**Key Points from the Document on RESTful API Design**

* **API** stands for Application Programming Interface.
* An API is an interface that allows different software applications to communicate with each other.
* The document mentions GUI (Graphical User Interface) and Shell as other ways a user can interact with a computer, contrasting them with API.
* **Socket Programming** is a fundamental concept for network communication, and the document explains the process of creating socket connections on both the client and server sides.
* The document briefly touches on the layers of the internet protocol suite, showing where HTTP, TCP, and IP fit in.
* **HTTP** (Hypertext Transfer Protocol) is used for communication between a web browser (client) and a web server.
* **RESTful API** is mentioned as an architectural style for designing networked applications.
* The document emphasizes the importance of the **HTTP verbs** (GET, POST, PUT, PATCH, DELETE) and how they map to actions on resources.
* **HTTP response codes** (e.g., 200 OK, 204 No Content, 401 Unauthorized, 404 Not Found) are used to indicate the status of a request.
* The document provides examples of how to use HTTP verbs to perform CRUD (Create, Read, Update, Delete) operations on a resource (e.g., /student, /voter).
* The document discusses the roles of different people involved in API development:
  + **Software Architect**: Designs the API (path, request variables, request body, response code, response body).
  + **Developer**: Implements the API based on the architecture.
  + **Maintenance**: Maintains and extends the API.
* The document lists several important aspects of RESTful API architecture:
  + **OpenAPI Specification**: A standard for describing RESTful APIs.
  + **Swagger**: A tool for interactive API documentation.
  + **Data Models**: How resources are modeled.
  + **Authentication**: Protecting API endpoints.
  + **Caching**: Improving performance by storing frequently accessed data.
  + **Circuit Breaker**: Handling failures and improving system robustness.
  + **Log and Audit**: Debugging and security.
* **✅ Get all voters**
* **Endpoint:** GET /voter  
  **Request Body:** None  
  **Response:**

HTTP/1.1 200 OK

[

{

"id": 1292671,

"name": "Abdus Salam",

"Address": {

"village": "..",

"district": ".."

}

}

]

**➕ Create a new voter**

**Endpoint:** POST /voter  
**Request Body:**

{

"name": "Abdus Salam",

"Address": {

"village": "..",

"district": ".."

}

}

HTTP/1.1 200 OK

[

{

"id": 1292671,

"name": "Abdus Salam",

"Address": {

"village": "..",

"district": ".."

},

"status": "active"

}

]

**🛠️ Update a specific voter**

**Endpoint:** PUT /voter?id=1292671  
**Request Body:**

{

"name": "Abdus Salam",

"Address": {

"village": "pahartali",

"district": "Chittagong"

}

}

HTTP/1.1 200 OK

[

{

"id": 1292671,

"name": "Abdus Salam",

"Address": {

"village": "pahartali",

"district": "Chittagong"

},

"status": "active"

}

]

**Update a specific voter (not found)**

**Endpoint:** PUT /voter?id=00000  
**Request Body:**

{

"name": "Abdus Salam",

"Address": {

"village": "pahartali",

"district": "Chittagong"

}

}

**🗑️ Delete a specific voter**

**Endpoint:** DELETE /voter?id=1292671  
**Request Body:** None  
**Response:**