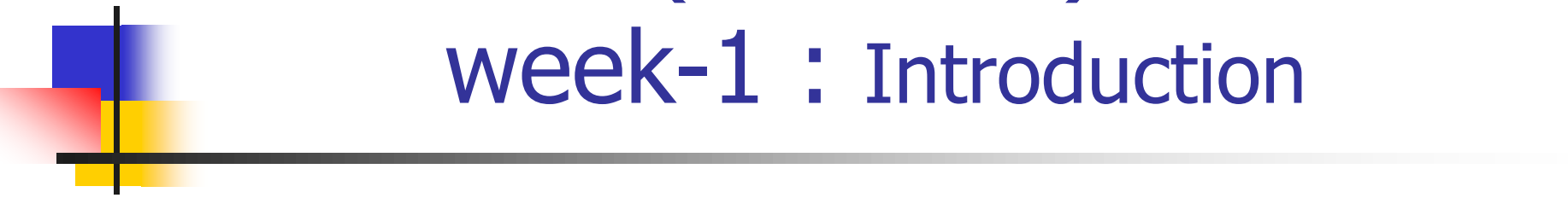
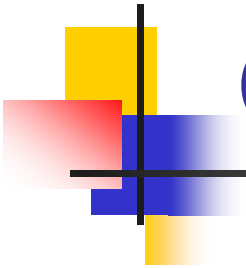
Advanced Application Development (CSE-214)
week-1 : Introduction



Dr. Alper ÖZCAN
Akdeniz University
alper.ozcan@gmail.com

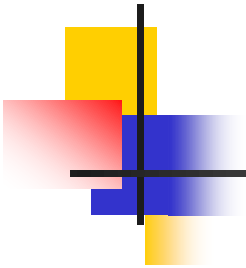
Course Schedule

- Introduction (February 11)
- Angular – Typescript (February 18)
- Angular Application Architecture(February 25)
- Data binding, Routing(March 4)
- Angular forms, RxJS (March 11)
- Introduction to SpringBoot (March 18)
- Spring Core (March 25)
- Hibernate / JPA (April 1)
- **Midterm (April 8)**
- Rest / CRUD (April 15)
- Spring MVC (April 22)
- JPA / Hibernate Advanced Mappings (April 29)
- **Project Presentations (Due on May 6)**
- **Project Presentations (Due on May 13)**
- Final examination review (May 20)



Grading

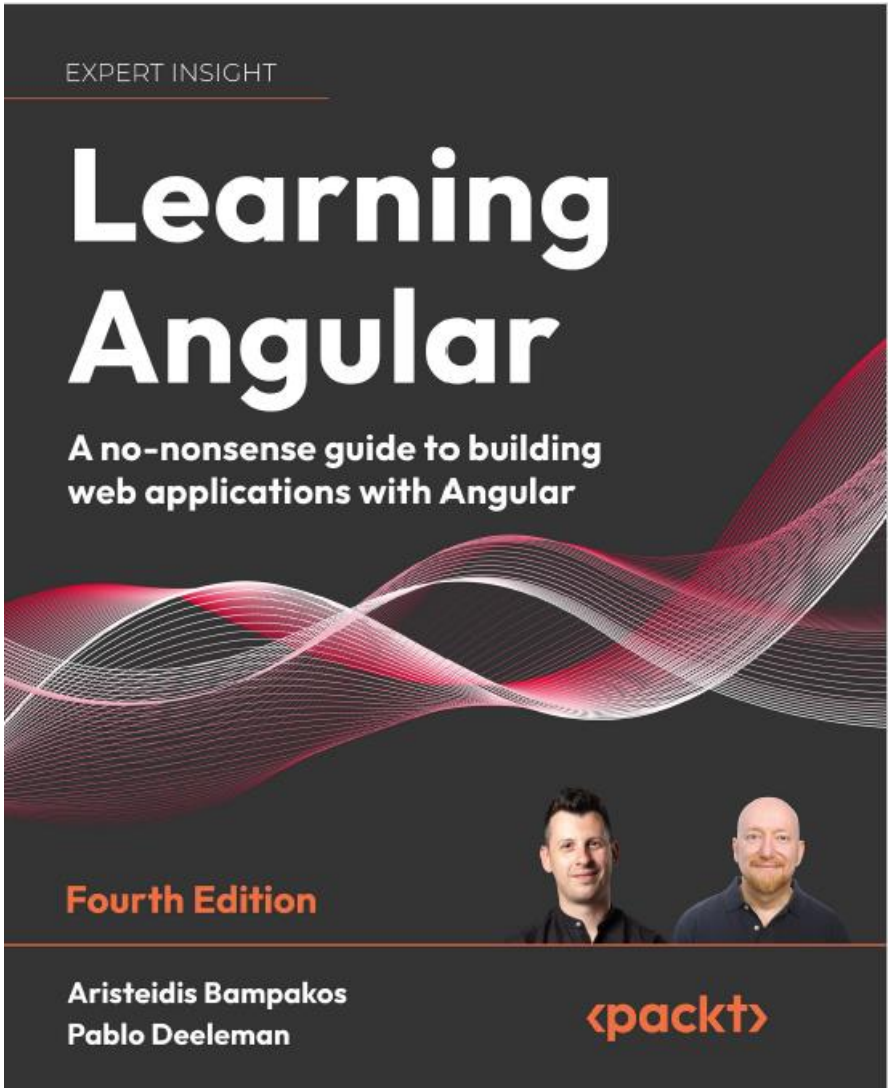
- Midterm (15%)
- Lab (20 %) (**Coding will be held on lab**)
- Project (25%) – Angular + SpringBoot
  - Proposal, final report, github repo
  - Each project team should consist of 2 students
- Final (40%)



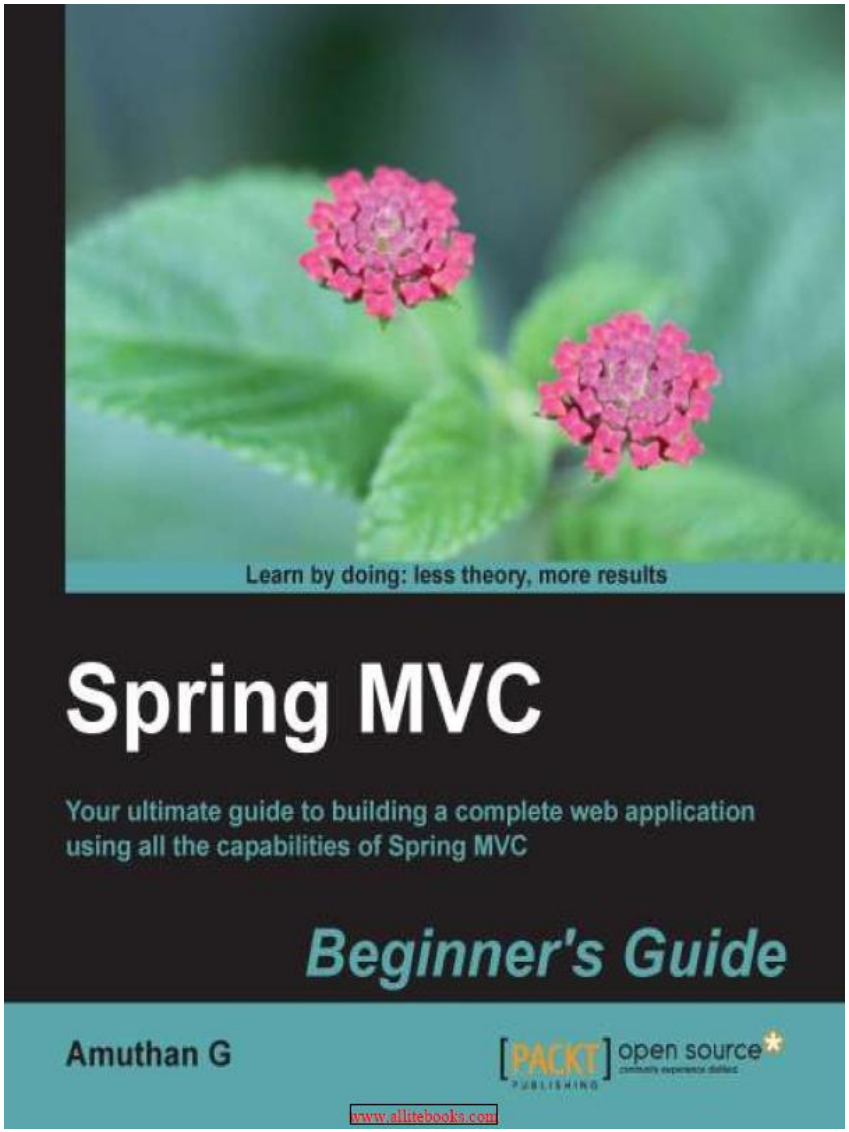
Prerequisites

- Background in
  - Programming
- Programming:
  - Angular - Typescript
  - SpringBoot - Java
  - VsCode , IntelliJ

Text Books - Angular



Text Books - SpringBoot



What's this course about

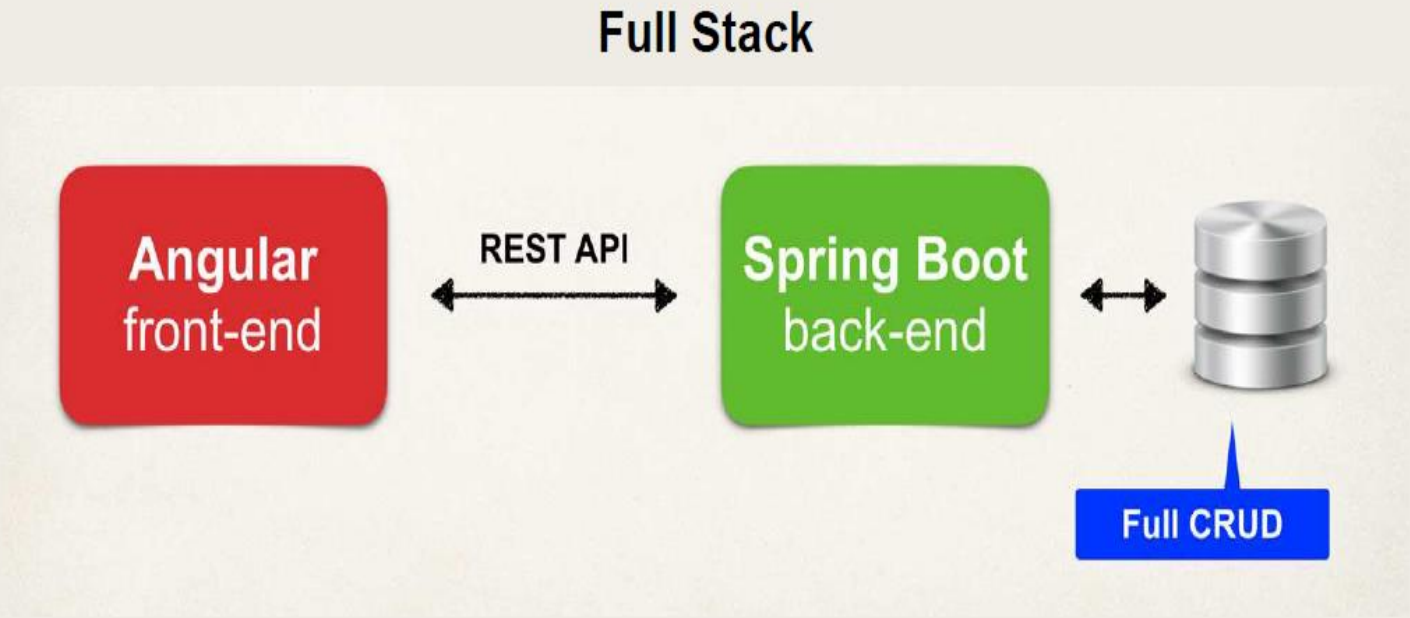
- **Front-End (Angular)**
  - **Data Binding:** Effective data exchange between templates and components.
  - **Component Lifecycle Hooks:** Lifecycle management for Angular components.
  - **Routing**: Efficient navigation between application views.
  - **Angular Forms:** Form handling and validation.
  - **Dependency Injection:** Modular and testable architecture.
  - **RxJS**: Reactive programming for asynchronous operations.

What's this course about

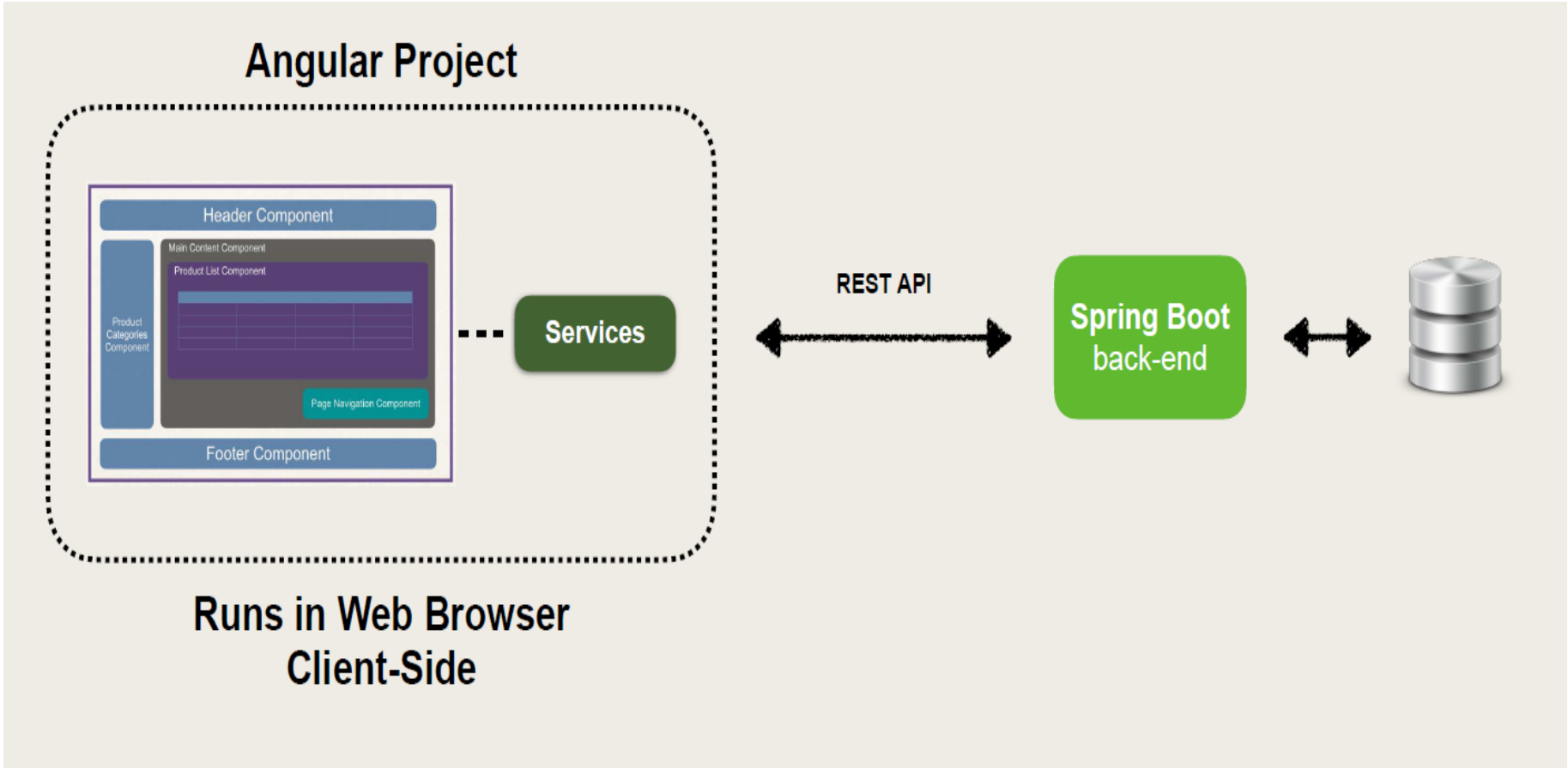
- **Back-End (Spring Boot)**
  - **Hibernate/JPA:** Data persistence and ORM (Object-Relational Mapping).
  - **REST API Development:** Secure and scalable APIs for CRUD operations.
  - **Spring MVC:** Clear separation of concerns using Model-View-Controller architecture.
  - **Advanced JPA Mappings:** Efficient representation of complex data relationships.
  - **Security:** Authentication and authorization using Spring Boot REST API Security.

Project

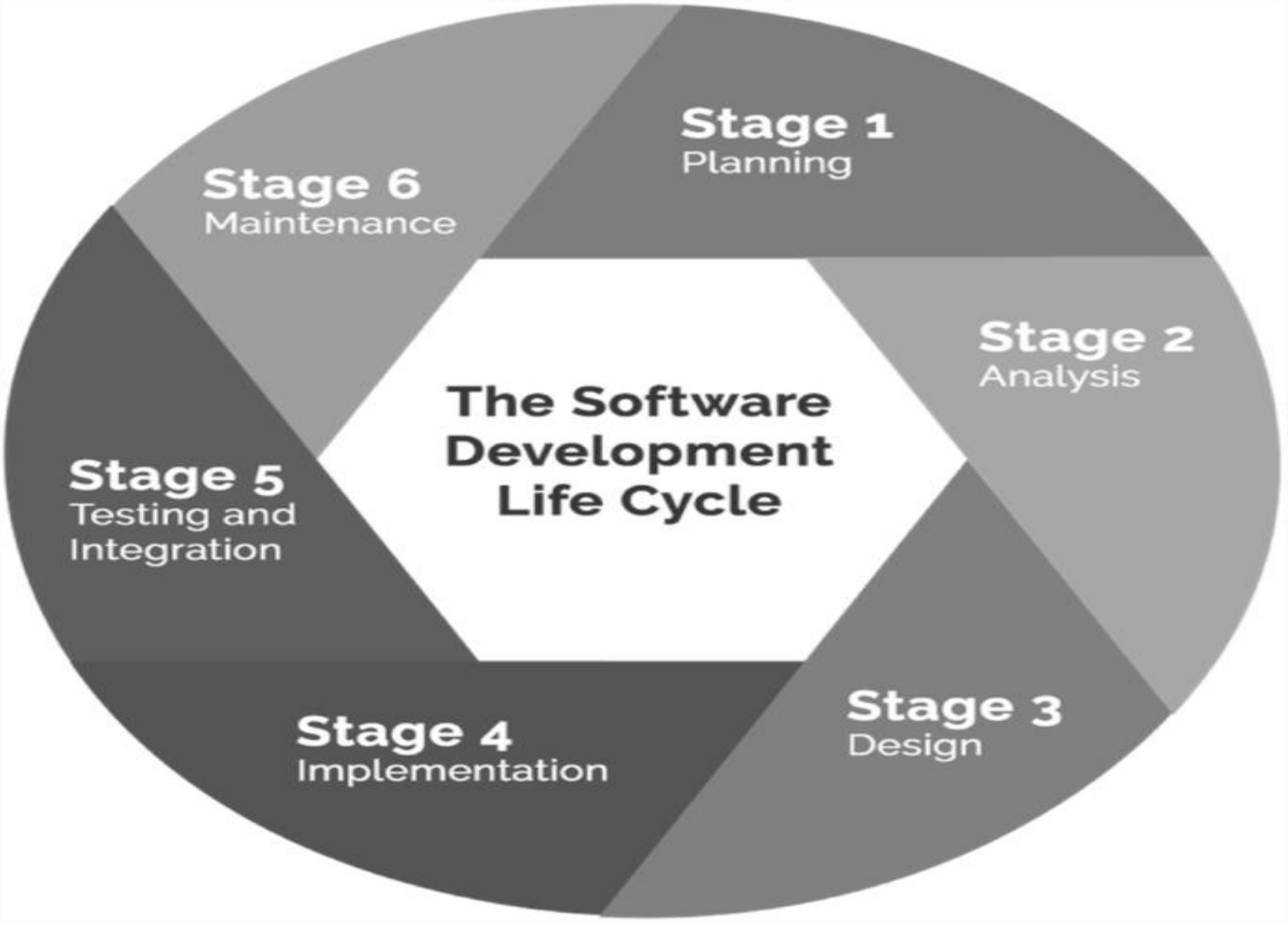
- Create Angular Front End components
- Retrieve data from Spring Boot REST APIs



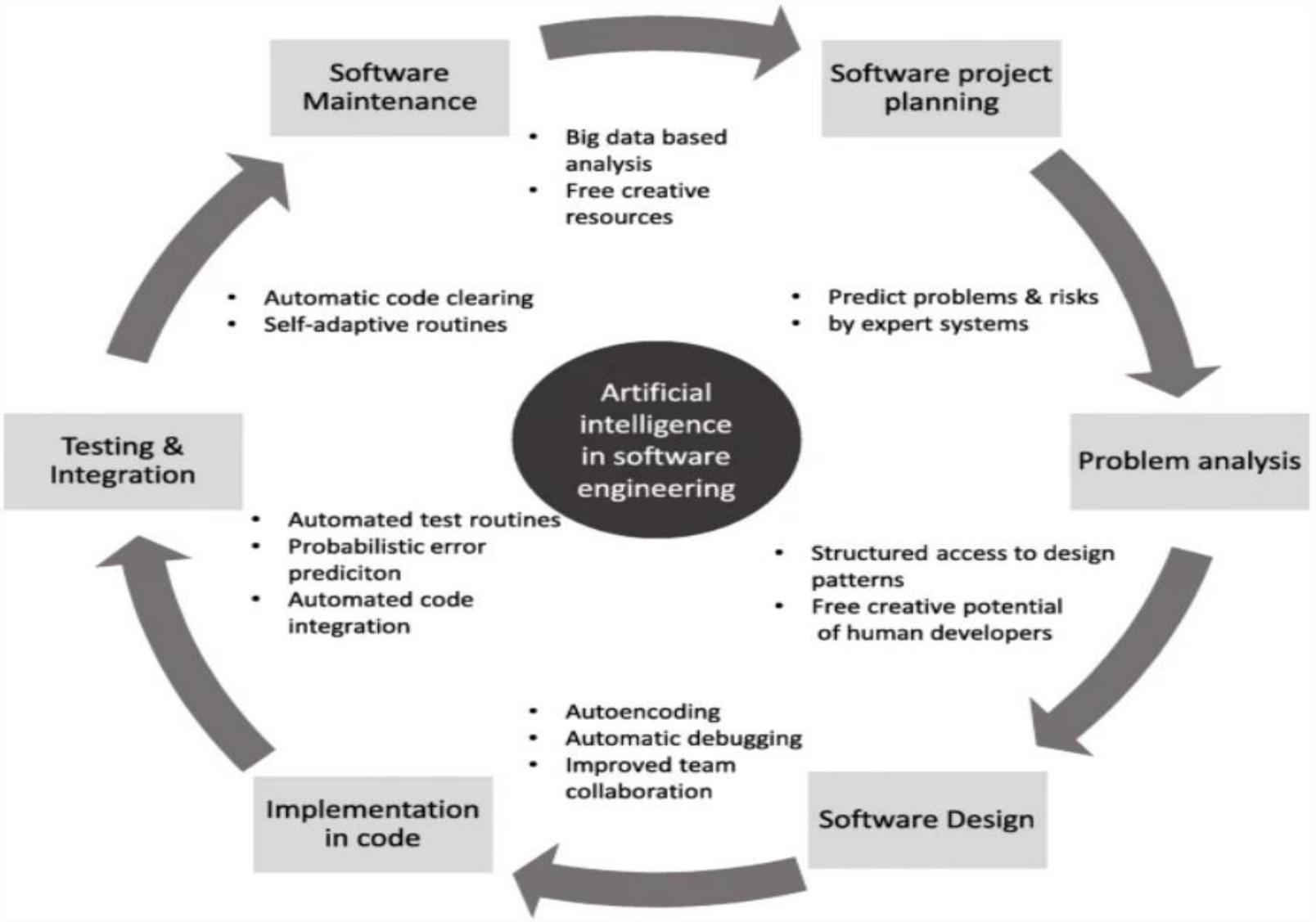
Project


Angular Project
Header Component
Main Content Component
Product List Component
Product Categories Component
Page Navigation Component
Footer Component
Services
REST API
Spring Boot
back-end
Runs in Web Browser
Client-Side

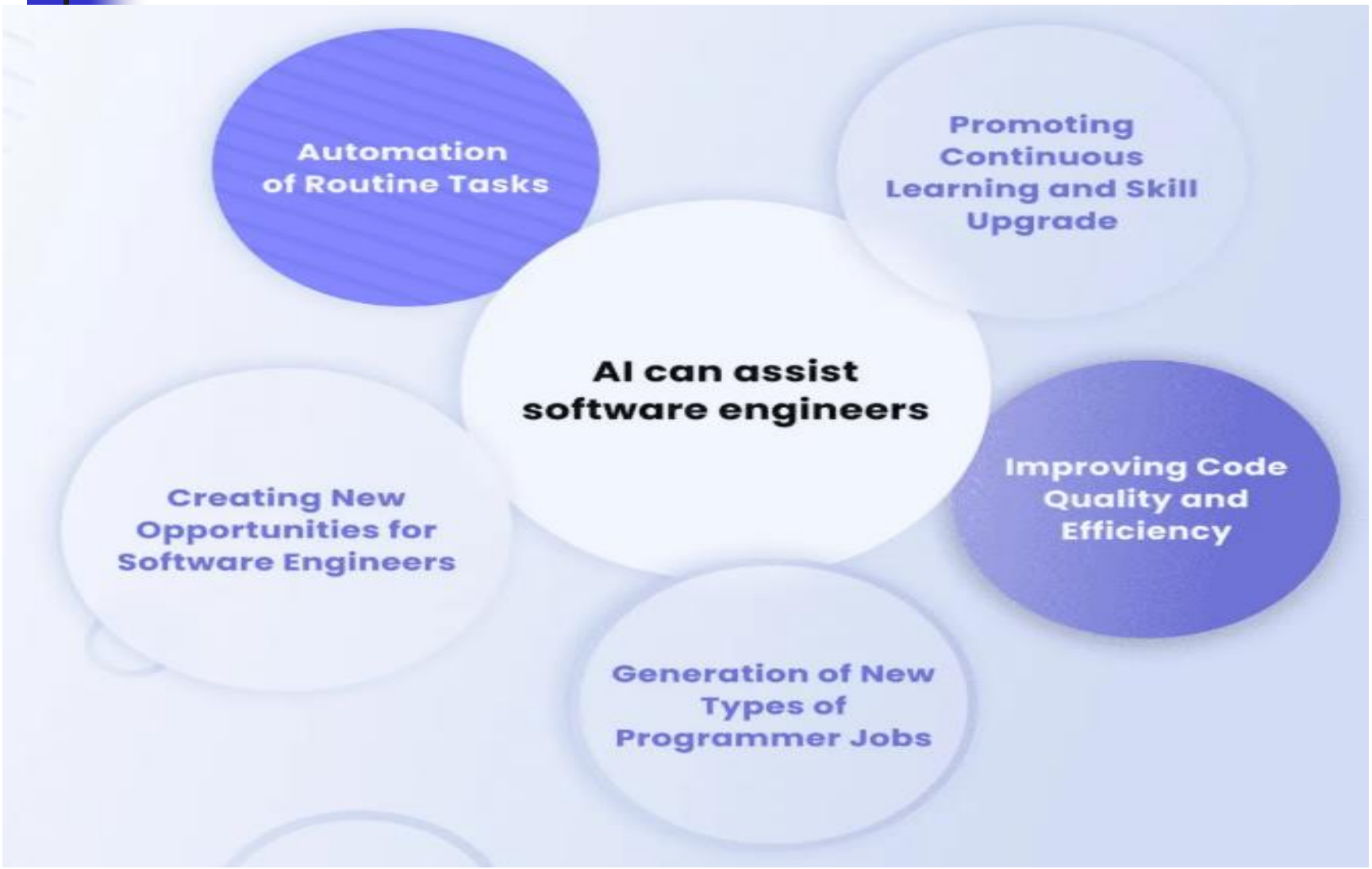
Software Development Life Cycle



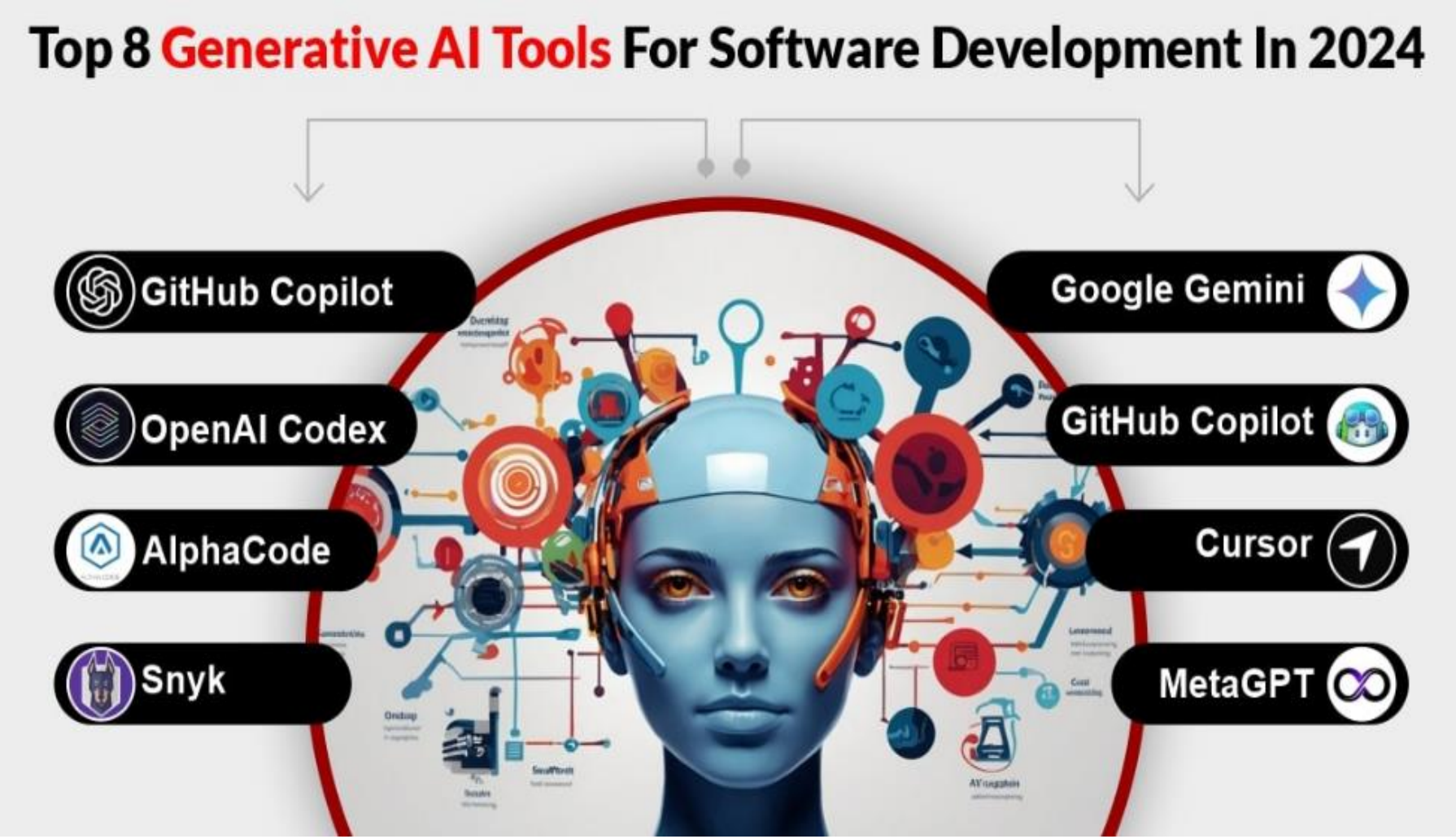
Artificial intelligence tools in the software development life cycle



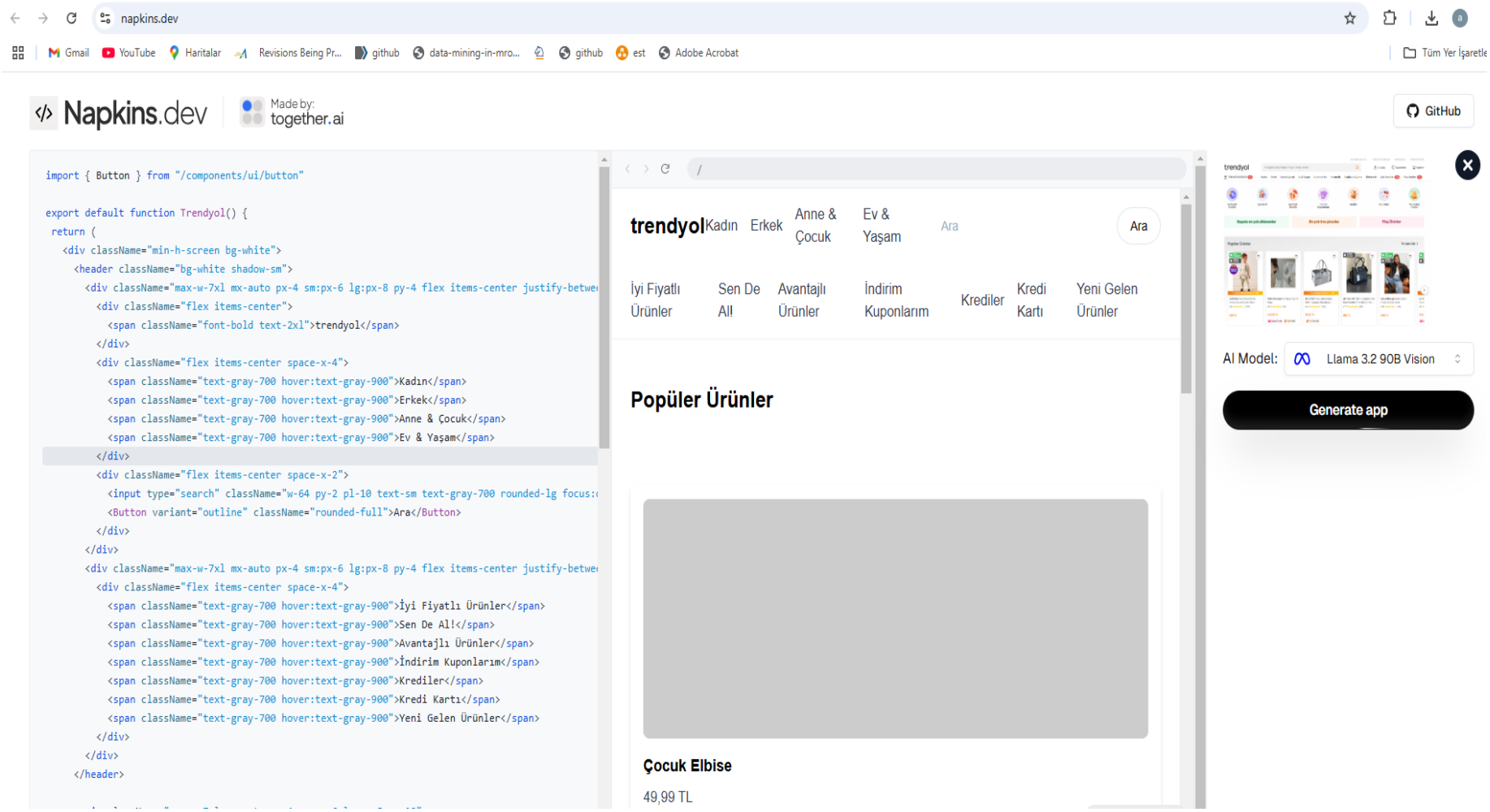
How Will AI Affect the Work of Developers?



How Will AI Affect the Work of Developers?

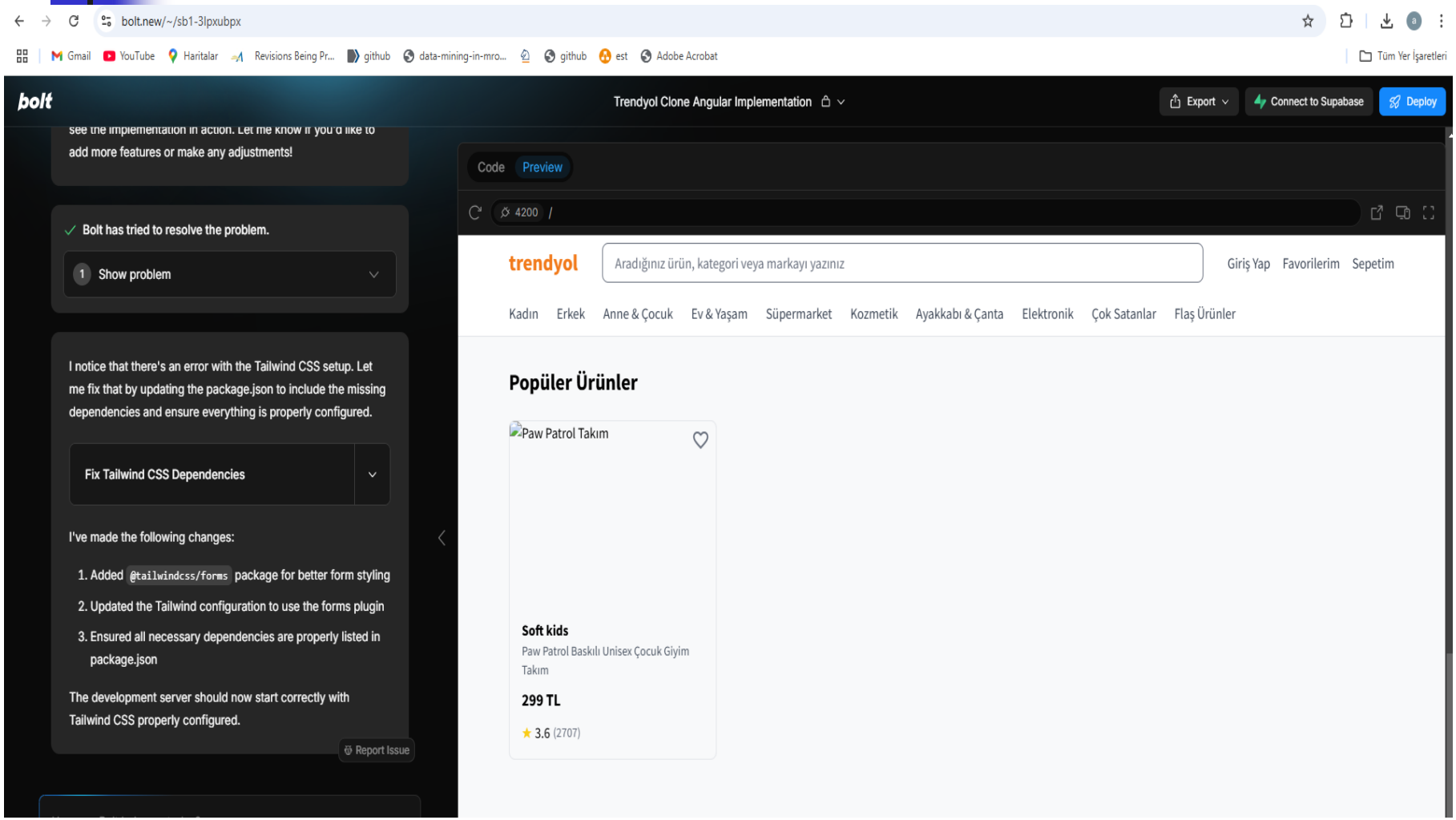


AI (LLM) based code generation (napkins)



https://www.napkins.dev/

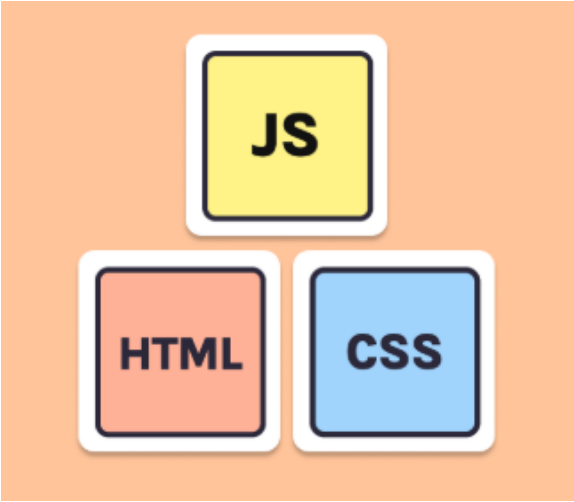
AI (LLM) based code generation (bolt)



https://bolt.new/

HTML, CSS, JS

HTML (Hypertext **Markup Language**) is an essential building block of any webpage. It describes the structure and semantics (meaning) of content rather than its appearance.



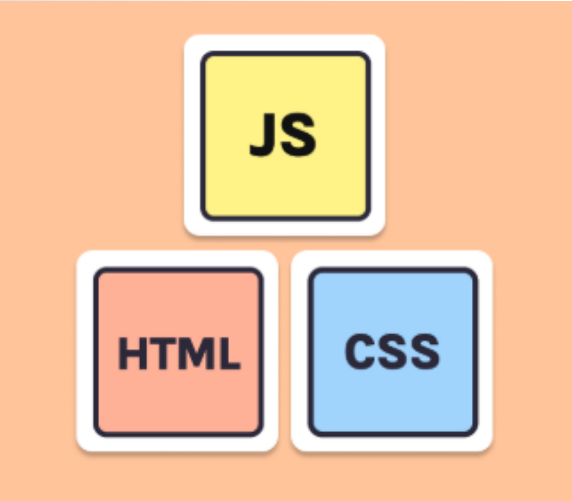

```
<!DOCTYPE HTML>
<HTML>
  <head>
    <title>This is a title</title>
  </head>
  <body>
    <p>Hello world!</p>
  </body>
</HTML>
```

HTML contains few predefined elements or tags each of which means differently for a browser.

https://html-css-js.com/

HTML, CSS, JS

Cascading Style Sheets which is known as CSS is a style sheet language that used to handle the presentation of the web page containing HTML. It makes our websites beautiful and modern looking.

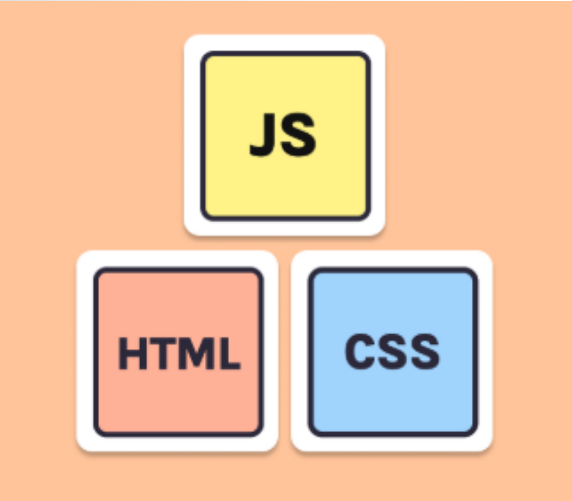



```
body {
  background-color: lightblue;
}

h1 {
  color: white;
  text-align: center;
}

p {
  font-family: verdana;
  font-size: 20px;
}
```

https://html-css-js.com/

HTML, CSS, JS

JavaScript which is often known as JS, is a high-level dynamic interpreted programming language that allows client-side scripting to create completely dynamic web applications and websites.




```
<script>
const myElement = document.getElementById("demo");
myElement.style.color = "red";
</script>
```

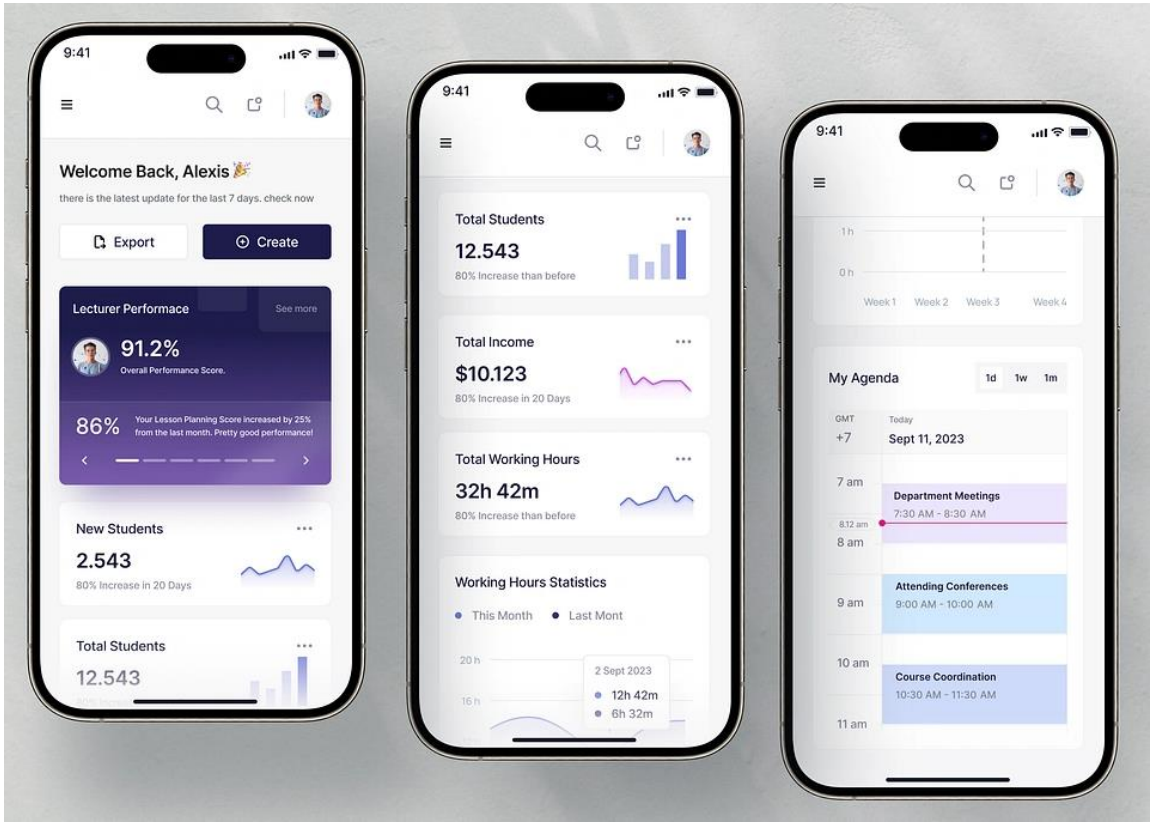
https://html-css-js.com/

Bootstrap

- Bootstrap is a free front-end framework for faster and easier web development
- Bootstrap includes HTML and CSS based design templates for forms, buttons, tables, navigation, modals, image carousels and many other, as well as optional JavaScript plugins
- Bootstrap also gives you the ability to easily create responsive designs

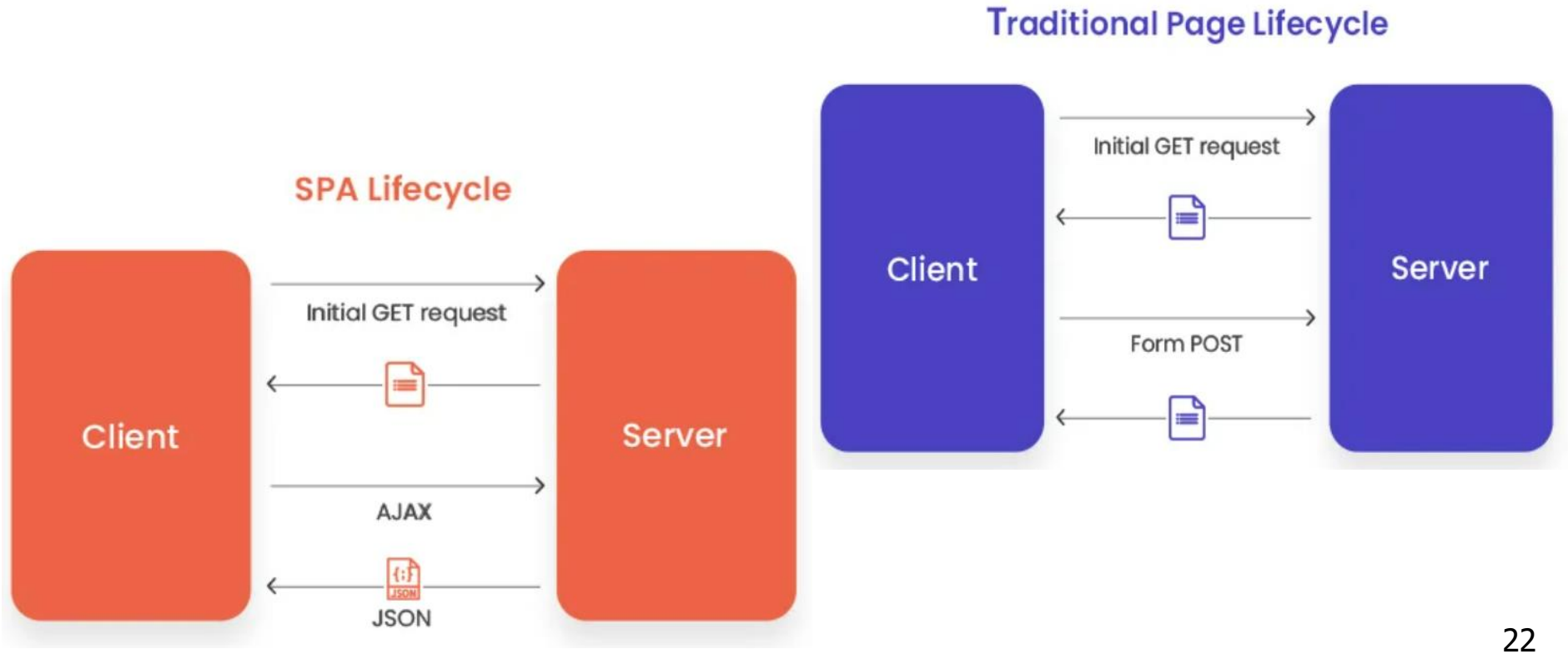
Responsive Design

Responsive Web Design is about using HTML and CSS to automatically resize, hide, shrink, or enlarge, a website, to make it look good on all devices (desktops, tablets, and phones)



What is a Single Page Application (SPA)?

A **Single Page Application** (SPA) dynamically updates the content of the page as users interact with it, instead of loading new pages from the server. This results in faster load times and a more responsive, app-like user experience.



What is a Single Page Application (SPA)?

In single page web apps, once the page gets loaded, the server does not send any more HTML or CSS. Whereas for the traditional one, every time when new data is requested, the whole page gets reloaded.

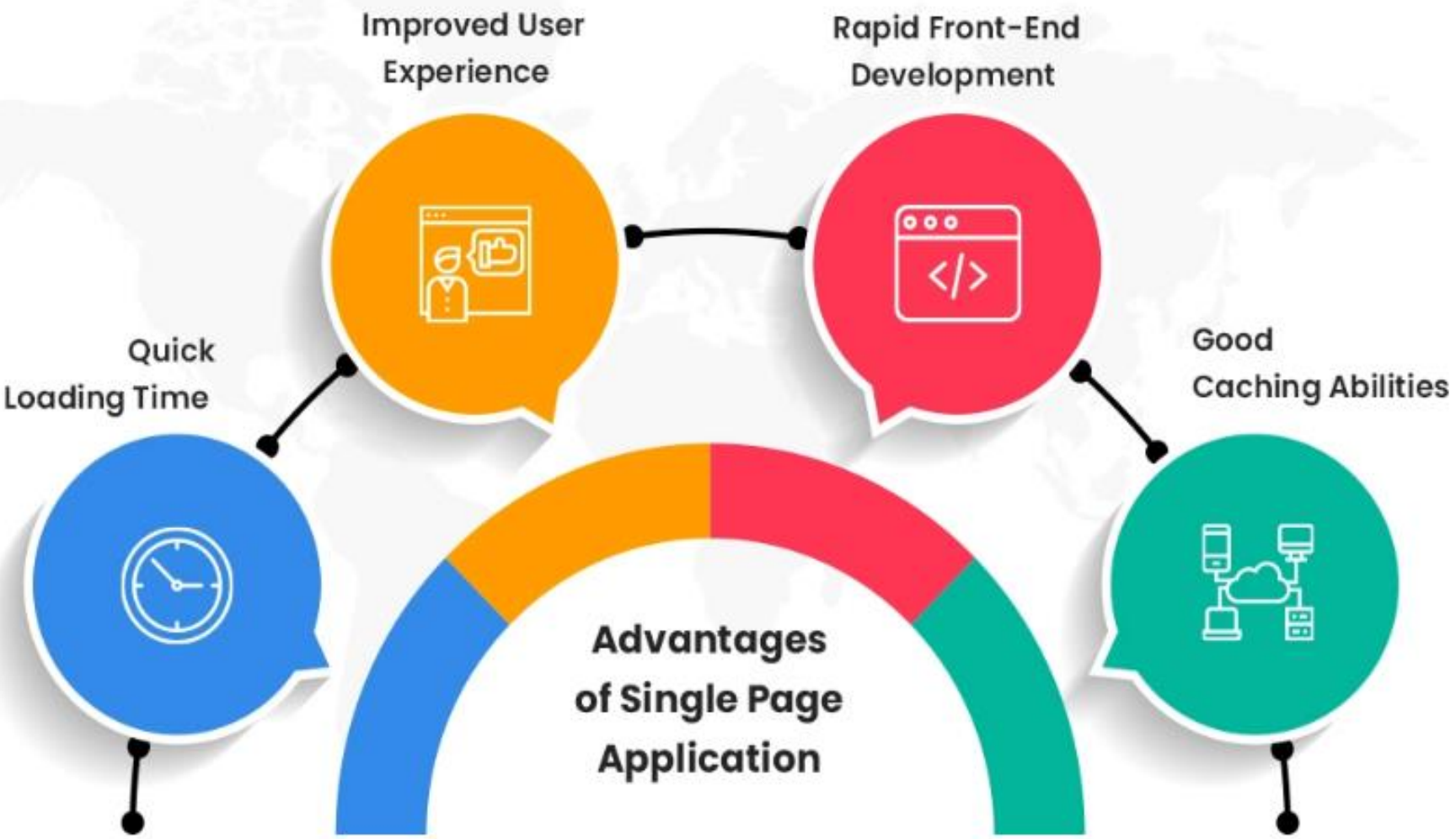
Traditional

Every request for new information gives you a new version of the whole page.

Single Page Application

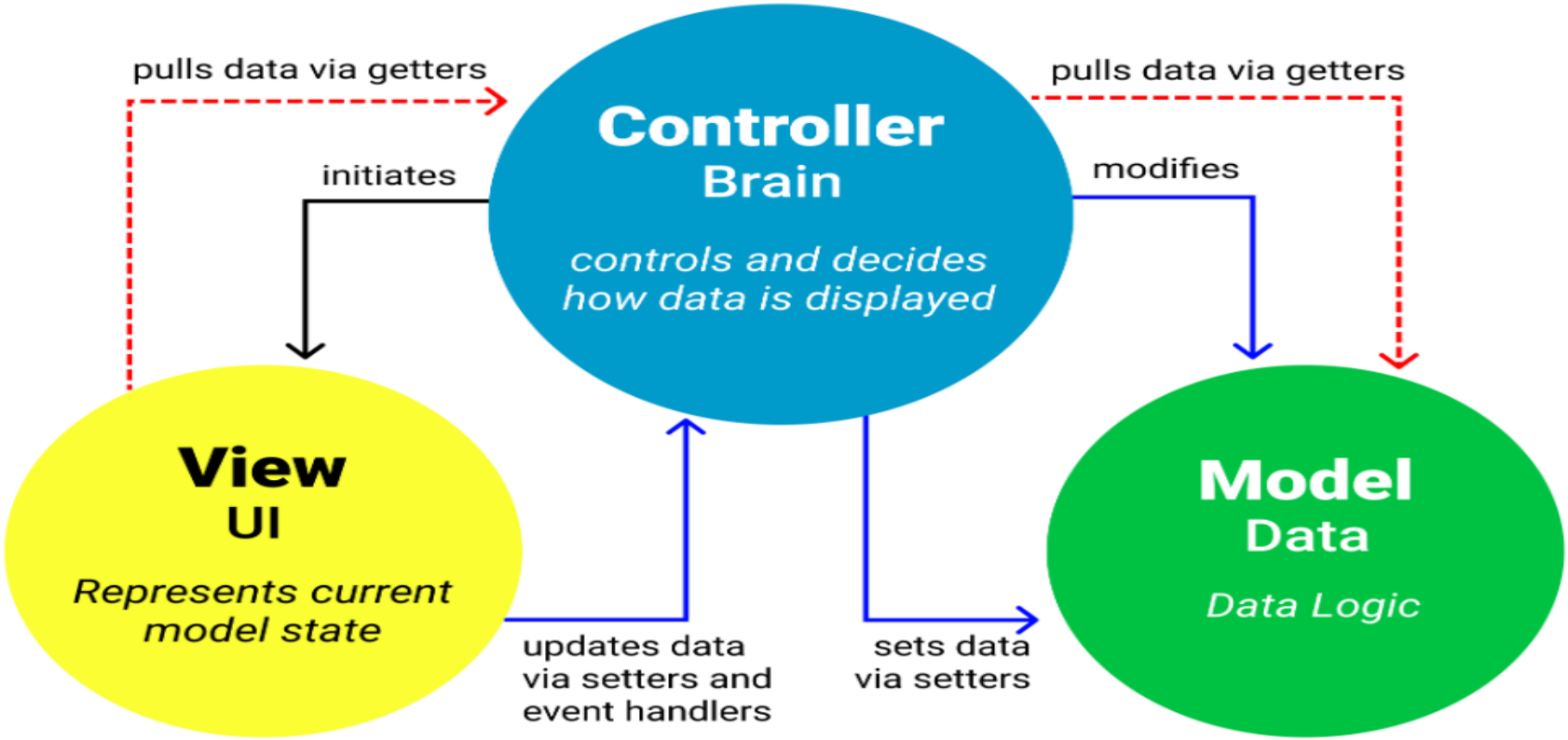
You request just the pieces you need.

Advantages of Single Page Application



Model-View-Controller (MVC) Design Pattern

Model: Manages the data, logic, and rules of the application.
View: Represents the user interface (UI) and displays data from the model.
Controller: Acts as a mediator between the Model and View, Handles user interactions and updates the model or view



What is Angular?

Angular is a popular **TypeScript-based front-end framework** developed by Google.

It allows developers to build **Single Page Applications (SPAs)**, which offer dynamic user experiences by updating the content seamlessly without reloading the entire page.

Key features of Angular include data binding, routing, dependency injection, and form handling.

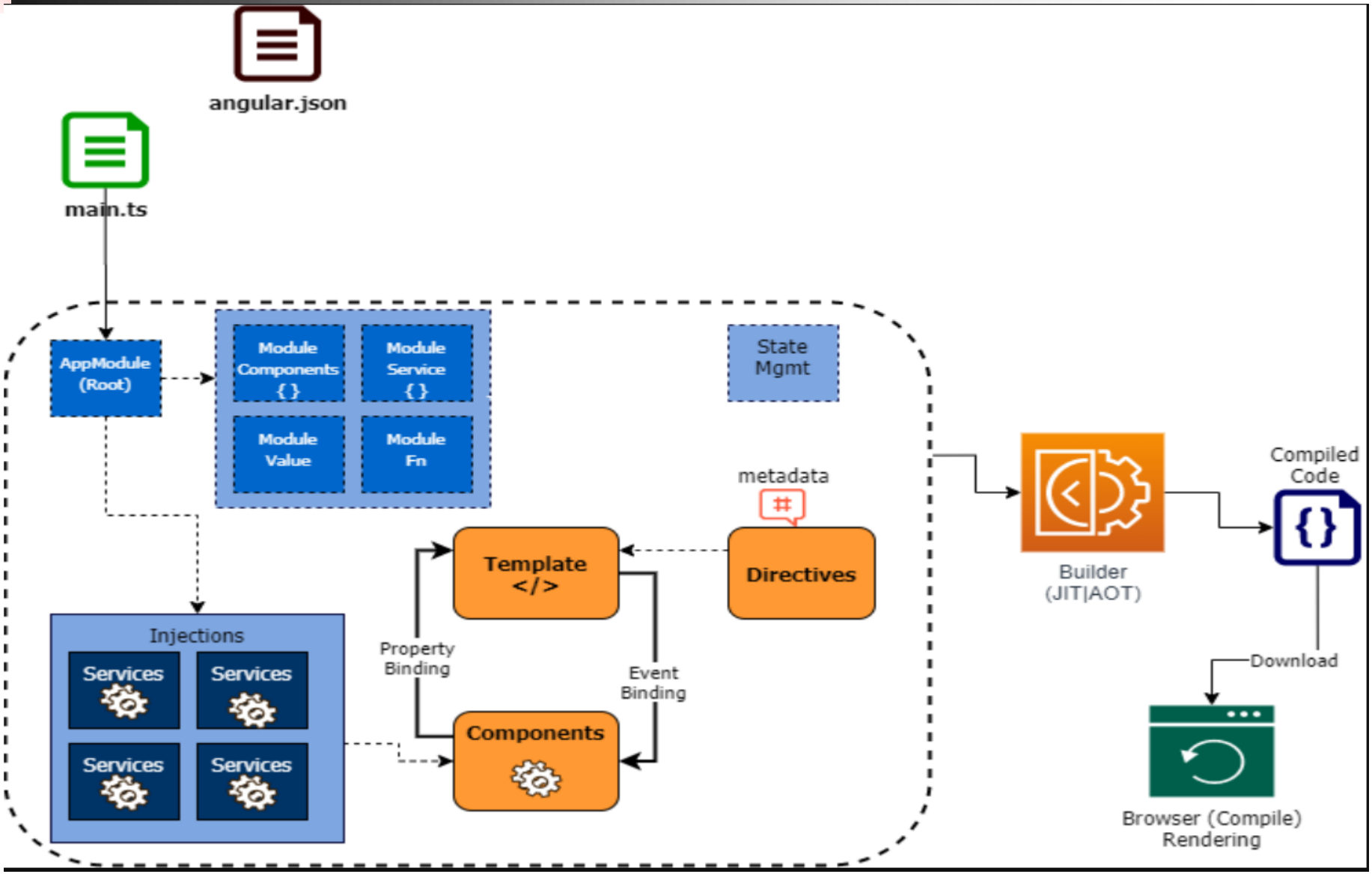
Structure of Angular Components

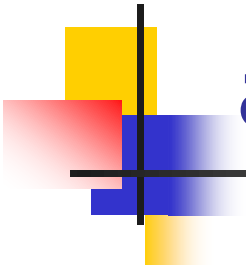
Components are the building blocks of an Angular application. Each component manages a specific part of the UI

- **TypeScript class (.ts)**: Contains logic and data for the component.
- **HTML template (.html)**: Defines the structure and layout of the view.
- **CSS or SCSS (.css or .scss)**: Styles for the view.
- **Metadata (@Component decorator)**: Provides component configuration

```typescript
import { Component } from '@angular/core';

@Component({
  selector: 'app-hello',
  templateUrl: './hello.component.html',
  styleUrls: ['./hello.component.css']
})
export class HelloComponent {
  message: string = 'Hello, Angular!';
}
```

Angular application architecture





angular.json

The angular.json file is the main configuration file for an Angular project. It defines various settings and configurations that are used by the Angular CLI (Command Line Interface) to build, serve, and test the application.

Central configuration: The angular.json file configures how the Angular CLI should behave (build, serve, test, etc.) for the entire project.

Project-specific settings: It holds project-specific settings for the apps or libraries within the project.

Customizable build process: It allows developers to customize aspects of the build, like optimization, file replacements, and file paths for assets and styles.

Multiple environments: Supports configuring different environments (like development and production) by using file replacements and different build settings.

Bootstrapping the AppModule (main.ts)

```
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
import { AppModule } from './app/app.module';

// Bootstrap the AppModule
platformBrowserDynamic().bootstrapModule(AppModule)
  .catch(err => console.error(err));
```

- Acts as the starting point (entry point) for the application lifecycle.
- This file is the **bootstrap** file of the Angular application. Angular uses main.ts to load the Angular module and start the app.
- ***platformBrowserDynamic***: Angular's function for bootstrapping applications in the browser.
- ***AppModule***: The root module of the application.

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>My Angular App</title>
  <base href="/">
</head>
<body>
  <!-- Angular will bootstrap and render AppComponent here -->
  <app-root></app-root>

  <!-- Angular application scripts will be injected here during the build process -->
</body>
</html>
```

- This is the entry point for the application.

- When the app starts, the browser loads index.html. It contains the root <app-root></app-root> element, AppComponent is rendered in the index.html file's <app-root></app-root> element

- it includes references to the stylesheets and JavaScript files, including the bundled Angular code.

Main Module (AppModule), app.module.ts

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import { HelloComponent } from './hello/hello.component';

@NgModule({
  declarations: [AppComponent, HelloComponent],
  imports: [BrowserModule],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule {}
```

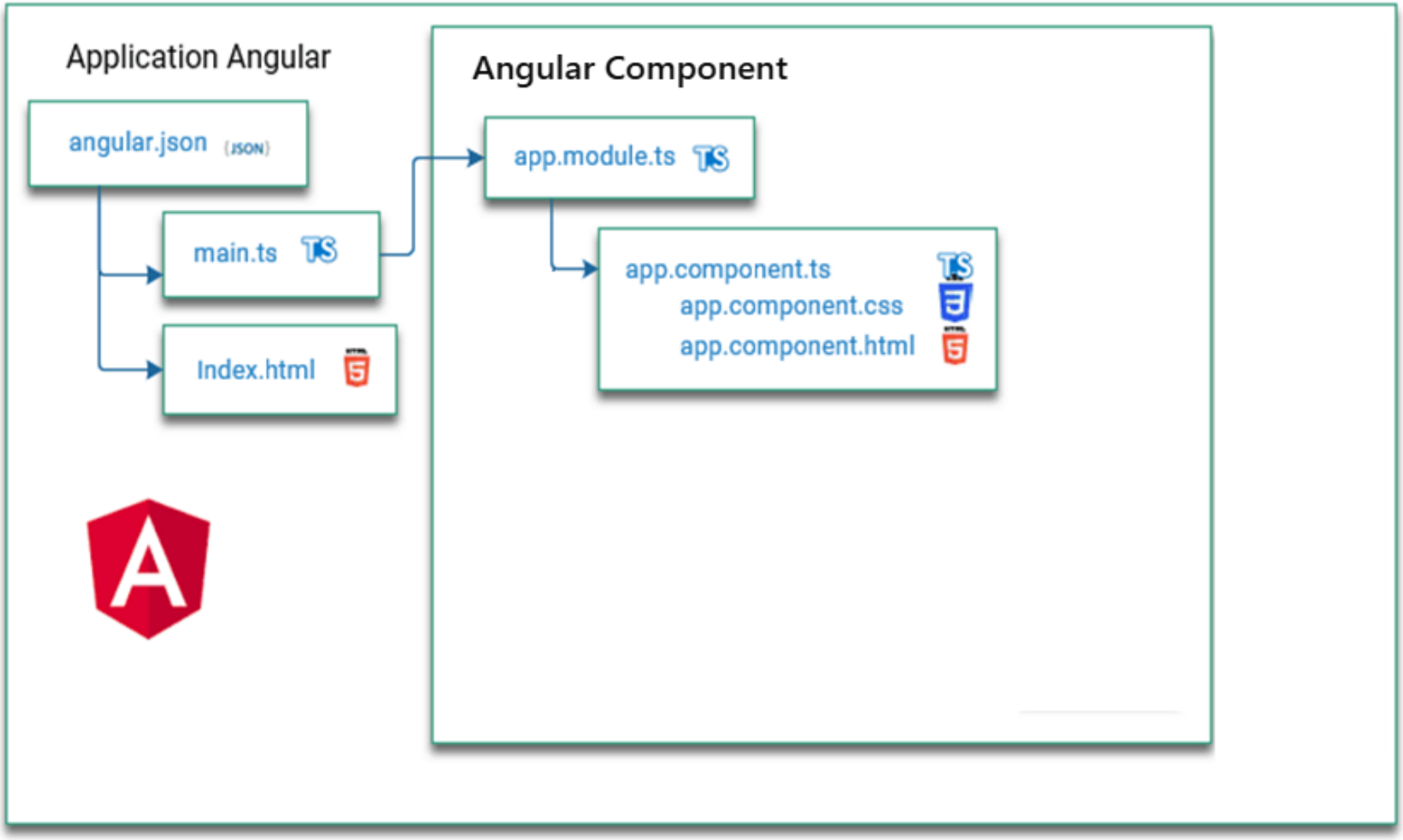
This file defines the root **NgModule** (Angular Module) of the application. It tells Angular which components, directives, and services are part of the app.

AppComponent (app.component.ts)

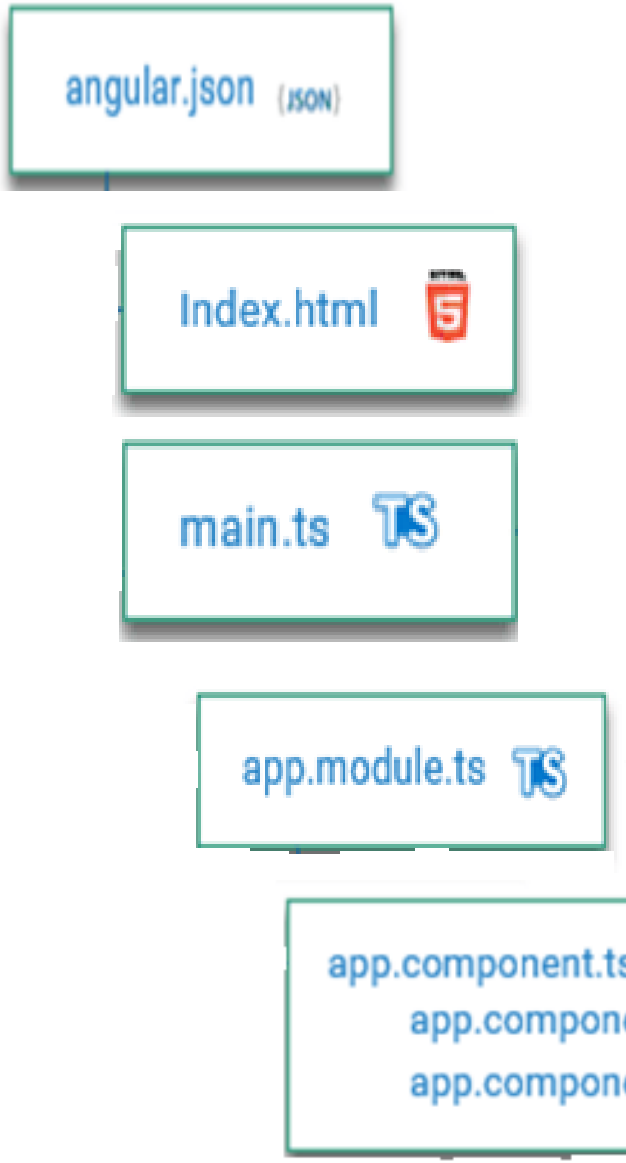
```
@Component({
  selector: 'app-root',  // Matches <app-root> in the HTML
  template: `<h1>Welcome to My Angular App!</h1>`,
})
export class AppComponent {}
```

- This is the root component that is rendered inside <app-root></app-root> in index.html.
- It is defined in app.component.ts, where the logic for the root component is implemented.
- The AppComponent acts as the main component that interacts with other components, binds data, and defines the application's structure.
- It is typically bootstrapped in app.module.ts under the bootstrap array.

Angular Files Execution Flow



Execution Flow



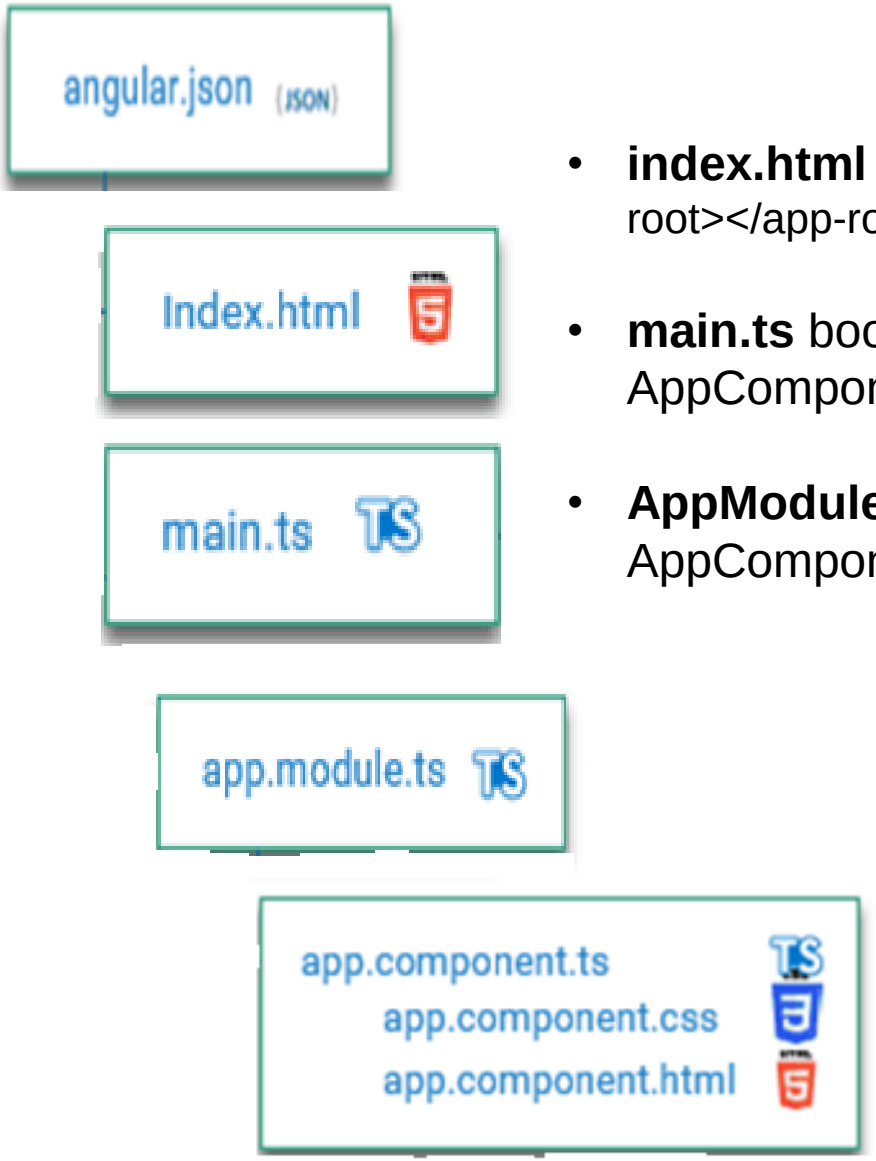
index.html is loaded first when the browser requests the app. It includes a reference to the Angular application and the root component (<app-root></app-root>).

main.ts is executed next. It bootstraps the Angular application by calling platformBrowserDynamic().bootstrapModule(AppModule) to initialize the Angular module (AppModule).

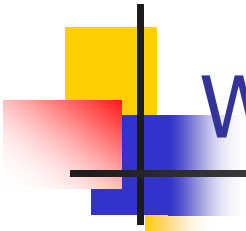
app.module.ts is processed, and Angular sets up the application module, registers components, imports other necessary modules, and prepares the app for rendering.

AppComponent is the root component declared in app.module.ts, and Angular renders this component inside the <app-root></app-root> element in index.html.

Relationships



- **index.html** references the root component <app-root></app-root>.
- **main.ts** boots the AppModule, which then uses AppComponent as the root component.
- **AppModule** imports and declares AppComponent, and AppComponent becomes the starting point for the UI.

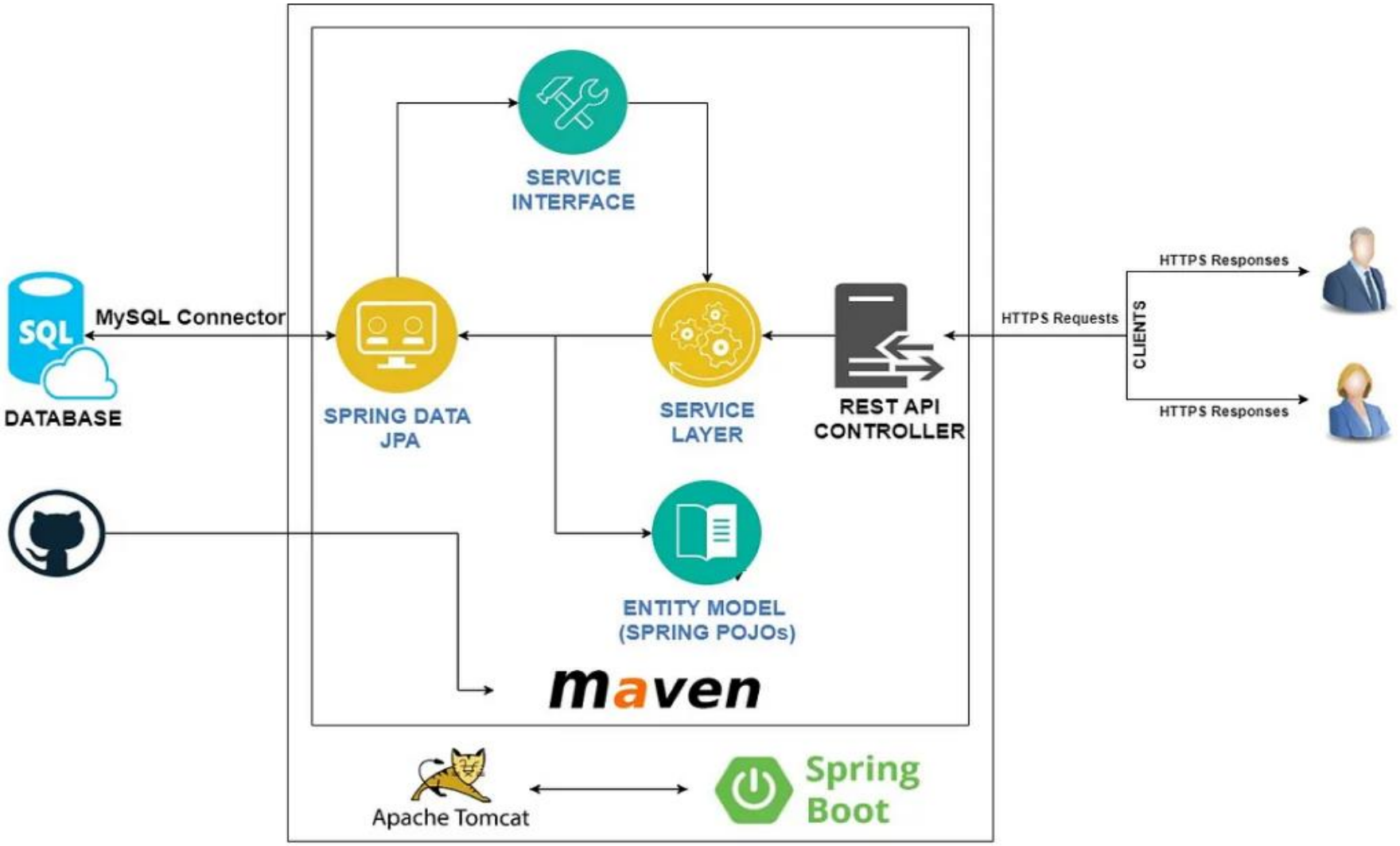


What is Spring Boot?

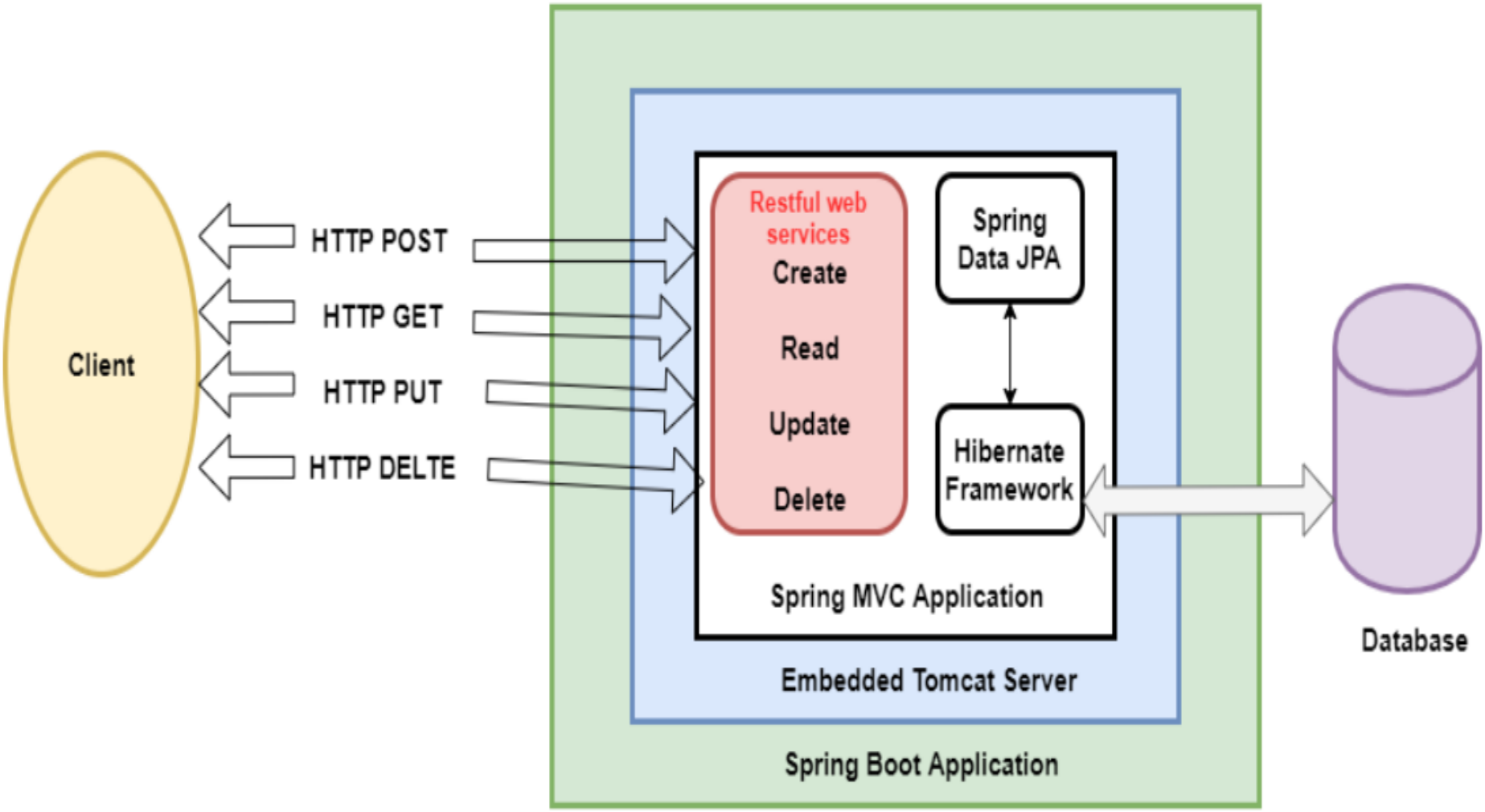
Spring Boot is a robust, production-ready **Java-based back-end framework** that simplifies application development by providing built-in configurations and features like **REST APIs**, database integration with **JPA/Hibernate**, and security services.

Its lightweight nature and modular architecture make it ideal for building scalable, secure server-side applications.

Technical Design (Spring Boot)



Technical Design (Spring Boot)

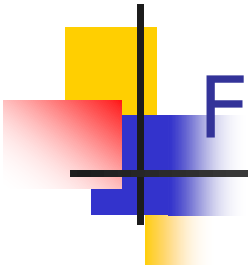


Why Full-Stack Development?

You will learn how to integrate Angular and Spring Boot to develop a complete e-commerce application.

The **front-end Angular application** will communicate with the **back-end Spring Boot application** through **REST APIs**, providing full **CRUD operations** and database management.

This architecture enables you to deliver a seamless, secure, and scalable user experience.



Full-Stack Development

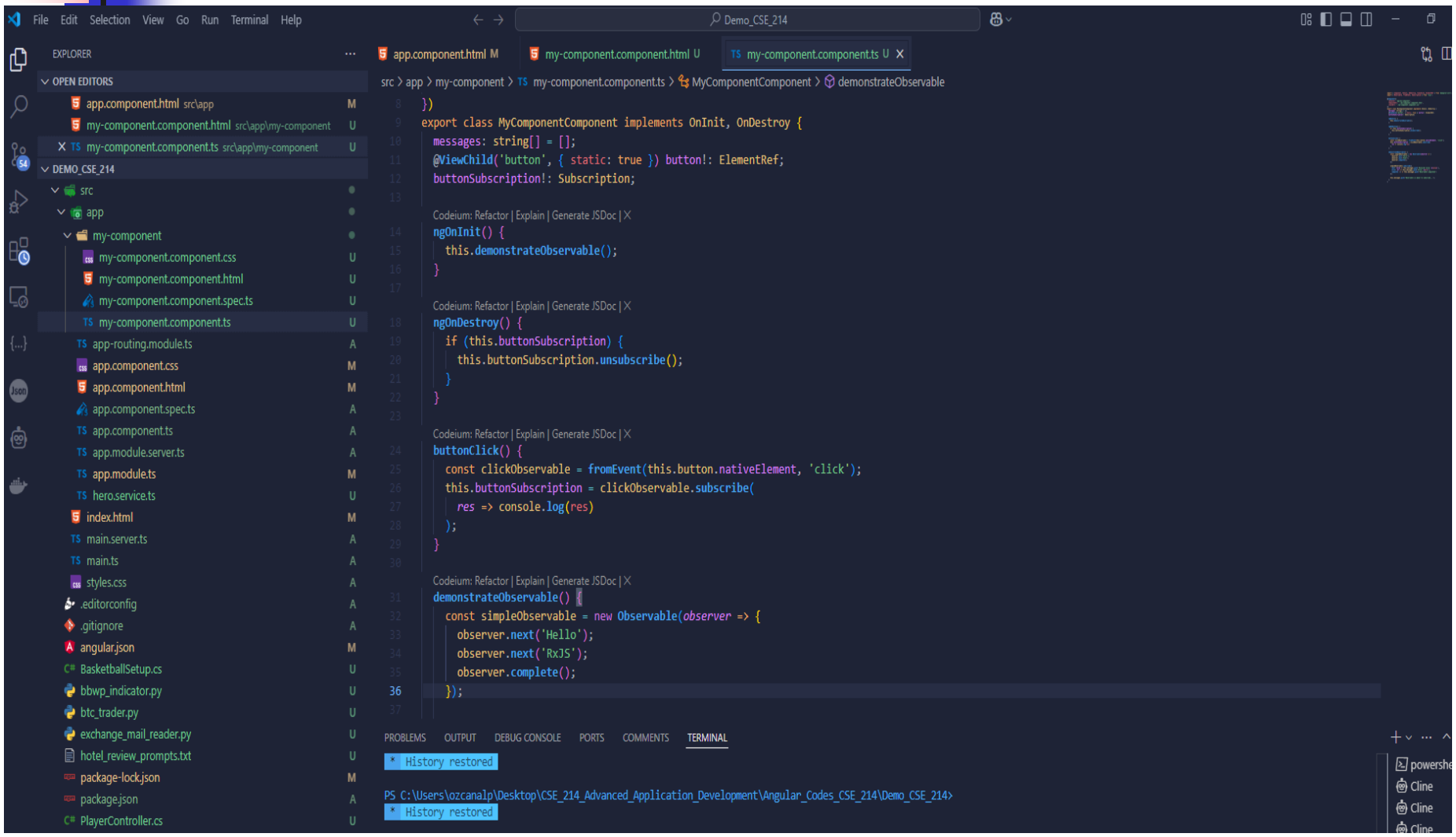
Building dynamic and responsive user interfaces with Angular

Managing back-end operations and databases using Spring Boot

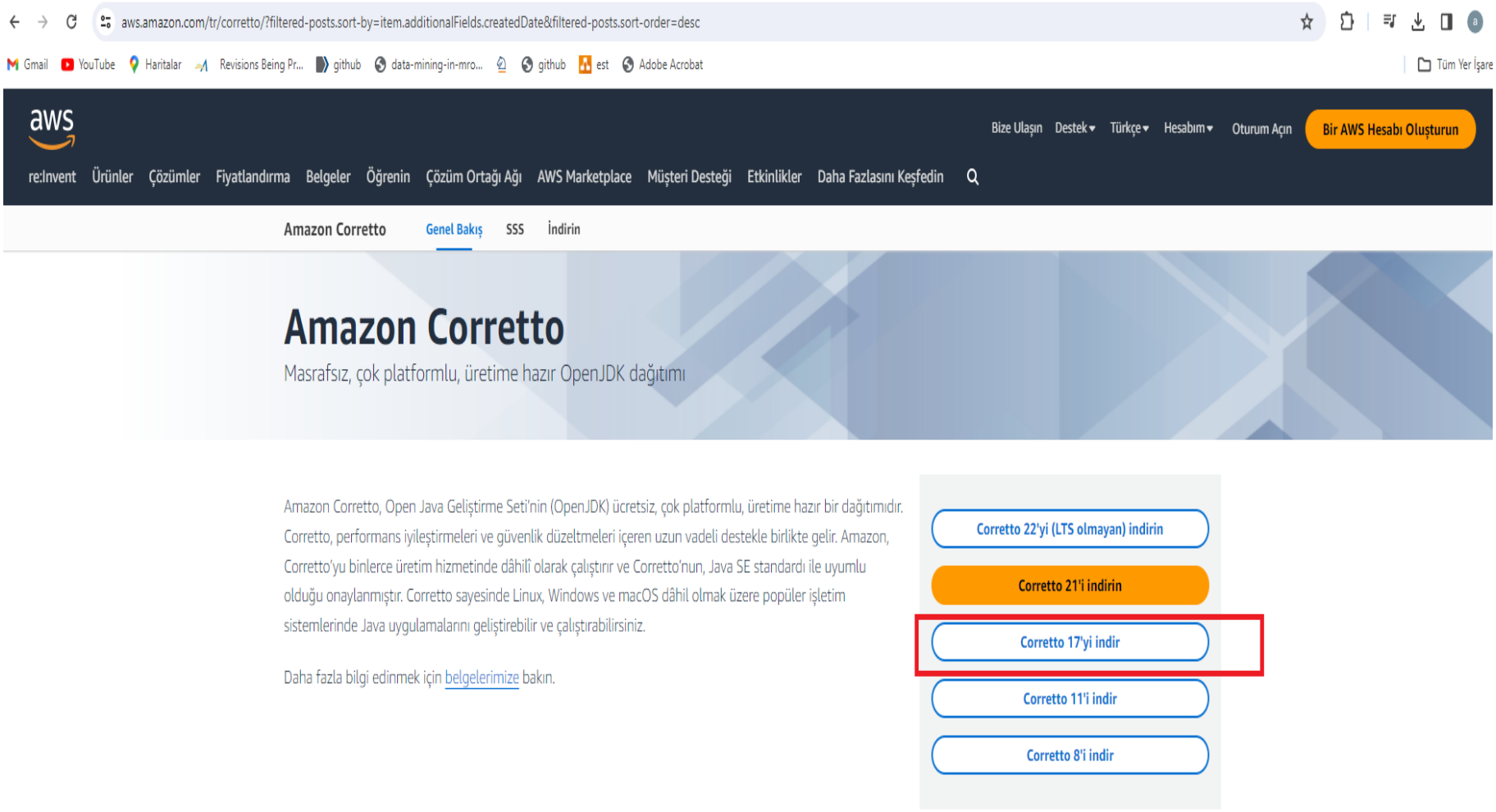
Establishing secure communication between front-end and back-end services

Implementing key features like authentication, routing, and reactive programming

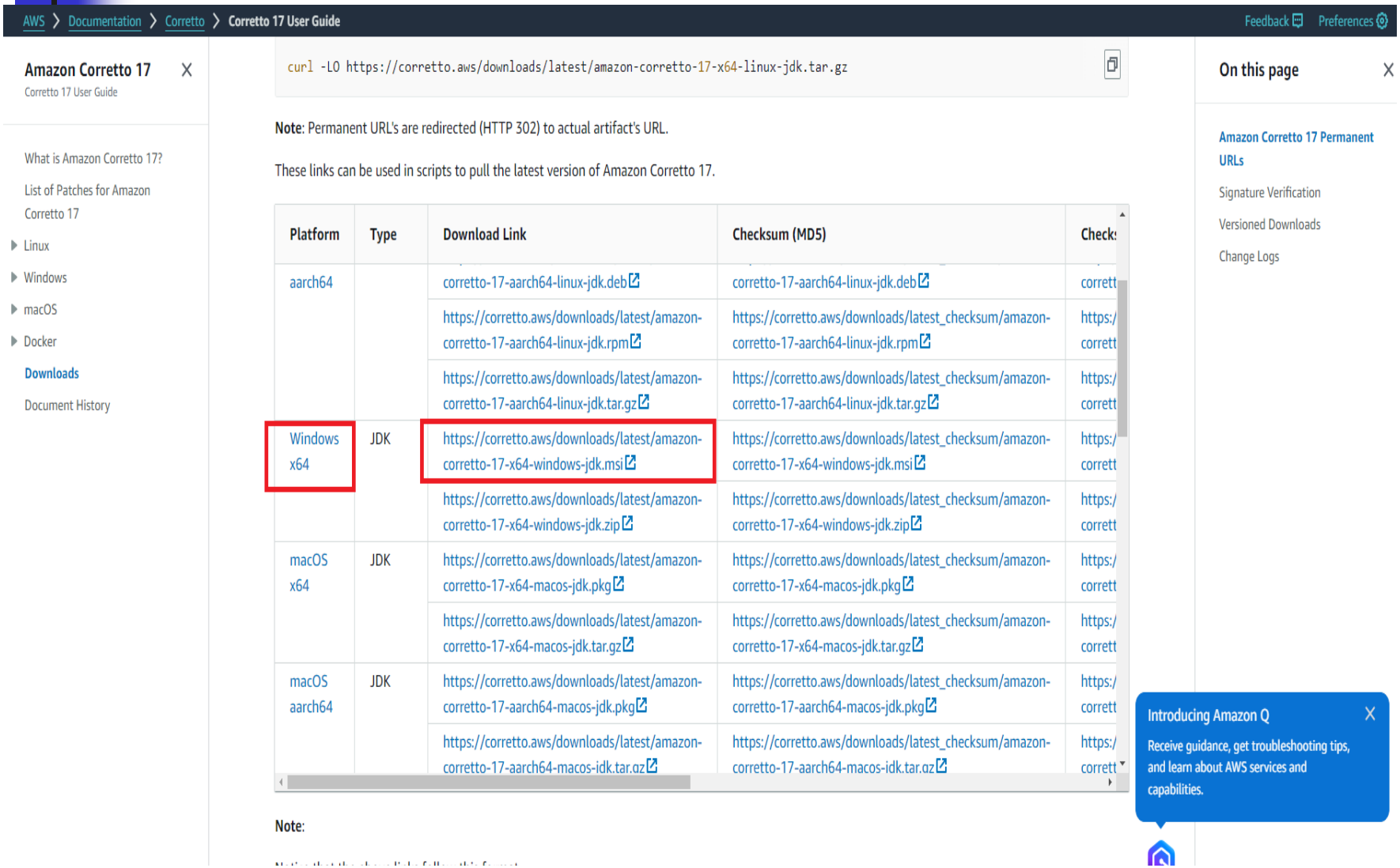
VScode


File Edit Selection View Go Run Terminal Help
Demo_CSE_214
EXPLORER
OPEN EDITORS
app.component.html src\app M
my-component.component.html src\app\my-component U
my-component.component.ts src\app\my-component U
DEMO_CSE_214
src
app
my-component
my-component.component.css U
my-component.component.html U
my-component.component.spec.ts U
my-component.component.ts U
app-routing.module.ts A
app.component.css M
app.component.html M
app.component.spec.ts A
app.component.ts A
app.module.server.ts A
app.module.ts M
hero.service.ts U
index.html M
main.server.ts A
main.ts A
styles.css A
.editorconfig A
.gitignore A
angular.json M
BasketballSetup.cs U
bbwp_indicator.py U
btc_trader.py U
exchange_mail_reader.py U
hotel_review_prompts.txt U
package-lock.json M
package.json A
PlayerController.cs U
app.component.html M
my-component.component.html U
my-component.component.ts U
src > app > my-component > my-component.component.ts > MyComponentComponent > demonstrateObservable
})
export class MyComponentComponent implements OnInit, OnDestroy {
messages: string[] = [];
@ViewChild('button', { static: true }) button!: ElementRef;
buttonSubscription!: Subscription;
Codeium: Refactor | Explain | Generate JSDoc | X
ngOnInit() {
this.demonstrateObservable();
}
Codeium: Refactor | Explain | Generate JSDoc | X
ngOnDestroy() {
if (this.buttonSubscription) {
this.buttonSubscription.unsubscribe();
}
}
Codeium: Refactor | Explain | Generate JSDoc | X
buttonClick() {
const clickObservable = fromEvent(this.button.nativeElement, 'click');
this.buttonSubscription = clickObservable.subscribe(
res => console.log(res)
);
}
Codeium: Refactor | Explain | Generate JSDoc | X
demonstrateObservable() {
const simpleObservable = new Observable(observer => {
observer.next('Hello');
observer.next('RxJS');
observer.complete();
});
PROBLEMS OUTPUT DEBUG CONSOLE PORTS COMMENTS TERMINAL
History restored
PS C:\Users\ozcanalp\Desktop\CSE_214_Advanced_Application_Development\Angular_Codes_CSE_214\Demo_CSE_214>
History restored
powershe
Cline
Cline
Cline

Installing Java



Installing Java



AWS > Documentation > Corretto > Corretto 17 User Guide

Feedback Preferences

Amazon Corretto 17

Corretto 17 User Guide

- What is Amazon Corretto 17?
- List of Patches for Amazon Corretto 17
- Linux
- Windows
- macOS
- Docker
- Downloads
- Document History

```
curl -LO https://corretto.aws/downloads/latest/amazon-corretto-17-x64-linux-jdk.tar.gz
```

Note: Permanent URL's are redirected (HTTP 302) to actual artifact's URL.

These links can be used in scripts to pull the latest version of Amazon Corretto 17.

Platform	Type	Download Link	Checksum (MD5)	Checks
aarch64		corretto-17-aarch64-linux-jdk.deb	corretto-17-aarch64-linux-jdk.deb	corrett
		https://corretto.aws/downloads/latest/amazon-corretto-17-aarch64-linux-jdk.rpm	https://corretto.aws/downloads/latest_checksum/amazon-corretto-17-aarch64-linux-jdk.rpm	https:/ corrett
		https://corretto.aws/downloads/latest/amazon-corretto-17-aarch64-linux-jdk.tar.gz	https://corretto.aws/downloads/latest_checksum/amazon-corretto-17-aarch64-linux-jdk.tar.gz	https:/ corrett
Windows x64	JDK	https://corretto.aws/downloads/latest/amazon-corretto-17-x64-windows-jdk.msi	https://corretto.aws/downloads/latest_checksum/amazon-corretto-17-x64-windows-jdk.msi	https:/ corrett
		https://corretto.aws/downloads/latest/amazon-corretto-17-x64-windows-jdk.zip	https://corretto.aws/downloads/latest_checksum/amazon-corretto-17-x64-windows-jdk.zip	https:/ corrett
macOS x64	JDK	https://corretto.aws/downloads/latest/amazon-corretto-17-x64-macos-jdk.pkg	https://corretto.aws/downloads/latest_checksum/amazon-corretto-17-x64-macos-jdk.pkg	https:/ corrett
		https://corretto.aws/downloads/latest/amazon-corretto-17-x64-macos-jdk.tar.gz	https://corretto.aws/downloads/latest_checksum/amazon-corretto-17-x64-macos-jdk.tar.gz	https:/ corrett
macOS aarch64	JDK	https://corretto.aws/downloads/latest/amazon-corretto-17-aarch64-macos-jdk.pkg	https://corretto.aws/downloads/latest_checksum/amazon-corretto-17-aarch64-macos-jdk.pkg	https:/ corrett
		https://corretto.aws/downloads/latest/amazon-corretto-17-aarch64-macos-idk.tar.gz	https://corretto.aws/downloads/latest_checksum/amazon-corretto-17-aarch64-macos-idk.tar.gz	https:/ corrett

Note:

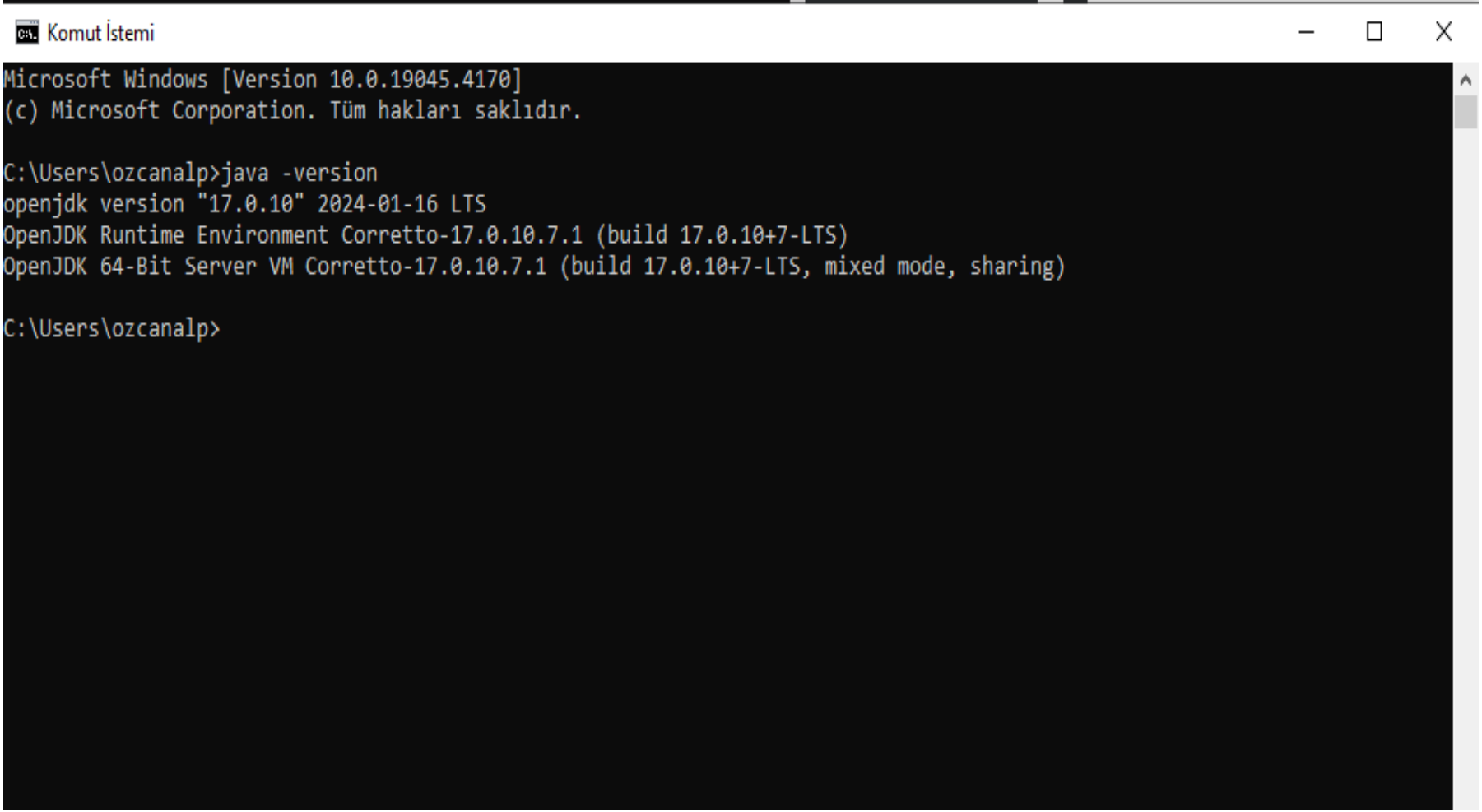
On this page

- Amazon Corretto 17 Permanent URLs
- Signature Verification
- Versioned Downloads
- Change Logs

Introducing Amazon Q

Receive guidance, get troubleshooting tips, and learn about AWS services and capabilities.

Installing Java



Installing Maven

Apache Maven Project
http://maven.apache.org/

Maven™

Apache / Maven / Download Apache Maven

Download | Get Sources | Last Published: 2024-03-26

- Welcome
- License

ABOUT MAVEN
- What is Maven?
- Features
- Download
- Use
- Release Notes

DOCUMENTATION
- Maven Plugins
- Maven Extensions
- Index (category)
- User Centre
- Plugin Developer Centre
- Maven Repository Centre
- Maven Developer Centre
- Books and Resources
- Security

COMMUNITY
- Community Overview
- Project Roles
- How to Contribute
- Getting Help
- Issue Management

Downloading Apache Maven 3.9.6

Apache Maven 3.9.6 is the latest release: it is the recommended version for all users.

System Requirements

Java Development Kit (JDK)	Maven 3.9+ requires JDK 8 or above to execute. It still allows you to build against 1.3 and other JDK versions by using toolchains.
Memory	No minimum requirement
Disk	Approximately 10MB is required for the Maven installation itself. In addition to that, disk space will be used for your local Maven repository. The size of your local repository will vary depending on usage but expect at least 500MB.
Operating System	No minimum requirement. Start up scripts are included as shell scripts (tested on many Unix flavors) and Windows batch files.

Files

Maven is distributed in several formats for your convenience. Simply pick a ready-made binary distribution archive and follow the installation instructions. Use a source archive if you intend to build Maven yourself.

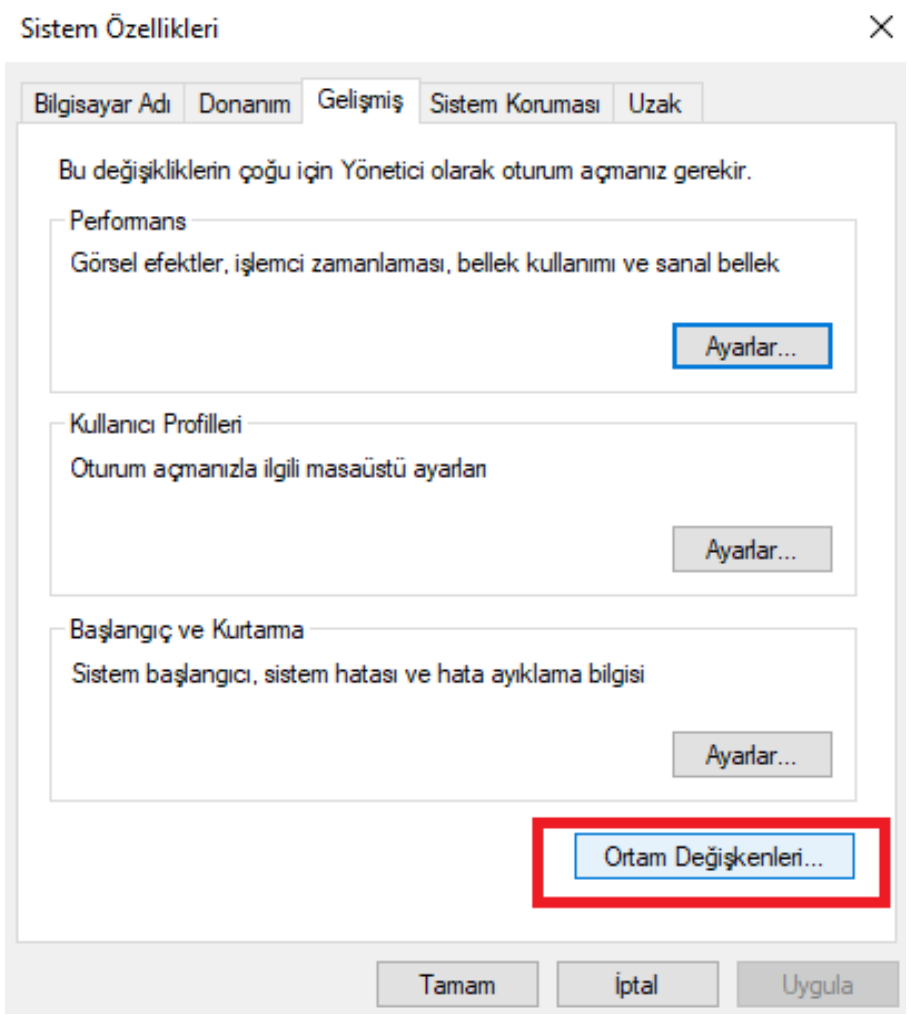
In order to guard against corrupted downloads/installations, it is highly recommended to verify the signature of the release bundles against the public KEYS used by the Apache Maven developers.

	Link	Checksums	Signature
Binary tar.gz archive	apache-maven-3.9.6-bin.tar.gz	apache-maven-3.9.6-bin.tar.gz.sha512	apache-maven-3.9.6-bin.tar.gz.asc
Binary zip archive	apache-maven-3.9.6-bin.zip	apache-maven-3.9.6-bin.zip.sha512	apache-maven-3.9.6-bin.zip.asc
Source tar.gz archive	apache-maven-3.9.6-src.tar.gz	apache-maven-3.9.6-src.tar.gz.sha512	apache-maven-3.9.6-src.tar.gz.asc
Source zip archive	apache-maven-3.9.6-src.zip	apache-maven-3.9.6-src.zip.sha512	apache-maven-3.9.6-src.zip.asc

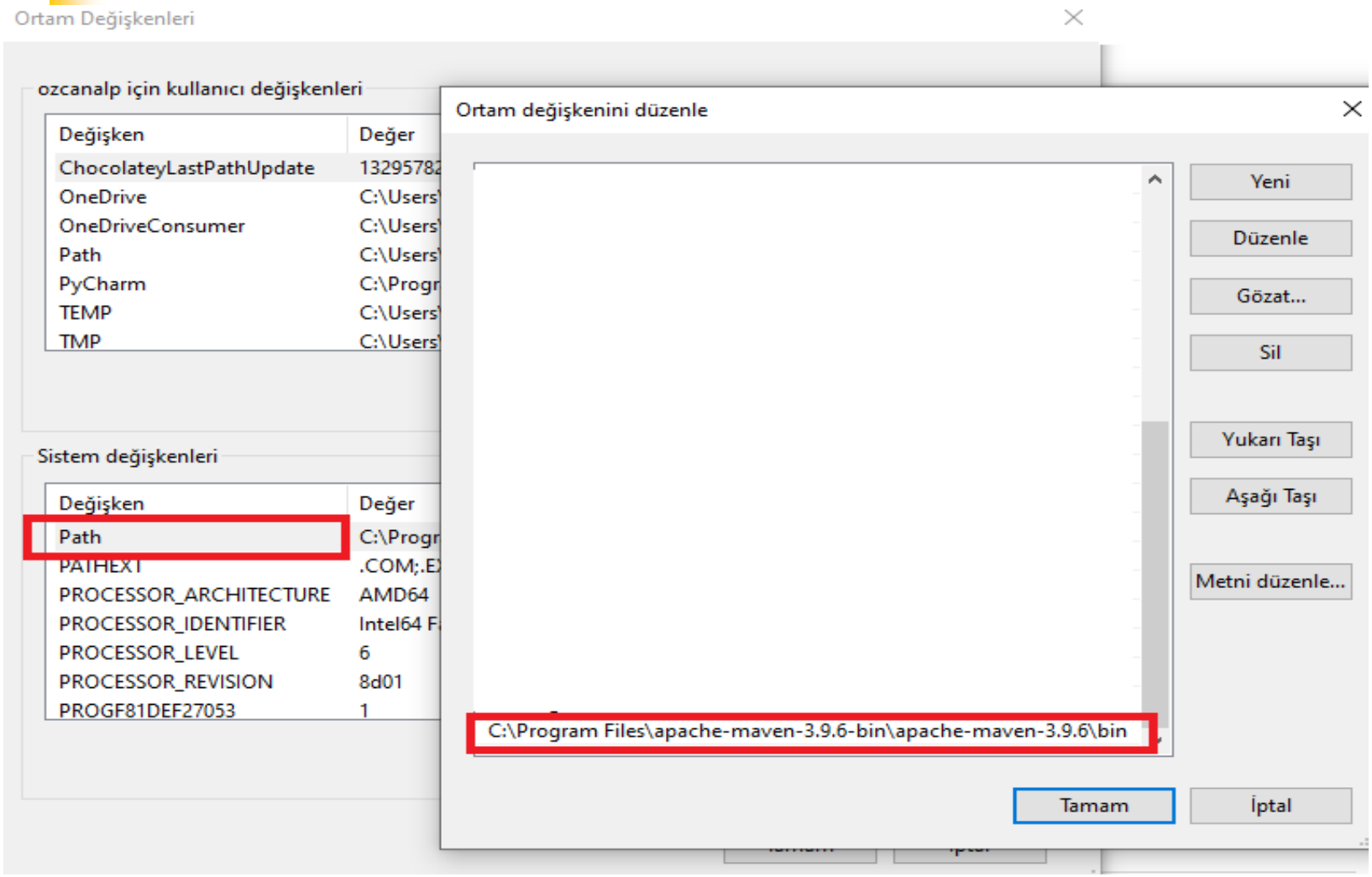
Installing Maven

C:\Program Files\apache-maven-3.9.6-bin

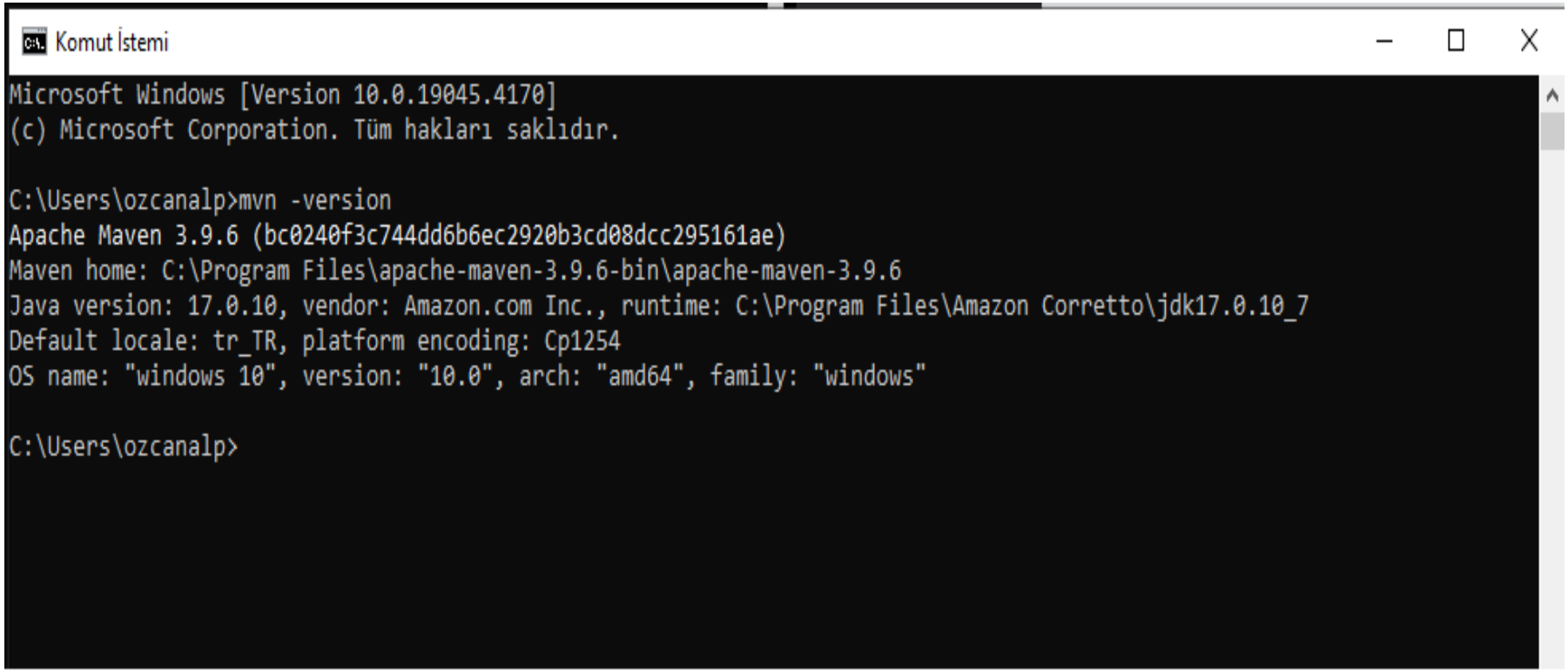
Installing Maven



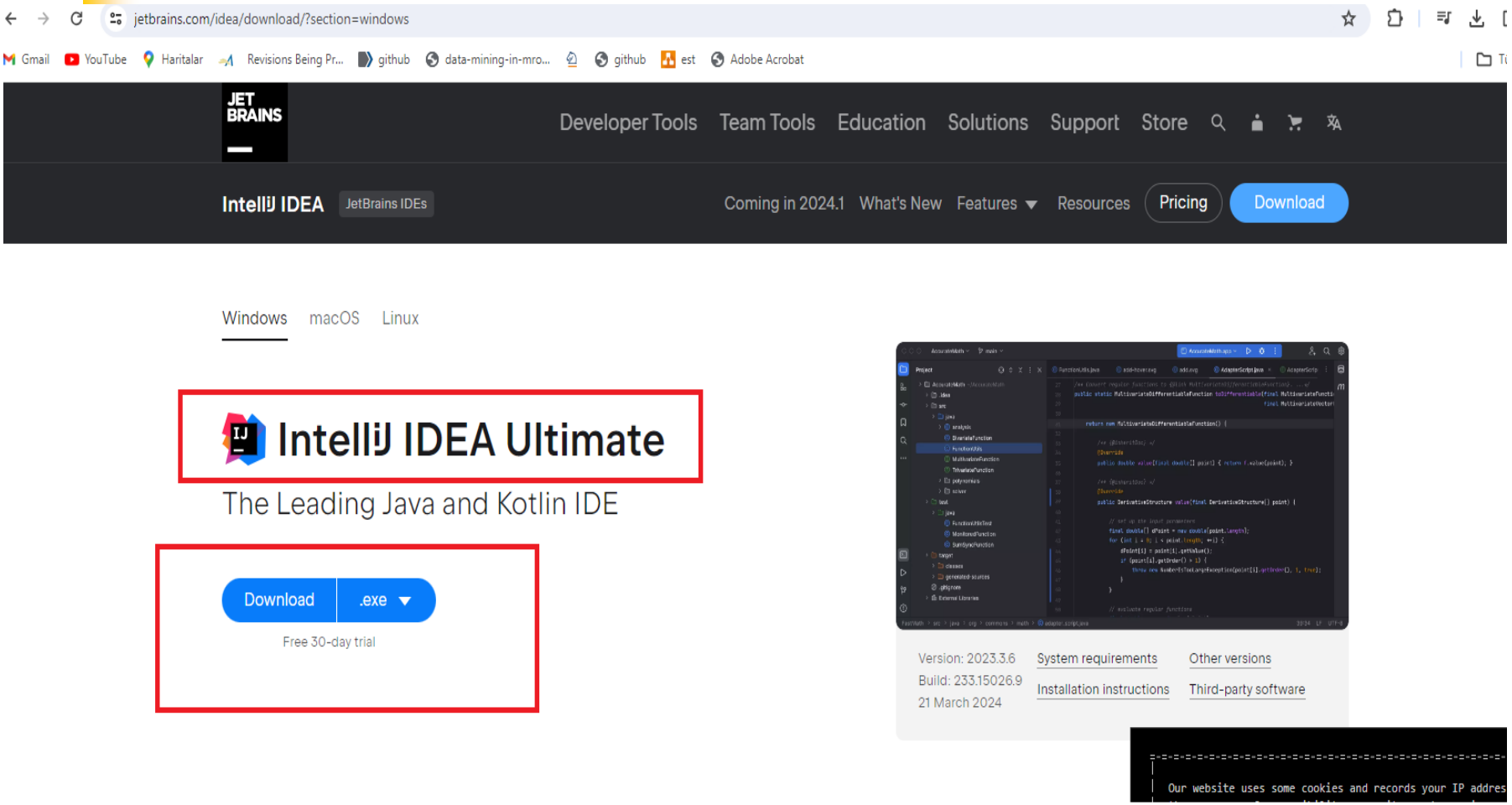
Installing Maven



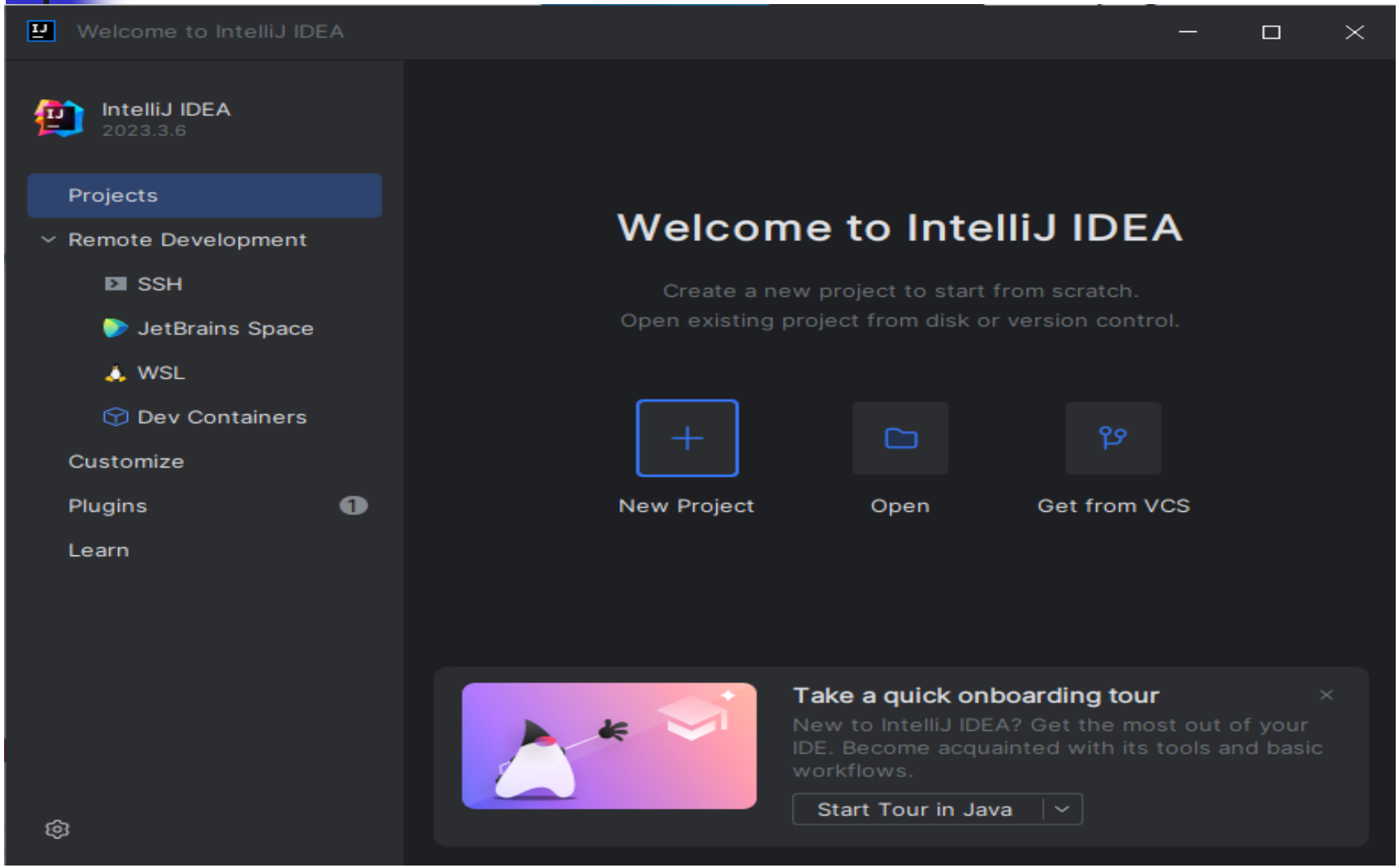
Installing Maven



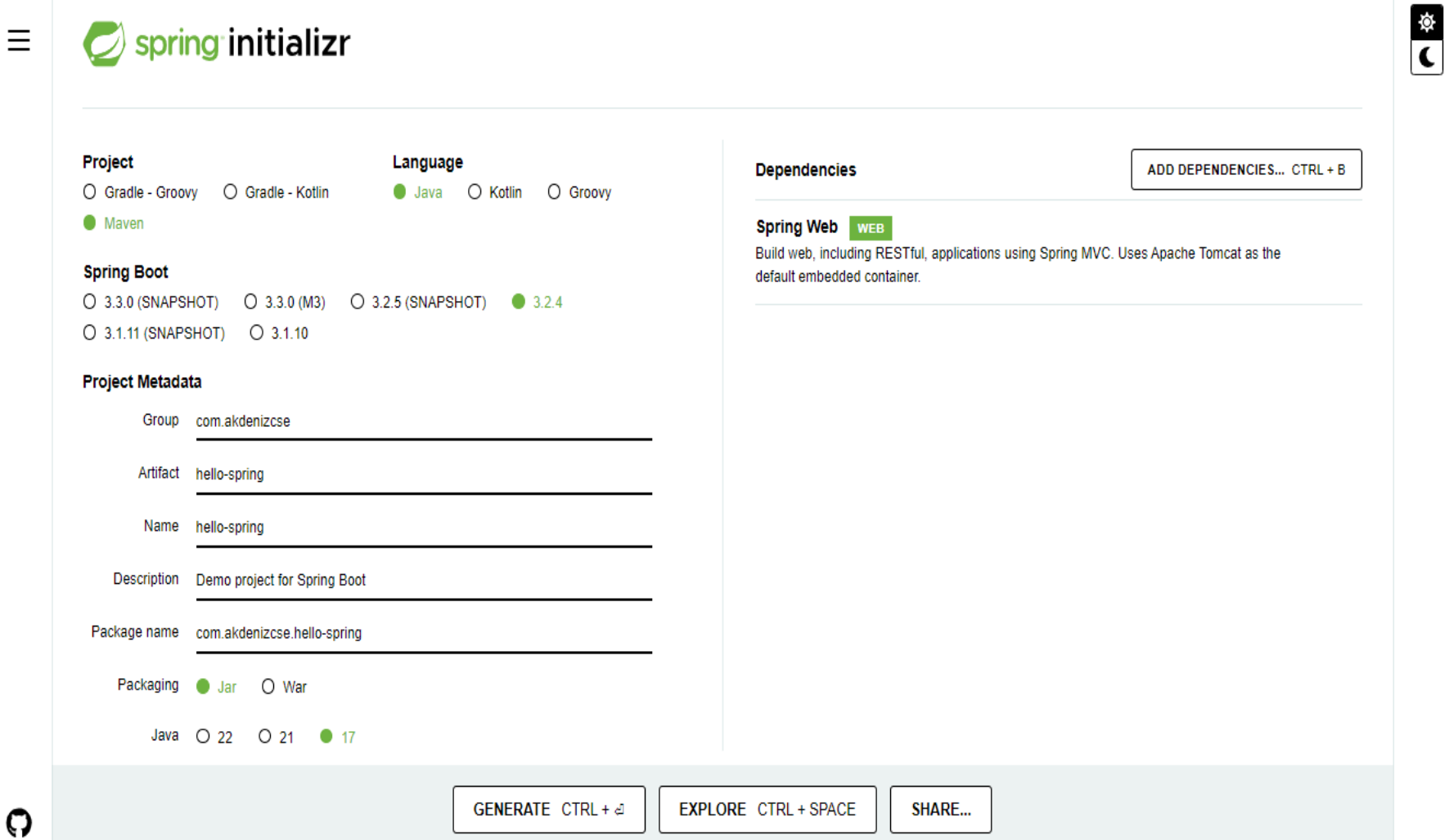
Installing IntelliJ



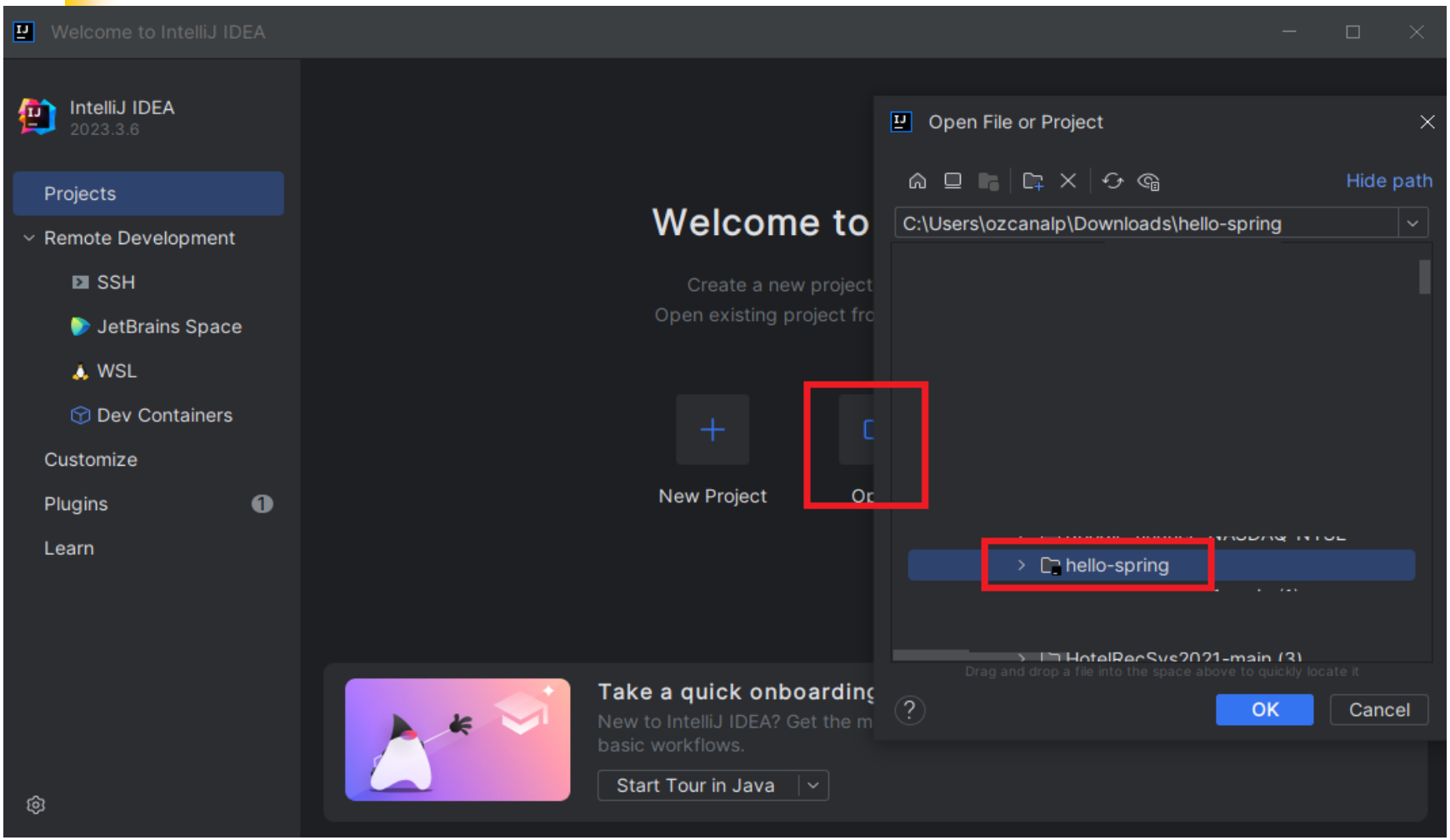
Installing IntelliJ



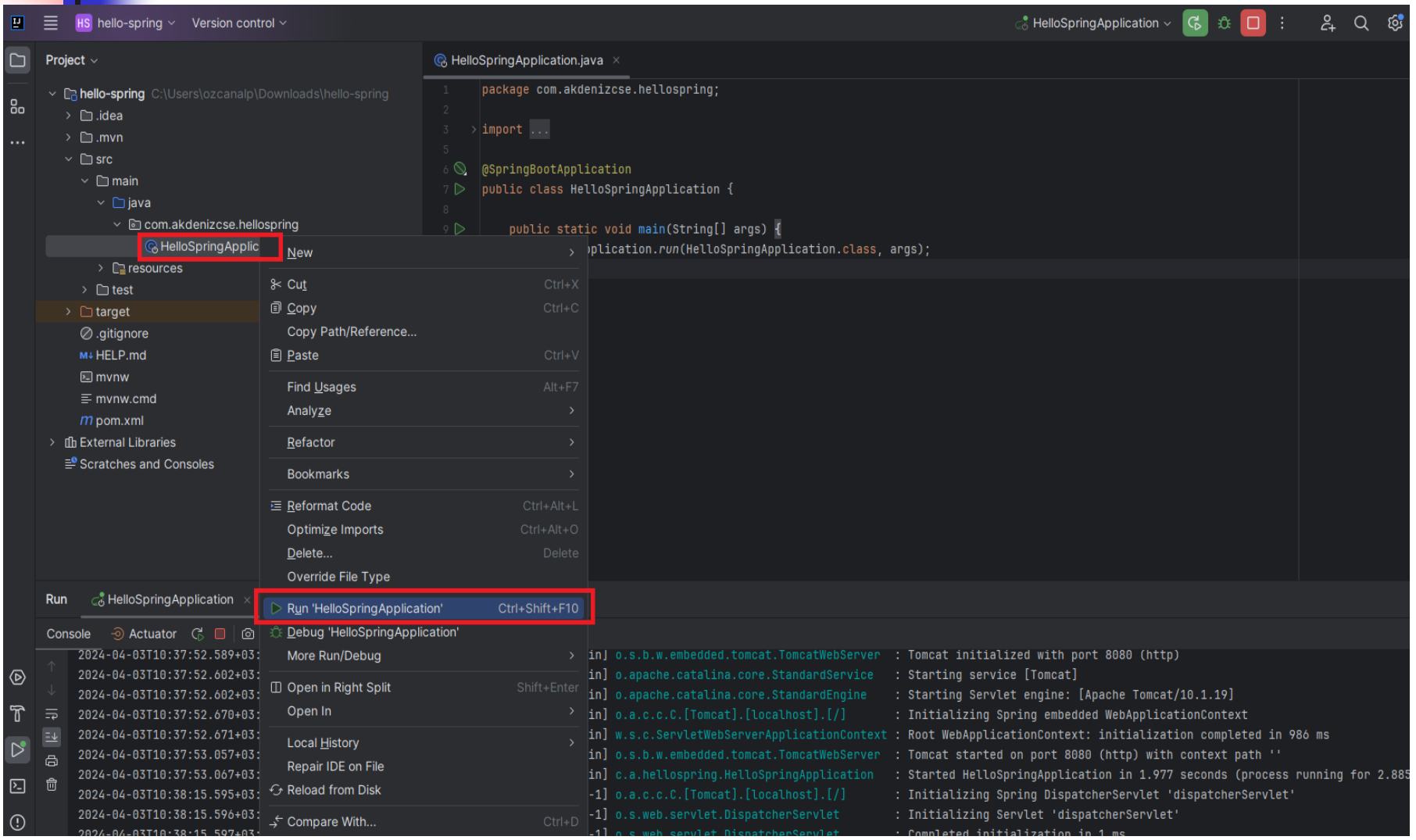
Installing IntelliJ



Installing IntelliJ



Spring Initializer



Spring Initializer

Whitelabel Error Page

This application has no explicit mapping for /error, so you are seeing this as a fallback.

Wed Apr 03 10:38:15 GMT+03:00 2024
There was an unexpected error (type=Not Found, status=404).