**ReqVista - Solution Architecture Description**

**1. Architecture Overview**

The ReqVista solution follows a modern, cloud-native architecture leveraging Microsoft Azure services. The architecture is designed with the following principles:

* **Layered Architecture**: Clear separation of concerns between UI, API, business logic, and data layers
* **Cloud-Native**: Leveraging Azure managed services for scalability and reliability
* **Shared Code**: Maximizing code reuse between web and mobile platforms
* **Offline-First Mobile**: Mobile application designed to function offline with synchronization
* **Security-First**: Security integrated at all architecture layers
* **DevOps Integration**: CI/CD pipelines for automated build, test, and deployment
* **Scalability**: Architecture designed to scale with growing user base and data volume
* **Observability**: Comprehensive monitoring and logging throughout the system

A diagram of a computer

AI-generated content may be incorrect.

**2. Architecture Layers**

**2.1 Client Layer**

**Blazor Web Application**

* **Technology**: Blazor WebAssembly
* **Description**: Browser-based application providing full system functionality
* **Key Features**:
  + Responsive design for various screen sizes
  + Rich interactive components
  + Client-side validation
  + Progressive Web App capabilities
  + Authentication integration with Azure AD

**.NET MAUI Mobile Application**

* **Technology**: .NET MAUI (Multi-platform App UI)
* **Description**: Cross-platform mobile application for iOS and Android
* **Key Features**:
  + Offline data access with LiteDB
  + Synchronization with backend when online
  + Native device feature integration
  + Responsive to different device form factors
  + Secure local data storage

**2.2 API Layer**

**Azure API Management**

* **Technology**: Azure API Management
* **Description**: API gateway providing unified access to backend services
* **Key Features**:
  + API versioning control
  + Rate limiting and throttling
  + API documentation with Swagger
  + Analytics and monitoring
  + Security policy enforcement

**ReqVista Web API**

* **Technology**: ASP.NET Core Web API (Serverless Function)
* **Description**: Primary API providing access to core system functionality
* **Key Features**:
  + RESTful API design
  + Entity Framework Core integration
  + JWT authentication
  + Role-based authorization
  + Comprehensive validation

**Authentication API**

* **Technology**: ASP.NET Core Web API (Serverless Function)
* **Description**: Specialized API for authentication and authorization
* **Key Features**:
  + Integration with Azure Active Directory
  + JWT token issuance and validation
  + User role and permission management
  + Secure password policies
  + Multi-factor authentication support

**Reporting API**

* **Technology**: ASP.NET Core Web API (Serverless Function)
* **Description**: API for generating and retrieving reports
* **Key Features**:
  + Report generation in multiple formats
  + Scheduled report execution
  + Query optimization for reporting
  + Caching of report data
  + Export capabilities

**Notification API**

* **Technology**: ASP.NET Core Web API (Serverless Function)
* **Description**: API for managing system notifications
* **Key Features**:
  + Email notifications
  + In-app notifications
  + Push notifications for mobile
  + Notification templates
  + Notification preferences management

**2.3 Business Logic Layer**

**Core Services**

* **Portfolio Service**: Management of portfolios and related entities
* **Project Service**: Project creation, management, and tracking
* **Requirement Service**: Requirement lifecycle management
* **Risk Service**: Risk identification, assessment, and mitigation
* **User Service**: User profile and preferences management
* **Milestone Service**: Milestone planning and tracking
* **Notification Service**: Notification generation and delivery
* **Reporting Service**: Report generation and scheduling

**Key Features**

* Domain-driven design implementation
* Business rule enforcement
* Transaction management
* Event-driven architecture for key operations
* Audit logging of business operations
* Caching strategy implementation

**2.4 Data Layer**

**Azure SQL Database**

* **Technology**: Azure SQL
* **Description**: Primary relational database for the system
* **Key Features**:
  + Entity Framework Core integration
  + Database-first approach for schema definition
  + Optimized indexing strategy
  + Transaction support
  + Data encryption at rest
  + Geo-replication for disaster recovery

**Azure Redis Cache**

* **Technology**: Azure Redis Cache
* **Description**: Distributed caching for performance optimization
* **Key Features**:
  + Caching of frequently accessed data
  + Session state management
  + Distributed locking mechanism
  + Cache invalidation strategy
  + Performance optimization

**Azure Blob Storage**

* **Technology**: Azure Blob Storage
* **Description**: Storage for documents, attachments, and large objects
* **Key Features**:
  + Secure file storage and retrieval
  + Access control for files
  + Metadata management
  + Direct upload/download capabilities
  + Tiered storage for cost optimization

**LiteDB (Mobile)**

* **Technology**: LiteDB
* **Description**: Embedded NoSQL database for mobile offline storage
* **Key Features**:
  + Document storage for offline operation
  + Data synchronization with backend
  + Conflict resolution strategy
  + Optimized for mobile performance
  + Secure local storage

**2.5 Shared Services**

**Shared Component Library**

* **Technology**: .NET Standard libraries
* **Description**: Reusable components shared between web and mobile
* **Key Features**:
  + UI components with platform-specific rendering
  + Business logic components
  + Utility functions and helpers
  + Extension methods
  + Common interfaces

**Shared ViewModels**

* **Technology**: .NET Standard with ObservableObject
* **Description**: ViewModels shared between platforms with MVVM pattern
* **Key Features**:
  + ObservableProperty attributes for change notification
  + Common validation logic
  + Platform-independent business logic
  + Command implementations
  + Navigation state management

**Data Transfer Objects**

* **Technology**: .NET Standard classes
* **Description**: Objects for data exchange between layers
* **Key Features**:
  + Immutable data structures
  + JSON serialization support
  + Validation attributes
  + Mapping profiles for entity conversion
  + Optimized for network transfer

**Validation Rules**

* **Technology**: .NET Standard libraries
* **Description**: Shared validation logic across platforms
* **Key Features**:
  + Common validation rules
  + Custom validators
  + Validation context handling
  + Error message management
  + Localization support

**2.6 External Integrations**

**Identity Provider/Azure AD**

* **Technology**: Azure Active Directory
* **Description**: Identity and access management
* **Key Features**:
  + User authentication
  + Role-based access control
  + Single sign-on capability
  + Security policy enforcement
  + Multi-factor authentication

**Email Service**

* **Technology**: SendGrid or similar
* **Description**: Email delivery service for notifications
* **Key Features**:
  + Transactional email delivery
  + Email templates
  + Delivery tracking
  + Bounce handling
  + Compliance with email regulations

**3rd Party Integrations**

* **Description**: Integration points with external systems
* **Key Features**:
  + API-based integration
  + Webhook support
  + Data import/export capabilities
  + Authentication with external systems
  + Error handling and retry logic

**2.7 DevOps/Infrastructure**

**Azure DevOps/CI CD**

* **Technology**: Azure DevOps
* **Description**: Continuous integration and deployment pipeline
* **Key Features**:
  + Automated build process
  + Automated testing
  + Deployment automation
  + Environment management
  + Release approval workflows

**Application Insights**

* **Technology**: Azure Application Insights
* **Description**: Application monitoring and telemetry
* **Key Features**:
  + Performance monitoring
  + Exception tracking
  + User behavior analytics
  + Distributed tracing
  + Availability monitoring

**Azure Monitor**

* **Technology**: Azure Monitor
* **Description**: Comprehensive monitoring of Azure resources
* **Key Features**:
  + Infrastructure monitoring
  + Log analytics
  + Alert management
  + Dashboard and visualization
  + Automated remediation

**Azure Key Vault**

* **Technology**: Azure Key Vault
* **Description**: Secure storage for secrets and certificates
* **Key Features**:
  + Centralized secret management
  + Certificate management
  + Access control for secrets
  + Encryption key management
  + Auditing of secret access

**3. Key Technical Decisions**

**3.1 Database Strategy**

* **Primary Database**: Azure SQL for relational data storage
* **Caching Layer**: Azure Redis Cache for performance optimization
* **Blob Storage**: Azure Blob Storage for document storage
* **Mobile Storage**: LiteDB for offline mobile data storage
* **Entity Framework Core**: ORM for data access with code-first approach
* **Concurrency Control**: Optimistic concurrency with row versioning

**3.2 Authentication and Authorization**

* **Identity Provider**: Azure Active Directory
* **Authentication Protocol**: OpenID Connect
* **Token Format**: JWT (JSON Web Tokens)
* **Authorization Model**: Role-based access control (RBAC)
* **API Security**: OAuth 2.0 for API authorization
* **Mobile Security**: Secure token storage with refresh capability

**3.3 API Design**

* **API Style**: RESTful with resource-oriented design
* **Data Format**: JSON for request/response
* **Versioning**: URL-based versioning (e.g., /api/v1/...)
* **Documentation**: OpenAPI/Swagger
* **Error Handling**: Standardized error responses
* **Rate Limiting**: Implemented at API Gateway level

**3.4 Offline Synchronization**

* **Sync Protocol**: Custom sync protocol based on timestamp and change tracking
* **Conflict Resolution**: Server-wins with client notification
* **Sync Scope**: Configurable by entity type and user context
* **Change Tracking**: Optimistic concurrency with timestamps
* **Data Prioritization**: Critical data synced first
* **Bandwidth Optimization**: Delta synchronization

**3.5 Deployment Strategy**

* **Environment Strategy**: Development, Testing, Staging, Production
* **Deployment Automation**: Azure DevOps pipelines
* **Infrastructure as Code**: ARM templates or Terraform
* **Container Strategy**: Containerized API services
* **Database Deployment**: Entity Framework migrations
* **Release Cadence**: Bi-weekly releases

**4. Security Architecture**

**4.1 Authentication**

* Azure AD integration for identity management
* OpenID Connect for authentication flows
* JWT tokens for maintaining authenticated state
* Secure token storage in web and mobile clients
* Multi-factor authentication support
* Password policy enforcement

**4.2 Authorization**

* Role-based access control for all system functions
* Fine-grained permissions mapped to API operations
* Resource-level security for entities
* Data-level security (row-level security) where needed
* Claims-based identity for authorization decisions
* Separate authorization service for centralized control

**4.3 Data Protection**

* Encryption of sensitive data at rest
* TLS/SSL for all data in transit
* Secure handling of personally identifiable information
* Data masking for sensitive information
* Secure storage of authentication credentials
* Regular security scanning and assessment

**4.4 API Security**

* OAuth 2.0 for API authorization
* API key management for external integrations
* Rate limiting to prevent abuse
* Input validation for all API requests
* Output encoding to prevent injection attacks
* Security headers for all API responses

**4.5 Mobile Security**

* Secure local storage of sensitive data
* Certificate pinning for API communication
* Biometric authentication option
* Secure offline data encryption
* Automatic session termination
* Secure data wiping capabilities

**5. Performance and Scalability**

**5.1 Performance Optimization**

* Redis caching for frequently accessed data
* Query optimization in Entity Framework
* Asynchronous processing for long-running operations
* Efficient API payload design
* Content delivery network for static assets
* Database indexing strategy

**5.2 Scalability Approach**

* Horizontal scaling for API services
* Vertical scaling for database (with optional sharding)
* Load balancing for web applications
* Stateless API design for scalability
* Queue-based processing for background operations
* Auto-scaling configuration for variable loads

**5.3 Capacity Planning**

* Initial sizing based on expected user load
* Scaling strategy for growth
* Database capacity planning
* Storage planning for documents and attachments
* Monitoring for capacity planning
* Performance testing to validate capacity

**6. Resilience and Availability**

**6.1 High Availability**

* Geographic redundancy for critical components
* Load balancing for traffic distribution
* Database failover configuration
* Health monitoring and auto-recovery
* Availability SLAs for each component
* Redundant design for critical paths

**6.2 Disaster Recovery**

* Regular database backups
* Point-in-time recovery capability
* Geo-replication for critical data
* Documented recovery procedures
* Recovery time objective (RTO) definition
* Recovery point objective (RPO) definition

**6.3 Error Handling and Resilience**

* Circuit breaker pattern for external dependencies
* Retry logic with exponential backoff
* Graceful degradation for non-critical features
* Comprehensive error logging
* User-friendly error messages
* Monitoring for error patterns

**7. Monitoring and Diagnostics**

**7.1 Logging Strategy**

* Centralized logging with Azure Application Insights
* Structured logging format
* Contextual logging with correlation IDs
* Log levels for different environments
* Sensitive data filtering in logs
* Log retention policy

**7.2 Monitoring Approach**

* Real-time monitoring with Azure Monitor
* Custom dashboards for key metrics
* Alerting for critical issues
* Performance counter monitoring
* User experience monitoring
* Synthetic transactions for availability testing

**7.3 Diagnostics**

* Distributed tracing across components
* Performance profiling capabilities
* Memory dump analysis for critical issues
* Debug information in development/test environments
* Diagnostic API endpoints (secure, non-production)
* Troubleshooting tools and procedures