Designing a RESTful API for User Profiles

URL Structure and HTTP Methods for Each Operattion

- **Create (POST):** '/api/users'
- **Read (GET):** '/api/users/{userId}'
- **Update (PUT/PATCH):** '/api/users/{userId}'
- **Delete (DELETE):** '/api/users/{userId}'

Data Format for Requests and Response

- Use **JSON** for both request and response bodies.

Handling Authentication and Authorization

- Use token-based authentication (JWT) for secure user authentication.
- Use OAuth2 for authorization for secure delegated access to resources.

Scalability

- Implement horizontal scaling by distributing the API across multiple servers or using load balancing.
- Use a caching layer for frequently accessed data.

Security

- Apply HTTPS to encrypt data in transit.
- Validate and sanitize user input to prevent injection attacks.

Integration with Frontend (Angular JS)

- Use Angular JS services to make HTTP requests to the API endpoints.
- Inject the "userService" into Angular JS controllers to interact with the API. To inject the "userService" into Angular JS controllers, first we have to declare the "userService" module and factory.Then, inject the "userService" into controller by specifying it as a dependency.

By following these design considerations, we can create a RESTful API that is scalable, secure and meets the requirements of managing user profiles.