MSc CA SEM-1 — Course 103-01: Web Development Frameworks (Angular + Express.js) Full Step-by-Step Solutions (Q1–Q5)

This document contains complete, step-by-step solutions for the practical paper 103-01 (Web Development Frameworks - Angular + Express.js). It includes exact file contents, terminal commands (PowerShell-friendly), how to run both backend and frontend, testing instructions (Postman / browser), and expected outputs. Follow the steps exactly.

Prerequisites

```
1. Install Node.js (LTS) from https://nodejs.org
2. Install Angular CLI globally (optional but recommended):
   npm install -g @angular/cli
3. Install Postman or use Thunder Client in VS Code for API testing.

Check versions:
  node -v
  npm -v
  ng version
```

Workspace & Overview

```
Project structure we will create:
workspace/
 ■■ backend/          (Express API)
 ■   ■■ server.js
 ■■ frontend/         (Angular app)
     ■■ src/
     ■  ■■ app/
     ■  ■  ■■ product/
     ■  ■  ■■ student-form/
     ■  ■  ■■ users/
     ■  ■  ■■ counter-app/
     ■  ■  ■■ todo-app/
     ■  ■  ■■ home/
     ■  ■  ■■ about/
     ■  ■  ■■ app-routing.module.ts
```

Backend — Express.js (Q5) — server.js

```
/* backend/server.js */
const express = require('express');
const cors = require('cors');

const app = express();
app.use(cors());
app.use(express.json()); // parse JSON body

let students = [
```

```
  { id: 1, name: "Rahul", marks: 80 },
  { id: 2, name: "Meena", marks: 25 }
];

// GET /students
app.get('/students', (req, res) => {
  res.json(students);
});

// POST /students
app.post('/students', (req, res) => {
  const { name, marks } = req.body;
  if (!name || typeof marks === 'undefined') return res.status(400).json({ error: 'name and marks r
  const id = students.length ? Math.max(...students.map(s => s.id)) + 1 : 1;
  const student = { id, name, marks };
  students.push(student);
  res.status(201).json(student);
});

// PUT /students/:id
app.put('/students/:id', (req, res) => {
  const id = parseInt(req.params.id);
  const { name, marks } = req.body;
  const idx = students.findIndex(s => s.id === id);
  if (idx === -1) return res.status(404).json({ error: 'Student not found' });
  students[idx] = { id, name: name ?? students[idx].name, marks: typeof marks === 'undefined' ? stu
  res.json(students[idx]);
});

// DELETE /students/:id
app.delete('/students/:id', (req, res) => {
  const id = parseInt(req.params.id);
  const before = students.length;
  students = students.filter(s => s.id !== id);
  if (students.length === before) return res.status(404).json({ error: 'Student not found' });
  res.json({ message: 'Deleted' });
});

// start server
const PORT = process.env.PORT || 3000;
app.listen(PORT, () => console.log(`Server running on http://localhost:${PORT}`));
```

Backend — Commands to create and run

```
Open PowerShell and run:
mkdir workspace
cd workspace
mkdir backend
cd backend
npm init -y
npm i express cors

Create server.js with the content above.

Start server:
node .\server.js
```

```
Expected output: Server running on http://localhost:3000

Test endpoints using Postman/Thunder Client:
GET    http://localhost:3000/students
POST   http://localhost:3000/students   (JSON body {"name":"Ankit","marks":55})
PUT    http://localhost:3000/students/2 (JSON body {"name":"Meena Updated","marks":45})
DELETE http://localhost:3000/students/2
```

Frontend — Create Angular Project

```
From workspace folder run:
ng new frontend --routing=true --style=css
cd frontend
npm install

Start dev server later with:
ng serve --open
```

app.module.ts (important imports & declarations)

```
Ensure these imports and declarations exist in src/app/app.module.ts:

import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule } from '@angular/forms';
import { HttpClientModule } from '@angular/common/http';
import { AppRoutingModule } from './app-routing.module';

@NgModule({
  declarations: [
    AppComponent,
    ProductComponent,
    StudentFormComponent,
    UsersComponent,
    CounterAppComponent,
    TodoAppComponent,
    HomeComponent,
    AboutComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    FormsModule,
    HttpClientModule
  ],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

app.component.ts and app.component.html (show all sections + routing)

```
src/app/app.component.ts
```

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

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  productList = [
    { id: 1, name: 'Pen', price: 20 },
    { id: 2, name: 'Notebook', price: 150 },
    { id: 3, name: 'Pencil', price: 8 },
    { id: 4, name: 'Bag', price: 950 },
     { id: 5, name: 'Coffee Mug', price: 120 }
  ];
}
```

src/app/app.component.html

```
<nav style="padding:12px; background:#f8f9fa; border-bottom:1px solid #e0e0e0;">
  <a routerLink="/home" routerLinkActive="active" style="margin-right:16px; text-decoration:none;">
  <a routerLink="/about" routerLinkActive="active" style="margin-right:16px; text-decoration:none;"
  <a href="#products" style="margin-right:16px; text-decoration:none;">Products</a>
  <a href="#form" style="margin-right:16px; text-decoration:none;">Student Form</a>
  <a href="#counter" style="margin-right:16px; text-decoration:none;">Counter</a>
  <a href="#users" style="margin-right:16px; text-decoration:none;">Users</a>
  <a href="#todos" style="margin-right:16px; text-decoration:none;">Todos</a>
</nav>

<main style="padding:20px; max-width:1000px; margin:0 auto;">

  <router-outlet></router-outlet>

  <section id="products" style="margin-top:20px;">
    <h2>Products</h2>
    <app-product [products]="productList"></app-product>
  </section>

  <hr style="margin:24px 0;" />

  <section id="form" style="margin-top:20px;">
    <h2>Student Registration (Template-driven Form)</h2>
    <app-student-form></app-student-form>
  </section>

  <hr style="margin:24px 0;" />

  <section id="counter" style="margin-top:20px;">
    <h2>Counter App</h2>
    <app-counter-app></app-counter-app>
  </section>

  <hr style="margin:24px 0;" />

  <section id="users" style="margin-top:20px;">
    <h2>API Users (jsonplaceholder)</h2>
```

```
    <app-users></app-users>
  </section>

  <hr style="margin:24px 0;" />

  <section id="todos" style="margin-top:20px;">
    <h2>Todo App</h2>
    <app-todo-app></app-todo-app>
  </section>

</main>
```

Product Component (Q1) — Files

src/app/product/product.component.ts

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

@Component({
  selector: 'app-product',
  templateUrl: './product.component.html',
  styleUrls: ['./product.component.css']
})
export class ProductComponent {
  @Input() products: {id:number, name:string, price:number}[] = [];
}
```

src/app/product/product.component.html

```
<h3>Products</h3>
<table style="width:100%; border-collapse: collapse;">
  <thead>
    <tr>
      <th style="border:1px solid #ddd;padding:8px">ID</th>
      <th style="border:1px solid #ddd;padding:8px">Name</th>
      <th style="border:1px solid #ddd;padding:8px">Price</th>
    </tr>
  </thead>
  <tbody>
    <tr *ngFor="let p of products" [ngStyle]="{ color: p.price < 100 ? 'red' : 'black' }">
      <td style="border:1px solid #ddd;padding:8px">{{p.id}}</td>
      <td style="border:1px solid #ddd;padding:8px">{{p.name}}</td>
      <td style="border:1px solid #ddd;padding:8px">{{p.price}}</td>
    </tr>
  </tbody>
</table>
```

Student Form Component (Q2) — Files

src/app/student-form/student-form.component.ts

```
import { Component } from '@angular/core';
import { NgForm } from '@angular/forms';
@Component({
```

```
  selector: 'app-student-form',
  templateUrl: './student-form.component.html'
})
export class StudentFormComponent {
  submittedData: any = null;

  onSubmit(form: NgForm) {
    if (form.valid) {
      this.submittedData = form.value;
      form.reset();
    }
  }
}
```

src/app/student-form/student-form.component.html

```
<form #f="ngForm" (ngSubmit)="onSubmit(f)">
  <div>
    <label>Name</label>
    <input name="fullName" ngModel required minlength="2" #name="ngModel" />
    <div *ngIf="name.invalid && name.touched" style="color:red">Name is required (min 2 chars).</di
  </div>

  <div>
    <label>Email</label>
    <input name="email" ngModel required email #email="ngModel" />
    <div *ngIf="email.invalid && email.touched" style="color:red">Enter a valid email.</div>
  </div>

  <div>
    <label>Mobile</label>
    <input name="mobile" ngModel required pattern="^[0-9]{10}$" #mobile="ngModel" />
    <div *ngIf="mobile.invalid && mobile.touched" style="color:red">Mobile must be 10 digits.</div>
  </div>

  <div>
    <label>Age</label>
    <input name="age" ngModel required type="number" min="18" max="60" #age="ngModel" />
    <div *ngIf="age.invalid && age.touched" style="color:red">Age must be between 18 and 60.</div>
  </div>

  <button type="submit" [disabled]="f.invalid">Submit</button>
</form>

<div *ngIf="submittedData">
  <h4>Submitted:</h4>
  <pre>{{ submittedData | json }}</pre>
</div>
```

Users Component & Service (Q3) — Files

src/app/user.service.ts

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
```

```
import { catchError, Observable, throwError } from 'rxjs';

@Injectable({ providedIn: 'root' })
export class UserService {
  private url = 'https://jsonplaceholder.typicode.com/users';
  constructor(private http: HttpClient) {}

  getUsers(): Observable<any> {
    return this.http.get(this.url).pipe(
      catchError(err => {
        return throwError(() => new Error('Error loading users'));
      })
    );
  }
}
```

src/app/users/users.component.ts

```
import { Component, OnInit } from '@angular/core';
import { UserService } from '../user.service';

@Component({ selector: 'app-users', templateUrl: './users.component.html' })
export class UsersComponent implements OnInit {
  users: any[] = [];
  loading = true;
  error: string | null = null;

  constructor(private userService: UserService) {}

  ngOnInit(): void {
    this.userService.getUsers().subscribe({
      next: data => { this.users = data; this.loading = false; },
     error: err => { this.error = err.message || 'Error loading users'; this.loading = false; }
    });
  }
}
```

src/app/users/users.component.html

```
<div *ngIf="loading">Loading...</div>
<div *ngIf="error" style="color:red">{{error}}</div>

<table *ngIf="!loading && !error" style="width:100%; border-collapse:collapse">
  <thead>
    <tr><th style="border:1px solid #ddd; padding:8px">Name</th><th style="border:1px solid #ddd; p
  </thead>
  <tbody>
    <tr *ngFor="let u of users">
      <td style="border:1px solid #ddd; padding:8px">{{u.name}}</td>
      <td style="border:1px solid #ddd; padding:8px">{{u.username}}</td>
      <td style="border:1px solid #ddd; padding:8px">{{u.email}}</td>
    </tr>
  </tbody>
</table>
```

Counter Component (Q3) — Files

src/app/counter-app/counter-app.component.ts

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

@Component({ selector: 'app-counter-app', templateUrl: './counter-app.component.html' })
export class CounterAppComponent {
  count = 0;
  increment() { this.count++; }
  decrement() { this.count = Math.max(0, this.count - 1); }
  reset() { this.count = 0; }
}
```

src/app/counter-app/counter-app.component.html

```
<div style="text-align:center">
  <h3>Counter: {{count}}</h3>
  <button (click)="increment()">Increment</button>
  <button (click)="decrement()" style="margin:0 10px">Decrement</button>
  <button (click)="reset()">Reset</button>
</div>
```

Todo Component (Q5 Frontend optional) — Files

src/app/todo-app/todo-app.component.ts

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

@Component({ selector: 'app-todo-app', templateUrl: './todo-app.component.html' })
export class TodoAppComponent {
  todos: any[] = [];
  text = '';
  add() { if(this.text.trim()) { this.todos.push({ id: Date.now(), text: this.text.trim(), complete
  toggle(t:any) { t.completed = !t.completed; }
  delete(id:number) { this.todos = this.todos.filter(t => t.id !== id); }
}
```

src/app/todo-app/todo-app.component.html

```
<input [(ngModel)]="text" placeholder="Enter todo" />
<button (click)="add()">Add</button>
<ul>
  <li *ngFor="let t of todos">
    <span (click)="toggle(t)" style="cursor:pointer">{{t.text}}</span>
    <button (click)="delete(t.id)" style="margin-left:8px">Delete</button>
  </li>
</ul>
```

Routing (app-routing.module.ts)

src/app/app-routing.module.ts

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { HomeComponent } from './home/home.component';
import { AboutComponent } from './about/about.component';

const routes: Routes = [
  { path: 'home', component: HomeComponent },
  { path: 'about', component: AboutComponent },
  { path: '', redirectTo: '/home', pathMatch: 'full' },
  { path: '**', redirectTo: '/home' }
];

@NgModule({ imports: [RouterModule.forRoot(routes)], exports: [RouterModule] })
export class AppRoutingModule { }
```

Run & Test — Full stack

```
1) Start backend:
   cd workspace/backend
   node .\server.js

2) Start frontend:
   cd workspace/frontend
   ng serve --open

Frontend: http://localhost:4200
Backend: http://localhost:3000

Testing tips:
- Use Postman to test backend endpoints (GET/POST/PUT/DELETE)
- In frontend, navigate to sections and take screenshots as evidence for practical exam
- If backend data is reset on server restart, explain it in your answer sheet (in-memory store)
```

Exam submission checklist

```
- Server running and screenshots of Postman requests/responses
- Browser screenshots showing Product table, Form validations, Users table (loading), Routing navig
- Short explanation for each question about what was implemented (2-3 lines)
- All source files included in a zip (backend and frontend)
```

End note

```
If you want I can generate a downloadable zip file containing the complete frontend and backend sou
```