WEB PROGRAMMING WITH MEAN

WEB PROGRAMMING WITH MEAN

BTech (AI) III Year I Semester					Dept. of Artificial Intelligence			
Code	Category	Hours / Week			Credits	Marks		
		L	T	P	C	CIE	SEE	Total
A55033	PCC	3	0	0	3	50	50	100

Course Objectives

1. To introduce Node.js for web server platform.
2. To introduce Express for the framework.
3. To introduce MongoDB for the Database.
4. To introduce Mongoose for data modeling.
5. To introduce Angular for front-end framework.

Course Outcomes

At the end of this course, students will be able to:

1. Gain knowledge on client-side scripting.
2. understand server-side scripting.
3. understand MongoDB and create Database.
4. understand Express frame work.
5. create multi-tier architecture web application.

UNIT-I

Architecture of WWW, HTTP, HTTPS, 2-Tier and multi-Tier web application architectures.
Introducing full-stack development: Introduction to the full-stack, history of web development, Introduction to MEAN stack. Node.js, Express, MongoDB and Angular, supporting cast.
Designing a MEAN Stack Architecture: Common MEAN stack architecture, Beyond SPAs, Designing flexible MEAN architecture, planning a real application, breaking the development into stages, hardware architecture.

UNIT-II

Building a Node Web Application
Creating and setting up a MEAN project: Creating an Express project, modifying Express for MVC, Importing Bootstrap for quick, responsive layouts, making it live on Heroku.
Building a static site with Node and Express: Defining the routes in Express, building basic controllers, creating some views, adding the rest of the views, taking the data out of the views and making them smarter.

UNIT-III

Building a data model with MongoDB and Mongoose: Connecting the Express application to MongoDB by using Mongoose, Benefits of modeling the data, defining simple mongoose schemas, using the MongoDB shell to create a MongoDB database and add data, getting database live.

Writing a REST API: Exposing the MongoDB database to the application: The rules of a REST API, setting up the API in Express, GET methods: Reading data from MongoDB, POST methods: Adding data to MongoDB, PUT methods: Updating data in MongoDB, DELETE method: Deleting data from MongoDB.

UNIT-IV

Adding Dynamic Front End with Angular

Creating an Angular application with Typescript: getting up and running with Angular, working with angular components, getting data from an API, putting and Angular application into production.

Building a single-page application with Angular: Foundations: Adding navigation in an Angular SPA, building a modular app using multiple nested components, adding geo-location to find places near you, and safely binding HTML content.

UNIT-V

Managing Authentication and User Sessions

Using an authentication API in Angular applications: Creating an Angular authentication service, creating the Register and Login pages, working with authentication in the Angular app.

Text Book

1. Simon Holmes, Clive Harber "Getting MEAN with Mongo, Express, Angular and Node", Second Edition, Manning Publications Co., 2019

References

1. Adam Bretz & Colin J. Ihrig "Full Stack Javascript Development With Mean"
2. Amos Q. Haviv, MEAN Web Development Second Edition, Packt Publishing, 2016

WEB PROGRAMMING WITH MEAN LAB

BTech (AI) III Year I Semester					Dept. of Artificial Intelligence			
Code	Category	Hours / Week			Credits	Marks		
		L	T	P	C	CIE	SEE	Total
A55214	PCC-Lab	0	0	3	1.5	50	50	100

The following tasks have to be performed week wise

Week 1:
Identification of the problem

Week 2:
a. Requirement specification
b. Architecture design

Week 3, 4 & 5:
Development of the front end.

Week 6, 7 & 8:
Development of backend

Week 9, 10 & 11:
Creation of Database

Week 12:
Integration and Testing

Week 13:
Presentation

Week 14:
Documentation and Submission

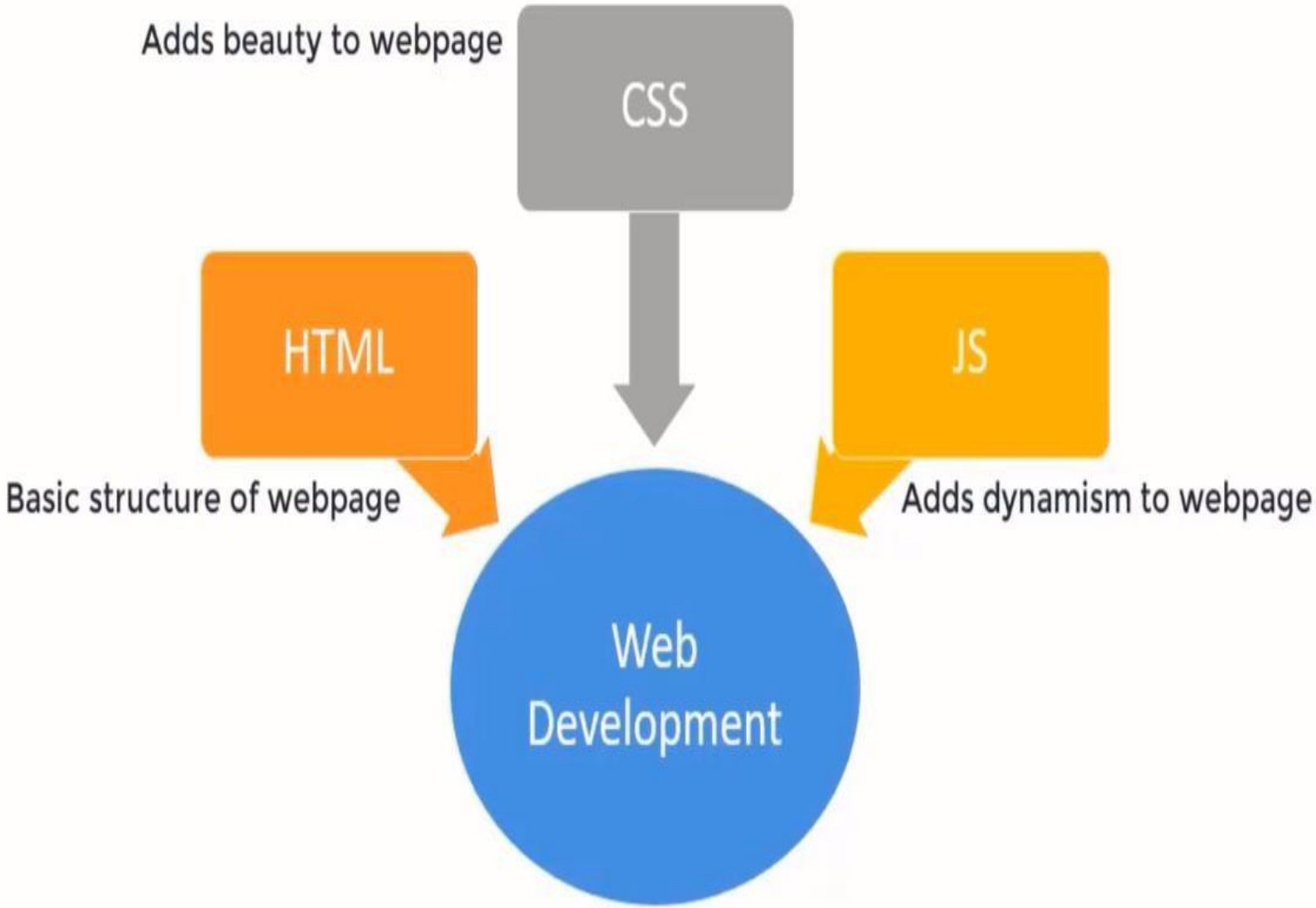
MEAN STACK DEVELOPMENT

UNIT-I

Architecture of WWW, HTTP, HTTPS, 2-Tier and multi-Tier web application architectures. **Introducing full-stack development:** Introduction to the full-stack, history of web development, Introduction to MEAN stack. Node.js, Express, MongoDB and Angular, supporting cast.

Designing a MEAN Stack Architecture: Common MEAN stack architecture, Beyond SPAs, Designing flexible MEAN architecture, planning a real application, breaking the development into stages, hardware architecture.

Core Web Technologies



Hypertext Markup Language (HTML5) – Learning Path

Getting started with web development

Introduction to web and basic web technologies: HTML, CSS, JS

HTML Basic Elements

Use HTML Semantic, Formatting elements, hyperlinks, etc

HTML Tables

Enhance content readability by applying HTML Tables

HTML Embedded Elements

Embed content using media and iframe HTML elements

HTML Document Structure

Need and use of HTML document structure

HTML Lists

Enhance content readability by applying HTML Lists

HTML Forms

Capture user inputs using HTML Forms and form controls

HTML Best Practices

Coding Standards and Security measures

ANURAG UNIVERSITY

WORLD WIDE WEB

What is World Wide Web?

- The **World Wide Web (WWW)** is a network of online content that is formatted in HTML and accessed via HTTP. The term refers to all the interlinked HTML pages that can be accessed over the Internet.
- It is the way of exchanging information between the computers on the Internet.
- A document on the web is called Web Page, identified by a unique address called Uniform Resource Locator (URL).

HISTORY

- 1989-1990 – Tim Berners-Lee invents World Wide Web at CERN.
- On 30 April 1993, CERN put the World Wide Web software in the public domain. Later, CERN made a release available with an open license, a more sure way to maximize its dissemination.
- Tim moved from CERN to the Massachusetts Institute of Technology in 1994 to found the World Wide Web Consortium (W3C), an international community devoted to developing open web standards.
- Established as a common language for sharing information on computers.

STRUCTURE

- Clients use browser application to send URIs via HTTP to servers requesting a Web Page.
- Web Pages constructed using HTML (or other markup language) and consists of text, sounds, graphics plus embedded files.
- Servers (or Caches) respond with requested Web Page
  - Or with Error message
- Client's browser renders Web Page returned by server
  - Page is written with Hyper Text Markup Language (HTML).
  - Displaying text, graphics and sound in browser.
  - Writing data as well.
- The entire system runs over standard networking protocols (TCP/IP, DNS etc.)

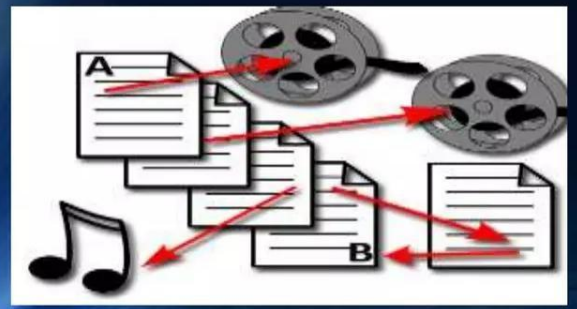
World Wide Web Components

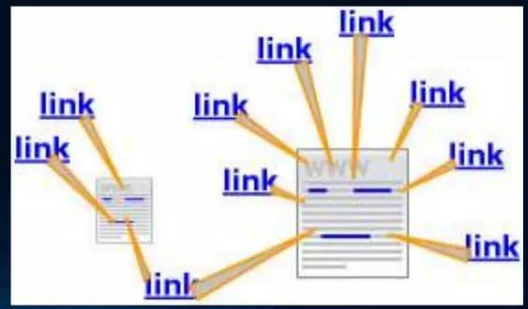
- Structural Components :
  - **Clients/Browsers** – to dominant implementations.
  - **Servers** – Run on sophisticated hardware.
  - **Caches –** Many interesting implementations.
  - **Internet** – The global infrastructure which facilitates data transfer.
- Semantic Components :
  - Hyper Text Transfer Protocol (HTTP)
  - Hyper Text Markup Language (HTML)
  - Extensible Markup Language (XML)
  - Uniform Resource Identifiers (URIs)

ANURAG UNIVERSITY

The Fundamental Concept of World Wide Web

- The Hypertext Concept :
  - **Hypertext** is text which contains links to other texts. The term was first coined by Ted Nelson around 1965 .
- The Hypermedia Concept :
  - **Hypermedia** is term used for Hypertext which is not constrained to be text; it can include video and sound.





Web Browser :

- A web browser is a software program that allows a user to access, and display web pages. Browsers are used primarily for displaying and accessing websites on the internet, as well as other content created using languages such as Hypertext Markup Language (HTML) and Extensible Markup Language (XML).

Web Server :

- This program that waits patiently for the browser to request a Web Page. The servers looks for the requested information, retrieves it and send it to the browser or sends an error message if the file is not found.

Uniform Resource Locator (URL) :

- These are the web addresses. The resource locator is an addressing system.

The Difference between Internet and World Wide Web

- Many people use the terms **Internet** and **World Wide Web**, but in fact the two terms are not synonymous. The **Internet** and the **Web** are two separate but related things.
- The **Internet** s a massive network of networks. It connects millions of computers together globally, forming a network in which any computer can communicate with any other computer as long as they are both connected to the Internet.
- The **World Wide Web**, or simply **Web**, is a way of accessing information over the medium if the Internet.
- So the **Web** is just a portion of the **Internet**, a larger portion, but two terms are not synonymous and should not be confused.

WEB 2.0

- Web 2.0 is the term given to describe a second generation of the World Wide Web. It refers to the features of the World Wide Web which allow people to collaborate and share information online. One see the transition rom static HTML.
- Web 2.0 provides a number of online tools and platforms which are now defining how people share their perspectives, opinions, thoughts and experiences.
- Examples :
  - **Social Networking Websites –** facebook.com, twitter.com, instagram.com
  - **Music and Video Websites –** youtube.com
  - **Image Websites –** imgur.com, flickr.com
  - **Blogging Websites –** blogger.com, wordpress.com


HTTP vs HTTPS
Secure vs Non-Secure Websites | Know The Difference
http://domain.com
https://domain.com

ANURAG UNIVERSITY

HTTP and HTTPS

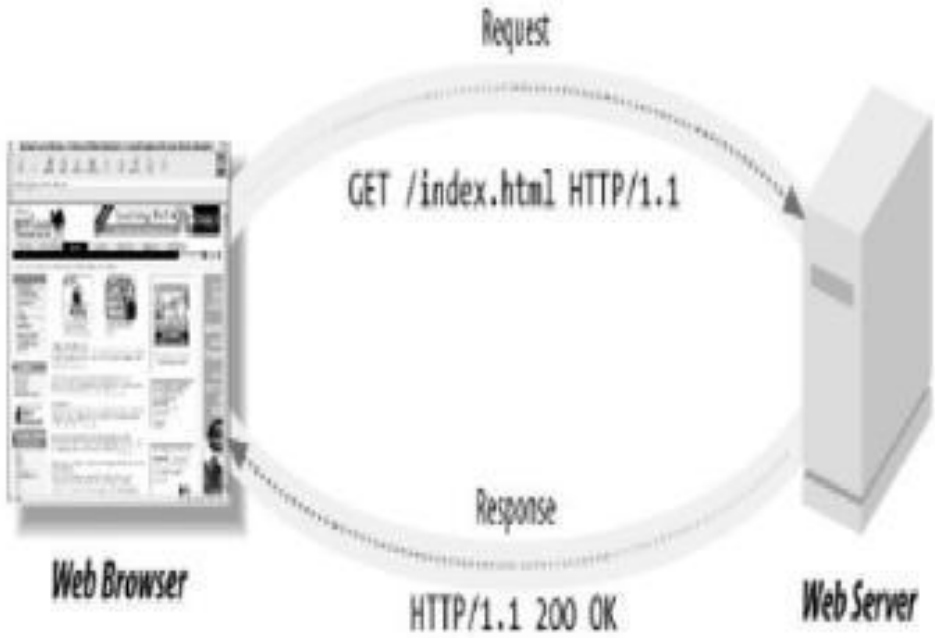
A Brief Introduction

- The World Wide Web (WWW) as we know it uses both HTTP and HTTPS protocols to send and receive data over the internet. But, what are the differences between HTTP vs HTTPS?
- The Hypertext Transfer Protocol (HTTP) is a protocol for sending and receiving data between a web server and a client. This protocol is used to send almost any type of file, including images, text, and video. The HTTP web server helps to connect with web browsers including Chrome, Mozilla, IE, etc.

HTTP

- HTTP stands for **Hypertext Transfer Protocol.**
- HTTP provides a set of rules and standards that govern how information is transmitted on the World Wide Web.
- Computers on the World Wide Web use the HyperText Transfer Protocol to talk with each other
- **http://www.google.co.in**

- The first part of an address (URL) of a site on the Internet, signifying a document written in Hypertext Markup Language (HTML).



HTTP

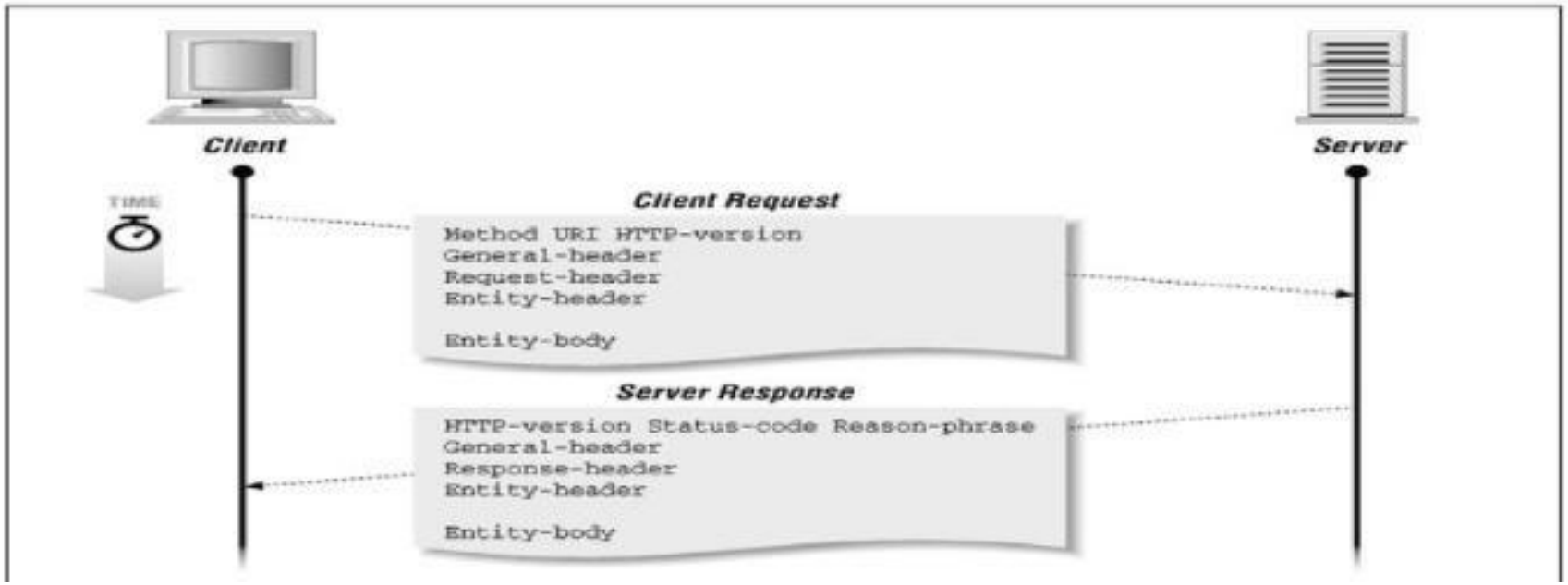
- HTTP is a client-server protocol by which two machines communicate using a reliable, connection-oriented transport service such as the TCP.
  - A browser is an *HTTP client* because it sends requests to an *HTTP server* (Web server), which then sends responses back to the client
  - An HTTP server is a program that sits listening on a machine's port for HTTP requests.
  - The standard (and default) port for HTTP servers to listen on is 80, though they can use any port.
- HTTP can be "implemented on top of any other protocol on the Internet, or on other networks."
- HTTP only presumes a reliable transport; any protocol that provides such guarantees can be used." e.g. TCP.

HTTP

- HTTP is stateless. The lifetime of a connection corresponds to a single request-response sequence
  - An HTTP client opens a TCP/IP connection to the server via a socket, transmits a request for a document, then waits for a reply from the server. Once the request-response sequence is completed, the socket is closed.
  - There is no "memory" between client connections.
  - The pure HTTP server implementation treats every request as if it was brand-new.

How HTTP Works

- HTTP Server is implemented by Apache HTTP Server · Microsoft IIS · Jigsaw · Zope etc.
- Each client-server transaction, whether a request or a response, consists of three main parts
  - A response or request line
  - Header information
  - The body



Advantages of HTTP

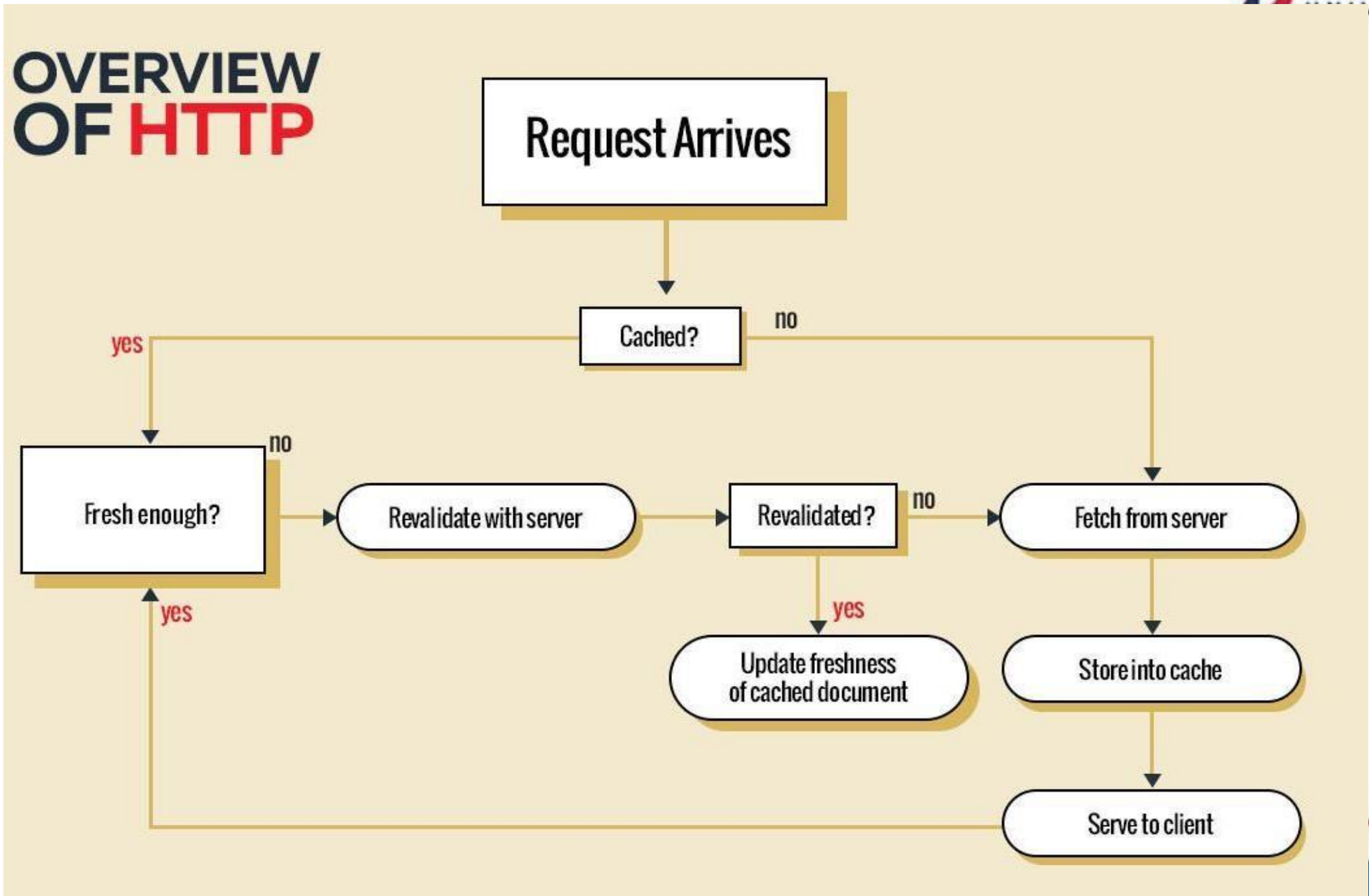
- Platform independent- Allows Straight cross platform porting.
- No Runtime support required to run properly.
- Usable over Firewalls! Global applications possible.
- Not Connection Oriented- No network overhead to create and maintain session state and information.

HTTP Limitations

Security Concerns

- Privacy -Anyone can see content
- Integrity -Someone might alter content. HTTP is insecure since no encryption methods are used. Hence is subject to man in the middle and eavesdropping of sensitive information.
- Authentication -Not clear who you are talking with. Anyone who intercepts the request can determine the username and password being used.
- Stateless - Need State management techniques to maintain the information across multiple request-response cycles.

OVERVIEW OF HTTP

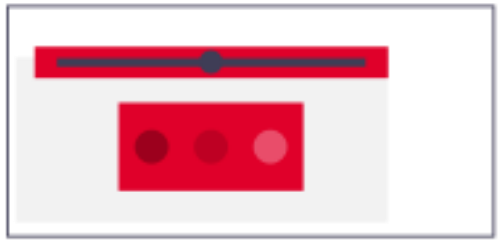


ANURAG UNIVERSITY

Here’s How it Works

The HTTP exchanges data in the simple text between the above-mentioned browsers and the server. You can view the information transferred if you have access to the network where it is transmitted.

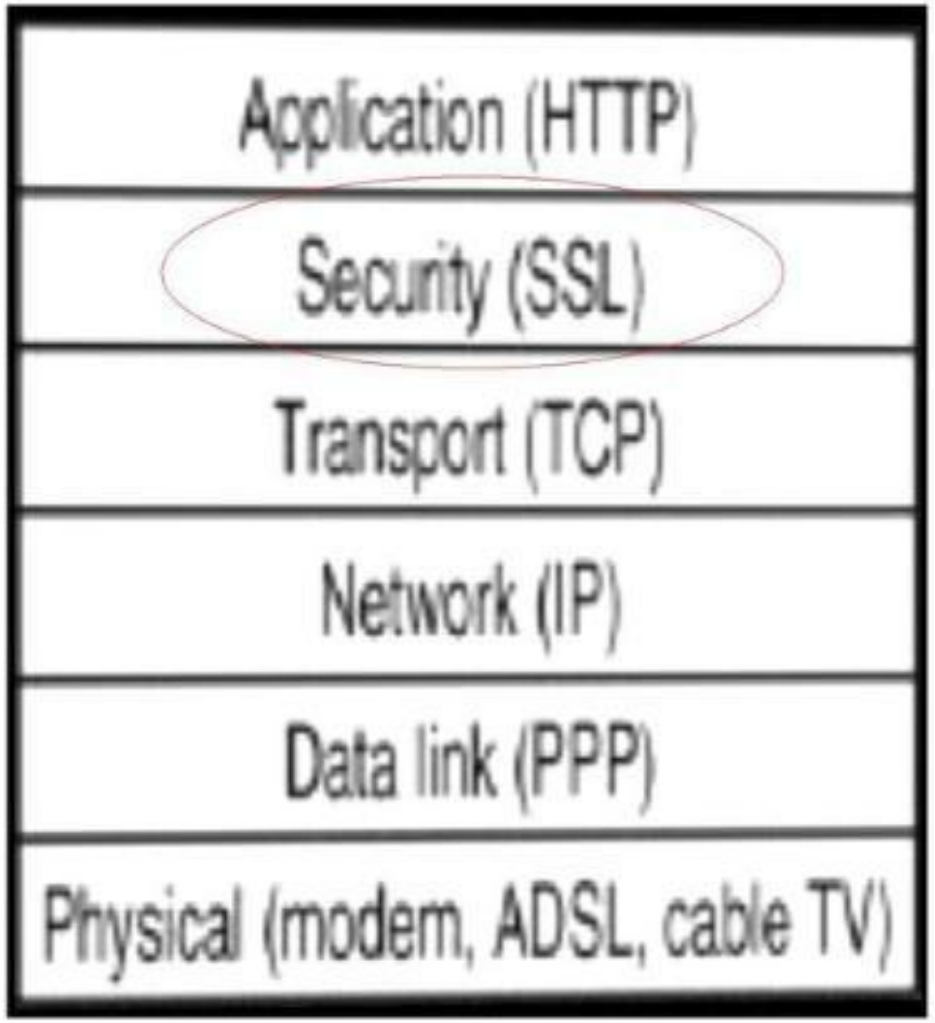
As a result, it’s quite probable that pushy advertising, security agencies, etc. are scrutinizing or storing this information somewhere along with this communication protocol.



ANURAG UNIVERSITY

HTTPS

- HTTPS stands for Hypertext Transfer Protocol over Secure Socket Layer, or HTTP over SSL.
- SSL acts like a sub layer under regular HTTP application layering.
- HTTPS encrypts an HTTP message prior to transmission and decrypts a message upon arrival.



HTTPS

- HTTPS by default uses port 443 as opposed to the standard HTTP port of 80.
- URL's beginning with HTTPS indicate that the connection between client and browser is encrypted using SSL

  e.g.: https://login.yahoo.com/config/login_verify2?&.src=ym
- SSL transactions are negotiated by means of a key based encryption algorithm between the client and the server, this key is usually either 40 or 128 bits in strength (the higher the number of bits the more secure the transaction).

HTTPS

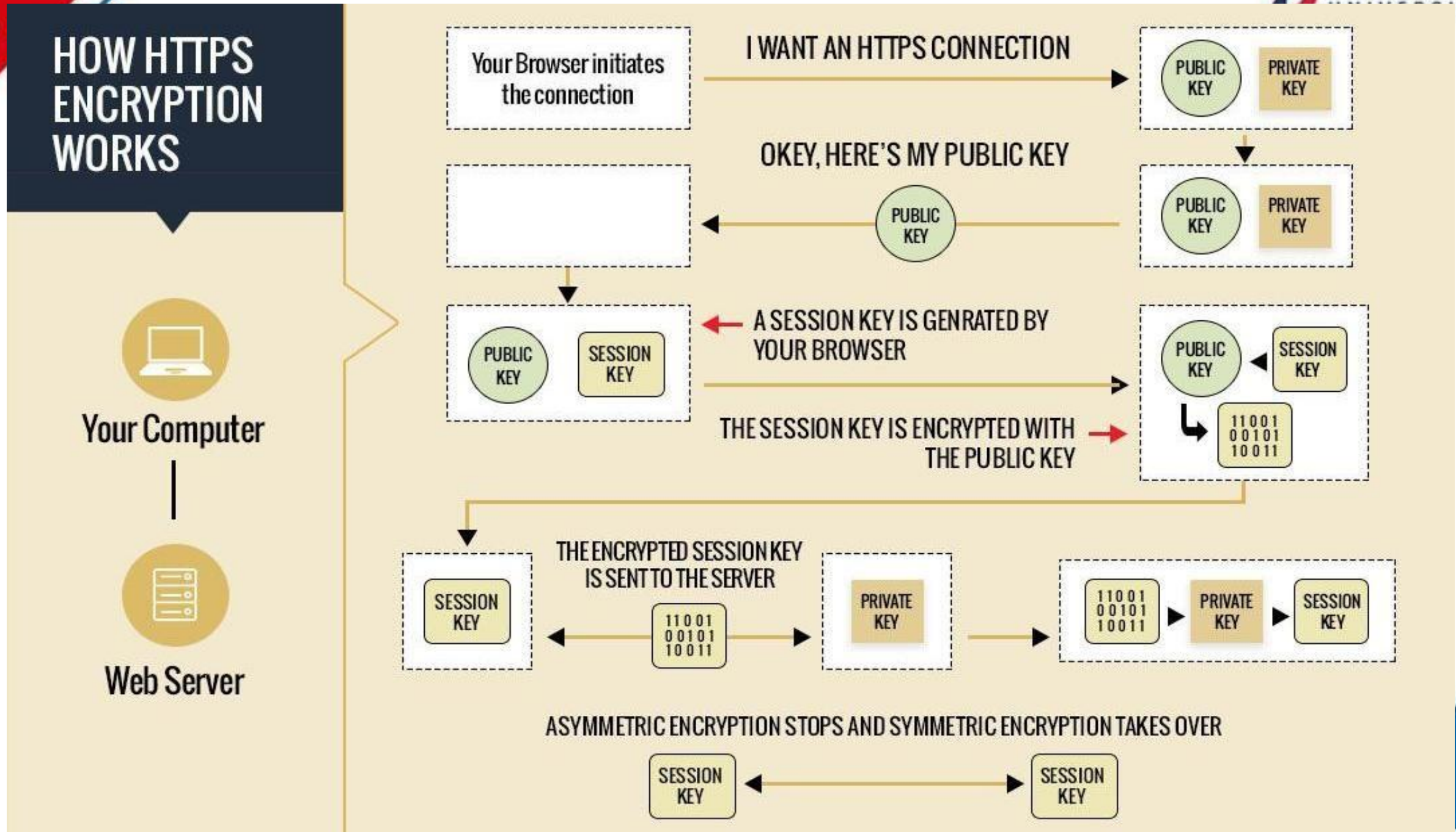
- Need SSL if...
  - you have an online store or accept online orders and credit cards
  - you offer a login or sign in on your site
  - you process sensitive data such as address, birth date, license, or ID numbers
  - you need to comply with privacy and security requirements

Certification Authority (CA) is an entity that issues digital certificates for use by other parties. It is an example of a trusted third party.

e.g. VeriSign, Thwate, Geotrust etc

HTTPS

- Ability to connect to server via HTTP secure consists of:
  - Generating key
  - Generating certificate signing request
  - Generating self signed certificate
  - Certificate Authority signed certificate
  - Configuring web server.


HOW HTTPS ENCRYPTION WORKS
Your Computer
Web Server
Your Browser initiates the connection
I WANT AN HTTPS CONNECTION
PUBLIC KEY
PRIVATE KEY
OKEY, HERE'S MY PUBLIC KEY
PUBLIC KEY
PRIVATE KEY
PUBLIC KEY
SESSION KEY
A SESSION KEY IS GENRATED BY YOUR BROWSER
PUBLIC KEY
SESSION KEY
11001 00101 10011
THE SESSION KEY IS ENCRYPTED WITH THE PUBLIC KEY
SESSION KEY
THE ENCRYPTED SESSION KEY IS SENT TO THE SERVER
11001 00101 10011
PRIVATE KEY
11001 00101 10011
PRIVATE KEY
SESSION KEY
ASYMMETRIC ENCRYPTION STOPS AND SYMMETRIC ENCRYPTION TAKES OVER
SESSION KEY
SESSION KEY

Limitations of HTTPS

- HTTPS cannot prevent stealing confidential information from the pages cached on the browser.
  - Since in SSL data is encrypted only during transmission on the network, it is in clear text in the browser memory
- HTTPS is slightly slower than HTTP.
  - HTTPS adds computational overhead as well as network overhead.

Conclusion

- The HTTP network protocol is fundamental to the way the World Wide Web works, and the encryption involved in HTTPS adds an essential layer if confidential information or sensitive data are to be exchanged over the public internet.

- Hence, If a website *ever* asks you to enter your credit card information, you should *automatically* look to see if the web address begins with **https://**. If it *doesn't*, there's no way you're going to enter sensitive information like a credit card number!

ANURAG UNIVERSITY

TWO-TIER ARCHITECHTURE

INTRODUCTION

Definition: Two-tier architecture is a client-server architecture where the application logic and data storage are divided into two distinct layers.

- Two Layers: The two layers are the client layer and the server layer.
- Communication: The client layer interacts directly with the server layer to retrieve data and perform business logic.

ANURAG UNIVERSITY

TWO-TIER ARCHITECHTURE

CLIENT TIER: The client layer consists of the user interface (UI) and presentation logic. It runs on the client machine and is responsible for displaying information to the user and accepting user input

SERVER TIER: The server layer contains the business logic and data storage. It runs on a server machine and processes client requests, retrieves data from the database, and performs any necessary computations

ANURAG UNIVERSITY

Two-tier Architechture

KEY COMPONENTS

1.CLIENT TIER

*USER INTERFACE

*PRESENTATION LOGIC

*DATA VALIDATION

2.SERVER TIER

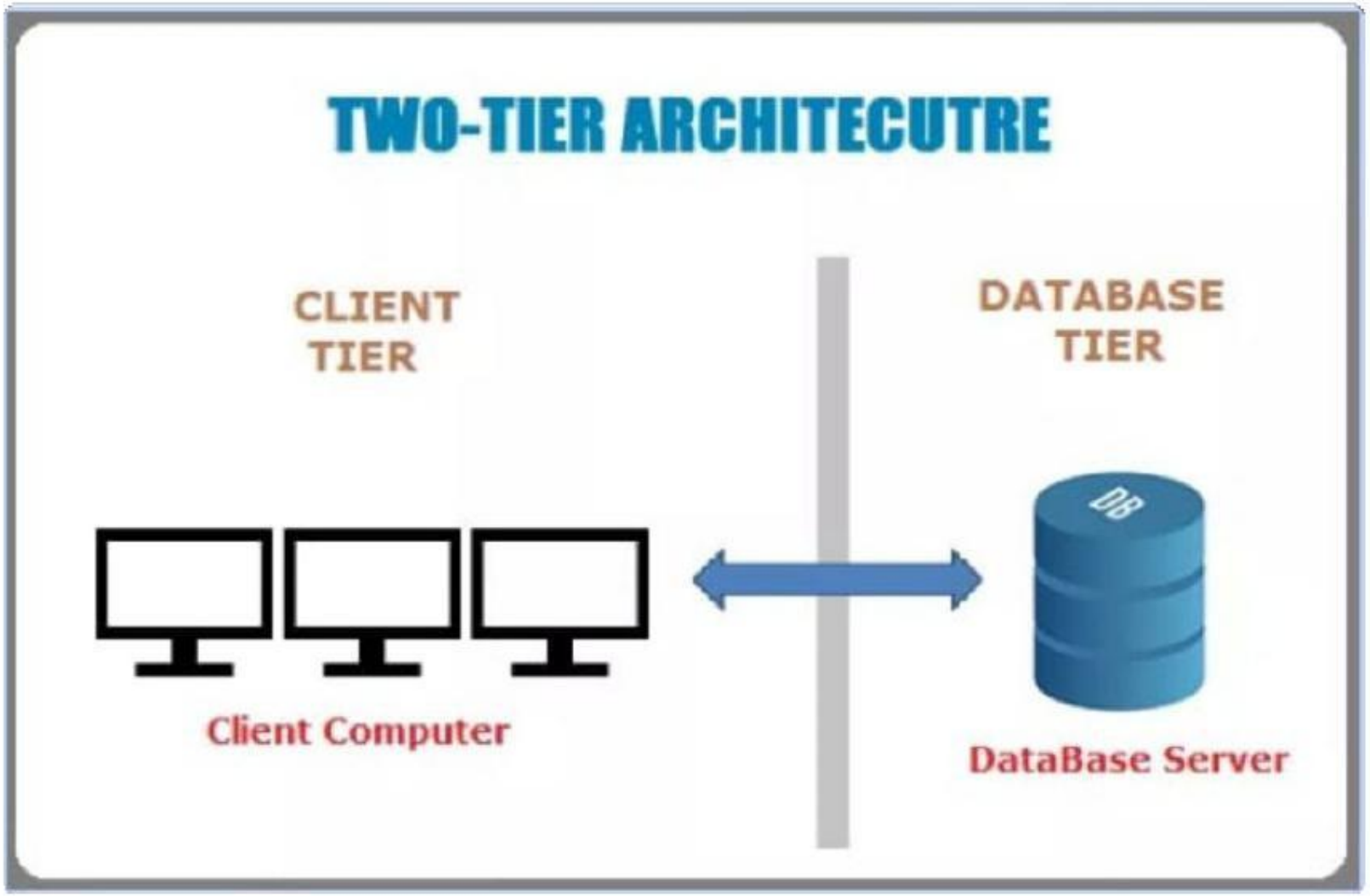
*DATABASE

*BUSINESS LOGIC

*DATA ACCESS LAYER

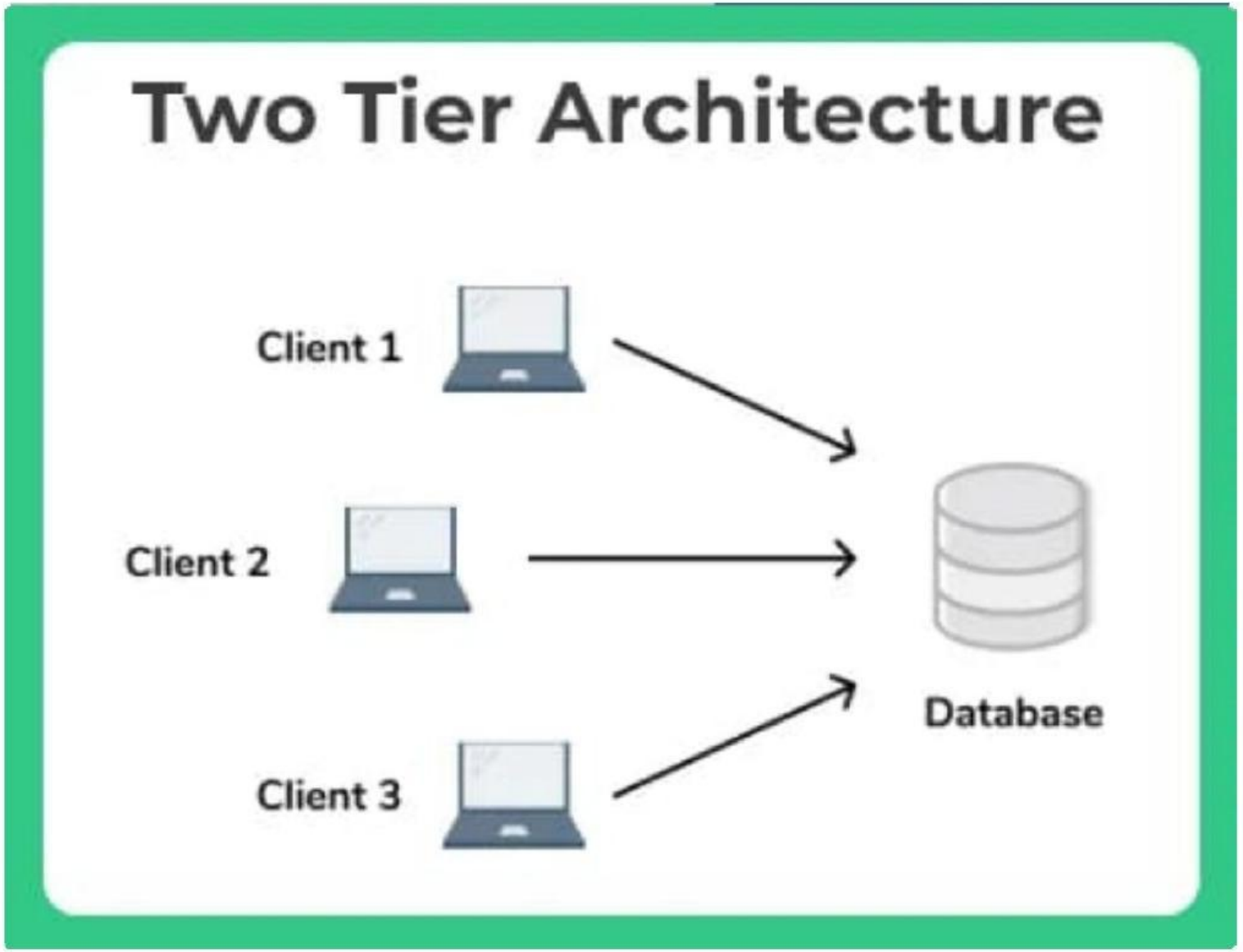
ANURAG UNIVERSITY

TWO-TIER ARCHITECHTURE



ANURAG UNIVERSITY

TWO-TIER ARCHITECHTURE



ANURAG UNIVERSITY

TWO-TIER ARCHITECHTURE

In a Two-tier Architecture, the server manages the database functionality. It makes it possible for the clients to use the Database through APIs (Application Programming Interfaces) over a direct internet connection. For example, ODBC (Open Database Connectivity) and JDBC(Java Database Connectivity).

ANURAG UNIVERSITY

TWO-TIER ARCHITECHTURE

1. SIMPLICITY
2. PERFORMANCE
3. SCALABILITY

ANURAG UNIVERSITY

TWO-TIER ARCHITECHTURE

1. LACK OF FLEXIBILITY
2. MAINTAINANCE CHALLENGES
3. LIMITED SECURITY

ANURAG UNIVERSITY

TWO-TIER ARCHITECHTURE

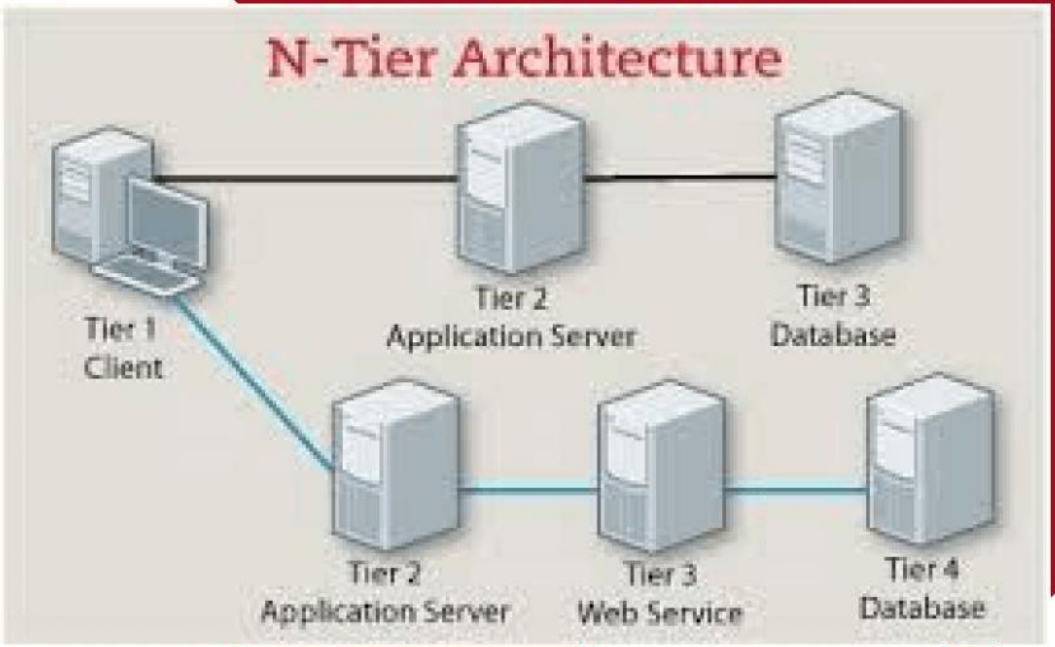
- Two-Tier Architecture provides a simple and cost-effective approach to client-server applications.
- It is suitable for small-scale systems and standalone applications.
- While it has limitations in terms of scalability and security, it can be a practical choice for specific use cases.

N-TIER ARCHITECTURE

WHAT IS N-TIER?

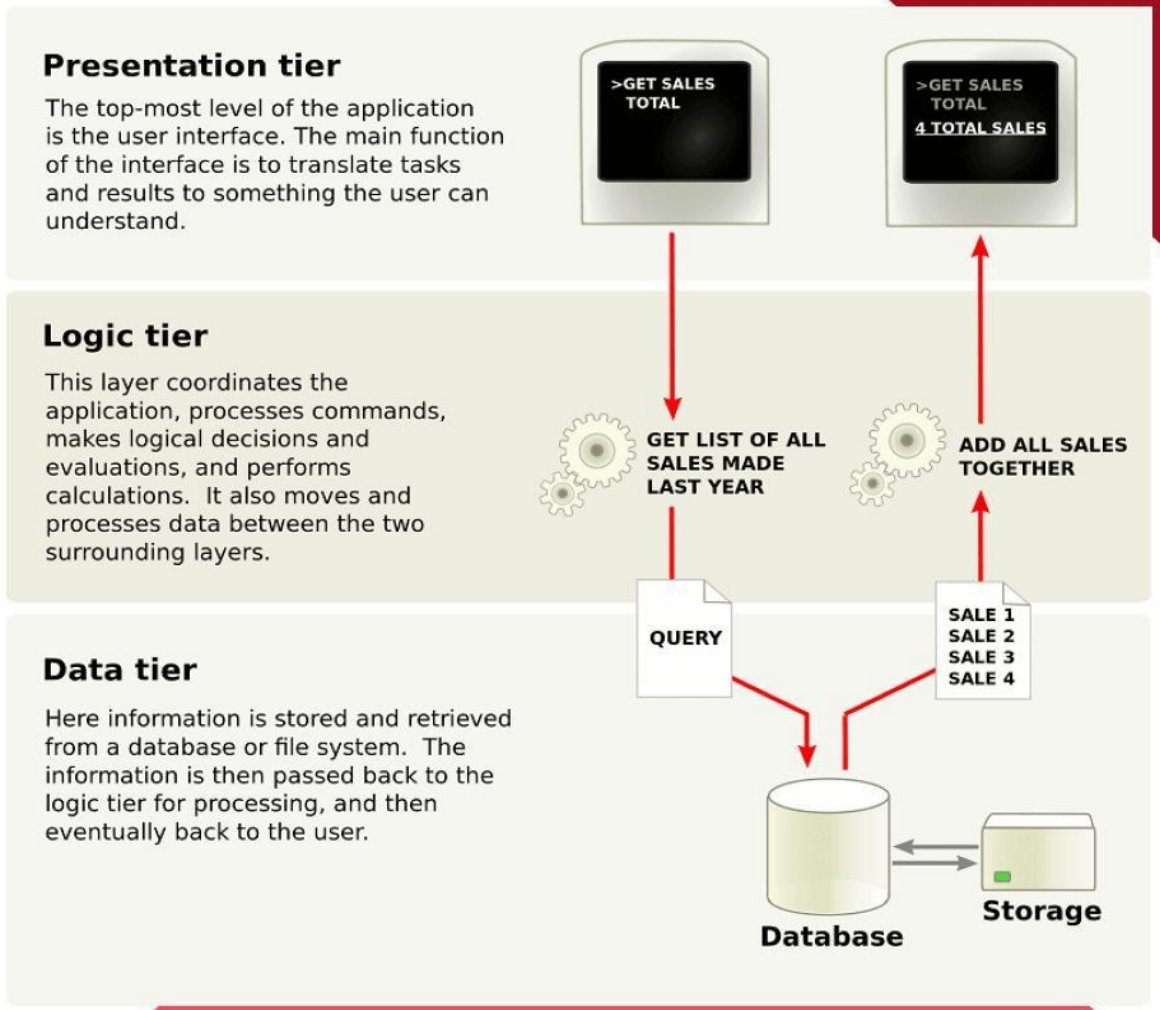
An N-Tier Application program is one that is distributed among three or more separate computers in a distributed network.
It is classified into three categories.

- User interface programming in the user's computer
- Business logic in a more centralized computer, and
- Required data in a computer that manages a database.



N-tier architecture would involve dividing an application into three different tiers. These would be the

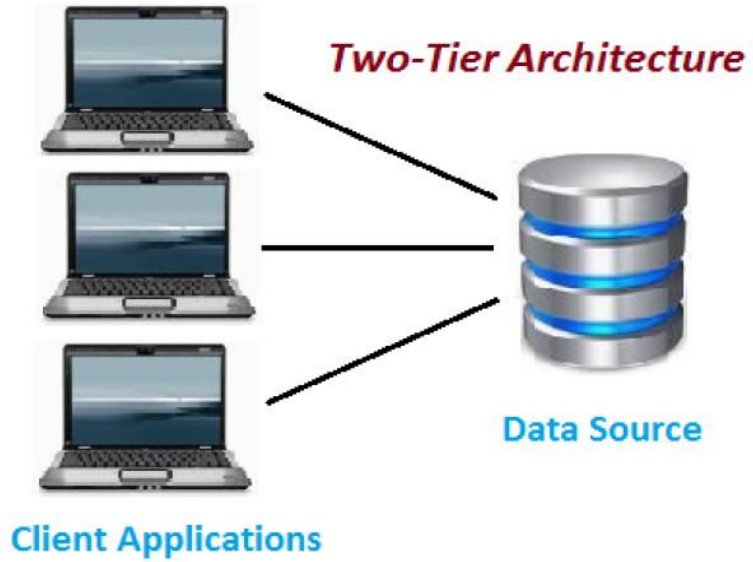
1. **Logic tier,**
2. **The presentation tier, and**
3. **The data tier.**

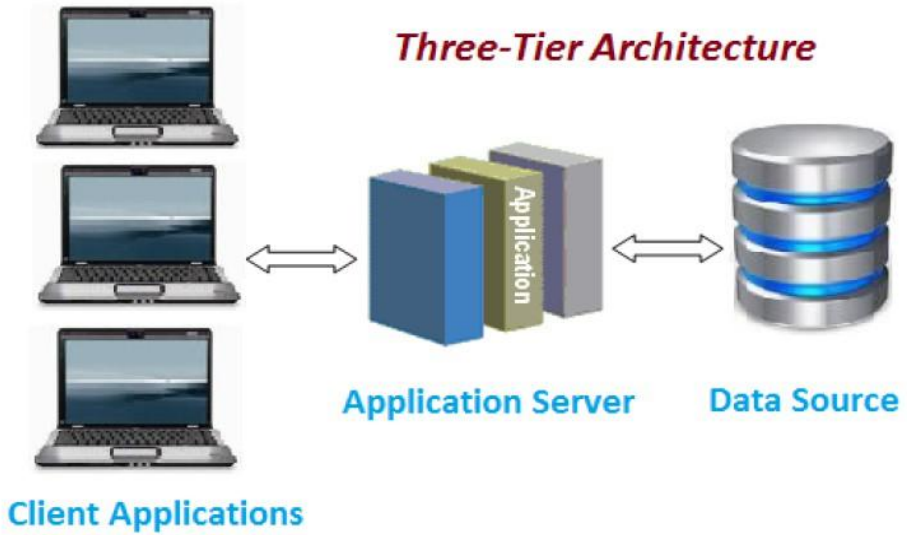


ANURAG UNIVERSITY

TYPES OF N-TIER ARCHITECTURES

In N-tier, “N” refers to a number of tiers or layers are being used like – 2-tier, 3-tier or 4-tier, etc. It is also called “Multi-Tier Architecture”.





2-TIER ARCHITECTURES

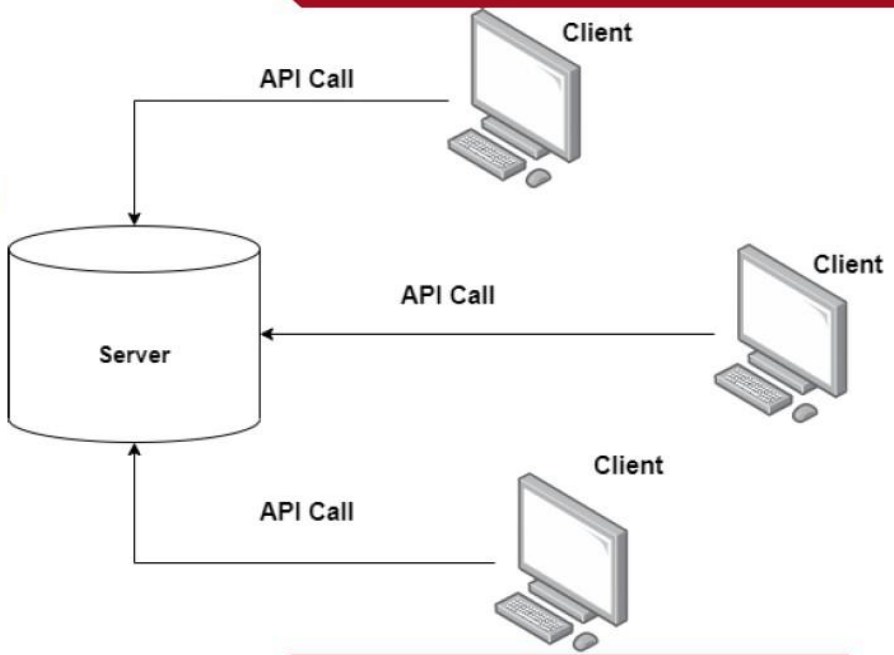
- The essential idea of client-server architecture is comparable to Two-tier DBMS Architecture.
- In a Two-tier Architecture, the server manages the database functionality.
- It makes it possible for the clients to use the Database through APIs (Application Programming Interfaces) over a direct internet connection.
- For example, ODBC (Open Database Connectivity) and JDBC(Java Database Connectivity).

2-TIER ARCHITECTURES

- Through the use of API calls, client-side applications can connect directly to the database server, separating the application from the Database in terms of functionality, programming, and design.

The following are the key benefits of a two-tier architecture over one tier:

- Due to the server's exclusive responsibility for managing database functions, it has significant processing power.
- It is simpler to maintain due to the two independent levels.
- It can be used concurrently by several people. It can therefore be applied within a company.
- Faster database access because of the direct link and enhanced efficiency.



2-TIER ARCHITECTURES

The following are the key drawbacks of the two-tier DBMS architecture:

1)Scalability:

- The server is under more pressure as there are more users. Decreasing the DBMS's performance, which in turn affects the client-side application's performance.

2)Security:

- This design is prone to assaults since the client and server systems are directly connected.

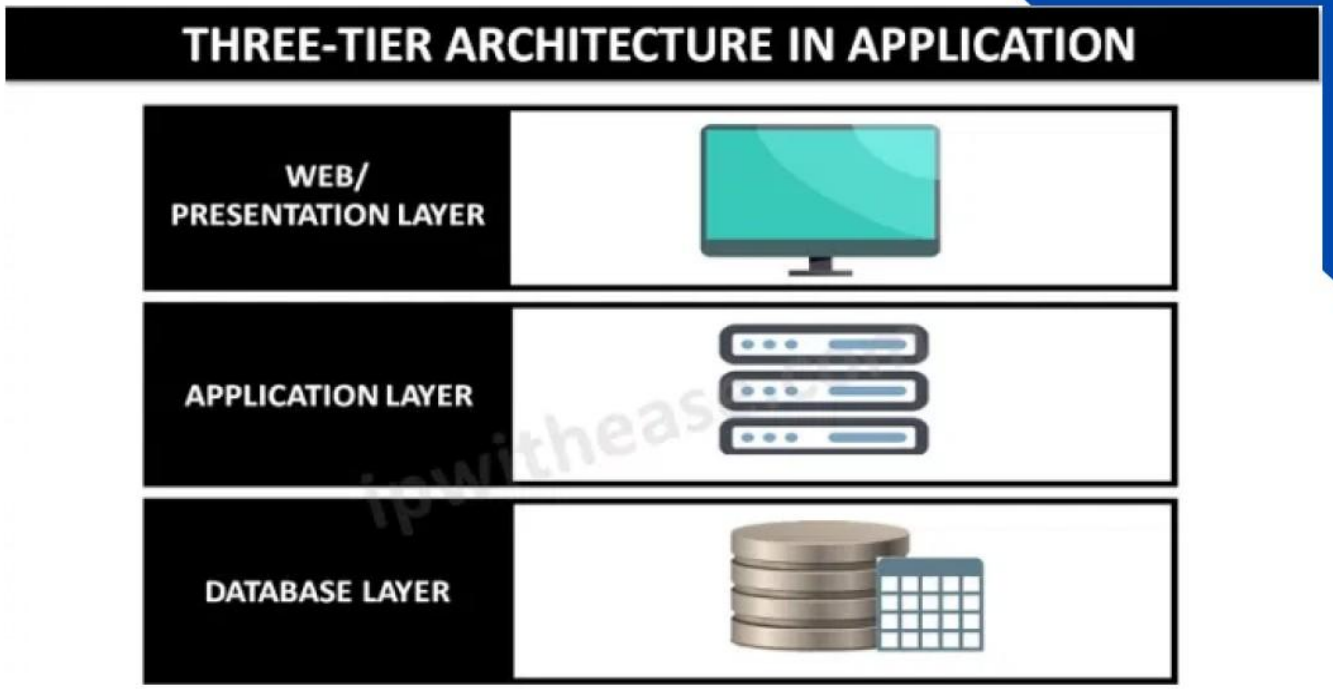
Highlights of Two-Tier Architecture:

- Faster access, simpler maintenance, and capacity for several concurrent users.
- Due to the direct client-server connection, it has scalability and security difficulties.
- It is comparable to a client-server setup.
- It is used when we want to use applications and APIs to access DBMS.

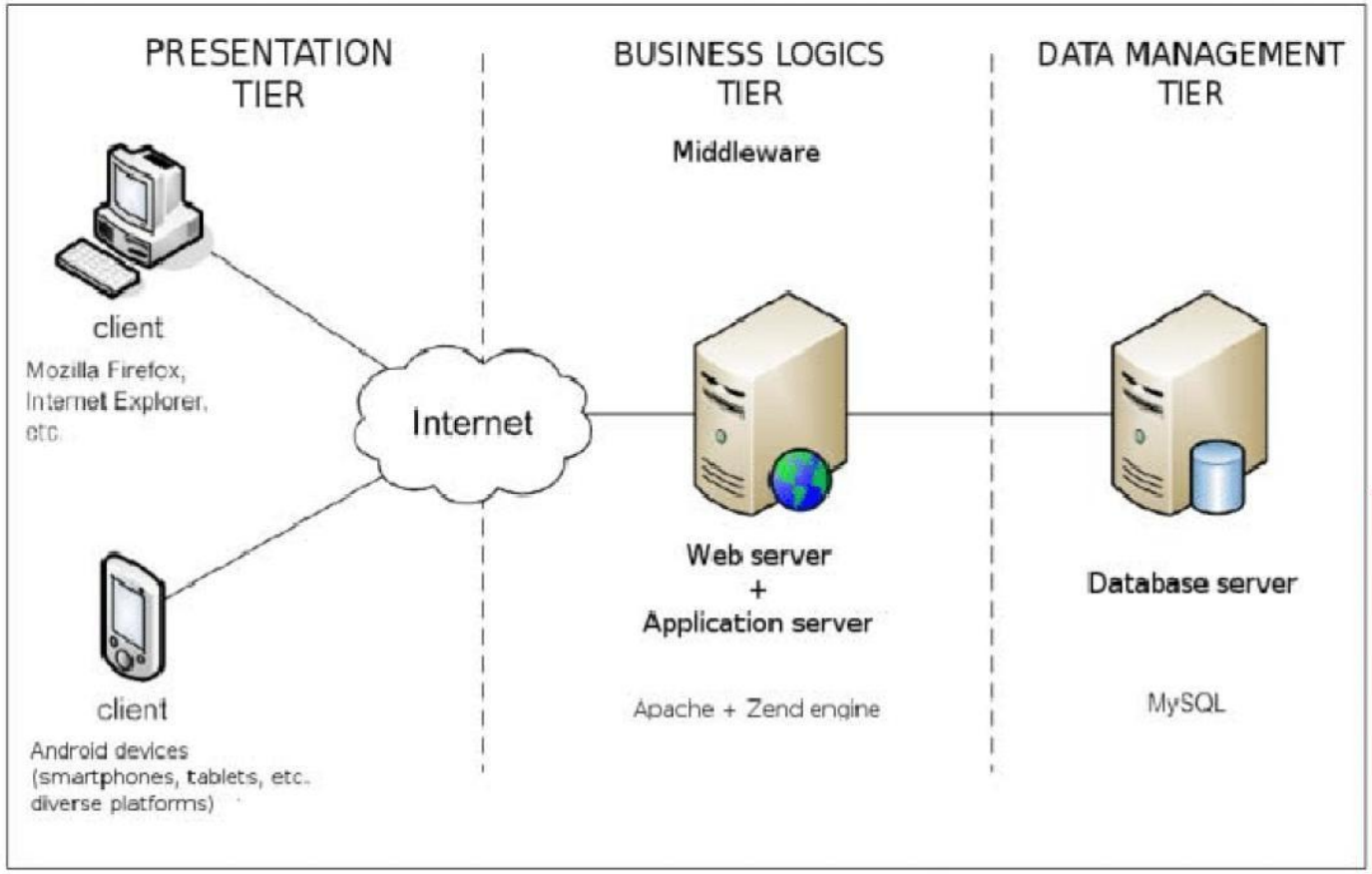
3-TIER ARCHITECTURES

By looking at the diagram, you can easily identify that 3-tier architecture has three different layers.

- Presentation layer
- Business Logic layer
- Database layer



OVERVIEW OF 3-TIER ARCHITECTURE



ANURAG UNIVERSITY

3-TIER ARCHITECTURE

Highlights of Two-Tier Architecture:

- Maintainability - Because each tier is independent of the other tiers, updates or changes can be carried out without affecting the application as a whole.
- Scalability - Because tiers are based on the deployment of layers, scaling out an application is reasonably straightforward.
- Flexibility - Because each tier can be managed or scaled independently, flexibility is increased.
- Availability - Applications can exploit the modular architecture of enabling systems using easily scalable components, which increases availability.
- Reusability - Components are reusable
- Faster development - Because of division of work web designer does presentation, software engineer does logic, DB admin does data model.

The following are the key drawbacks of the two-tier DBMS architecture:

- High installation cost.
- Structure is more complex as compare to 1 & 2 tier architectures.

Applications

- E-commerce Websites
- Database related Websites

ANURAG UNIVERSITY

Introducing full-stack development

ANURAG UNIVERSITY

MEAN STACK

ANURAG UNIVERSITY

Introduction

- **MEAN** is an opinionated full stack JavaScript framework which simplifies and accelerates web application development.
- **MEAN** represents a major shift in architecture and mental models — from relational databases to NoSQL and from server-side Model-View-Controller to client-side, single-page applications.
- **MEAN** is an acronym for **MongoDB**, **ExpressJS**, **AngularJS**, and **Node.js**.

What is LAMP ?

- **L**inux
- **A**pache
- **M**ySQL
- **P**HP
- **LAMP** stack is a popular open-source web platform commonly used to run dynamic websites and servers.
- It includes **Linux, Apache, MySQL, and PHP/Python/Perl**, and is considered by many the platform of choice for the development and deployment of high-performance web applications which require a solid and reliable foundation.

ANURAG UNIVERSITY

Problems with LAMP?

- Apache is not the fastest web server around
- It’s hard to write good-to-read, reusable and fast PHP code
- Frontend works with other languages than the backend
- Too many conversions (XML to PHPto HTML, model to SQL)
- There is no separation between server-side and client-side development

ANURAG UNIVERSITY

Requirements for a modern web?

Requirements for a Modern Web

- Customers want fast websites / fast response times
- No page reloads
- Enterprises want to go virtual
  - One box + several virtual images ⇒ Shared hardware
  - System with minimal memory footprint / overhead needed
- As many concurrent requests as possible
- Only load resources when needed (conditional loading)
- Mobile / Responsive UIs

ANURAG UNIVERSITY

📌 What is MEAN Stack?

MEAN Stack is a full-stack JavaScript solution that helps you build fast, robust, and maintainable production web applications using:

- MongoDB
- Express
- AngularJS
- Node.js

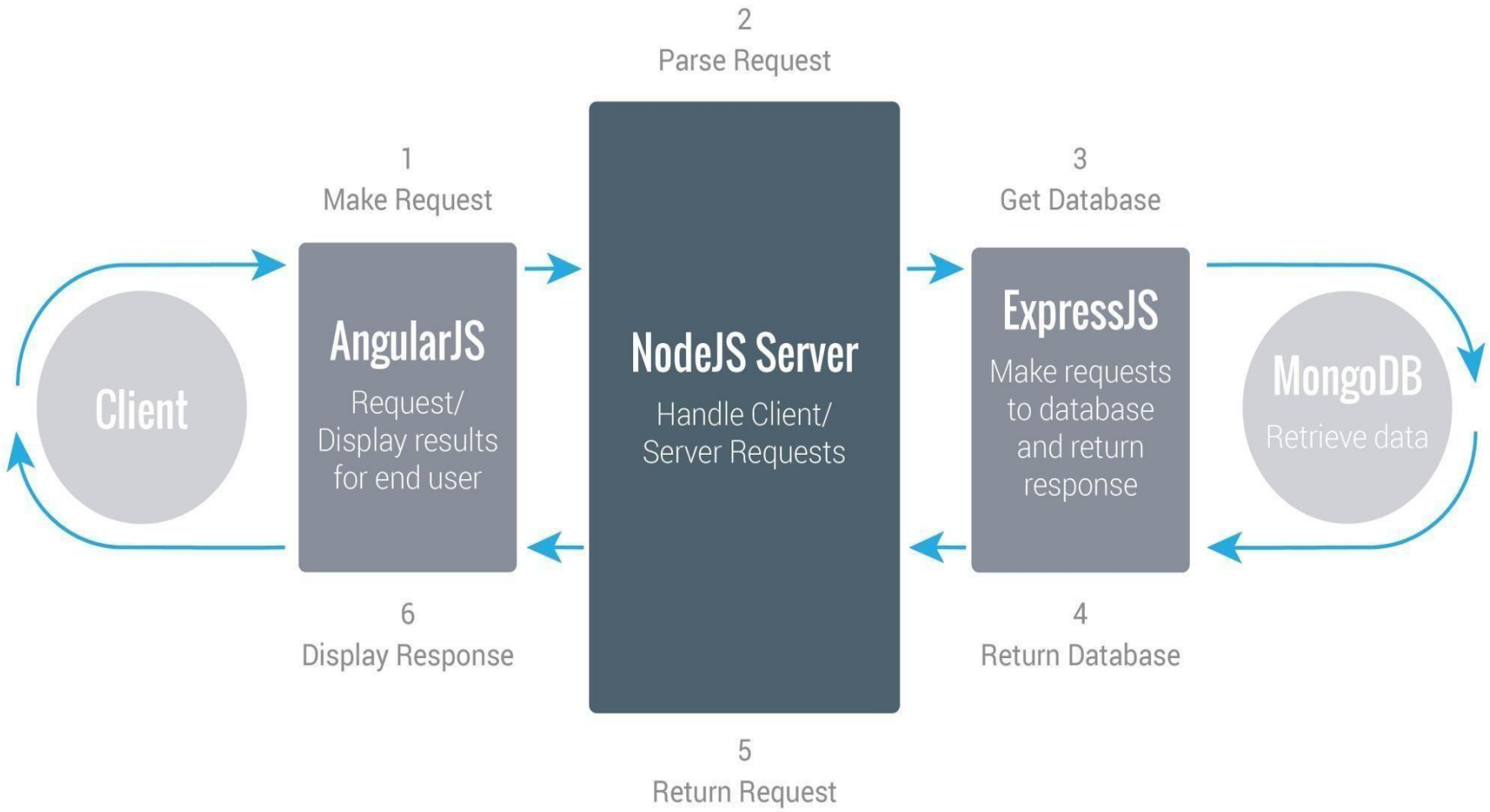
ANURAG UNIVERSITY

✅ Advantages of MEAN Stack

- 💯 **100% free, 100% Open Source**
- 💻 **100% JavaScript** (along with JSON and HTML)
- 🌐 **100% Web Standards**
- 🔁 **Consistent models** from backend to frontend
- 🧩 **Single language across the stack**:
  - JavaScript (language of the web)
  - JSON (data format)
  - No database format conversion needed
- 🚀 **Great JavaScript framework** (better than jQuery)
- 🧱 **Frontend-first development**
- 🪶 **Low memory footprint / overhead**

ANURAG UNIVERSITY

Processing model



ANURAG UNIVERSITY

MongoDB

MongoDB is a **cross-platform, document-oriented** database classified as a **NoSQL** database.

- It **avoids traditional table-based relational structures**
- Stores **JSON-like documents**
- Uses **dynamic schemas**

ANURAG UNIVERSITY

What is MongoDB

- Developed by software company **10gen** (now **MongoDB Inc.**)
- A **fast, schemaless NoSQL** database written in **C++**
- **Document-Oriented Storage**
  - JSON-style documents with dynamic schemas
- **Full Index Support**
  - Index any attribute
  - Supports replication and high availability
- **Auto-Sharding**
  - Enables horizontal scaling without sacrificing functionality

ANURAG UNIVERSITY

MongoDB	RDBMS
Collection	Table
Document	Row
Index	Index
Embedded Document	Join
Reference	Foreign Key

ANURAG UNIVERSITY

Example

```
>db.mycol.insert(
{ _id: ObjectId(7df78ad8902c),
title: 'MongoDB Overview',
description: 'MongoDB is no sql database', by: 'tutorials point',
url: 'http://www.tutorialspoint. com',
tags: ['mongodb','database', 'NoSQL'], likes: 100
})
```

ANURAG UNIVERSITY

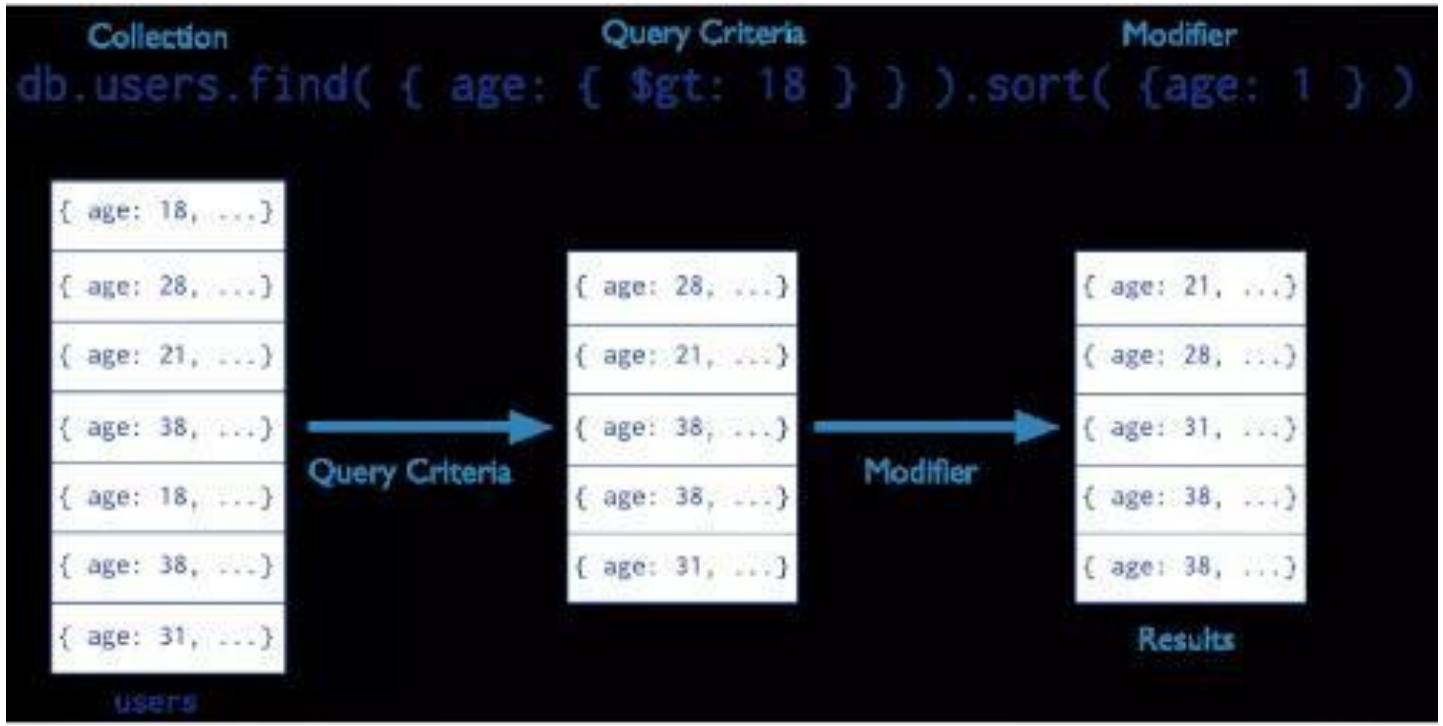
MongoDB Document

ANURAG UNIVERSITY

MongoDB Collection

ANURAG UNIVERSITY

MongoDB – Query a database



ANURAG UNIVERSITY

Advantages and Disadvantages

Advantages:

- Lightning fast
- Auto sharding
- Replication is very easy
- You can perform rich queries, can create on the fly indexes with a single command

Disadvantages:

- Very unreliable
- Indexes take up a lot of RAM
- B-tree indexes

ANURAG UNIVERSITY

Express JS

Express is a **minimal** and **flexible** node.js web application framework, providing a **robust** set of features for building **single, multi-page**, and **hybrid** web applications.

- **What is Express?**:
  - Node.js-based web framework
  - Based on connect middleware
  - Makes usage of Node.js even easier
  - Easy to implement REST API
  - Easy to implement session management
  - Supports several template rendering engines (Jade, EJS)
  - Supports partials -> so you can split your HTML in fragments
  - Asynchronous
  - Implements MVC pattern

- **What is Express? (Cont..)**:
  - Allows to setup middlewares to respond to HTTP requests
  - Defines a routing table which is used to perform different action based on HTTP method and URL
  - Allows to dynamically render HTML pages based on passing arguments to templates

ANURAG UNIVERSITY

Example

```
var express = require('express');
var app = express();
app.get('/', function(req, res) {
    res.send('Hello World');
});
app.listen(3000, function() {
    console.log('Example app listening on port 3000');
});
```

ANURAG UNIVERSITY

Advantages and Disadvantages

- **Advantages**:
  - Regardless of complexity, there should be very few roadblocks if you know JavaScript well
  - Supports concurrency well
  - Fast and the performance is comparable with Golang micro frameworks and Elixir's Phoenix
- **Disadvantages**:
  - There is no built-in error handling methods

ANURAG UNIVERSITY

What is Angular ?

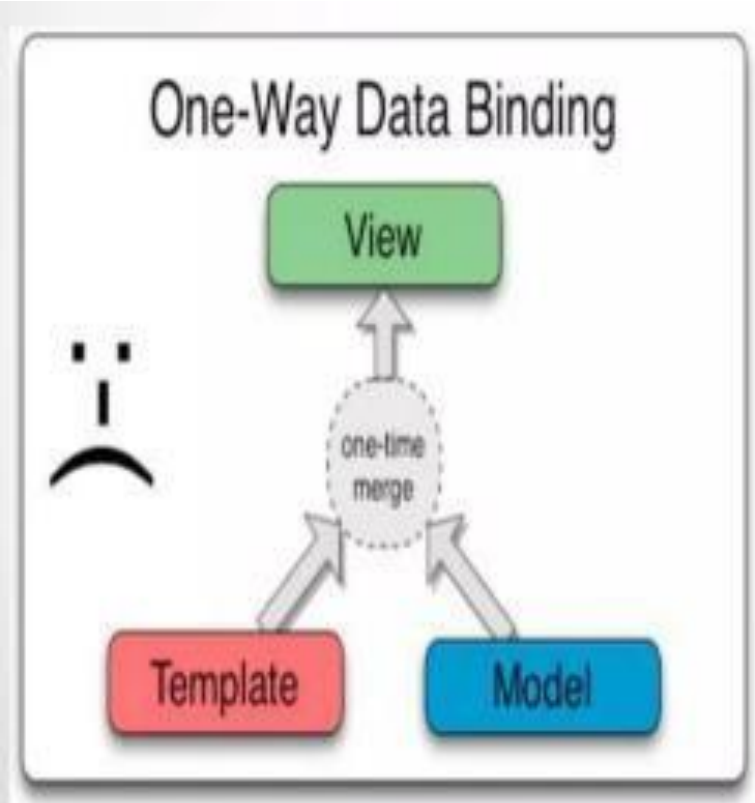
- AngularJS is an open-source JavaScript framework, maintained by Google, that assists with running single-page applications
- Its goal is to augment browser-based applications with model-view-controller (MVC) capability, in an effort to make both development and testing easier
- JavaScript framework developed by Google
- Based on the model-view pattern (Client-side), MVC/MVVM, Bi-directional Data Binding
- In declarative programming (focus on what—not the how!), Directives are integrated in HTML directly
- DOM manipulations are completely hidden, Great for front-end development, SPA (Single Page Applications), and mobile Apps

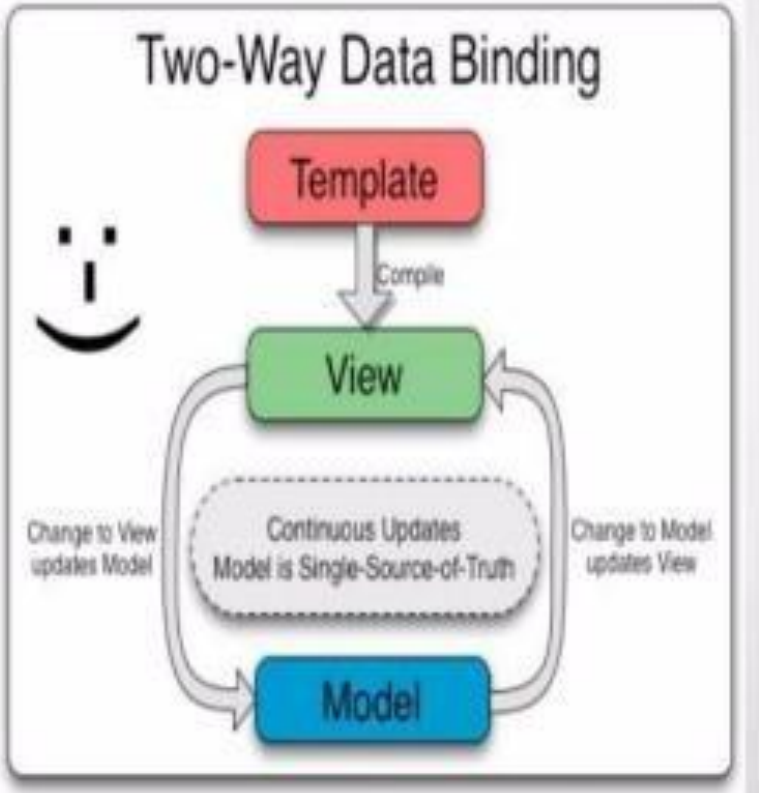
ANURAG UNIVERSITY

AngularJS Features

- Very modular and extensible
- Makes testing an ease
- Great browser support (>IE8)
- Well documented
- Two Way Data-binding:
  - **One-Way Data Binding**: Template updates Model one-time or on-merge
  - **Two-Way Data Binding**: Template and View continuously update; Model is Single-Source-of-Truth; Changes to View update Model and vice versa

Two Way Data-binding





ANURAG UNIVERSITY

Angular Directives

- `ng-app`: Declares an element as a root element of the application allowing behavior to be modified through custom HTML tags
- `ng-bind`: Automatically changes the text of an HTML element to the value of a given expression
- `ng-model`: Similar to `ng-bind`, but allows two-way data binding between the view and the scope
- `ng-controller`: Specifies a JavaScript controller class that evaluates HTML expressions
- `ng-repeat`: Instantiate an element once per item from a collection
- `ng-show & ng-hide`: Conditionally show or hide an element, depending on the value of a Boolean expression
- `ng-switch`: Conditionally instantiate one template from a set of choices, depending on the value of a selection expression
- `ng-view`: The base directive responsible for handling routes that resolve JSON before rendering templates driven by specified controllers

ANURAG UNIVERSITY

Advantages and Disadvantages

- **Advantages**:
  - Fast development
  - Makes developing SPA easy
  - Awesome performance
  - Make app scalable
- **Disadvantages**:
  - Good for IO-driven apps only (not games)

ANURAG UNIVERSITY

Node JS

- **Node.js** is a platform built on **Chrome's JavaScript runtime** for easily building **fast, scalable network applications**.
- It uses an **event-driven**, **non-blocking I/O model** that makes it:
  - **Lightweight**
  - **Efficient**
  - Ideal for **data-intensive, real-time applications** that run across **distributed devices**.

ANURAG UNIVERSITY

What is Node.js?

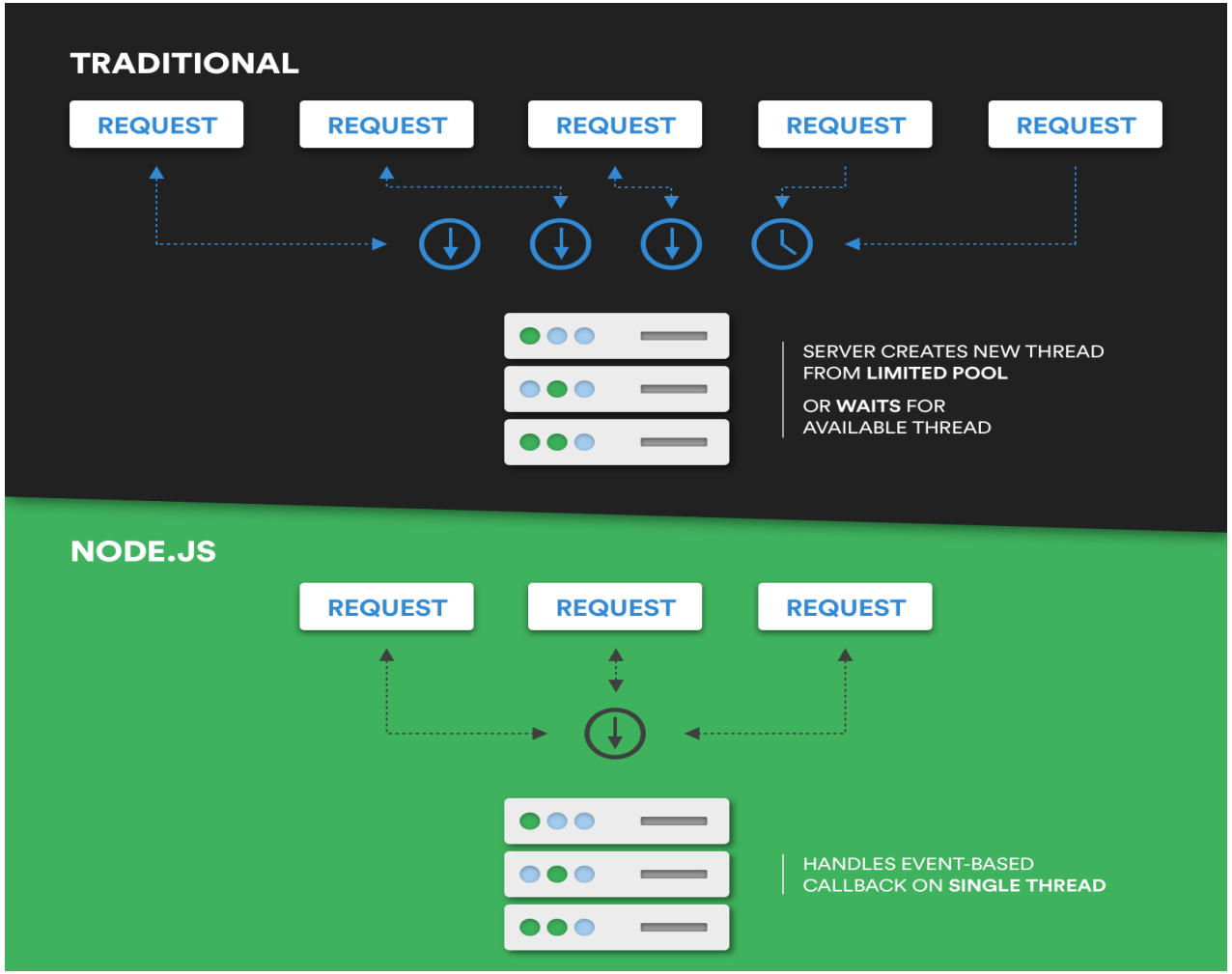
- Written in **C/C++** (can use C libraries)
- Based on **ECMAScript 5**
- Framework for **asynchronous I/O applications**
- **Single-threaded** (no concurrency bugs/deadlocks)
- One Node process = one CPU core

Why Node.js?

- Handles **10K+ concurrent connections**
- **No concurrency issues**
- Low **CPU/Memory overhead**
- **Scalable** with clusters
- **Very fast** (mostly C code)
- Starts server in under **5 minutes**
- REST API setup possible in **<5 minutes**
- Not a web framework (just runtime environment)

ANURAG UNIVERSITY

Blocking I/O vs Non-Blocking I/O



ANURAG UNIVERSITY

Example

```
var http = requrie('http');
http.createServer(function(req, res)
{
    res.writeHead(200, {'Content-Type':'text/plain'});
    res.end('Hello World');
}).listen(80);
console.log('Server listening on port 80');
```

ANURAG UNIVERSITY

Example

```
var http = require('http'); // Imports the built-in Node.js http module

// Creates an HTTP server
http.createServer(function(req, res) {
    // Sets the HTTP header with a status code (200 OK) and content type (plain text)
    res.writeHead(200, {'Content-Type': 'text/plain'});

    // Sends the response body "Hello World" and ends the response
    res.end('Hello World');

// The server listens for incoming requests on port 80
}).listen(80);

// Logs a message to the console once the server starts listening
console.log('Server listening on port 80');
```

ANURAG UNIVERSITY

Advantages:

- **Node.js is fast:** Refers to its non-blocking, event-driven architecture which makes it very efficient for I/O-bound operations.
- **The ever-growing NPM:** NPM (Node Package Manager) is the world's largest software registry, offering a vast ecosystem of open-source libraries and tools that accelerate development.
- **Real-time web apps:** Node.js is well-suited for real-time applications like chat apps, online games, and collaboration tools due to its ability to handle many concurrent connections efficiently.
- **Productivity:** Using JavaScript across the entire stack (frontend with Angular/React/Vue, backend with Node.js) can lead to increased developer productivity as context switching between languages is minimized.

Disadvantages:

- JavaScript's semantics and culture

Disadvantages:

- **JavaScript's semantics and culture:** This point is a bit broad, but it could refer to:
  - **Callback Hell/Asynchronous Complexity:** While modern JavaScript (ES6+) has improved with Promises and Async/Await, dealing with asynchronous operations can still be challenging for some developers.
  - **Single-threaded Nature:** While Node.js uses an event loop to achieve non-blocking I/O, CPU-bound tasks can still block the single thread, requiring careful handling (e.g., using worker threads).
  - **Maturity (compared to older languages):** Historically, Node.js was newer than some established backend technologies, though it has matured significantly.
  - **Lack of strong typing (historically):** Pure JavaScript is dynamically typed, which can lead to runtime errors. This is often mitigated by using TypeScript in larger projects.

Disadvantages:
• JavaScript's semantics and culture

ANURAG UNIVERSITY

Disadvantages of MEAN Stack

- There are still no general JS coding guidelines.
- MongoDB is not as robust as SQL server
  - This security is what they sacrifice to gain speed
- Once you have created the first site with this technology, it's hard to go back to the old approach

Conclusion

- MEAN is a full-stack, JavaScript web application framework. If you require a fast, easy, simple way to create a modern, responsive, dynamic website, then MEAN would be a great solution

Designing mean stack architecture

ANURAG UNIVERSITY

A common MEAN stack architecture

- A common way to architect a MEAN stack application is to have a representational state transfer (REST) API feeding a single-page application (SPA).
- The API is typically built with MongoDB, Express, and Node.js, with the SPA being built in Angular.

Figure 2.1. A common approach to MEAN stack architecture, using MongoDB, Express, and Node.js to build a REST API that feeds JSON data to an Angular SPA run in the browser

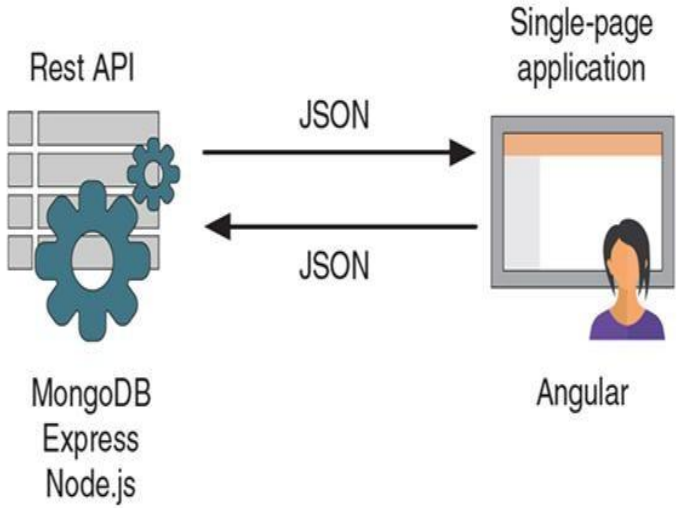
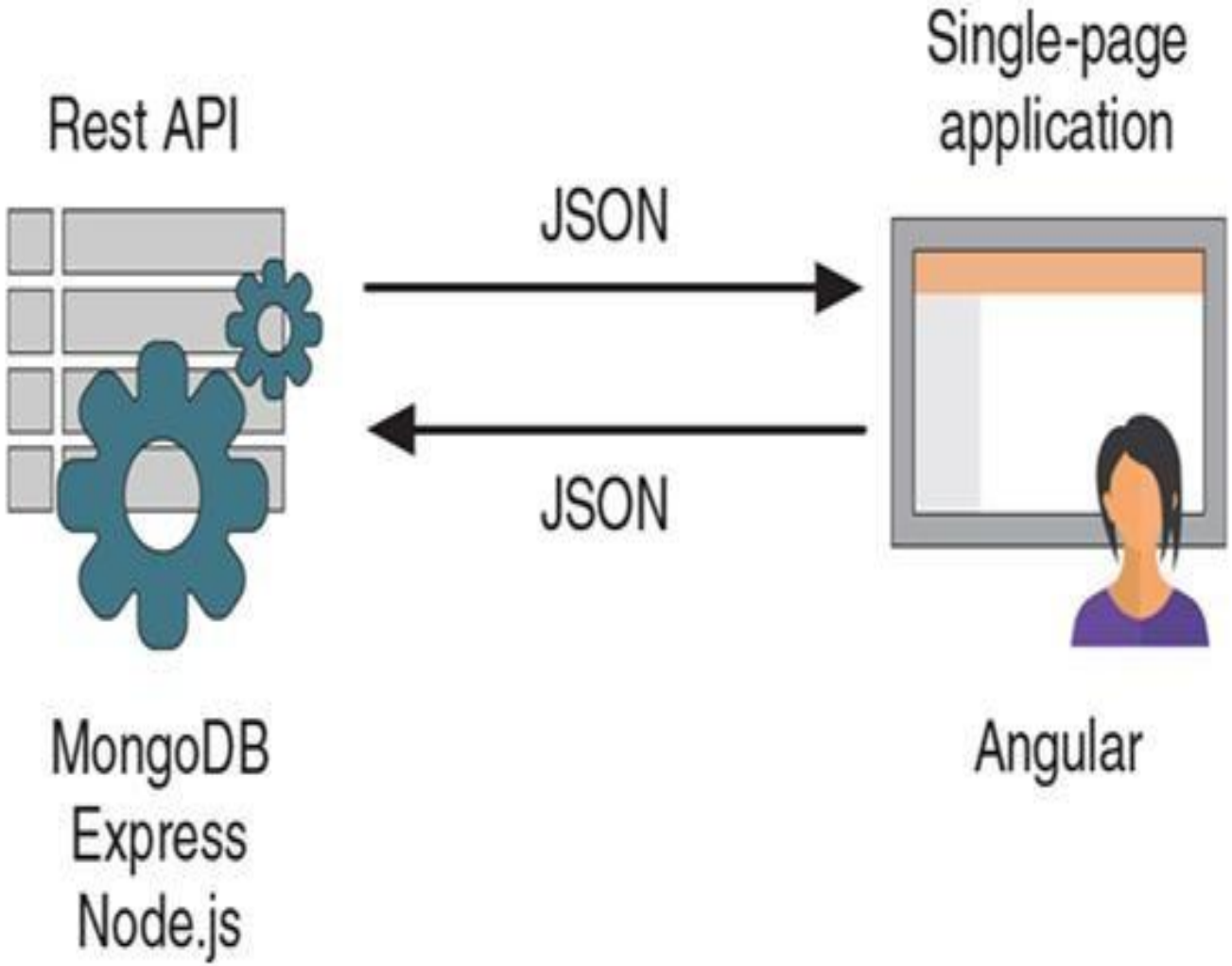


Figure 2.1. A common approach to MEAN stack architecture, using MongoDB, Express, and Node.js to build a REST API that feeds JSON data to an Angular SPA run in the browser



ANURAG UNIVERSITY

REST API

- *REST* stands for *REpresentational State Transfer*, which is an architectural style rather than a strict protocol.
- REST is stateless; it has no idea of any current user state or history.
- In the case of the web, an API is normally a set of URLs that respond with data when called in the correct manner with the correct information.
- A REST API is a stateless interface to your application.

ANURAG UNIVERSITY

REST API

- In the case of the MEAN stack, the REST API is used to create a stateless interface to your database, enabling a way for other applications, such as an Angular SPA, to work with the data. In other words, you create a collection of structured URLs that return specific data when called.
  - ❖Angular is designed with a focus on building SPAs, pulling in data from a REST API as well as pushing it back.
  - ❖MongoDB, Express, and Node.js are also extremely capable when it comes to building an API, using JSON all the way through the stack, including the database itself.

ANURAG UNIVERSITY

Looking beyond SPAs

🧠 Main Idea

Angular SPAs are excellent for creating fast, responsive web apps. But like a sports car, they're not ideal for every situation. Sometimes a more traditional server-rendered approach is better.

⚠️ Drawbacks of SPAs

1. **Hard to Crawl**
   - Search engines struggle to index JavaScript-heavy sites.
   - Social media previews (e.g., Facebook, LinkedIn) don't render JS, so shared links may appear blank.
   - Solutions like server-side rendering (SSR) or headless browsers (e.g., PhantomJS) are available but complex and fragile .

ANURAG UNIVERSITY

Looking beyond SPAs

2. **Analytics and Browser History**
    - SPAs break conventional analytics because there are no full page loads after the initial one.
    - Requires additional setup using the HTML5 History API and integrations to track usage properly .
3. **Slow Initial Load**
    - SPAs need to download the full app and framework before displaying anything, leading to perceived slowness.
    - This can result in users bouncing before any content appears, especially problematic for content-driven sites like blogs .

ANURAG UNIVERSITY

Looking beyond SPAs

✅ When SPAs Make Sense

- Applications with high interactivity, such as admin dashboards, internal tools, or apps where users stay logged in for long sessions.
- Projects where SEO is not a primary concern.

❌ When SPAs May Be Overkill

- Content-heavy websites needing fast first loads and good SEO (e.g., blogs, landing pages).
- Projects where users quickly consume content and bounce.

🔁 Alternatives and Hybrids

- Use Angular for admin areas (as SPAs).
- Use Express with server-rendered HTML for public-facing parts.
- Combine both in a **hybrid architecture**, with a single REST API serving multiple front-ends

ANURAG UNIVERSITY

Designing a flexible MEAN architecture

- If Angular is like having a Porsche, the rest of the stack is like also having an Audi RS6 in the garage. A lot of people may be focusing on your sports car out front and not give a second glance to the estate car in your garage. But if you do go into the garage and have a poke around, you'll find that there's a Lamborghini V10 engine under the hood. There's a lot more to that estate car than some people think.
  - MongoDB can store and stream binary information.
  - Node.js is particularly good for real-time connections using web sockets.
  - Express is a web application framework with templating, routing, and session management built in.

ANURAG UNIVERSITY

Requirements for a blog engine

A blog engine typically has two sides:

1. Public-Facing Blog

- Users: Anonymous visitors
- Needs:
  - **Fast loading**
  - **SEO optimized**
  - Low interaction
  - Content should be easy to share
  - Short visits

2. Admin Interface

- Users: Blog authors/editors
- Needs:
  - **High interactivity**
  - Responsive UI
  - Secure login
  - Long sessions

⚠️ These two sides have **conflicting needs**, which means using a **single architecture (SPA)** might not suit both.

Requirements for a blog engine

Blog entries

Characteristics

- Content-rich
- Low interaction
- Fast first load
- Short user duration
- Public and shareable

Admin interface

Characteristics

- Feature-rich
- High interaction
- Fast response to actions
- Long user duration
- Private

Figure 2.2 Conflicting characteristics of the two sides of a blog engine: the public-facing blog entries and the private admin interface

A blog engine architecture

📌 Admin Interface

- Built as an **Angular SPA**
- Communicates with a **REST API** built using:
  - **Node.js** (runtime)
  - **Express** (framework)
  - **MongoDB** (database)
- Angular fetches/post data as JSON from the backend

This is a **classic MEAN setup**: Angular ⟷ REST API ⟷ MongoDB

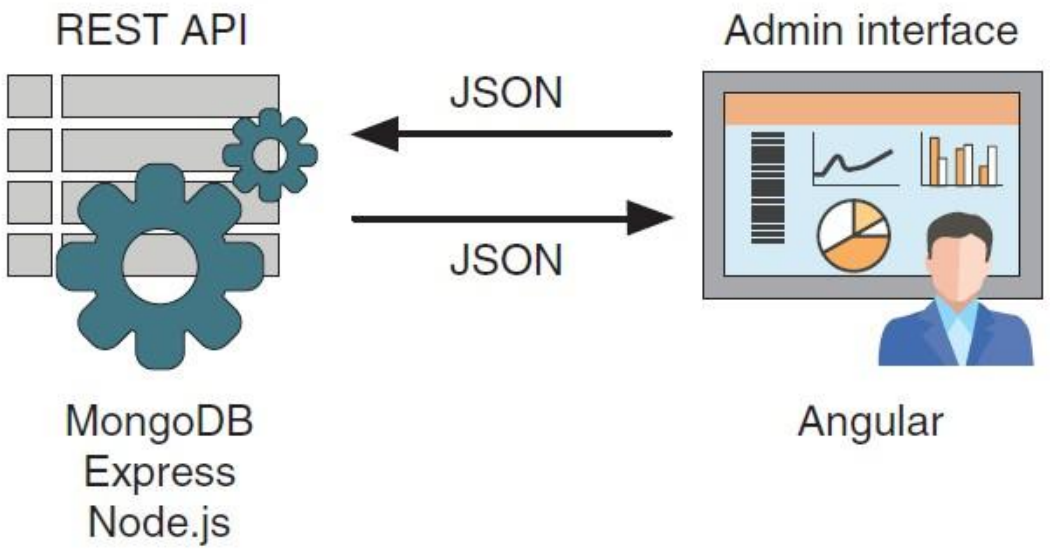


Figure 2.3 A familiar sight: the admin interface is an Angular SPA making use of a REST API built with MongoDB, Express, and Node.js.

A blog engine architecture

📌 Public Blog Interface

- Deliver fully rendered HTML directly to the browser
- Use **Express** and **server-side rendering** (via Pug or another template engine)
- Fetch data from the same REST API

Why? It's **faster**, **SEO-friendly**, and avoids SPA drawbacks (like delayed rendering or poor link previews).

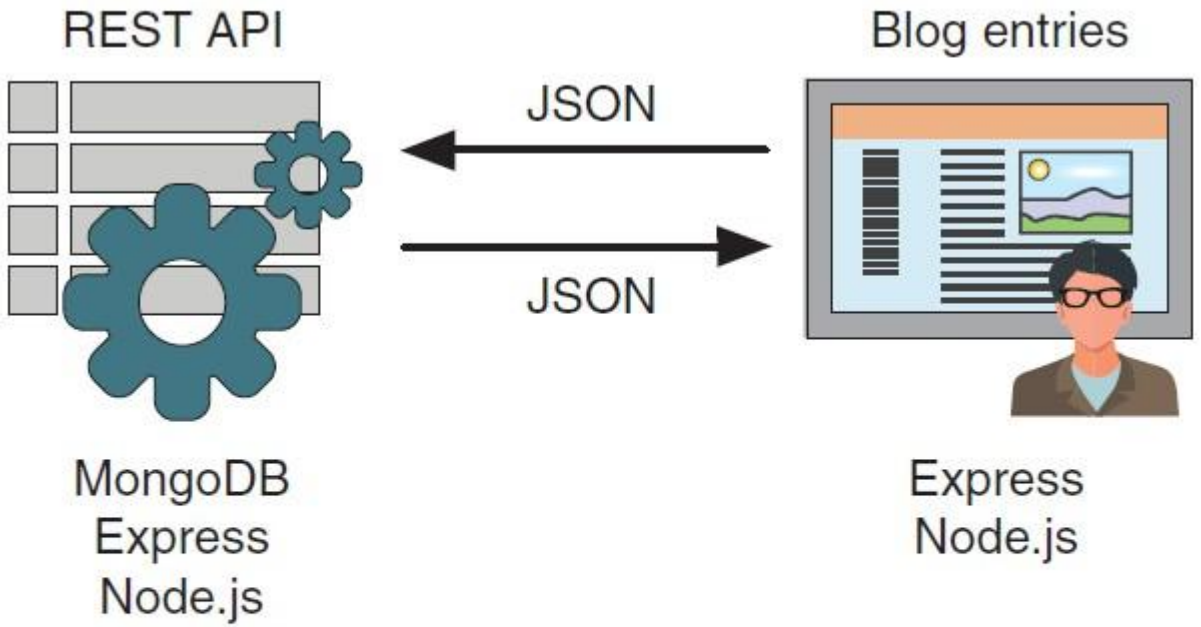


Figure 2.4 An architecture for delivering HTML directly from the server: an Express and Node.js application at the front, interacting with a REST API built in MongoDB, Express, and Node.js

ANURAG UNIVERSITY

A blog engine architecture

Blog Entries: Using More Of The Stack

- Add **Angular** components to the public site for:
  - Search boxes with type-ahead
  - Real-time comment updates
- Use **MongoDB** for session storage if visitors can log in (e.g., to comment)

This hybrid approach mixes the **best of server-side and client-side rendering** — performance + interactivity.

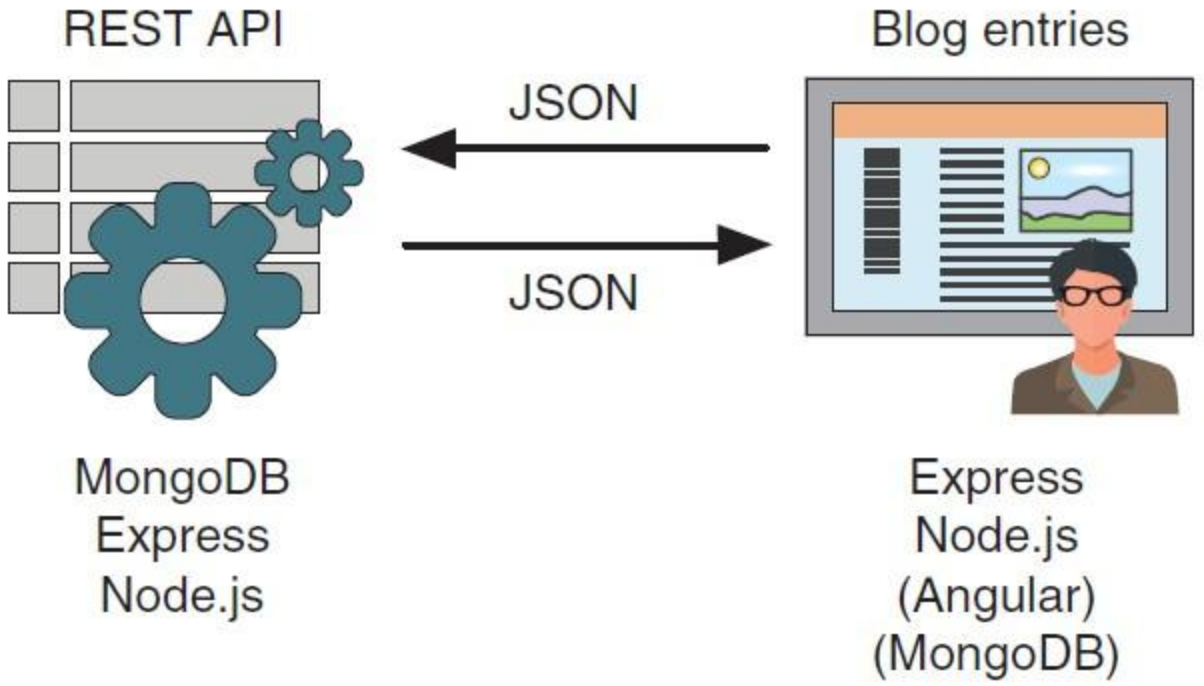
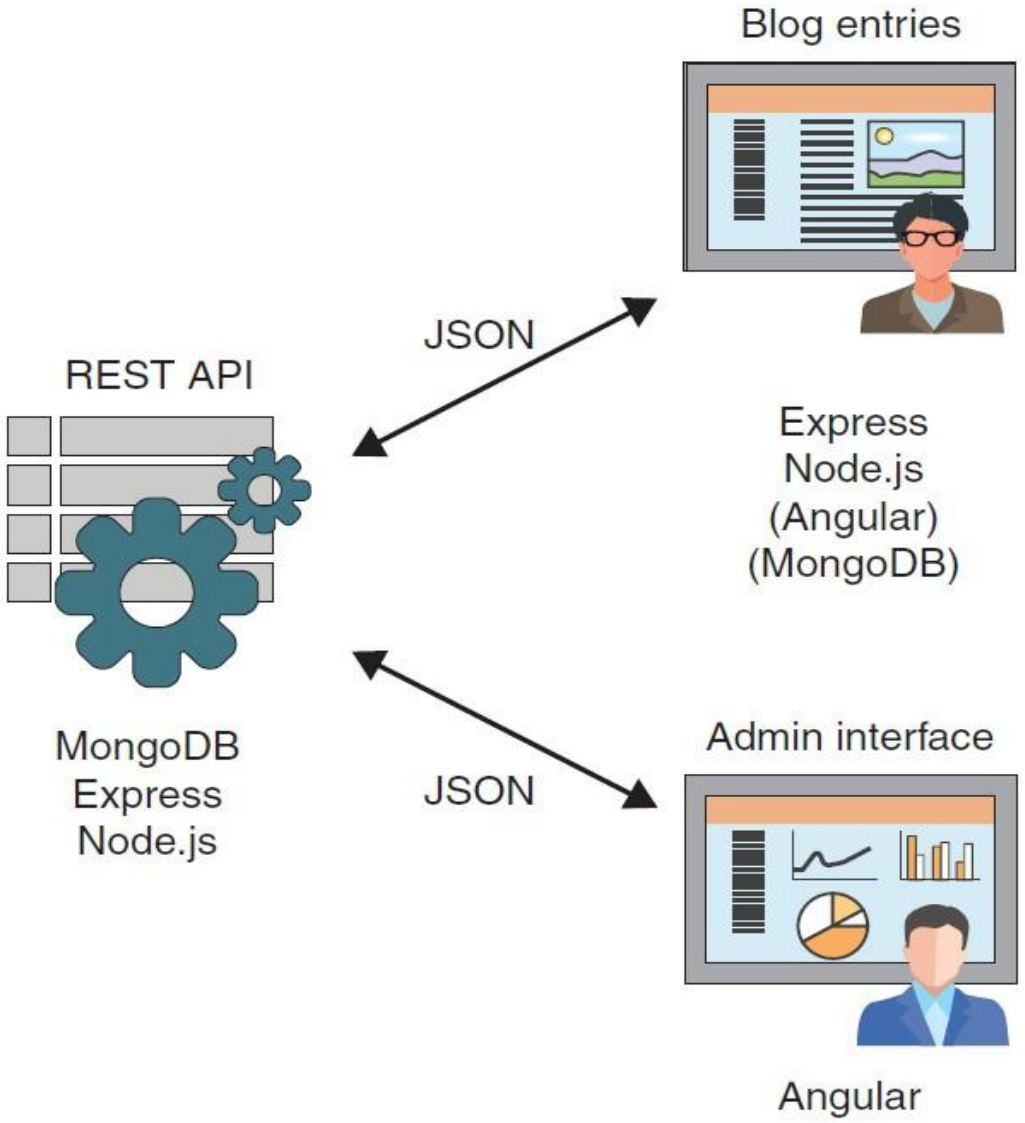


Figure 2.5 Adding the options of using Angular and MongoDB as part of the public-facing aspect of the blog engine, serving the blog entries to visitors

ANURAG UNIVERSITY

A Blog Engine Architecture



Hybrid Architecture

A **single shared REST API** powers:

- An Angular SPA for admin (private)
- An Express-rendered public site

Both use the same database via this **central API**.

Figure 2.6 A hybrid MEAN stack architecture: a single REST API feeding two separate user-facing applications, built using different parts of the MEAN stack to provide the most appropriate solution

ANURAG UNIVERSITY

Best practice: Building an internal API for a data layer

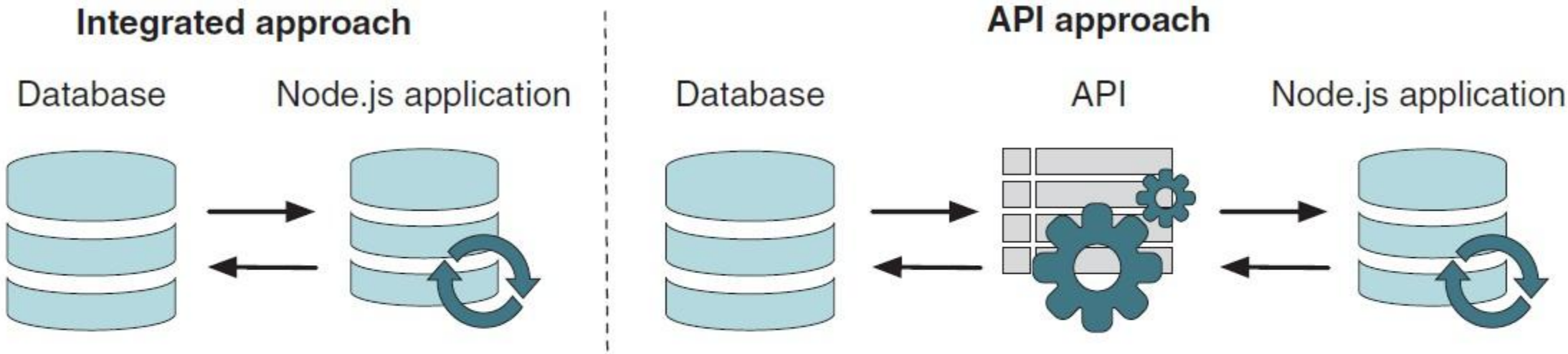


Figure 2.7 The short-term view of integrating data into your Node.js application. You can set up your Node.js application to talk directly to your database, or you can create an API that interacts with the database, and have your Node.js application talk only with the API.

ANURAG UNIVERSITY

Best practice: Building an internal API for a data layer

❌ The Wrong Way (Integrated Approach)

- Directly connecting Express routes to the database
- Tightly couples business logic with rendering logic
- Hard to reuse across clients (web, mobile, etc.)

✅ The Right Way (API Approach)

- **Create a REST API layer** (Express + Mongoose) that all clients can consume
- Clients can be:
  - Angular SPA
  - Express site
  - iOS/Android app
  - Command-line tools

> 💡 This API doesn't have to be public — it can be internal, secure, and private.

ANURAG UNIVERSITY

Best practice: Building an internal API for a data layer

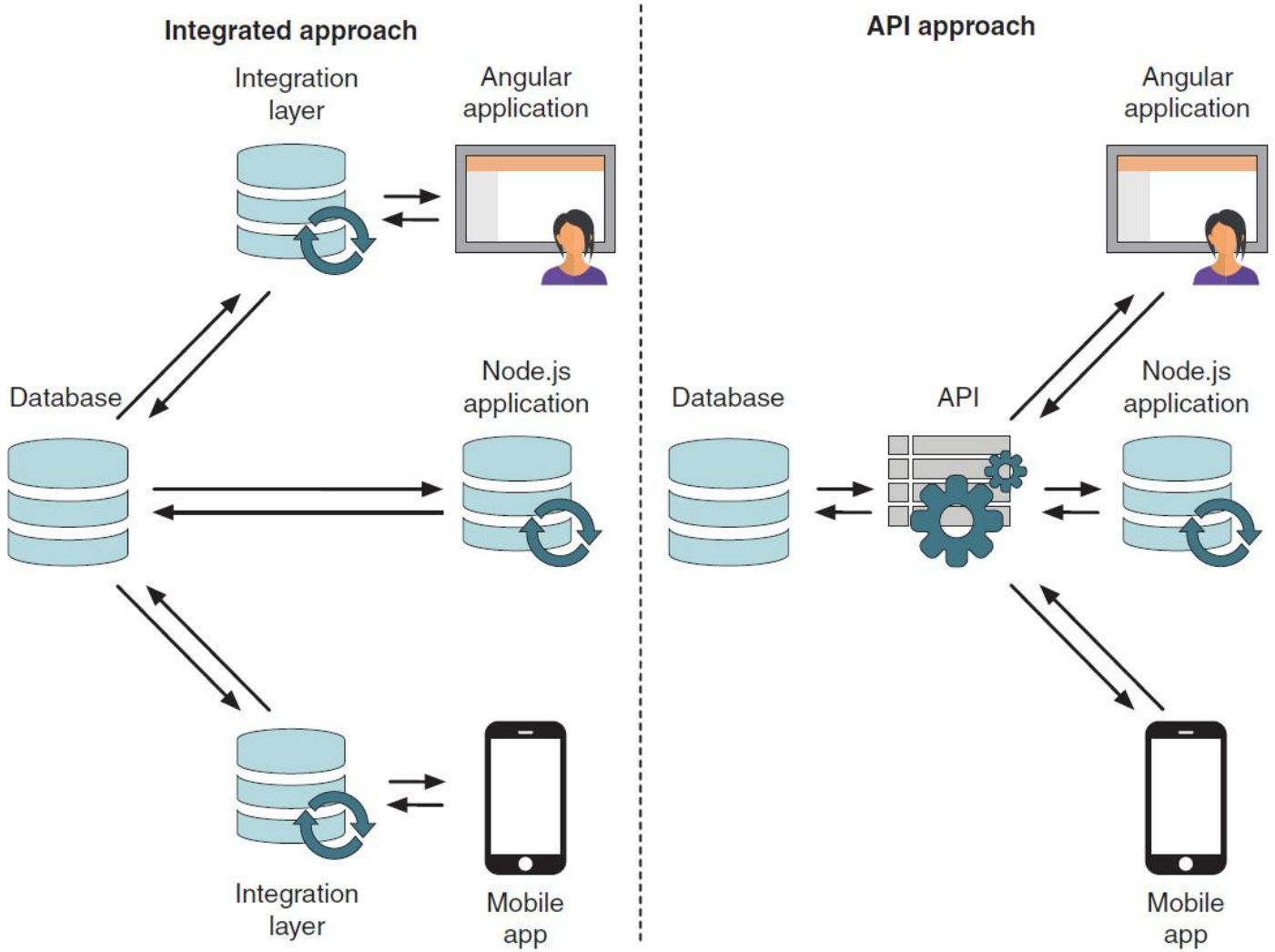


Figure 2.8 The long-term view of integrating data into your Node.js application and additional Angular and iOS applications. The integrated approach has become fragmented, whereas the API approach is simple and maintainable.

ANURAG UNIVERSITY

Planning a real application

🎯 Project Overview

Loc8r is a demo MEAN stack application designed to:

- List nearby places with WiFi
- Show facilities, ratings, opening hours, and maps
- Allow users to leave ratings and reviews
- Built step-by-step to explore all MEAN technologies

ANURAG UNIVERSITY

Planning a real application

✅ 1. Introduction

Before starting to code, proper planning is essential for building reliable and scalable web applications. This phase involves:

- Understanding **what to build**
- Identifying **key screens and features**
- Designing **how components interact**
- Choosing the right **architecture strategy**

🧩 Purpose

Begin with high-level planning:

- Identify **key screens/pages**
- Map **user journeys**
- Don't worry about details or data at this stage

ANURAG UNIVERSITY

Planning the Application at a High Level

Planning a real application

Key Screens Identified

1. **List View** – Shows nearby locations
2. **Detail View** – Info about a specific place
3. **Review Form** – Submit a review
4. **About Page** – App background/info

Screen Collections

Grouped logically:

- **Locations** (List → Details → Review)
- **Others** (About Us)

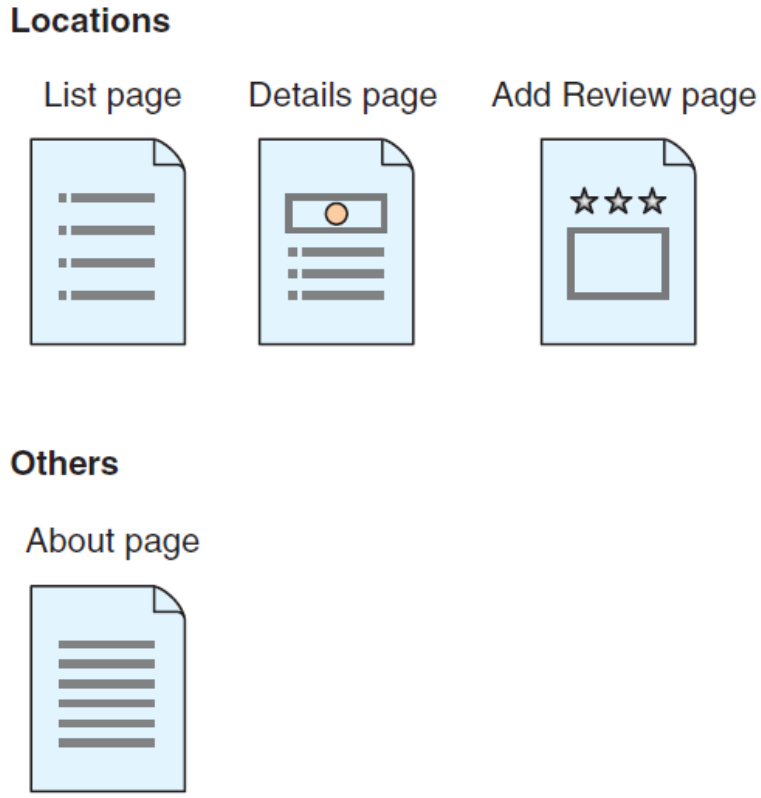


Figure 2.9 Collate the separate screens for your application into logical collections.

> Sketching these helps clarify flow and scope before architectural decisions are made.

ANURAG UNIVERSITY

Planning a real application

Architecting the Application

4. Start with the API

Why?

- The application will rely heavily on reading and writing data
- A solid API helps ensure data consistency and security

Build a **RESTful API** using:

- **Node.js** and **Express.js** for backend logic
- **MongoDB** as the database

Start with the API

- Core of the app: **REST API using Node.js, Express, and MongoDB**
- Handles **data access**, **CRUD operations**, and **business logic**

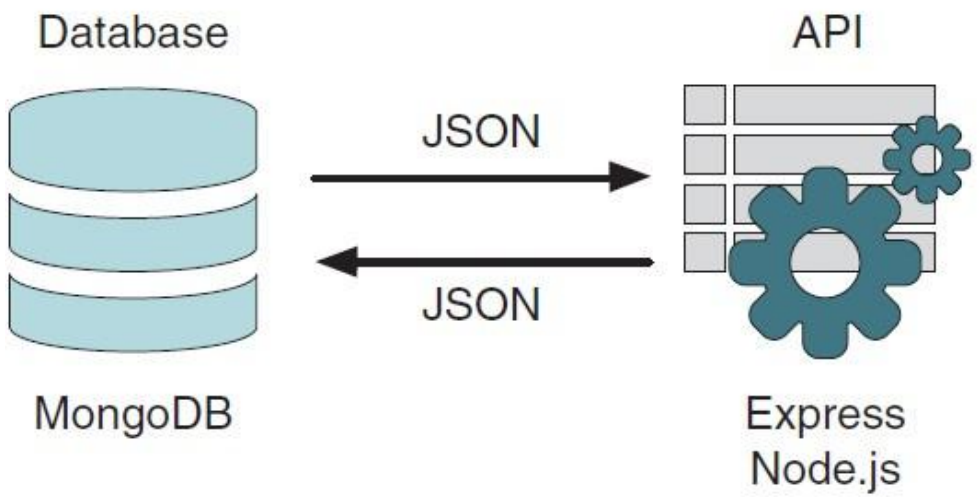


Figure 2.10 Start with the standard MEAN REST API, using MongoDB, Express, and Node.js.

ANURAG UNIVERSITY

Planning a real application

Architecting the Application

5. Application Architecture Options

There are three options:

Option 1: Node.js + Express

- Server-side rendering
- Simple architecture
- Suitable for small apps

Option 2: Node.js + Express + Angular

- Angular adds dynamic interactivity
- Better for responsive UIs

Option 3: Angular SPA

- Frontend is completely decoupled from backend
- Backend serves only the API
- Ideal for scalable apps

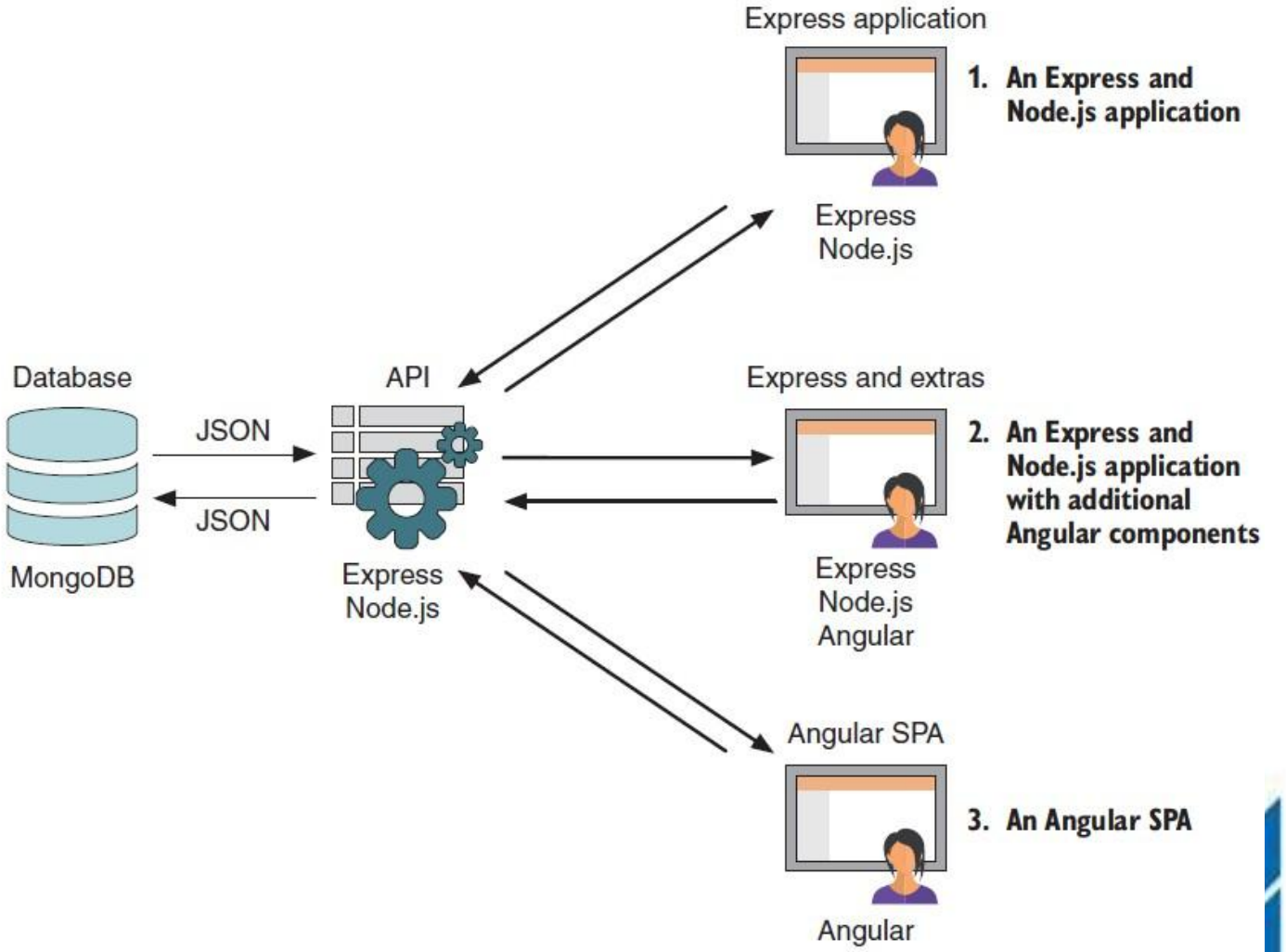


Figure 2.11 Three options for building the Loc8r application, ranging from a server-side Express and Node.js application to a full client-side Angular SPA

Planning a real application

📦 6. Wrapping in an Express Project

Two approaches:

1. **Modular/Separate Projects:**
   - API and Application Logic in separate codebases
   - Better for large-scale apps
2. **Single Express Project:**
   - All logic in one project folder
   - Easier to manage initially
   - Good for MVPs and small teams

📌 Organize code well to keep features modular.

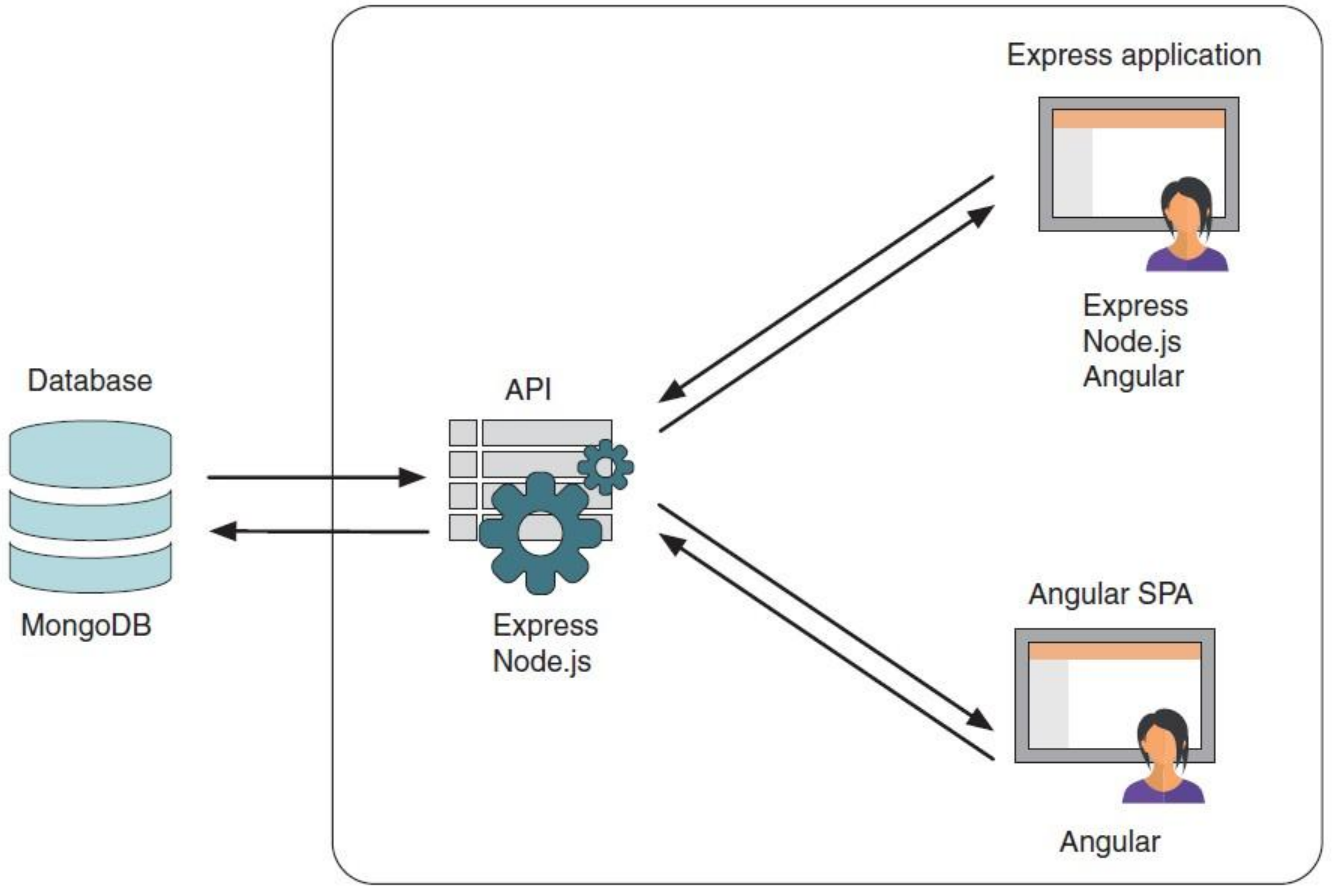


Figure 2.12 The architecture of the application with the API and application logic wrapped inside the same Express project

ANURAG UNIVERSITY

Planning a real application

Final Product Goal

- Responsive layout with Bootstrap
- Cross-device compatibility
- Functional:
  - Browse places
  - View maps/details
  - Submit reviews
  - Auth system

ANURAG UNIVERSITY

Breaking Development into Stages

🧠 Approach: Build + Learn

> Build the app step-by-step, each focusing on one part of the MEAN stack

Five Stages of Rapid Prototype Development

Stage 1: Build a Static Site

- Just HTML pages for:
  - Visual layout
  - User flow
- No data, no database, no interactivity

Stage 2: Design Data Model + Create DB

- Identify:
  - What data is needed
  - How objects relate
- Use **MongoDB** + **Mongoose**

ANURAG UNIVERSITY

Breaking Development into Stages

Stage 3: Build REST API

- API allows the frontend to interact with the DB
- Created with **Express**, **Node.js**, and **Mongoose**

Stage 4: Connect App to API

- Replace hard-coded content with **real data**
- Application becomes **data-driven**

Stage 5: Augment the App

- Add:
  - Input validation
  - Interactivity
  - Error handling
  - AngularJS widgets or components

ANURAG UNIVERSITY

Building Loc8r: 7 Steps

Breaking Development into Stages

Step	Goal	Tech Focus
1. Static Site	UX/UI structure	HTML, Bootstrap, Express views
2. Data Modeling	Schema + DB	MongoDB, Mongoose
3. REST API	Backend logic	Express, Node.js
4. Hook API	Link frontend to data	Express server
5. Add Angular	Add dynamic UI	AngularJS
6. Convert to SPA	Move logic to frontend	AngularJS SPA
7. Auth System	User login + secure APIs	All MEAN technologies

ANURAG UNIVERSITY

Breaking Development into Stages

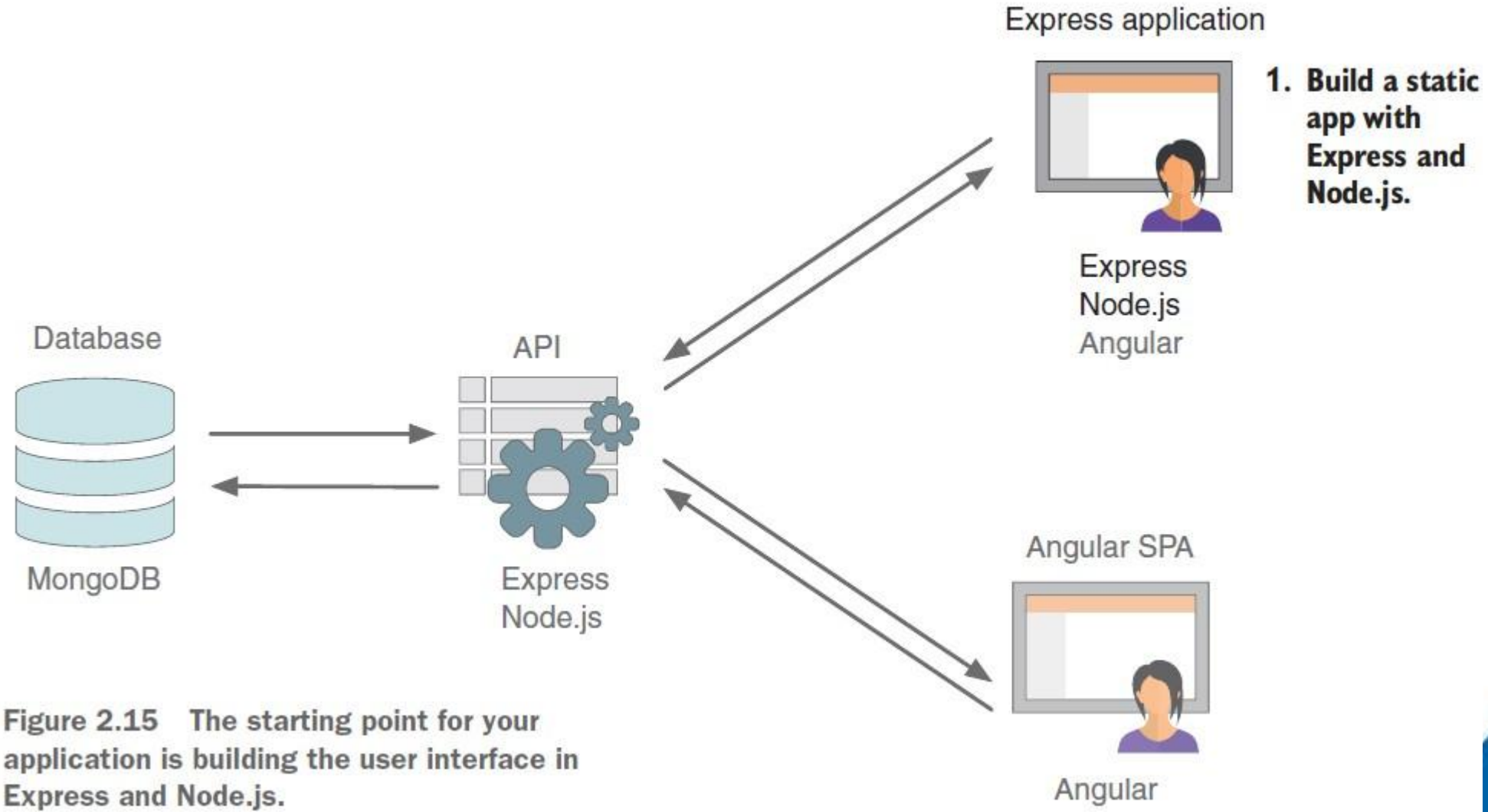


Figure 2.15 The starting point for your application is building the user interface in Express and Node.js.

Breaking Development into Stages

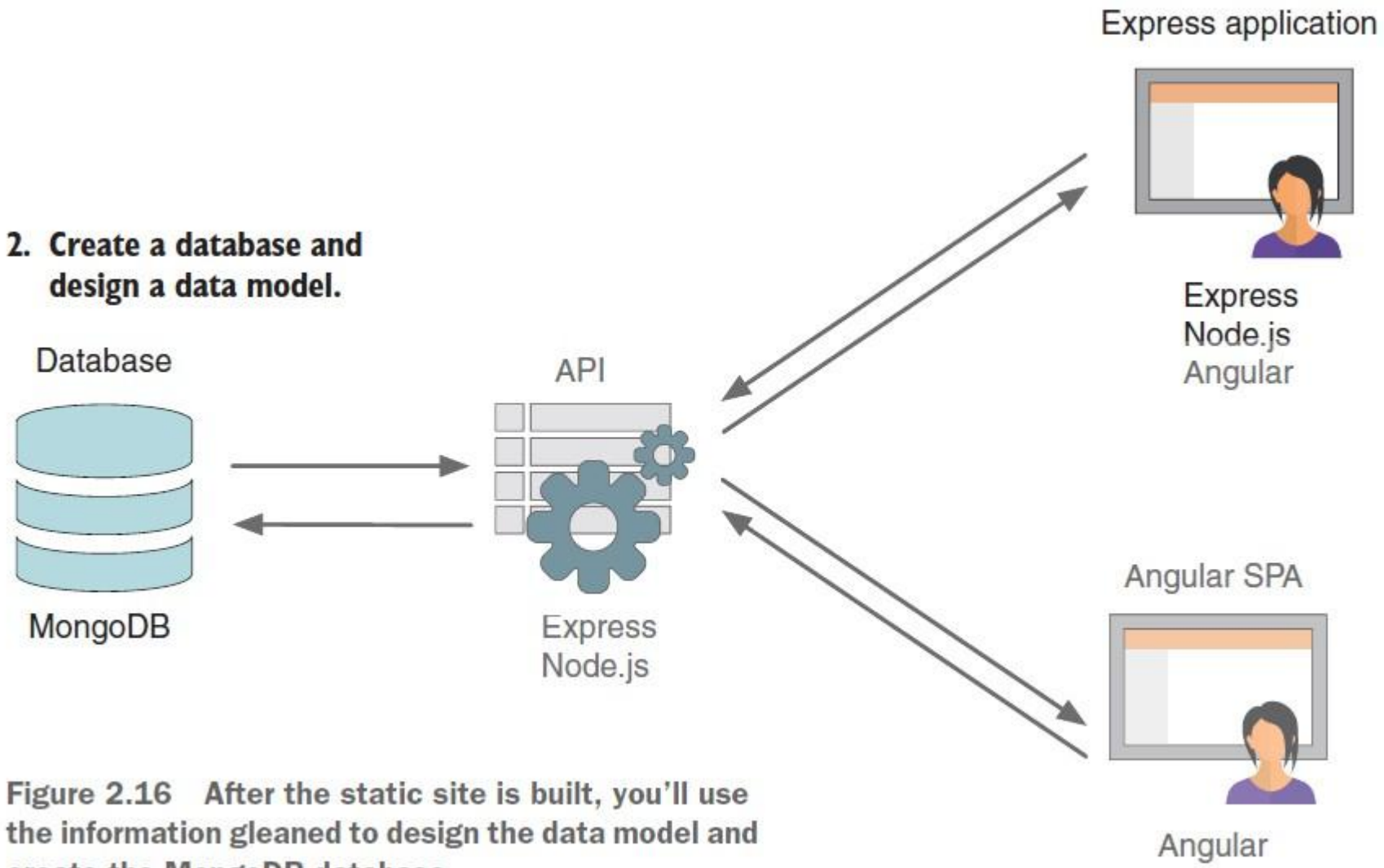


Figure 2.16 After the static site is built, you'll use the information gleaned to design the data model and create the MongoDB database.

Breaking Development into Stages

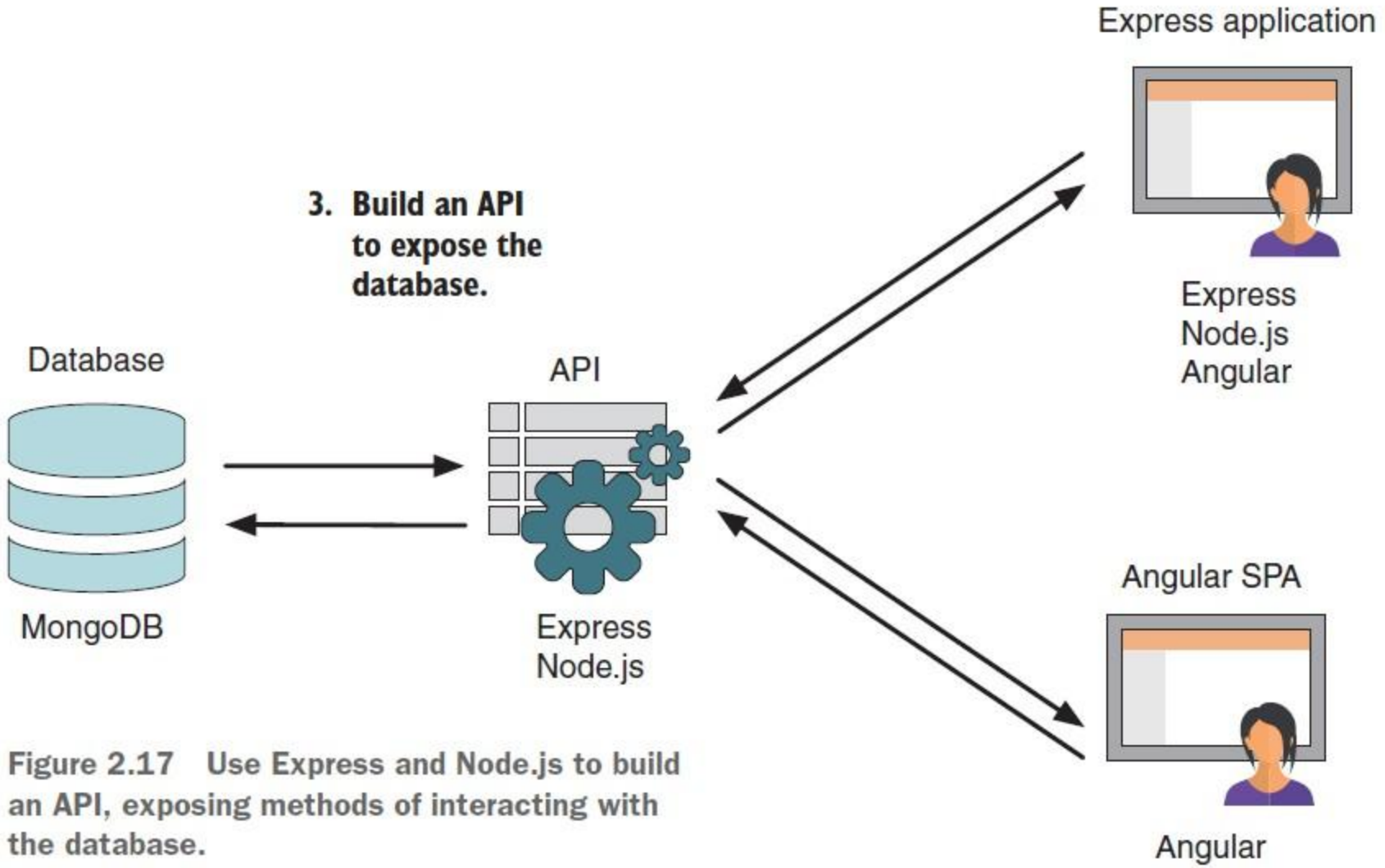


Figure 2.17 Use Express and Node.js to build an API, exposing methods of interacting with the database.

ANURAG UNIVERSITY

Breaking Development into Stages

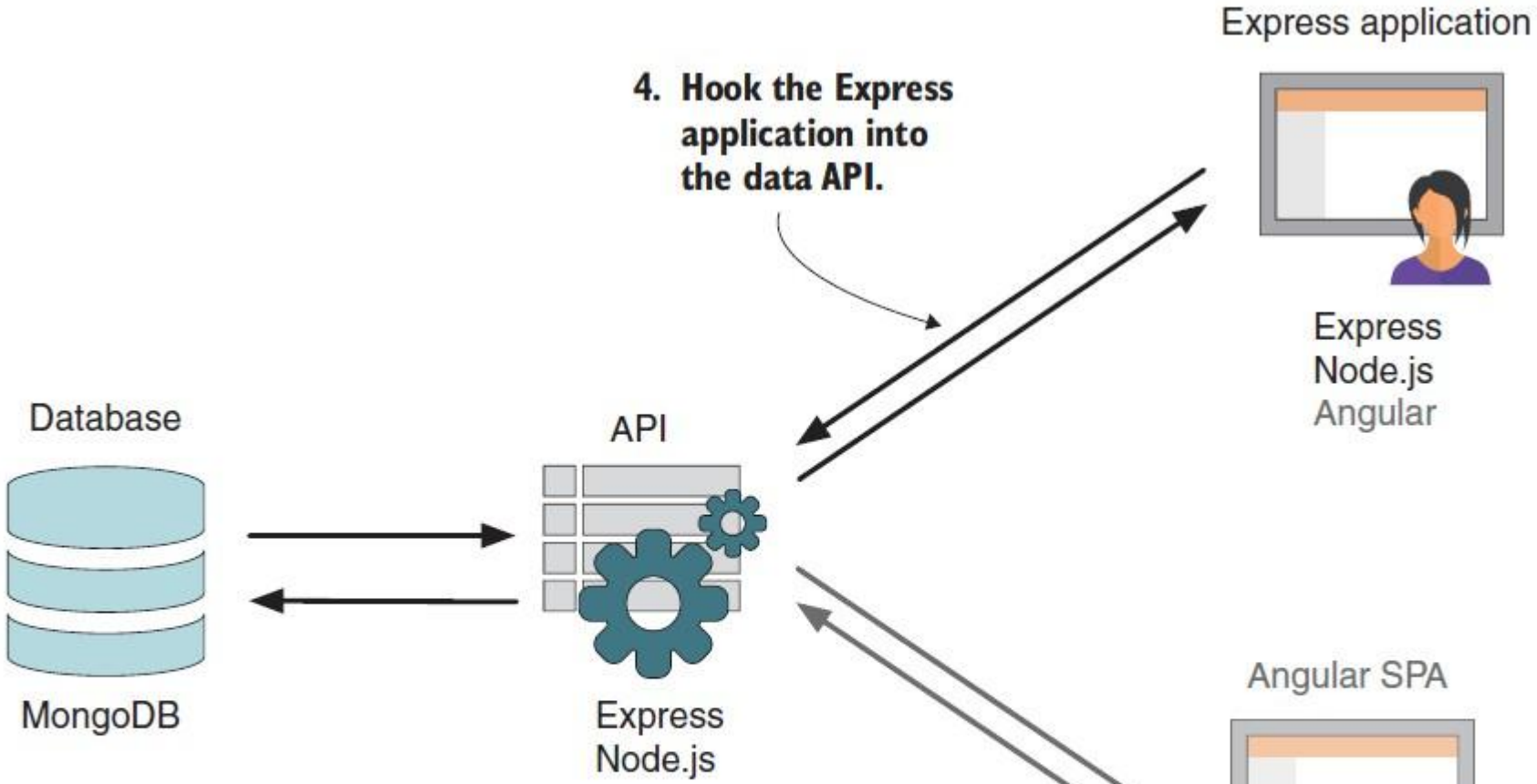


Figure 2.18 Update the static Express application by hooking it into the data API, allowing the application to be database-driven.

Breaking Development into Stages

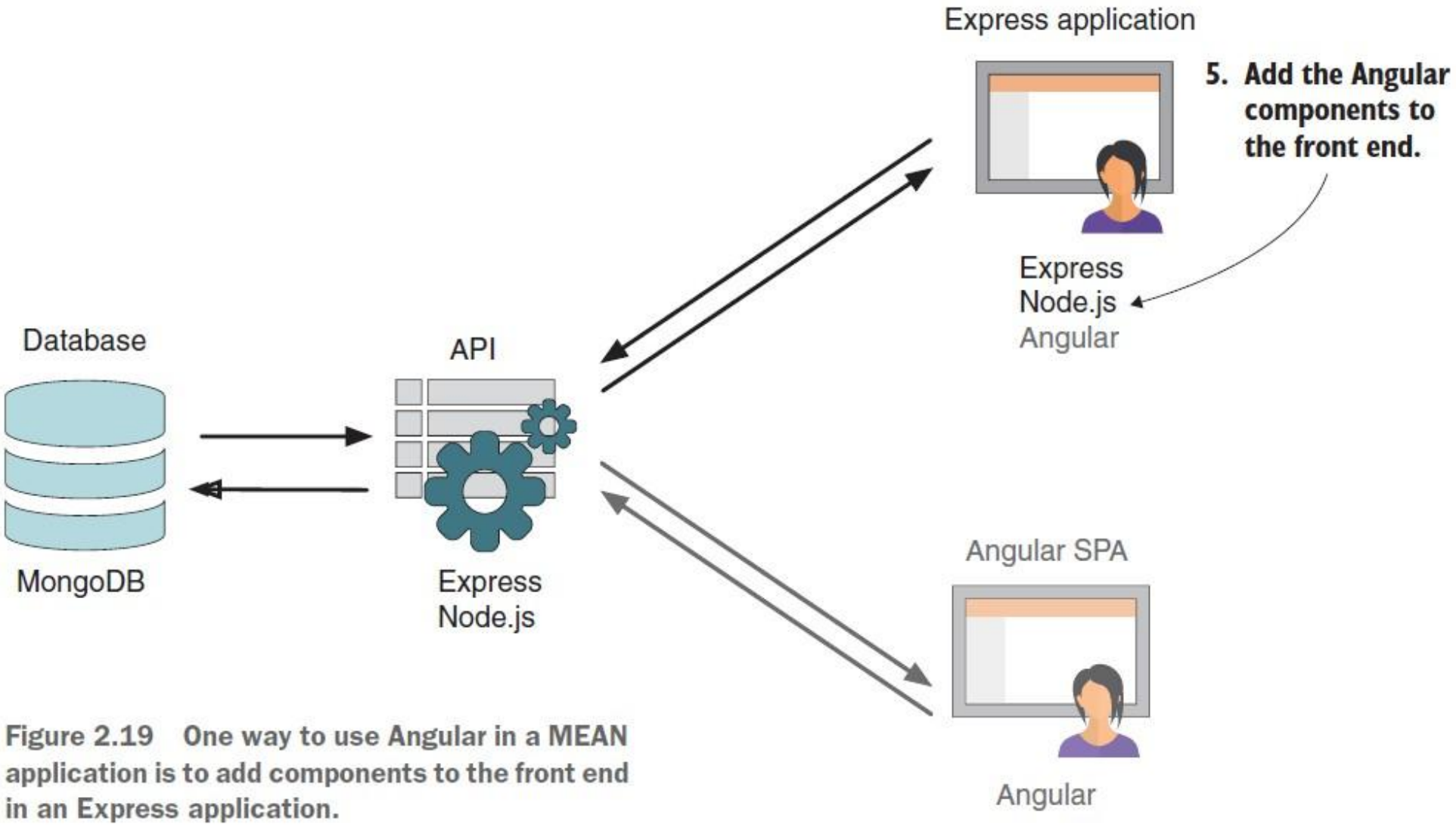


Figure 2.19 One way to use Angular in a MEAN application is to add components to the front end in an Express application.

ANURAG UNIVERSITY

Breaking Development into Stages

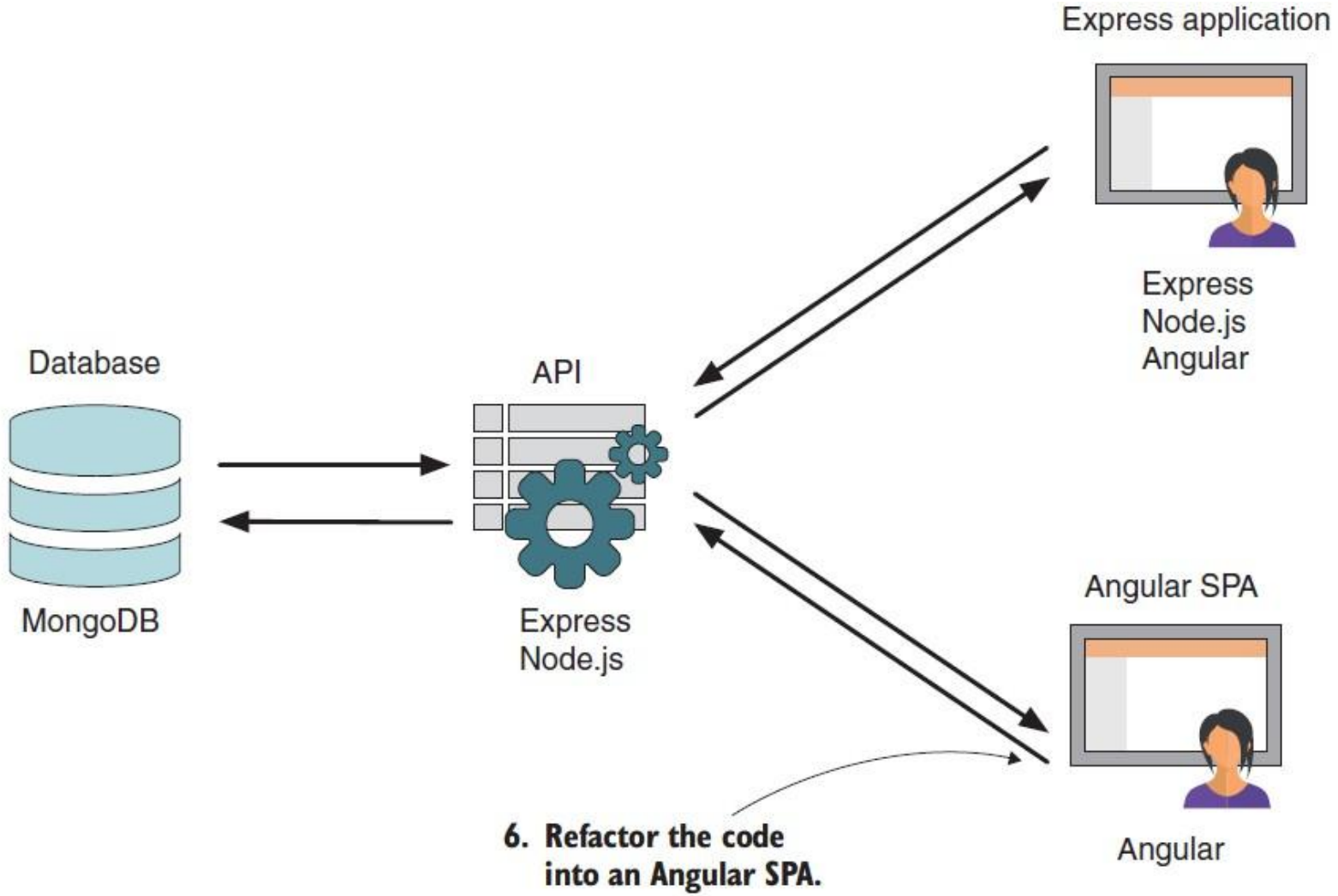


Figure 2.20 Effectively rewriting the application as an Angular SPA

Breaking Development into Stages

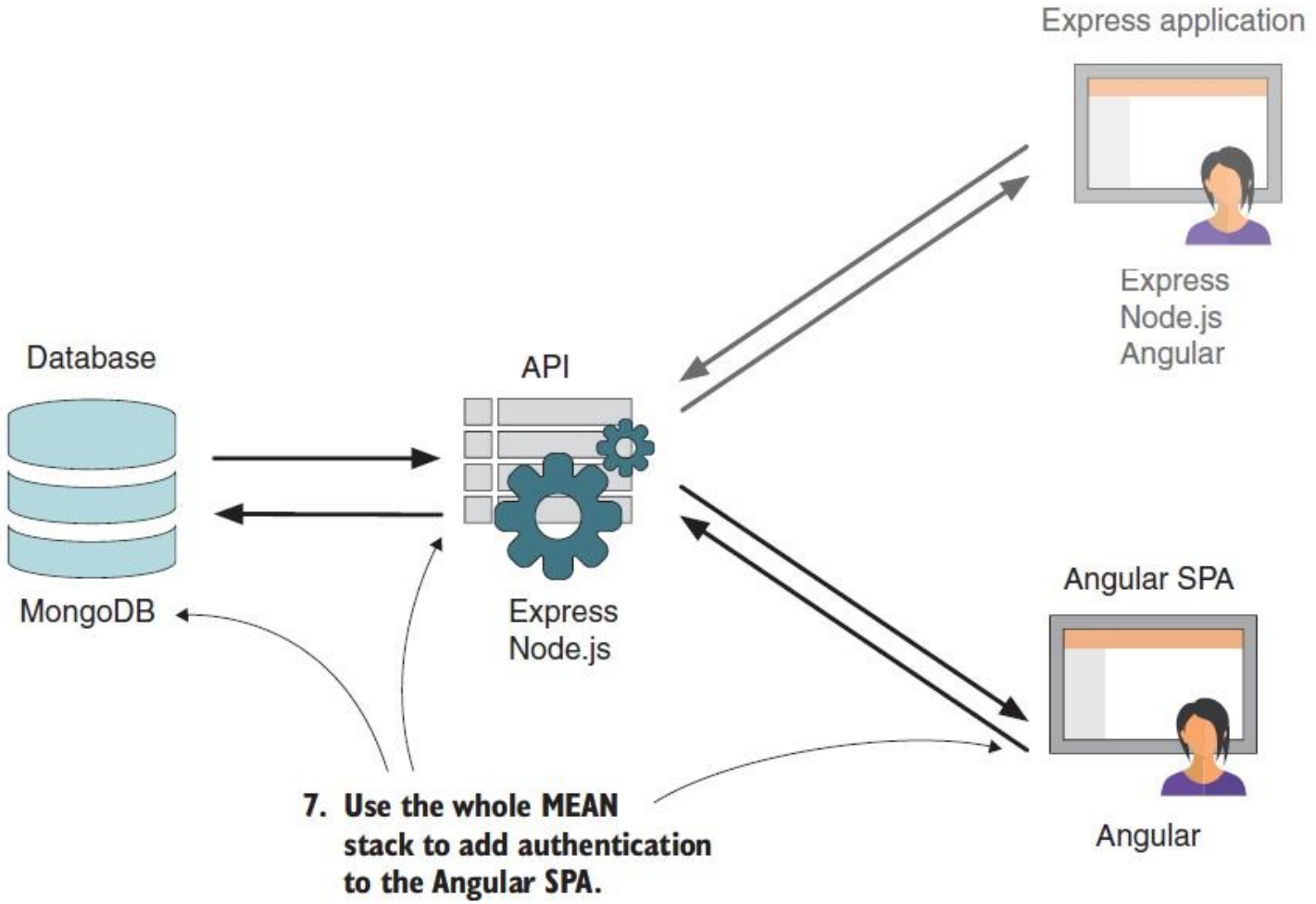


Figure 2.21 Using all the MEAN stack to add authentication to the Angular SPA

ANURAG UNIVERSITY

Hardware Architecture

💻 Development

- Any laptop (Win/Mac/Linux)
- Can install entire MEAN stack locally
- VM or local network works too

ANURAG UNIVERSITY

Hardware Architecture

🌐 Production

Type	Description
Starter	All-in-one server
Scaled	Separate DB server
Full-scale	3 servers (API, App, DB)
Clustered	Multiple instances per role

> Use cloud services (like Heroku, Render, MongoDB Atlas) for scaling with ease.

ANURAG UNIVERSITY

Hardware Architecture

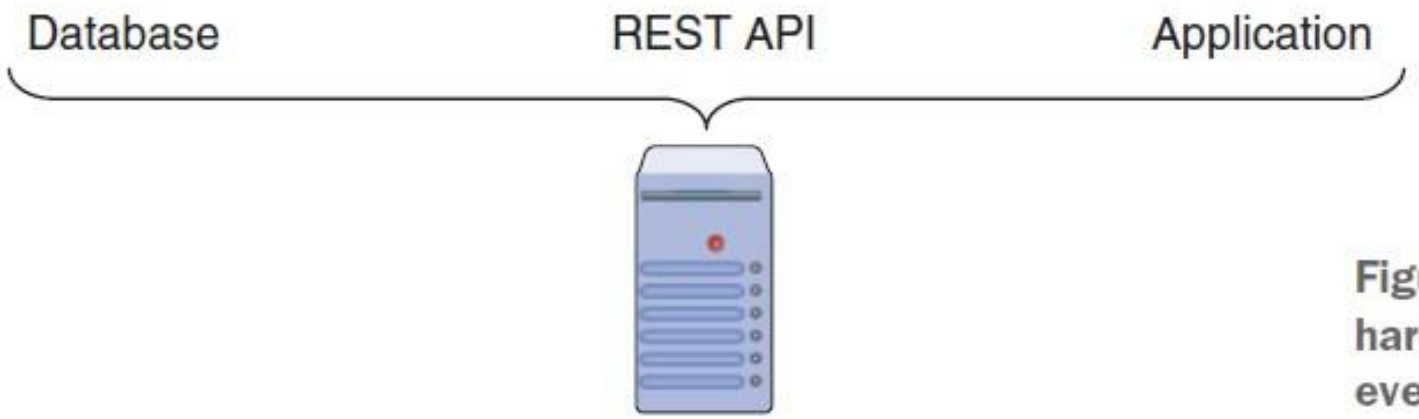


Figure 2.22 The simplest of hardware architectures, with everything on a single server

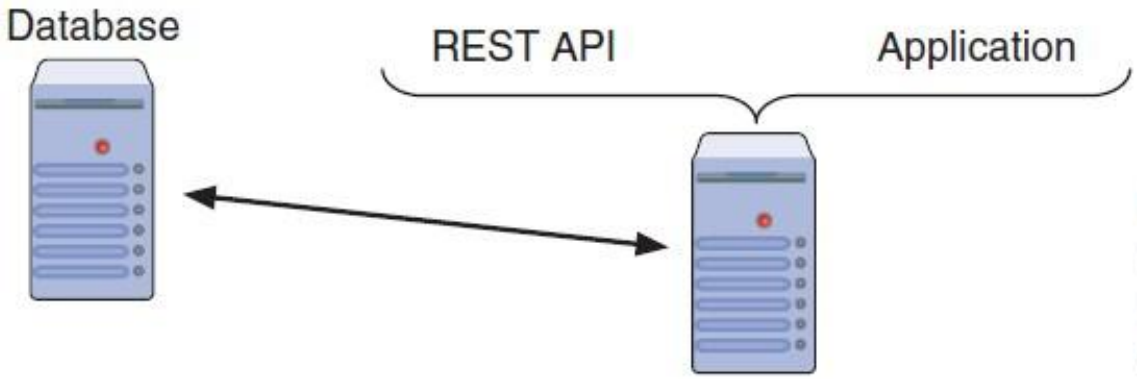


Figure 2.23 A common hardware architecture approach: one server to run the application code and API and a second, separate database server

ANURAG UNIVERSITY

Hardware Architecture

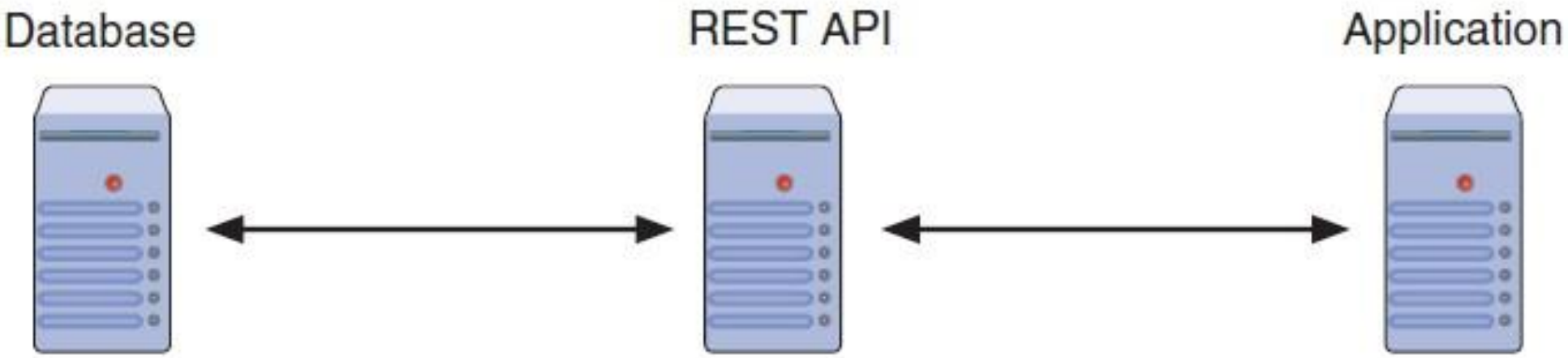


Figure 2.24 A decoupled architecture using three servers: one for the database, one for the API, and one for the application code

ANURAG UNIVERSITY

Hardware Architecture

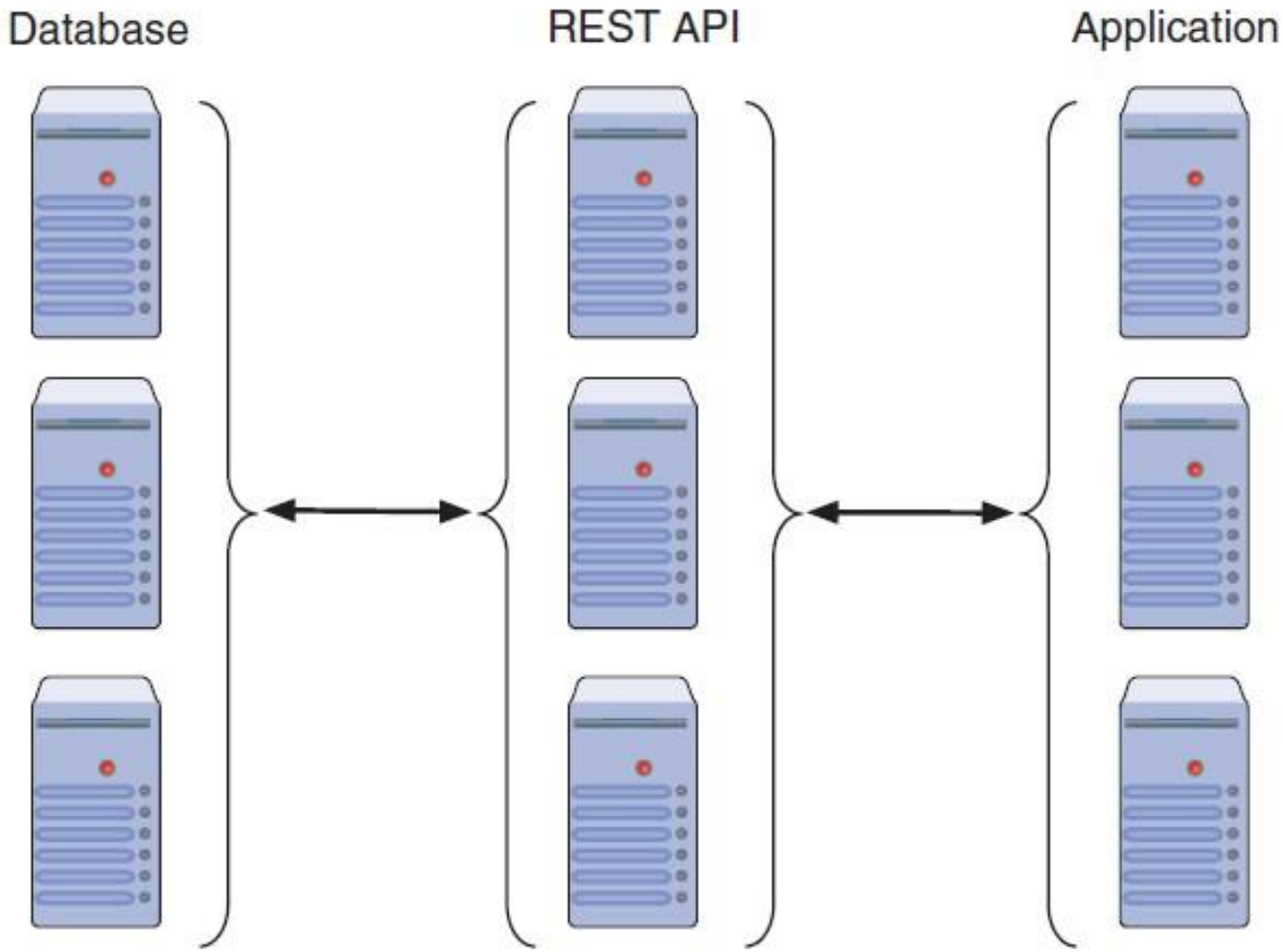


Figure 2.25 You can scale MEAN applications by having clusters of servers for each part of your entire application.

ANURAG UNIVERSITY

Final Notes

- **Loc8r** is not just an app — it's a **learning framework**.
- The book's approach teaches **progressive layering**, helping you understand and master each part of the MEAN stack.
- You learn **how**, **why**, and **when** to use each component.

ANURAG UNIVERSITY

Thank you