**Service Layer Design Document**

**Service Layer** is an abstraction over domain logic or over application's business logic. It defines application's boundary with a layer of services that establishes a set of available operations and coordinates the application's response in each operation.

**Use the Service Layer pattern when we need:**

* To encapsulate domain logic under API.
* To implement multiple interfaces with common logic and data.

**Final application Service end points:**

Base URL - <http://3.15.233.65/api/>

Post ('login' - to login and create session

Post ('signup' - to register

Get ('logout' - to end session

**Dashboard**

get('dashboard/status/{id}'- to change the status of any bill

get('dashboard/pending/{userid}' - get all pending bills limit 4

get('dashboard/pending2/{userid}' - get all pending bills no limit

get('dashboard/upcoming/{userid}' - get all upcoming bills limit 4

get('dashboard/upcoming2/{userid}' - get all upcoming bills no limit

get('bill/category', - fetch all categories

get('user/{email}' - fetch all user detail

get('/dashboard/allactivebills/{userid}' - return active bills with status 1 in 'pending bill' database-table for billing event calendar with different response keys

get('/dashboard/allactivebills2/{userid}' - return active bills with status 1 in 'pending bill' database-table

get('/dashboard/categorygraph/{userid}' - for graph

get('/subscriber/{email}' - to save email ids of contact and subscription

**Billing details crud operations**

get('bill/show/{id}',; - show bill for that id

get('bill/showall/{userid}' - all bills for that user

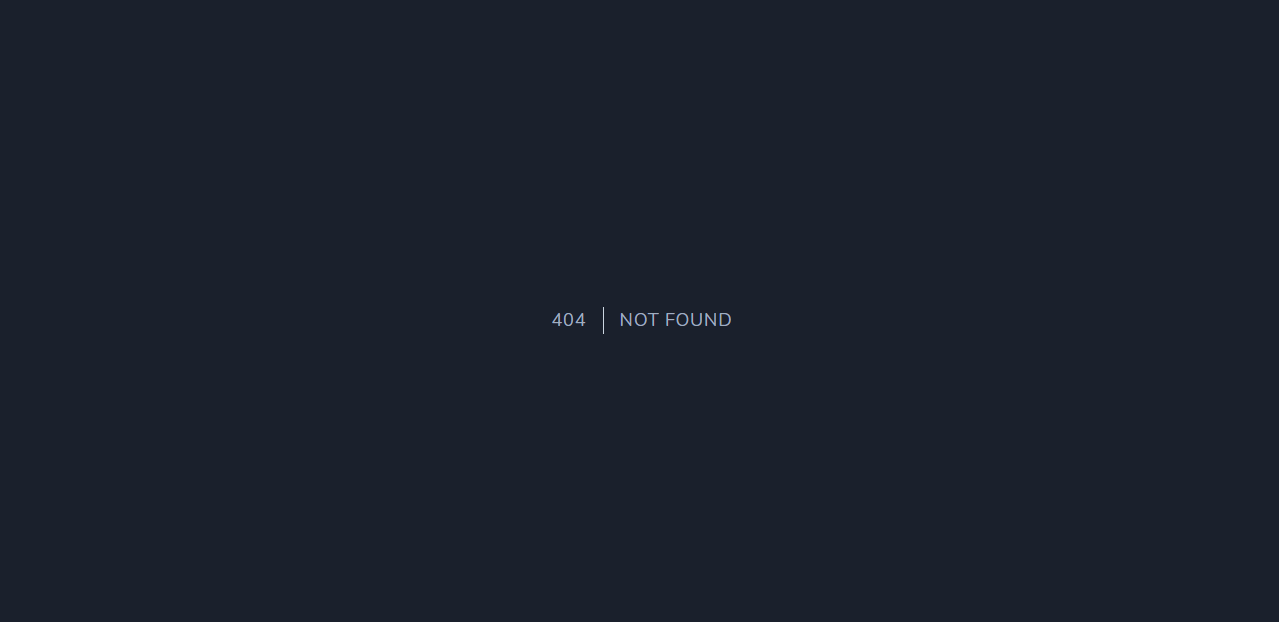
post('bill/create', - add new billing item

post('bill/edit' - edit bill

get('bill/delete/{id}', - delete bill

**Response code:**

1. 404: The 404 error status code indicates that the REST API can’t map the client’s URI to a resource but may be available in the future. Subsequent requests by the client are permissible



1. 401 unauthorised.
2. A 401 error response indicates that the client tried to operate on a protected resource without providing the proper authorization.
3. Internal server error: 500 is the generic REST API error response. Most web frameworks automatically respond with this response status code whenever they execute some request handler code that raises an exception.
4. 4.400 bad request- it is the generic client-side error status, used when no other 4xx error code is appropriate. Errors can be like malformed request syntax, invalid request message parameters, or deceptive request routing etc.
5. 5.200 ok: It indicates that the REST API successfully carried out whatever action the client requested and that no more specific code in the 2xx series is appropriate.

**Diagrams for mapping endpoint to user interface:**

Login page

Host/api/login

Sign-up

page

Host/api/register

Dashboard

Home

page

Host/api/dashboard/home

Manage

Bills page

Host/api/dashboard/manage\_bills

Billing Events

Host/api/dashboard/Billing\_events