**Software Requirements Specification (SRS)**

**Project Title:** Job Application Tracker App (MAUI + Web API)

**Author:** Susan Crownhart  
**Date:** April 14, 2025

**1. Introduction**

**1.1 Purpose**

This document outlines the functional requirements, system architecture, and design elements for a mobile application built using .NET MAUI, with a backend Web API. The application enables users to track job applications, interviews, companies, contacts at companies, and interview preparation.

**1.2 Scope**

The application will:

* Allow users to add and manage job applications, interviews, and interview preparation
* Track applications and interviews
* Store company, company contact details, and interview preparation notes
* Communicate with a secure Web API for data persistence
* Use MVVM architecture with dependency injection

**1.3 Target Audience**

* Individuals searching for jobs across multiple platforms like Indeed, Monster, LinkedIn, ZipRecruiter, etc.

**2. System Overview**

The system is composed of three layers:

* **Frontend (MAUI app):** Handles user interaction and view logic
* **API Service Layer:** Generic HTTP clients for calling the backend
* **Backend (Web API + Repositories):** Handles business logic and data persistence using Entity Framework Core

**3. Functional Requirements**

**3.1 Application Management**

* Add new job applications
* View/edit/delete applications
* Search applications by status

**3.2 Interview Tracking**

* Add/view/edit interview details
* Track follow-up on applications
* Add/view/edit/delete interview preparation notes

**3.3 Company Management**

* Add/view/edit/delete companies
* Add/view/edit/delete company contacts

**4. Non-Functional Requirements**

* Cross-platform compatibility (Android)
* Scalable backend architecture
* Secure API communication
* Clean separation of concerns (MVVM + Repository Pattern)

**5. Architecture Overview**

**5.1 Diagram Reference**

The UML class diagram defines:

* **Models** such as Application, Company, Interview, etc.
* **DbContext**: JobAppDbContext exposes DbSet<T> for all entity types
* **Repositories**: JobAppRepository<T> provides base CRUD methods; specific repositories inherit it
* **Controllers**: JobAppApi<T> defines RESTful endpoints
* **API Service Layer**: JobAppApiService<T> wraps HTTP communication
* **ViewModels**: Each entity has a ViewModel (e.g., ApplicationViewModel) consuming JobAppApiService<T>

**5.2 MVVM Pattern**

* ViewModels are responsible for view logic and UI bindings
* Services are injected into ViewModels
* ViewModels communicate with API via the JobAppApiService<T> class

**5.3 Dependency Injection**

All services and repositories are registered in the DI container. Services and DbContext are injected where needed.

**6. Class Responsibilities**

**6.1 Models**

* Define schema and relationships

**6.2 DbContext**

* Manages EF Core data access

**6.3 Repositories**

* Base and specific logic for data manipulation

**6.4 Web API**

* RESTful endpoints for frontend interaction

**6.5 API Services**

* Generic client-side services wrapping HTTP logic

**6.6 ViewModels**

* Manage ObservableCollections, user commands, and service calls

**7. Future Enhancements**

* Authentication/authorization using JWT
* Cloud-hosted API (Azure App Service)
* SQLite offline caching
* Notifications for scheduled interviews
* Save Resumes and Cover Letters specifically for jobs applied to

**8. Glossary**

* **MVVM:** Model-View-ViewModel
* **EF Core:** Entity Framework Core
* **API:** Application Programming Interface
* **DI:** Dependency Injection