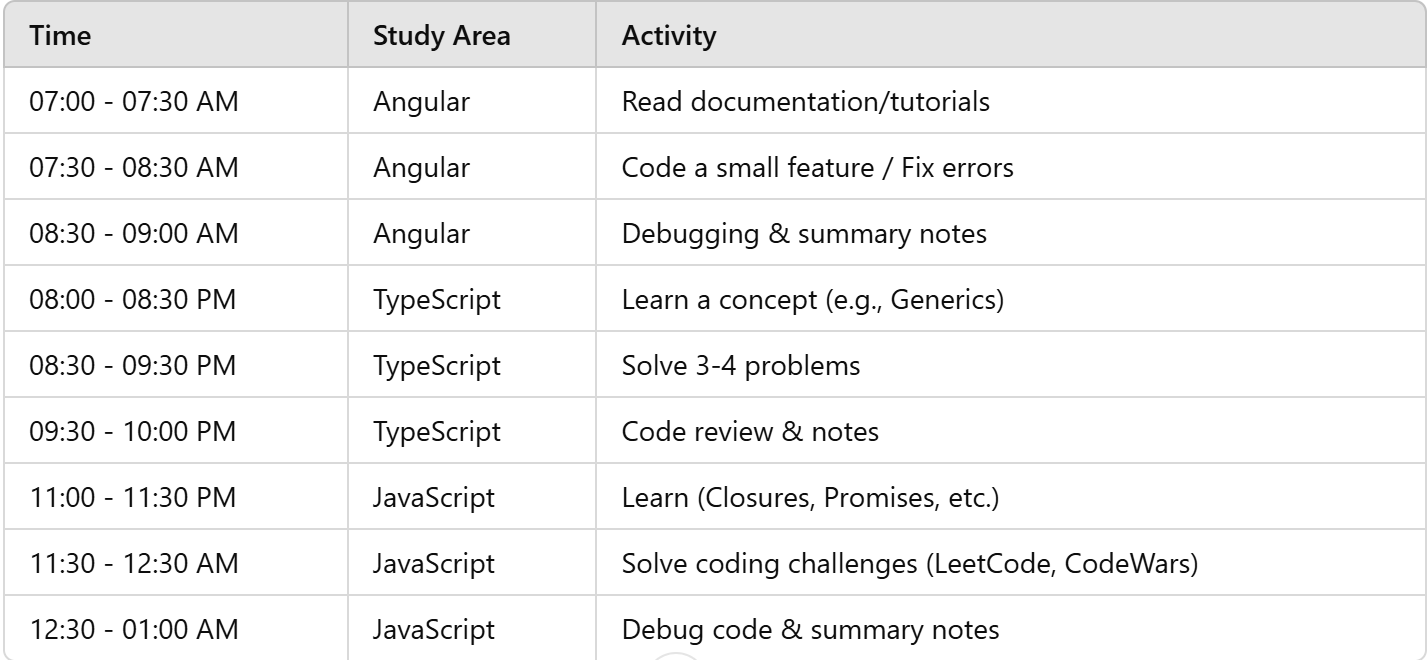
**Full Introduction**

\*\*"Good evening, thank you for this opportunity. My name is Afzal Hussain, a front-end developer with 3+ years of experience in Angular. I currently work at Bajaj Finance Ltd.

My expertise in Angular, TypeScript, JavaScript, Angular Material and API integration. I’ve worked on projects like an Asset Tracking System and a Payment Gateway.

Since I’m serving my notice period, I’m looking for a new opportunity."\*\*



To build a strong understanding of **Angular** concepts for real-world projects, you should focus on the following areas:

**1. Angular Core Concepts**

🔹 **Modules and Architecture**

* **NgModules** (@NgModule)
* **Standalone Components** (from Angular 17+)
* **Feature Modules & Lazy Loading**
* **Shared Modules**
* **Core Module (Singleton Services)**

🔹 **Components and Templates**

* **Component Lifecycle Hooks** (ngOnInit, ngOnChanges, ngAfterViewInit, etc.)
* **Change Detection Strategies** (Default & OnPush)
* **ViewEncapsulation** (Emulated, None, ShadowDom)
* **Dynamic Components (ViewContainerRef, ComponentFactoryResolver)**

🔹 **Directives**

* **Structural Directives** (\*ngIf, \*ngFor, \*ngSwitch)
* **Attribute Directives** ([ngClass], [ngStyle])
* **Custom Directives**

🔹 **Pipes**

* **Built-in Pipes** (date, currency, uppercase, async, etc.)
* **Custom Pipes** (Pure & Impure Pipes)
* **Pipe Chaining & Performance Considerations**

**2. Data Binding & Forms**

🔹 **Data Binding**

* **One-way Binding** ({{}}, [], (), bind-\*)
* **Two-way Binding** ([(ngModel)])
* **Event Binding ((click), (keyup))**
* **Template Reference Variables (#var)**

🔹 **Forms**

* **Reactive Forms (FormGroup, FormControl, FormBuilder)**
* **Template-driven Forms**
* **Validation (Built-in & Custom)**
* **Async Validators**
* **Dynamic Forms Handling**

**3. Services & Dependency Injection**

* **Singleton Services (providedIn: 'root')**
* **Injection Tokens**
* **Multi-provider Services**
* **Hierarchical Dependency Injection**
* **Inter-component Communication (@Input, @Output, EventEmitter, BehaviorSubject)**

**4. State Management**

* **RxJS & Observables (subscribe, pipe, map, filter, etc.)**
* **BehaviorSubject vs Subject**
* **NgRx Store (Redux for Angular)**
* **Component Store (Lightweight State Management)**

**5. Routing & Navigation**

* **Router Module (RouterModule.forRoot, RouterModule.forChild)**
* **Lazy Loading & Preloading Strategies**
* **Route Guards (CanActivate, CanDeactivate, Resolve, CanLoad)**
* **Route Parameters & Query Params**
* **Nested Routing & Auxiliary Routes**
* **Router Events & Navigation Interception**

**6. HTTP Client & API Integration**

* **HttpClientModule**
* **GET, POST, PUT, DELETE requests**
* **Handling Observables with RxJS Operators**
* **Interceptors (Request & Response Interceptors)**
* **Error Handling & Retry Mechanism**
* **Caching & Optimizing API Calls**

**7. Angular Material & UI Components**

* **Material Components (Table, Dialog, Form Field, Snackbar, etc.)**
* **Theming & Custom Styles**
* **CDK (Component Dev Kit) - Drag & Drop, Overlay, Scrolling, Accessibility**
* **Flex Layout & Grid System**

**8. Performance Optimization**

* **OnPush Change Detection Strategy**
* \**TrackBy in ngFor*
* **Lazy Loading Modules**
* **Virtual Scrolling**
* **Debouncing & Throttling (for API Calls)**
* **Caching & Memoization**
* **Optimizing Large Forms & Tables**
* **Web Workers for Heavy Computation**

**9. Security Best Practices**

* **Sanitization (DomSanitizer)**
* **Cross-Site Scripting (XSS) Prevention**
* **Cross-Origin Resource Sharing (CORS)**
* **Authentication & Authorization (JWT, OAuth)**
* **Role-Based Access Control (RBAC)**
* **Rate Limiting & API Security**

**10. Testing & Debugging**

* **Unit Testing (Jasmine & Karma)**
* **Component Testing with TestBed**
* **Service Testing with HttpTestingController**
* **Mocking Dependencies (spyOn, fakeAsync, flush, tick)**
* **E2E Testing with Cypress or Playwright**
* **Debugging with Chrome DevTools & Augury**

**11. Micro Frontends (MFE)**

* **Module Federation (Webpack 5)**
* **Communication between MFEs**
* **Sharing Dependencies**
* **Deployment Strategies**

**12. Server-Side Rendering (SSR) & PWA**

* **Angular Universal (SSR)**
* **Progressive Web Apps (PWAs)**
* **Service Workers**
* **Lazy Loading & Caching Strategies**