User Management — Function and API Reference

This document summarizes the main classes and functions in the solution, grouped by project.

UserManagement.Data

- **Entities**
  - `UserManagement.Models.User`
    - Properties:
      - `long Id`
      - `string Forename`
      - `string Surname`
      - `string Email`
      - `bool IsActive`
      - `DateTime? DateOfBirth`
  - `UserManagement.Models.LogEntry`
    - Properties:
      - `long Id`
      - `long? UserId`
      - `string Action`
      - `string? Description`
      - `DateTime CreatedAtUtc`
- **Data Abstraction**
  - `UserManagement.Data.IDataContext`
    - `IQueryable<TEntity> GetAll<TEntity>()`
    - `void Create<TEntity>(TEntity entity)`
    - `Task CreateAsync<TEntity>(TEntity entity)`
    - `void Update<TEntity>(TEntity entity)`
    - `Task UpdateAsync<TEntity>(TEntity entity)`
    - `void Delete<TEntity>(TEntity entity)`
    - `Task DeleteAsync<TEntity>(TEntity entity)`
- **EF Core Implementation**
  - `UserManagement.Data.DataContext : DbContext, IDataContext`
    - Overrides:
      - `OnConfiguring(DbContextOptionsBuilder options)` — uses InMemory provider
      - `OnModelCreating(ModelBuilder model)` — seeds demo users; registers `LogEntry`
    - DbSets:
      - `DbSet<User>? Users`
      - `DbSet<LogEntry>? LogEntries`
    - CRUD (implements `IDataContext`): `GetAll`, `Create`/`CreateAsync`, `Update`/`UpdateAsync`, `Delete`/`DeleteAsync`

- **DI Registration**
  - `Microsoft.Extensions.DependencyInjection.ServiceCollectionExtensions` (in Data)
    - `IServiceCollection AddDataAccess(this IServiceCollection services)` — registers `IDataContext -> DataContext` (scoped)

UserManagement.Services

- **Interfaces**
  - `UserManagement.Services.Domain.Interfaces.IUserService`
    - Sync:
      - `IEnumerable<User> FilterByActive(bool isActive)`
      - `IEnumerable<User> GetAll()`
      - `User? GetById(long id)`
      - `void Add(User user)`
      - `void Update(User user)`
      - `void Delete(long id)`
    - Async:
      - `Task<IEnumerable<User>> GetAllAsync()`
      - `Task<User?> GetByIdAsync(long id)`
      - `Task AddAsync(User user)`
      - `Task UpdateAsync(User user)`
      - `Task DeleteAsync(long id)`
  - `UserManagement.Services.Domain.Interfaces.ILogService`
    - Sync:
      - `IEnumerable<LogEntry> GetAll(int skip = 0, int take = 50)`
      - `IEnumerable<LogEntry> GetByUser(long userId, int skip = 0, int take = 100)`
      - `LogEntry? GetById(long id)`
      - `void Log(string action, string? description = null, long? userId = null)`
    - Async:
      - `Task<IEnumerable<LogEntry>> GetAllAsync(int skip = 0, int take = 50)`
      - `Task<IEnumerable<LogEntry>> GetByUserAsync(long userId, int skip = 0, int take = 100)`
      - `Task<LogEntry?> GetByIdAsync(long id)`
      - `Task LogAsync(string action, string? description = null, long? userId = null)`
- **Implementations**
  - `UserManagement.Services.Domain.Implementations.UserService`

: IUserService

* Filters/users: FilterByActive
* CRUD (sync): GetAll, GetById, Add, Update, Delete
* CRUD (async): GetAllAsync, GetByIdAsync, AddAsync, UpdateAsync, DeleteAsync

– UserManagement.Services.Domain.Implementations.LogService : ILogService

* Query logs: GetAll, GetByUser, GetById
* Create log: Log
* Async wrappers: GetAllAsync, GetByUserAsync, GetByIdAsync, LogAsync

- **DI Registration**
  - Microsoft.Extensions.DependencyInjection.ServiceCollectionExtensions (in Services)
    * IServiceCollection AddDomainServices(this IServiceCollection services) — registers IUserService and ILogService (scoped)

UserManagement.Web

- **Startup**
  - Program.cs
    * Registers: AddDataAccess(), AddDomainServices(), AddControllersWithViews()
    * Middleware (Development vs non-Development): HSTS/HTTPS in non-dev; Static files; Routing; Authorization
    * Routing: MapDefaultControllerRoute()
- **Controllers**
  - HomeController
    * ViewResult Index() — returns home page
  - UsersController
    * ViewResult List() — list all users (async service call)
    * ViewResult ListActive() — list active users
    * ViewResult ListInactive() — list inactive users
    * ViewResult Add() — GET create form
    * Task<IActionResult> Add(UserCreateViewModel model) — POST create
    * Task<IActionResult> View(long id) — user details; logs a "Viewed" action
    * Task<IActionResult> Edit(long id) — GET edit form
    * Task<IActionResult> Edit(UserEditViewModel model) — POST update; logs "Updated"

    - `Task<IActionResult> Delete(long id)` — GET delete confirmation
    - `Task<IActionResult> DeleteConfirmed(long id)` — POST delete; logs "Deleted"
  - `LogsController`
    - `ViewResult Index(int page = 1, int pageSize = 25)` — list logs with paging
    - `IActionResult Details(long id)` — log details
    - `ViewResult ForUser(long userId, int page = 1, int pageSize = 25)` — logs for a specific user
- **Views (not exhaustive)**
  - `Users/List` — table with actions; filter buttons
  - `Users/Add`, `Users/Edit`, `Users/Delete`, `Users/View` — forms and details
  - Partial: `Users/_UserRecentLogs` — shows recent logs for a user
  - `Logs/Index`, `Logs/Details` — list and details for logs

Endpoints (quick index)

- `/` — Home
- `/users` — List
- `/users/active` — Active
- `/users/inactive` — Inactive
- `/users/add` (GET/POST) — Create
- `/users/{id}/view` — Details
- `/users/{id}/edit` (GET/POST) — Edit
- `/users/{id}/delete` (GET/POST) — Delete
- `/logs` — Logs list
- `/logs/{id}` — Log details
- `/logs/user/{userId}` — Logs for a user