Track 1: Java + Angular — Detailed Curriculum (29 Hours Core)

> *Total with AI + Agile + Project: 58 hours*
>
> *Backend: Java 21+ with Spring Boot 3.4*
>
> *Frontend: Angular 20*
>
> *Database: PostgreSQL + MongoDB*
>
> *Target: Final-year BE/BTech/BCA/MCA students*

Module 1: Development Setup & Git (2 Hours)

Session 1 (2 hrs) — Dev Environment & Version Control

Topics:

- Installing JDK 21+, IntelliJ IDEA / VS Code with Java extensions
- Node.js, Angular CLI setup
- PostgreSQL installation & pgAdmin
- Git fundamentals refresher — init, add, commit, branch, merge
- GitHub account setup, SSH keys
- Pull request workflow — branch → commit → push → PR → review → merge
- `.gitignore` for Java/Angular projects
- GitHub Copilot installation and activation

Hands-on:

- Create a Spring Boot project with Spring Initializr and run it
- Create a GitHub repo, make a branch, submit a PR, merge it
- First Copilot interaction — let it autocomplete a simple Java class

---

Module 2: Java & Spring Boot Backend (11 Hours)

Session 2 (2 hrs) — Java Refresher & Spring Boot Introduction

Topics:

- Java 21 features students may not know — records, sealed classes, pattern matching, text blocks, virtual threads
- Collections framework — List, Map, Set, Stream API
- Lambda expressions and functional interfaces
- Spring Boot project setup with Spring Initializr

- Project structure — controllers, services, repositories, models
- `@SpringBootApplication`, `@RestController`, `@GetMapping`
- Running the application, understanding embedded Tomcat

Hands-on:

- Create a Spring Boot project with a "Hello World" REST endpoint
- Use Stream API to filter/map a collection
- Explore project structure

Session 3 (2 hrs) — REST API Fundamentals & Basic Security

Topics:

- REST principles — resources, HTTP methods (GET, POST, PUT, DELETE), status codes
- `@RequestMapping`, `@PathVariable`, `@RequestBody`, `@RequestParam`
- Request/Response DTOs — why not expose entities directly
- Input validation with `@Valid`, `@NotNull`, `@Size`, `@Email`
- Exception handling — `@ControllerAdvice`, `@ExceptionHandler`, custom error responses
- Basic API security — why APIs need protection, authentication vs authorization
- JWT overview — what tokens are, how they secure APIs (detailed setup in Module 5)
- API testing with Postman

Hands-on:

- Build a complete CRUD API for a `Student` resource (create, read, update, delete)
- Add validation — name required, email format, age range
- Custom error responses for validation failures
- Test all endpoints in Postman

Session 4 (2 hrs) — Database Integration with Spring Data JPA

Topics:

- Spring Data JPA setup — `spring-boot-starter-data-jpa`, PostgreSQL driver
- `application.properties` — datasource config, ddl-auto, show-sql
- JPA entities — `@Entity`, `@Id`, `@GeneratedValue`, `@Column`
- Relationships — `@OneToMany`, `@ManyToOne`, `@JoinColumn`
- Spring Data repositories — `JpaRepository`, derived query methods
- Custom queries with `@Query` (JPQL)
- Database migrations concept (Flyway intro)

Hands-on:

- Connect Student API to PostgreSQL
- Add a `Course` entity with `@ManyToOne` to Student
- Write custom query: find students by course name
- Verify data in pgAdmin

Session 5 (2 hrs) — Service Layer Design & Dependency Injection

Topics:

- Service layer design — why separate from controllers
- Dependency Injection — `@Service`, `@Autowired`, constructor injection
- Interface-based design — service interfaces and implementations, testability benefits
- Repository pattern — generic repository concepts
- Project structure — organizing by feature vs by layer
- `@Qualifier` and `@Primary` — resolving multiple implementations

Hands-on:

- Refactor Student API to proper service layer pattern
- Implement interface-based services with DI
- Create multiple implementations and switch with `@Qualifier`
- Reorganize project structure by feature

Session 6 (2 hrs) — Pagination, Filtering & Data Mapping

Topics:

- MapStruct — entity-to-DTO mapping, mapper interfaces, custom mappings
- Pagination and sorting — `Pageable`, `Page<T>`, `Sort`, query parameters
- Dynamic filtering — specifications pattern with `Specification<T>`
- Search across fields — combining multiple filter criteria
- Response envelope — standardized API response format (`ApiResponse<T>`)
- `@ControllerAdvice` for consistent response wrapping

Hands-on:

- Add MapStruct for entity-DTO mapping
- Add pagination and sorting to list endpoints
- Add dynamic filtering (by name, course, date range) with specifications
- Implement standardized response envelope
- Build a search endpoint that combines multiple filters

Session 7 (2 hrs) — Testing & Code Quality

Topics:

- Why testing matters — confidence, refactoring safety, CI/CD readiness
- Unit testing with JUnit 5 — `@Test`, `@DisplayName`, `@ParameterizedTest`, assertions
- Mocking with Mockito — `@Mock`, `@InjectMocks`, `when().thenReturn()`, `verify()`
- Testing service layer — isolate business logic from infrastructure
- Testing controllers — `@WebMvcTest`, `MockMvc`
- Test organization — unit vs integration, naming conventions
- Code coverage basics — what to measure, what coverage is meaningful

Hands-on:

- Write 5 unit tests — service layer tests with mocked repository
- Write 2 controller tests with `@WebMvcTest`

- Test CRUD operations end-to-end
- Practice test-first: write a test, then implement the feature

Session 8 (1 hr) — API Documentation & Best Practices

Topics:

- OpenAPI / Swagger — `springdoc-openapi` setup
- Documenting endpoints with annotations
- API versioning strategies
- CORS configuration for frontend integration
- Environment profiles — dev, test, prod
- Logging best practices — SLF4J, log levels

Hands-on:

- Add Swagger UI to the project
- Document all Student API endpoints
- Configure CORS for `http://localhost:4200` (Angular dev server)

Module 3: Database Deep Dive (3 Hours)

Session 9 (2 hrs) — PostgreSQL Mastery

Topics:

- Schema design principles — normalization (1NF, 2NF, 3NF), when to denormalize
- Data types — VARCHAR, INTEGER, BOOLEAN, TIMESTAMP, JSONB, UUID
- Constraints — PRIMARY KEY, FOREIGN KEY, UNIQUE, CHECK, NOT NULL
- Indexes — B-tree, when to create, covering indexes, `EXPLAIN ANALYZE`
- Complex queries — JOINs (INNER, LEFT, RIGHT), subqueries, CTEs (`WITH`)
- Aggregate functions — GROUP BY, HAVING, window functions intro
- Transactions — ACID properties, isolation levels

Hands-on:

- Design a schema for the capstone project (3-4 tables with relationships)
- Write 5 complex queries with JOINs and aggregations
- Create indexes and compare `EXPLAIN ANALYZE` before/after
- Practice with CTEs for readable complex queries

Session 10 (1 hr) — Advanced ORM Patterns

Topics:

- N+1 query problem — what it is, how to detect, how to fix
- Fetch strategies — LAZY vs EAGER loading, `@EntityGraph`
- Projections — interface-based, DTO-based for performance
- Auditing — `@CreatedDate`, `@LastModifiedDate`, `@CreatedBy`

- Connection pooling — HikariCP configuration

Hands-on:

- Identify and fix N+1 query in Student-Course relationship
- Implement audit fields on all entities
- Compare JPA generated SQL vs hand-written for a report query

---

Module 4: Angular Frontend (9 Hours)

Session 11 (2 hrs) — Angular Fundamentals

Topics:

- Angular architecture — modules, components, templates, services
- Angular CLI — `ng new`, `ng generate`, `ng serve`
- TypeScript essentials for Angular — types, interfaces, decorators, generics
- Components — `@Component`, templates, styles, lifecycle hooks
- Data binding — interpolation `{{ }}`, property `[attr]`, event `(click)`, two-way `[(ngModel)]`
- Directives — `*ngIf`, `*ngFor`, `[ngClass]`, `[ngStyle]`
- Pipes — built-in pipes, custom pipes

Hands-on:

- Create Angular project, generate 3 components
- Build a student list with `*ngFor`, conditional display with `*ngIf`
- Create a custom pipe (e.g., capitalize name, format date)

Session 12 (2 hrs) — Routing & Navigation

Topics:

- Angular Router — `RouterModule`, route configuration, `<router-outlet>`
- Route parameters — `ActivatedRoute`, `:id` params
- Lazy loading modules — `loadChildren`, performance benefits
- Route guards — `CanActivate`, `CanDeactivate`, auth guards
- Navigation — `routerLink`, programmatic navigation with `Router`
- Layout components — shared header, sidebar, footer
- Child routes and nested layouts

Hands-on:

- Set up routing: Dashboard, Student List, Student Detail, Login
- Implement lazy-loaded feature modules
- Create an auth guard that redirects unauthenticated users to login
- Build a layout with header navigation

Session 13 (2 hrs) — State Management & Forms

Topics:

- Reactive Forms — `FormGroup`, `FormControl`, `FormBuilder`, validators
- Custom validators — sync and async
- Form submission and error display patterns
- Component communication — `@Input()`, `@Output()`, `EventEmitter`
- Services as state containers — `BehaviorSubject`, shared state
- RxJS essentials — `Observable`, `Subject`, `pipe`, `map`, `filter`, `switchMap`

Hands-on:

- Build a student registration form with reactive forms and validation
- Display inline validation errors
- Create a shared auth service with `BehaviorSubject` for login state
- Use RxJS operators to transform API responses

Session 14 (2 hrs) — HTTP & API Integration

Topics:

- `HttpClient` — GET, POST, PUT, DELETE
- Interceptors — adding JWT token to every request
- Error handling — global error interceptor, user-friendly messages
- Loading states — spinners, skeleton screens
- Environment configuration — `environment.ts` for API URLs
- CORS — understanding and debugging cross-origin issues
- File uploads — `FormData`, multipart requests

Hands-on:

- Connect Angular to Spring Boot API
- Implement JWT auth flow — login, store token, attach to requests
- Build Student CRUD screens — list (with pagination), create form, edit, delete
- Add loading spinner and error toast notifications

Session 15 (1 hr) — UI Polish & Responsive Design

Topics:

- Angular Material or Tailwind CSS — quick setup for professional UI
- Responsive layouts — CSS Grid, Flexbox, media queries
- Table components — sorting, filtering, pagination
- Charts — ng2-charts or Chart.js for dashboards
- Build optimization — `ng build --prod`, bundle analysis
- Deployment basics — building for production, static hosting

Hands-on:

- Style the student management app with Angular Material or Tailwind
- Add a dashboard with 2 chart widgets (e.g., students per course, enrollment trends)
- Verify responsive design on mobile viewport

- Run production build

Module 5: Authentication & Authorization (2 Hours)

Session 16 (2 hrs) — Full Authentication & Authorization

> *Taught after frontend so students can implement the complete auth flow end-to-end (backend + frontend).*

Topics:

- Why authentication matters — stateless APIs, tokens vs sessions
- JWT (JSON Web Tokens) — structure (header, payload, signature), how they work
- Spring Security setup — `SecurityFilterChain`, `UserDetailsService`
- Password hashing with BCrypt
- JWT generation, validation, and filter chain
- Role-based access — `@PreAuthorize("hasRole('ADMIN')")`, `@Secured`
- Protecting endpoints — public vs authenticated vs admin-only
- Frontend auth flow — login page, token storage, route guards, interceptors

Hands-on:

- Add user registration and login endpoints to backend
- Implement JWT token generation and validation
- Protect Student CRUD — only authenticated users can create/update/delete
- Add ADMIN role — only admins can delete students
- Connect Angular login page to backend auth API
- Test full flow: register → login → get token → access protected routes

---

Module 6: AI Tools for Developers (15 Hours)

> *See AI-TOOLS-MODULE.md for detailed curriculum.*

> *This module is shared across all three tracks.*

Module 7: Agile & Scrum Workshop (2 Hours)

> *See CAPSTONE-PROJECT.md for the Agile workshop session.*

> *Teaches Scrum framework, user stories, estimation, GitHub Projects setup.*

> *Shared across all three tracks.*

Module 8: Capstone Project Sprint (12 Hours)

> *See CAPSTONE-PROJECT.md for project guidelines and sprint structure.*

> *Track 1 stack: Java (Spring Boot) + Angular + PostgreSQL + AI features*

> *Run in 2 real Agile sprints with standups, feature branches, PR reviews, and retrospective.*

Track 1 Summary

Module	Hours	Key Technologies
Dev Setup & Git	2	JDK 21, IntelliJ/VS Code, Git, GitHub, Copilot
Java & Spring Boot	13	Spring Boot 3.4, JPA, JUnit 5, Mockito, Swagger
Database	3	PostgreSQL, Flyway, HikariCP, Spring Data
Angular	9	Angular 20, TypeScript, RxJS, Reactive Forms, Angular Material
Authentication	2	JWT, Spring Security, role-based access
AI Tools	15	Copilot, ChatGPT, Prompt Eng, OpenAI API, RAG, MCP
Agile & Scrum	2	Scrum, user stories, estimation, GitHub Projects
Capstone Project	12	Full-stack app in Agile sprints with AI features
Total	58	