User Management API – Project Documentation

Project Overview

TechHive Solutions required a **User Management API** to support HR and IT departments in managing user records. The API needed to provide CRUD operations, robust validation, error handling, and middleware for logging, authentication, and auditing. Microsoft Copilot was used throughout the project to scaffold, enhance, and debug the code.

Phase 1: Initial API Setup

- **Task:** Create a new ASP.NET Core Web API project named `UserManagementAPI`.
- **Implementation:**
  - Used Copilot to scaffold the project setup in `Program.cs`.
  - Configured `WebApplication.CreateBuilder` and `WebApplication` pipeline.
- **Copilot Contribution:** Suggested boilerplate code for project setup and HTTPS redirection.

Phase 2: CRUD Endpoints

- **Task:** Implement endpoints for user management.
- **Implementation:**
  - `GET /users` → Retrieve all users (with optional pagination).
  - `GET /users/{id}` → Retrieve a specific user by ID.
  - `POST /users` → Add a new user with validation.
  - `PUT /users/{id}` → Update an existing user.
  - `DELETE /users/{id}` → Remove a user by ID.
  - `GET /users/search/{term}` → Search users by name or email.
- **Copilot Contribution:** Generated endpoint scaffolding, suggested validation attributes (`[Required]`, `[StringLength]`, `[EmailAddress]`), and helped implement pagination logic.

Phase 3: Debugging & Reliability

- **Task:** Fix bugs reported in initial deployment.
- **Issues Identified:**
  - Users added without proper validation.
  - Errors when retrieving non-existent users.
  - Occasional crashes due to unhandled exceptions.
- **Fixes Implemented:**
  - Added validation checks using `Validator.TryValidateObject`.
  - Returned `BadRequest` for invalid IDs and `NotFound` for missing users.
  - Implemented global exception handling middleware.
- **Copilot Contribution:** Suggested validation logic, standardized error responses, and added `try/catch` middleware to prevent crashes.

Phase 4: Middleware Integration

- **Task:** Add middleware for corporate compliance.
- **Implementation:**
  - **Error-handling middleware:** Catches unhandled exceptions and returns JSON `{ "error": "Internal server error." }`.
  - **Authentication middleware:** Validates `Authorization: Bearer valid-token` header, returns `401 Unauthorized` if invalid.
  - **Logging middleware:** Logs HTTP method, request path, and response status code.
  - **Pipeline order:** Configured as error handling → authentication → logging.
- **Copilot Contribution:** Generated middleware scaffolding, ensured correct pipeline order, and standardized JSON error responses.

Phase 5: Testing

- **Task:** Validate API functionality.
- **Implementation:**
  - Tested CRUD endpoints with Postman.
  - Verified validation errors for invalid input.
  - Confirmed `401 Unauthorized` for invalid tokens.
  - Triggered exceptions to validate error-handling middleware.
  - Checked console logs for auditing accuracy.
- **Copilot Contribution:** Suggested test cases and edge scenarios (invalid IDs, missing tokens, malformed input).

Summary of Copilot's Contributions

- **Scaffolding:** Generated boilerplate project setup and CRUD endpoint structures.
- **Validation:** Suggested data annotations and runtime validation logic.
- **Error Handling:** Helped implement global exception middleware with consistent JSON responses.
- **Authentication:** Provided token-based middleware logic.
- **Logging:** Generated middleware to log requests and responses.
- **Debugging:** Identified missing validation and unhandled exceptions, suggested fixes.
- **Testing Guidance:** Proposed edge cases and scenarios for validation.