**ReqVista System - Statement of Work**

**1. Project Overview**

ReqVista is a comprehensive requirements management system designed to track and manage a hierarchical structure of portfolios, projects, and requirements. The system will provide stakeholder management, risk tracking, milestone planning, requirement lifecycle management, user management, notifications, and reporting capabilities.

The solution will be built using C# .NET with a multi-platform approach:

* Web application using Blazor
* Cross-platform mobile application using .NET MAUI with offline capabilities
* RESTful API for integration and data access
* Azure-hosted backend with cloud-native architecture

**2. Workstreams**

**2.1 Analysis and Planning**

* **Requirements Gathering and Validation**: Review and refine the provided ER diagram and requirements list
* **System Architecture Design**: Define the overall architecture for web, mobile, and API components
* **Database Schema Finalization**: Review and refine the database schema from the ER diagram
* **Project Planning**: Create detailed project plan, milestones, and resource allocation
* **Risk Assessment**: Identify potential project risks and mitigation strategies

**2.2 Design**

* **UI/UX Design**: Create wireframes, mockups, and design system for web and mobile applications
* **Database Design**: Finalize database schema and entity relationships
* **API Design**: Define API endpoints, request/response structures, and authentication mechanisms
* **Architecture Design**: Detailed design of system components, libraries, and services
* **Security Design**: Authentication, authorization, data protection, and security controls

**2.3 Implementation**

* **Database Implementation**: Create database schema, migrations, and seed data
* **Core Domain Models**: Implement entity models with ObservableObject inheritance
* **API Development**: Build RESTful API endpoints and services
* **Web Application Development**: Implement Blazor web application
* **Mobile Application Development**: Implement .NET MAUI mobile application with offline sync
* **Shared Component Libraries**: Develop reusable components shared between web and mobile
* **Integration Services**: Implement external system integrations
* **Authentication and Authorization**: Implement security features

**2.4 Testing**

* **Unit Testing**: Test individual components and services
* **Integration Testing**: Test component interactions and API endpoints
* **End-to-End Testing**: Test complete user workflows and scenarios
* **Performance Testing**: Validate system performance under various loads
* **Security Testing**: Validate security controls and identify vulnerabilities
* **User Acceptance Testing**: Validate system meets user requirements
* **Mobile Device Testing**: Test across various mobile platforms and devices

**2.5 DevOps and Infrastructure**

* **Azure Infrastructure Setup**: Provision and configure Azure resources
* **CI/CD Pipeline Setup**: Configure continuous integration and deployment pipelines
* **Container Configuration**: Set up container environments with Podman
* **Environment Configuration**: Set up development, testing, staging, and production environments
* **Monitoring and Alerting**: Configure monitoring, logging, and alerting systems
* **Backup and Recovery**: Implement backup and disaster recovery procedures

**2.6 Deployment**

* **Deployment Planning**: Create deployment strategy and rollback procedures
* **Database Deployment**: Deploy database schema and initial data
* **API Deployment**: Deploy API services to Azure
* **Web Application Deployment**: Deploy Blazor web application
* **Mobile Application Release**: Publish mobile application to app stores
* **Documentation Deployment**: Publish system documentation

**2.7 Post-Production Support**

* **Bug Fixes**: Address identified issues and bugs
* **Performance Optimization**: Optimize system performance
* **User Support**: Provide support for user questions and issues
* **System Enhancements**: Implement minor enhancements and improvements
* **Maintenance Updates**: Regular maintenance and updates

**3. Key Design Considerations**

**3.1 Technical Architecture**

* **Microservices vs. Monolith**: Consider the appropriate architectural approach based on scale and complexity
* **API Design Pattern**: RESTful API design with proper versioning and documentation
* **Shared Code Strategy**: Approach for sharing code between web and mobile applications
* **Offline Synchronization**: Strategy for mobile offline data synchronization with LiteDB
* **Authentication and Authorization**: Implementation of secure authentication and role-based access control
* **Data Access Layer**: Efficient implementation using Entity Framework Core

**3.2 User Experience**

* **Responsive Design**: Ensure web application works across devices and screen sizes
* **Mobile Experience**: Native mobile experience that works offline
* **Accessibility**: Ensure system meets accessibility standards
* **Performance**: Fast loading times and responsive interactions
* **Consistency**: Consistent design across web and mobile platforms

**3.3 Data Management**

* **Data Integrity**: Ensure data integrity across all system components
* **Data Migration**: Strategy for data migration if replacing an existing system
* **Backup and Recovery**: Regular backups and efficient recovery procedures
* **Data Retention**: Define data retention policies
* **Data Security**: Encryption, access controls, and data protection

**3.4 Scalability and Performance**

* **Horizontal Scaling**: Ability to scale with increased load
* **Database Performance**: Optimize for read and write operations
* **Caching Strategy**: Implement appropriate caching mechanisms
* **Resource Optimization**: Efficient use of compute and storage resources
* **Performance Monitoring**: Tools and metrics for ongoing performance evaluation

**3.5 Security**

* **Authentication**: Secure user authentication mechanisms
* **Authorization**: Role-based access control
* **Data Protection**: Encryption of sensitive data at rest and in transit
* **Audit Trail**: Comprehensive logging of system activities
* **Compliance**: Meet relevant security and privacy compliance requirements

**4. Deliverables**

**4.1 Software Components**

* **Azure SQL Database**: Fully implemented database with Entity Framework Core integration
* **API Services**: RESTful API endpoints with authentication and authorization
* **Blazor Web Application**: Full-featured web client
* **.NET MAUI Mobile Application**: Cross-platform mobile application with offline capabilities
* **Shared Component Libraries**: Reusable UI and business logic components

**4.2 Documentation**

* **System Architecture Documentation**: Detailed system architecture and component design
* **API Documentation**: Comprehensive API documentation with examples
* **Database Schema Documentation**: Detailed database schema and relationships
* **User Manuals**: End-user documentation for web and mobile applications
* **Administrator Guides**: System administration and configuration guides
* **Deployment Guides**: Instructions for deployment and configuration

**4.3 Infrastructure**

* **Azure Resources**: Configured Azure resources for hosting the system
* **CI/CD Pipelines**: Automated build and deployment pipelines
* **Container Configurations**: Containerized components with Podman configurations
* **Environment Configurations**: Development, testing, staging, and production environments

**4.4 Testing Assets**

* **Test Plans**: Comprehensive test plans for all system components
* **Test Scripts**: Automated test scripts for unit, integration, and end-to-end testing
* **Test Reports**: Test execution reports and issue tracking
* **Performance Test Results**: Reports on system performance under various loads

**4.5 Training Materials**

* **User Training Materials**: Training guides and videos for end users
* **Administrator Training Materials**: Training for system administrators
* **Developer Handover Documentation**: Documentation for future development and maintenance

**5. Timeline and Milestones**

**5.1 High-Level Timeline**

| **Phase** | **Duration** | **Start Date** | **End Date** |
| --- | --- | --- | --- |
| Inception and Planning | 4 weeks | Week 1 | Week 4 |
| Analysis and Design | 8 weeks | Week 5 | Week 12 |
| Development - Iteration 1 | 6 weeks | Week 13 | Week 18 |
| Development - Iteration 2 | 6 weeks | Week 19 | Week 24 |
| Development - Iteration 3 | 6 weeks | Week 25 | Week 30 |
| System Testing | 4 weeks | Week 31 | Week 34 |
| User Acceptance Testing | 3 weeks | Week 35 | Week 37 |
| Deployment Preparation | 2 weeks | Week 38 | Week 39 |
| Go-Live and Stabilization | 3 weeks | Week 40 | Week 42 |
| **TOTAL** | **42 weeks** | **Week 1** | **Week 42** |

**5.2 Key Milestones**

* **Project Plan Approved**: Week 4
* **Design Documentation Approved**: Week 12
* **Core Functionality Demonstrated**: Week 18
* **Feature Complete - Portfolio and Project Management**: Week 24
* **Feature Complete - All Requirements**: Week 30
* **System Testing Complete**: Week 34
* **UAT Complete**: Week 37
* **Ready for Deployment**: Week 39
* **Project Complete**: Week 42

**6. Team Structure and Estimates**

**6.1 Team Composition**

| **Role** | **Count** | **Allocation** | **Duration** |
| --- | --- | --- | --- |
| Project Manager | 1 | 100% | 42 weeks |
| Solution Architect | 1 | 100% | 42 weeks |
| Backend Developers | 3 | 100% | 36 weeks |
| Web Developers | 2 | 100% | 36 weeks |
| Mobile Developers | 2 | 100% | 36 weeks |
| Shared Component Developers | 1 | 100% | 36 weeks |
| QA Engineers | 3 | 100% | 32 weeks |
| DevOps Engineers | 1 | 100% | 36 weeks |
| UX/UI Designers | 1 | 100% | 20 weeks |
| Database Specialists | 1 | 50% | 36 weeks |
| Documentation Specialists | 1 | 50% | 36 weeks |

**6.2 Effort Summary**

| **Workstream** | **Estimated Effort (Person-Days)** |
| --- | --- |
| Analysis and Planning | 45-60 |
| Design | 80-100 |
| Database and API Development | 120-150 |
| Web Application Development | 150-180 |
| Mobile Application Development | 180-220 |
| Shared Component Development | 60-80 |
| Integration Development | 40-60 |
| Testing | 140-170 |
| DevOps and Infrastructure | 60-80 |
| Documentation | 50-70 |
| Deployment and Rollout | 30-40 |
| Project Management | 90-110 |
| **TOTAL** | **1045-1320** |

**7. Project Roles and Responsibilities**

**7.1 Key Roles**

* **Project Sponsor**: Provides executive support, secures resources, approves major changes
* **Project Manager**: Manages project plan, schedule, budget, risks, and communications
* **Product Owner**: Maintains backlog, defines acceptance criteria, represents user needs
* **Technical Lead/Solution Architect**: Designs technical architecture, guides development team
* **Development Team**: Implements database, API, web, and mobile components
* **QA Team**: Develops test plans, performs testing, reports defects
* **DevOps Team**: Configures CI/CD pipelines, environments, and infrastructure
* **UX Team**: Creates user interface designs and user experience workflows
* **Subject Matter Experts**: Provide domain knowledge and validate requirements
* **Support Team**: Configures production systems and provides first-level support

**7.2 RACI Matrix**

A detailed RACI matrix defining Responsible, Accountable, Consulted, and Informed roles for all major deliverables is included in the roles and responsibilities document.

**8. Assumptions and Constraints**

**8.1 Key Assumptions**

* **Stakeholder Availability**: Key stakeholders will be available for requirements validation and sign-offs
* **Technical Environment**: Azure subscriptions and development tools will be available
* **Methodology**: The project will follow an Agile development methodology with 2-week sprints
* **Resource Availability**: Required resources with appropriate skills will be available
* **Technology Stack**: The defined C# .NET stack must be used as specified
* **Requirements Stability**: Core feature set will remain stable, with change management for modifications

**8.2 Key Constraints**

* **Schedule Constraints**: Fixed deadlines, resource availability windows, deployment windows
* **Technical Constraints**: Defined technology stack, cloud platform, security requirements
* **Resource Constraints**: Team size limitations, specialized skill availability, budget constraints
* **Dependencies**: Environment provisioning, third-party services, external system interfaces

**9. Solution Architecture**

The solution follows a layered architecture with clear separation between client, API, business logic, and data layers:

* **Client Layer**: Blazor Web Application and MAUI Mobile Application
* **API Layer**: API Gateway, Web API, Authentication API, Reporting API, Notification API
* **Business Logic Layer**: Core domain services implementing business rules
* **Data Layer**: Azure SQL Database, Redis Cache, Blob Storage, and LiteDB for mobile
* **Shared Services**: Component libraries, ViewModels, DTOs, and validation rules
* **External Systems**: Identity Provider, Email Service, 3rd Party Integrations
* **DevOps/Infrastructure**: CI/CD, Monitoring, Logging, and Security services

See the attached architecture diagram for a visual representation of the system components and their relationships.

**10. Acceptance Criteria**

* All functional and non-functional requirements are met
* System passes all test cases with at least 95% coverage
* Web and mobile applications perform within defined performance benchmarks
* All deliverables are completed and approved
* System is deployed to production environment and stable for at least 2 weeks
* User and administrator training is completed
* Documentation is complete and approved

**11. Support and Maintenance**

* 90-day warranty period for bug fixes after production deployment
* Optional ongoing support and maintenance agreement available
* Regular system updates and security patches
* Monitoring and incident response procedures

**12. Change Management**

* Formal change request process for scope changes
* Impact assessment for all proposed changes
* Approval process for change requests
* Change implementation and validation procedures

**13. Risk Management**

* Proactive risk identification and assessment
* Risk mitigation strategies for identified risks
* Regular risk monitoring and reporting
* Contingency plans for critical risks