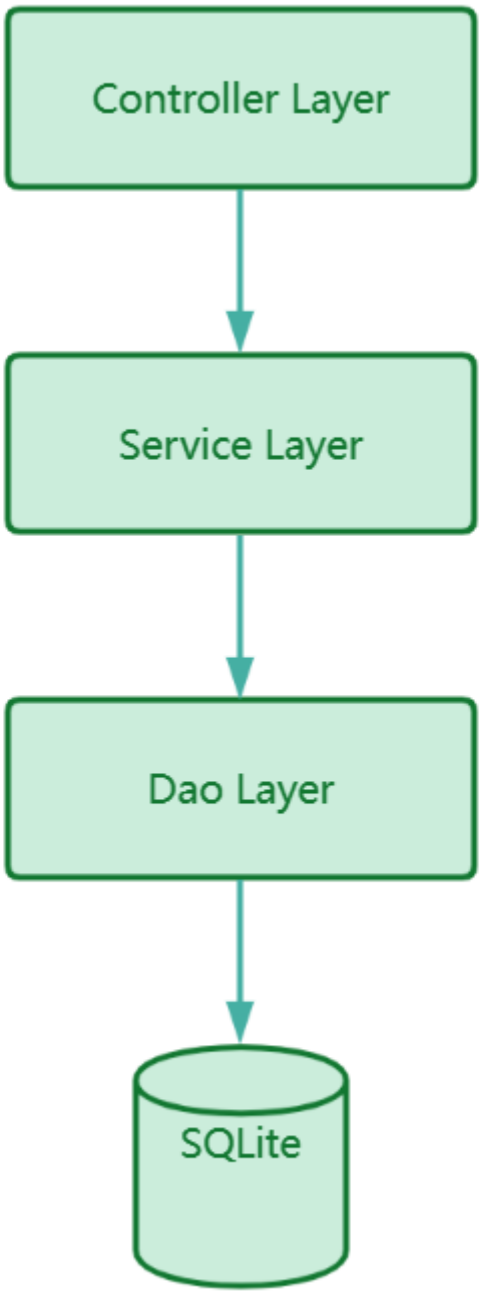

http://127.0.0.1
Coursework2
Overall
Country
Indicator
Year
Indicator name
Title



Application design

1. MVC

In the back-end application design, I adopted the MVC architecture, which also saved me a lot of time modifying the code when coding, and had more time to expand the code functionality, in addition, the code has good readability and the speed of finding the source of the error in the event of a crash.



1. The control layer is mainly responsible for the receipt of requests, packaging and processing of return messages, etc.
2. The service layer is mainly responsible for providing a variety of services, and is only the logical presenter. This job simplifies the extensibility of the service layer and only establishes a class for service.
3. The DAO layer is mainly responsible for the acquisition of data, is the main data source, and is responsible for dealing with the database.

2. REST API

An API, or application programming interface, is a set of rules that define how applications or devices can connect to and communicate with each other. A REST API is an API that conforms to the design principles of

the REST, or representational state transfer architectural style. Since we use flask for back-end coding, it makes it easy to implement a RESTful-style API with just a few lines of code:

```
self.api.add_resource(DataController, '/data')
self.api.add_resource(CountryController, '/country/<country>')
self.api.add_resource(IndicatorController, '/indicator/<name>')
self.api.add_resource(ControllerYearController,'/bar/<country>/<year>')
```

In this way, we can map the request path to our control layer, which is very convenient.

Since we only provide query function, our request method adopts GET request, the specific information is as follows:

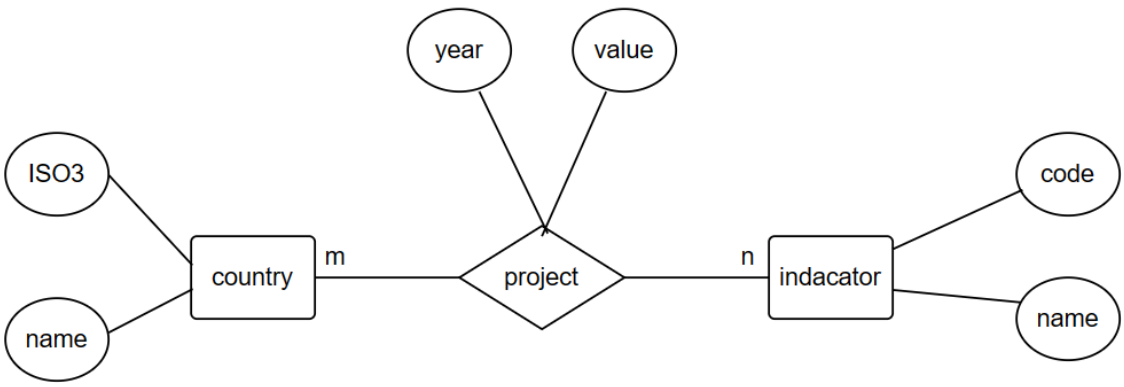
Method	Url	Parameter	Responce
GET	/data	None	Json Object
GET	/country/{country}	None	Json Object
GET	/indicator/{name}	None	Json Object
GET	/bar/{country}/{year}	None	Json Object

Database design

Our dataset is provided by the teacher, i.e. education_chn.csv file. After preliminary analysis, I came up with the following information:

1. Each $Country ISO3$ corresponds to a unique $Country Name$.
2. Each $Indicator Code$ corresponds to a unique $Indicator Name$ and $Country Name$.
3. $Country ISO3$, $Indicator Code$, $Year$ three fields determine $Year$.

According to the above analysis, our database design should be divided into three tables: country, indicator, and project. The ER diagram is as follows:



The constraints are as follows:

1. $ISO 3$ is the primary key of country.
2. $code$ is the primary key of indacator.
3. $ISO 3$, $code$, $Year$ are the primary keys of the project.
4. The $ISO3$ of project is the foreign key of the $ISO3$ of the country.
5. Project's $code$ is a foreign key to Indacator's $code$.