Name: Jay Singhvi

Jen –Chieh Lu

Karan Doshi

Vidhi Rathod

Assignment name and number: TW1 and Team 7

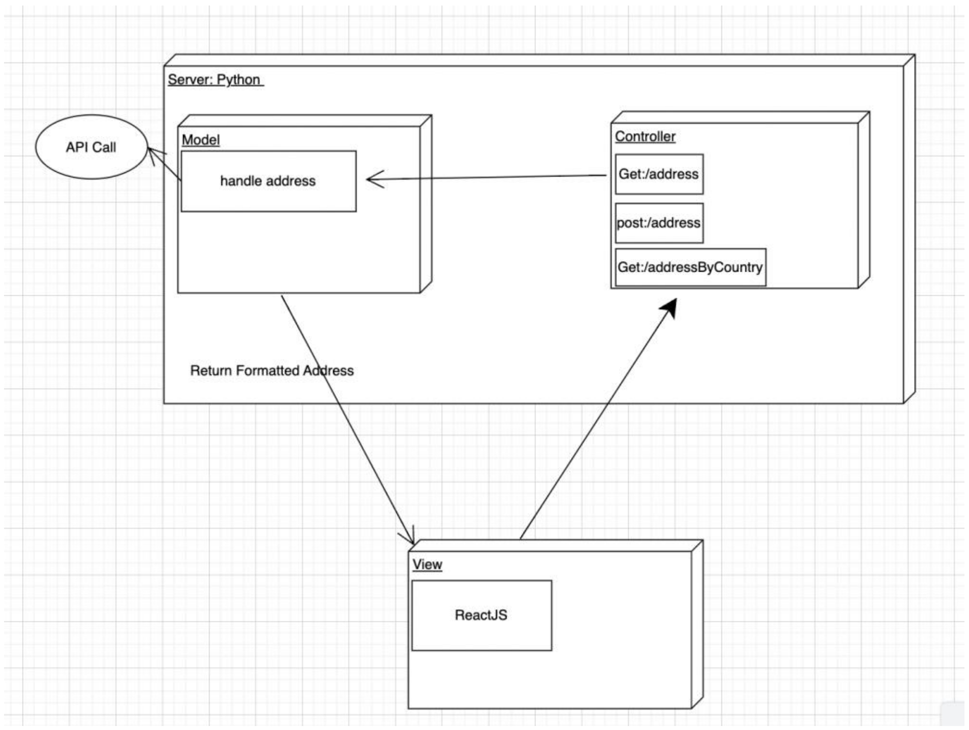
Date submitted: Feb 15, 2023

CPSC 5200-02 23WQ

- Your system context (what talks to your system, what are the inputs and outputs?)
    - simple System Context DiagramLinks to an external site.? (your system is represented by one big component-bubble on a system context diagram, showing inputs and outputs).
    - Maybe you want one more diagram showing inside your system, your components and how they interact?
- What architectural style are you choosing, and why?

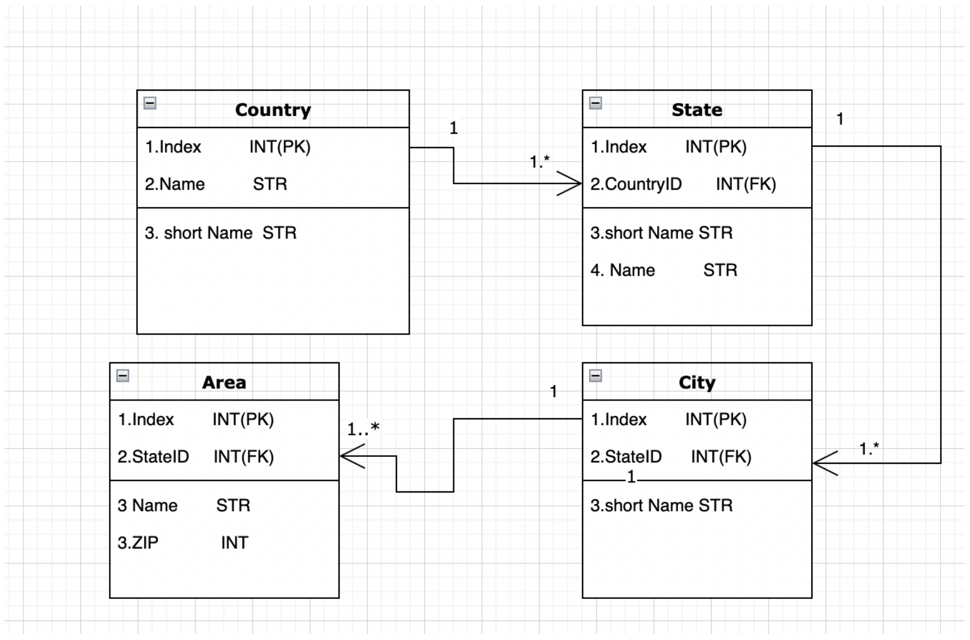
The architectural style we consider is MVC pattern, which stands for Model-View-Controller, is a software architecture pattern that separates an application data (model), user interface (view), and control logic (controller) into separate components. This separation of concerns has several advantages that make it an excellent choice for building websites:

1. Maintainability: By separating the different components, our team can work on each part independently without affecting the others. This makes it easier to update or modify individual components without affecting the entire application.
2. Scalability: As websites grow in complexity, it becomes more important to have a modular architecture that can scale with the application. With MVC, the separation of concerns makes it easier to add or remove features without having to rewrite the entire application.
3. Testability: With MVC, each component can be tested independently, which makes it easier to identify and fix issues. This also makes it easier to write automated tests, which can help catch issues early on.



MVC Diagram

- Where are you storing data? (In your diagram, not on which DB, or server...)
    - High level info on what data you are storing, how it is broken out?
        - Address is going to be broken into 4 different parts, country, state, city and area.
        - Each Table involved primary Key and foreign key to connect with other tables

- 

- What data moves between your components, and the external entities that talk to your system? (the gist, not the tiny details)
- What assumptions are you making? And why?
- What top-level components are you planning to build?
  - Can I understand what capabilities you are giving to users of your API?

The top-level components we will be dealing with are the user interface, the API and the database.

User Interface: A web-based form that captures the country specific address format entered by an end-user. The form will adjust dynamically to capture the address, and it will validate the address format and related data in real-time. We will be using React JS as our front-end framework.

API: Our RESTful API interacts with the database, retrieves, and stores address data, and provides access to this data through HTTP requests. The API will include methods for searching for addresses, retrieving address data, and adding new address data to the database.

We will be using verbs such as POST, GET, PUT.

Verb : GET
uri : /addresses/

Responses :
Code 200 : success

Code 400: Invalid format

Verb : POST
uri : /addresses/

Response :
{

“country” : “USA”,

“Address1” : “901, 12th Ave”,

“Address2” : “USA”,

“City” : “Seattle”,

“State” : “WA”,

“Postal Code” : “98122”,

}

Verb : GET
uri : /addresses/country/{country}

In this we will search for the address based on the country

Responses :
Code 200 : success

Code 400: Invalid format

Verb: PUT

Uri:/addresses/{address_id}

In this we will search for the address based on the address id identifier

{

“Address2”: “Apartment 111”

}

Verb : GET

Uri:/addresses/{address_id}

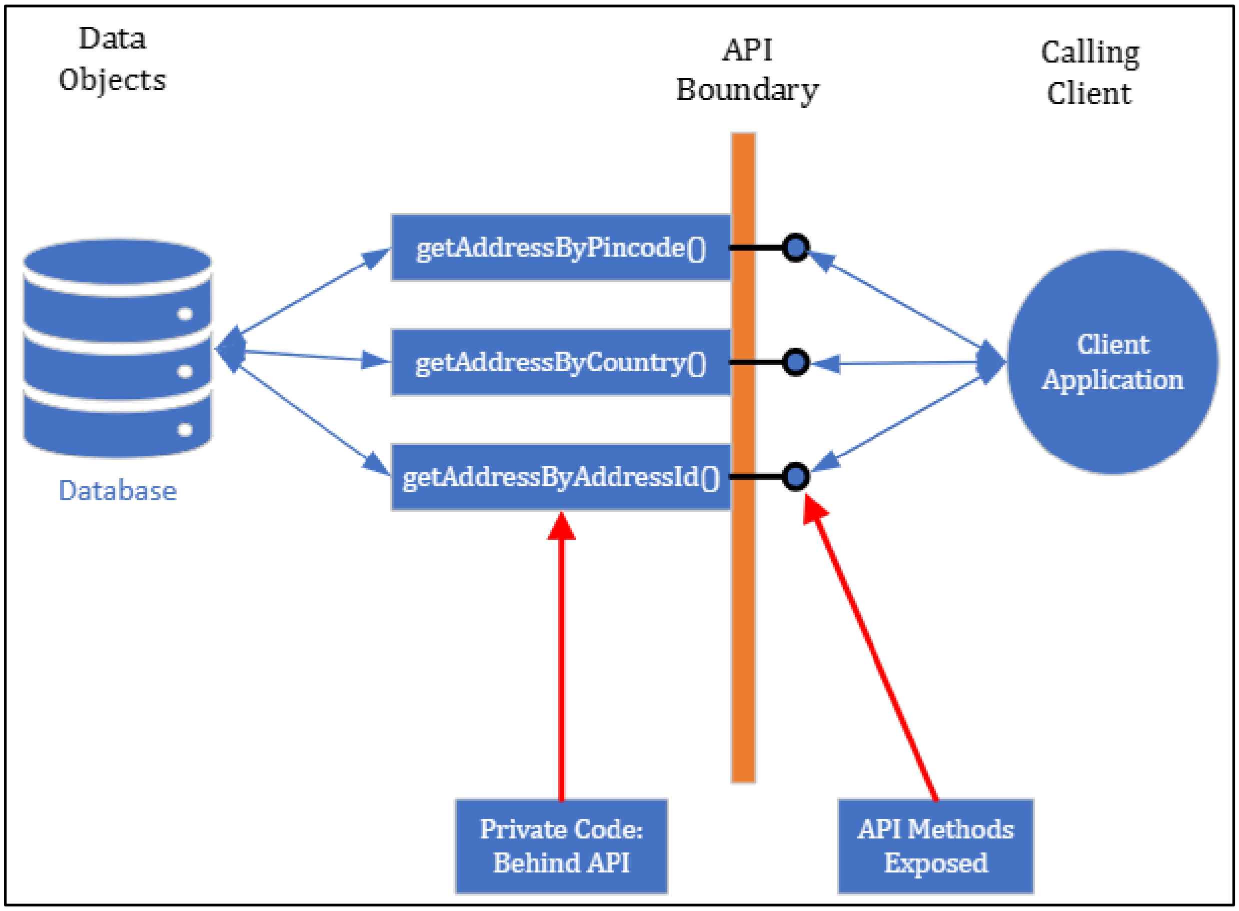
Responses :
Code 200 : success

Code 400: Invalid format

Database: The database component of the postal address handling system is a centralized repository that stores all the address data. It is designed to handle large volumes of data and concurrent requests, ensuring optimal performance and scalability.

When a user searches for an address via the API, the system first looks for the address in the database. If the address is not found, the API retrieves the information from an external source and then stores it in the database. This approach ensures that subsequent searches for the same address are retrieved from the database, rather than requiring an additional request to the external source.

By utilizing this approach, the system provides fast and efficient searches for postal addresses while minimizing the number of external requests required. This allows the system to scale to handle large volumes of users and requests without sacrificing performance.


Data Objects
API Boundary
Calling Client
Database
getAddressByPincode()
getAddressByCountry()
getAddressByAddressId()
Client Application
Private Code: Behind API
API Methods Exposed