# SECTION 1 — Introduction to Backend Development

## 1. What Is a Backend?

In web development, the backend refers to the server-side part of an application responsible for handling logic, data processing, authentication, and communication with the database.

When a user interacts with an application through a browser or mobile app, the visible interface is the frontend. However, operations such as logging in, placing orders, saving data, or retrieving records are processed by the backend.

The backend acts as the core engine of the application.

## 2. How Backend Integrates with Frontend and Database

A typical web application consists of three main layers:

Frontend → Backend → Database

**Frontend**

- Built using technologies such as React, Angular, or Vue.
- Responsible for user interface and user experience.
- Sends requests to the backend.

**Backend**

- Built using technologies such as Node.js, Java, Python, or .NET.
- Handles:
    - Authentication
    - Authorization
    - Business logic
    - Data validation
    - Security enforcement
    - Database communication

**Database**

- Stores application data permanently.
- Examples include MongoDB, MySQL, PostgreSQL.

**Example: Login Process Flow**

1. User enters email and password on the frontend.
2. Frontend sends a request to the backend.
3. Backend verifies credentials using the database.
4. Backend sends a response (success or error).
5. Frontend displays the result to the user.

The backend is responsible for validating, processing, and securing the request.

## 3. What Is an API?

API stands for Application Programming Interface.

An API is a communication interface that allows two systems to interact with each other.

In web applications:

- The frontend communicates with the backend using APIs.
- The backend communicates with the database using APIs.
- The backend may also communicate with external services such as payment gateways using APIs.

An API defines:

- What can be requested
- What data must be sent
- What response will be returned

**Example of an API Endpoint**

POST /api/auth/login

Frontend sends:

```
{
  "email": "user@example.com",
  "password": "Password@123"
}
```

Backend responds:

```
{
  "accessToken": "generated_token_here"
}
```

This structured communication happens through APIs.

## 4. What Is a REST API?

REST stands for Representational State Transfer.

A REST API is a standardized way of designing APIs using HTTP principles.

RESTful systems follow key rules:

- Use HTTP methods properly
- Represent resources clearly in URLs
- Maintain stateless communication
- Return structured responses (usually JSON)

### Common HTTP Methods

| Method | Purpose |
|---|---|
| GET | Retrieve data |
| POST | Create new data |
| PUT | Replace data |
| PATCH | Update partial data |
| DELETE | Remove data |

### RESTful Example

```
GET    /api/products
POST   /api/products
GET    /api/products/123
DELETE /api/products/123
```

Each endpoint represents a resource and follows predictable behavior.

## 5. Difference Between Library, Framework, and API

Understanding these differences is critical in backend development.

### Library

A library is a collection of reusable functions that you call when needed.

You control when and how to use it.

Examples:

- bcryptjs (password hashing)
- jsonwebtoken (JWT handling)
- mongoose (MongoDB interaction)

Example usage:

bcrypt.hash(password, 10);

You explicitly call the library.

### Framework

A framework provides structure and controls the application flow.

In this case, the framework calls your code at the appropriate time.

Examples:

- Express (Node.js framework)
- Spring Boot (Java)
- Django (Python)

Example in Express:

app.get("/login", controllerFunction);

Here, Express calls your controller when a request arrives.

This is known as inversion of control — the framework controls execution flow.

**API**

An API defines how two systems communicate.

It is not a tool or framework. It is a contract that specifies:

- What request can be made
- What response will be returned

Example:

POST /api/auth/login

The API defines the structure of request and response.

## 6. Concept Summary

| Concept | Definition |
|---|---|
| Backend | Server-side logic and processing |
| API | Communication interface between systems |
| REST API | Standard method for designing APIs |
| Library | Tool you call in your code |
| Framework | Structure that controls application flow |

## 7. Why This Foundation Matters

In this project:

- Node.js is the runtime environment.
- Express is the backend framework.
- Mongoose is used for database interaction.
- JSON Web Tokens (JWT) handle authentication.
- MongoDB stores data.
- We build REST APIs for communication.

Understanding these foundations ensures students can correctly implement structured backend systems rather than writing unorganized code.