QAVerse Software Requirements Specification (SRS)

Version 1.0
Date: July 2025
Prepared By: Muhammad Zohaib, Team Leader
Team Members: Maham Sheikh, Malik Umair Khalid
Institution: Sindh Madressatul Islam University
Project: Final Year Project (FYP)

Table of Contents

Table of Contents

---

1. Introduction

1.1 Purpose

This Software Requirements Specification (SRS) document provides a complete description of the QAVerse web application. QAVerse is a collaborative platform that connects student testers with project maintainers to facilitate crowd-sourced quality assurance testing while providing students with practical QA experience and skill development.

1.2 Scope

QAVerse is a web-based application designed to:

- Enable students to gain hands-on QA testing experience
- Help project maintainers get comprehensive testing coverage
- Provide a gamified learning environment for quality assurance
- Create a community-driven platform for bug reporting and resolution

Key Benefits:

- **For Students:** Practical QA experience, skill development, portfolio building
- **For Maintainers:** Cost-effective testing, diverse user perspectives, bug identification
- **For Academic Institutions:** Bridge between theoretical learning and practical application

1.3 Definitions, Acronyms, and Abbreviations

Term	Definition
QA	Quality Assurance
XP	Experience Points
MVP	Minimum Viable Product
UI	User Interface
API	Application Programming Interface
CRUD	Create, Read, Update, Delete
JWT	JSON Web Token
SRS	Software Requirements Specification
Tester	Student user who tests projects
Maintainer	Project owner who submits projects for testing
Bug Report	Documented issue found during testing
Severity	Impact level of a bug (Low, Medium, High, Critical)

1.4 References

- IEEE Standard 830-1998 for Software Requirements Specifications
- Web Content Accessibility Guidelines (WCAG) 2.1
- OWASP Security Guidelines
- Academic FYP Guidelines - [Your University]

1.5 Overview

This SRS document is organized into six main sections describing the system's functionality, performance requirements, design constraints, and external interfaces. The document serves as a contract between the development team and stakeholders.

---

2. Overall Description

2.1 Product Perspective

QAVerse is a standalone web application that operates as a centralized platform for QA testing collaboration. The system consists of:

- **Web Application:** Primary user interface accessible via web browsers
- **Database System:** Stores user data, projects, bug reports, and system analytics
- **File Storage System:** Manages screenshots, videos, and project assets
- **Notification System:** Real-time updates and email notifications

2.2 Product Functions

The major functions of QAVerse include:

User Management:

- User registration and authentication
- Profile management for testers and maintainers
- Role-based access control

Project Management:

- Project submission and management
- Project categorization and filtering
- Project status tracking

Testing & Bug Reporting:

- Bug report creation and management
- File upload for evidence (screenshots/videos)
- Bug status workflow (pending, approved, rejected, resolved)

Gamification System:

- XP (Experience Points) calculation and tracking

- Leaderboard system
- Badge and achievement system
- Progress tracking and analytics

Communication:

- Comment system for bug reports
- Notification system for updates
- Feedback mechanism between users

2.3 User Classes and Characteristics

2.3.1 Student Testers

- **Primary Users:** Computer science students, QA enthusiasts
- **Technical Expertise:** Beginner to intermediate
- **Frequency of Use:** Regular (daily/weekly)
- **Key Goals:** Learn QA skills, gain experience, build portfolio
- **Characteristics:** Motivated learners, tech-savvy, time-constrained

2.3.2 Project Maintainers

- **Primary Users:** Developers, project managers, students with projects
- **Technical Expertise:** Intermediate to advanced
- **Frequency of Use:** Periodic (project-based)
- **Key Goals:** Get projects tested, identify bugs, improve product quality
- **Characteristics:** Quality-focused, deadline-driven, resource-conscious

2.3.3 System Administrators

- **Primary Users:** Development team, system administrators
- **Technical Expertise:** Advanced
- **Frequency of Use:** As needed for maintenance
- **Key Goals:** System maintenance, user support, analytics monitoring
- **Characteristics:** Technical experts, problem-solvers

2.4 Operating Environment

- **Client-side:** Modern web browsers (Chrome, Firefox, Safari, Edge)
- **Server-side:** Node.js runtime environment
- **Database:** PostgreSQL with Redis for caching
- **Operating System:** Linux-based server (Ubuntu/CentOS)
- **Deployment:** Docker containers on cloud platforms (Heroku, AWS, or similar)

2.5 Design and Implementation Constraints

2.5.1 Technical Constraints

- Web-based application (no mobile app in MVP)
- Must be responsive for mobile browsers
- Support for modern browsers only (last 2 versions)
- File upload size limit: 10MB per file
- Database storage optimization required

2.5.2 Business Constraints

- Academic project timeline (16-20 weeks)
- Limited budget for third-party services
- Team size: 3 developers
- Must demonstrate educational value

2.5.3 Regulatory Constraints

- GDPR compliance for user data
- University academic integrity policies
- Data privacy and security standards

2.6 Assumptions and Dependencies

2.6.1 Assumptions

- Users have stable internet connection
- Users have basic computer literacy
- Modern web browser availability
- Email system for notifications

2.6.2 Dependencies

- Third-party services (Cloudinary for file storage)
- Email service provider (SendGrid/Mailgun)
- Database hosting service
- Web hosting platform

---

3. System Features

3.1 User Authentication and Authorization

3.1.1 Description

Secure user registration, login, and role-based access control system.

3.1.2 Functional Requirements

FR-1.1: User Registration

- System shall allow users to register with email and password
- System shall validate email format and password strength
- System shall send email verification for new accounts
- System shall allow users to select role (Tester or Maintainer)

FR-1.2: User Login

- System shall authenticate users with email and password
- System shall maintain user sessions using JWT tokens
- System shall provide "Remember Me" functionality
- System shall implement secure logout functionality

FR-1.3: Password Management

- System shall allow users to reset forgotten passwords
- System shall enforce password complexity requirements
- System shall hash passwords using bcrypt

FR-1.4: Profile Management

- System shall allow users to update profile information
- System shall display user statistics (XP, badges, testing history)
- System shall allow profile picture upload

3.2 Project Management

3.2.1 Description

Comprehensive project submission, management, and discovery system for maintainers and testers.

3.2.2 Functional Requirements

FR-2.1: Project Submission (Maintainers)

- System shall allow maintainers to submit projects for testing
- System shall collect project metadata (title, description, tech stack, testing URL)
- System shall allow maintainers to specify testing focus areas
- System shall support project categorization (Web, Mobile, Desktop)

FR-2.2: Project Discovery (Testers)

- System shall display available projects in a searchable list
- System shall provide filtering options (category, tech stack, difficulty)
- System shall show project details including testing requirements
- System shall track project testing statistics

FR-2.3: Project Status Management

- System shall track project status (Active, Paused, Completed)
- System shall allow maintainers to pause/resume projects
- System shall archive completed projects

3.3 Bug Reporting System

3.3.1 Description

Comprehensive bug reporting, tracking, and management system.

3.3.2 Functional Requirements

FR-3.1: Bug Report Creation (Testers)

- System shall provide bug report form with required fields
- System shall allow file uploads (screenshots, videos, logs)
- System shall support multiple bug categories (UI, Functionality, Performance, Security)
- System shall require steps to reproduce the bug
- System shall allow severity level selection (Low, Medium, High, Critical)

FR-3.2: Bug Report Management (Maintainers)

- System shall display all bug reports for owned projects
- System shall allow maintainers to approve/reject bug reports
- System shall provide commenting system for bug reports
- System shall track bug resolution status

FR-3.3: Bug Report Workflow

- System shall automatically assign "Pending" status to new reports
- System shall notify maintainers of new bug reports
- System shall update bug status based on maintainer actions
- System shall prevent duplicate bug reporting

3.4 Gamification System

3.4.1 Description

XP-based gamification system to motivate and engage student testers.

3.4.2 Functional Requirements

FR-4.1: XP System

- System shall award XP points for approved bug reports
- System shall implement XP multipliers based on bug severity
- System shall track total XP per user
- System shall display XP history and transactions

FR-4.2: Leaderboard

- System shall maintain real-time leaderboard rankings
- System shall display top testers by XP
- System shall show weekly and monthly leaderboards
- System shall highlight user's current rank

FR-4.3: Badge System

- System shall award badges for specific achievements
- System shall display earned badges on user profiles
- System shall implement badge categories (Bug Hunter, UI Expert, etc.)
- System shall track badge progress

3.5 Communication System

3.5.1 Description

Communication features to facilitate interaction between testers and maintainers.

3.5.2 Functional Requirements

FR-5.1: Commenting System

- System shall allow comments on bug reports
- System shall support threaded discussions
- System shall notify relevant users of new comments
- System shall allow comment editing and deletion

FR-5.2: Notification System

- System shall send real-time notifications for important events
- System shall support email notifications
- System shall allow users to customize notification preferences
- System shall maintain notification history

3.6 Analytics and Reporting

3.6.1 Description

System analytics and reporting capabilities for users and administrators.

3.6.2 Functional Requirements

FR-6.1: User Analytics

- System shall track user activity and engagement
- System shall display personal statistics dashboard
- System shall show testing history and progress
- System shall calculate success rates and performance metrics

FR-6.2: Project Analytics

- System shall track project testing metrics
- System shall display bug report statistics
- System shall show project popularity and engagement
- System shall generate testing quality reports

---

4. External Interface Requirements

4.1 User Interfaces

4.1.1 General UI Requirements

- **Responsive Design:** UI shall adapt to desktop, tablet, and mobile screens
- **Accessibility:** UI shall comply with WCAG 2.1 AA standards
- **Modern Design:** Clean, intuitive interface following current web design trends
- **Performance:** UI shall load within 3 seconds on standard connections

4.1.2 Key User Interface Screens

Login/Registration Screen:

- Email and password fields
- Role selection (Tester/Maintainer)
- Social login options (optional)
- Password reset functionality

Dashboard (Tester):

- Available projects grid
- Personal statistics widget
- Recent activity feed
- XP and badge display

Dashboard (Maintainer):

- Project management interface
- Bug report summary
- Project statistics
- Quick actions panel

Project Details Screen:

- Project information display
- Testing instructions
- Bug report submission form
- Project statistics

Bug Report Screen:

- Bug report form with rich text editor
- File upload area
- Severity and category selection
- Steps to reproduce section

Profile Screen:

- User information display
- XP and badge showcase
- Activity history
- Settings panel

4.2 Hardware Interfaces

- **Standard Web Browser:** Compatible with desktop and mobile browsers
- **Camera Access:** For screenshot/video capture (optional)
- **File System Access:** For file upload functionality

4.3 Software Interfaces

4.3.1 Database Interface

- **PostgreSQL Database:** Primary data storage
- **Redis Cache:** Session management and caching
- **Connection Pool:** Efficient database connection management

4.3.2 External Service Interfaces

- **Cloudinary API:** File upload and storage service
- **Email Service API:** Notification delivery (SendGrid/Mailgun)
- **Authentication Service:** JWT token management

4.3.3 Web Service Interfaces

- **RESTful API:** JSON-based API for frontend-backend communication
- **File Upload API:** Multipart form data handling
- **Real-time API:** WebSocket for live notifications

4.4 Communication Interfaces

- **HTTPS Protocol:** Secure communication between client and server
- **WebSocket Protocol:** Real-time bidirectional communication
- **Email Protocol:** SMTP for email notifications
- **File Transfer Protocol:** HTTP multipart for file uploads

---

5. Non-Functional Requirements

5.1 Performance Requirements

5.1.1 Response Time

- **Page Load Time:** < 3 seconds for initial page load
- **API Response Time:** < 1 second for standard operations
- **File Upload Time:** < 30 seconds for files up to 10MB
- **Search Response Time:** < 2 seconds for project/bug searches

5.1.2 Throughput

- **Concurrent Users:** Support 100 concurrent users minimum
- **Database Operations:** Handle 1000 database queries per minute
- **File Uploads:** Support 10 concurrent file uploads

5.1.3 Capacity

- **User Accounts:** Support 1000+ user accounts
- **Projects:** Support 500+ active projects
- **Bug Reports:** Support 10,000+ bug reports
- **File Storage:** Support 10GB+ total file storage

5.2 Safety Requirements

- **Data Backup:** Automated daily database backups
- **Error Handling:** Graceful error handling without system crashes
- **Data Validation:** Input validation to prevent malicious data
- **Recovery:** System recovery within 24 hours of failure

5.3 Security Requirements

5.3.1 Authentication and Authorization

- **Password Security:** Bcrypt hashing with salt rounds ≥ 12
- **Session Management:** Secure JWT token implementation
- **Access Control:** Role-based access control (RBAC)
- **API Security:** Rate limiting and input validation

5.3.2 Data Protection

- **Data Encryption:** HTTPS encryption for all communications
- **Personal Data:** GDPR-compliant data handling
- **File Security:** Secure file upload with type validation
- **Database Security:** SQL injection prevention

5.3.3 Privacy

- **User Data:** Minimal data collection principle
- **Data Retention:** Clear data retention policies
- **User Consent:** Explicit consent for data processing
- **Data Export:** User data export functionality

5.4 Availability Requirements

- **Uptime:** 99% uptime during business hours
- **Maintenance Window:** Scheduled maintenance during low-usage periods
- **Failover:** Automatic failover for critical system components
- **Monitoring:** Real-time system health monitoring

5.5 Usability Requirements

5.5.1 Ease of Use

- **Learning Curve:** New users should be productive within 15 minutes
- **Navigation:** Intuitive navigation with max 3 clicks to any feature
- **Error Messages:** Clear, actionable error messages
- **Help System:** Contextual help and tutorials

5.5.2 Accessibility

- **Screen Readers:** Compatible with screen reader software
- **Keyboard Navigation:** Full keyboard accessibility
- **Color Contrast:** WCAG 2.1 AA contrast ratios
- **Font Size:** Scalable font sizes for vision impairment

5.6 Reliability Requirements

- **Mean Time Between Failures (MTBF):** > 720 hours
- **Mean Time To Recovery (MTTR):** < 4 hours
- **Error Rate:** < 0.1% error rate for user operations
- **Data Integrity:** 99.99% data integrity guarantee

5.7 Maintainability Requirements

- **Code Quality:** Modular, well-documented code
- **Testing:** 80%+ code coverage with automated tests
- **Documentation:** Comprehensive system documentation
- **Deployment:** Automated deployment with rollback capability

5.8 Portability Requirements

- **Browser Compatibility:** Chrome, Firefox, Safari, Edge (latest 2 versions)
- **Operating Systems:** Windows, macOS, Linux client support
- **Mobile Compatibility:** Responsive design for mobile browsers
- **Cloud Deployment:** Deployable on multiple cloud platforms

---

6. Other Requirements

6.1 Legal Requirements

- **Terms of Service:** Clear terms governing platform usage
- **Privacy Policy:** GDPR-compliant privacy policy
- **User Agreement:** Academic use agreement for students
- **Intellectual Property:** Clear IP ownership policies

6.2 Standards Compliance

- **Web Standards:** HTML5, CSS3, ECMAScript 2020+ compliance
- **API Standards:** RESTful API design principles
- **Security Standards:** OWASP security guidelines
- **Accessibility Standards:** WCAG 2.1 AA compliance

6.3 Cultural Requirements

- **Language:** English primary language with UTF-8 support
- **Time Zones:** UTC timestamp storage with local display
- **Cultural Sensitivity:** Inclusive design principles
- **Academic Context:** Alignment with educational objectives

6.4 Environmental Requirements

- **Energy Efficiency:** Optimized code for minimal server resource usage
- **Sustainable Practices:** Efficient database queries and caching
- **Green Hosting:** Preference for renewable energy-powered hosting

---

7. Appendices

Appendix A: Glossary

Bug Severity Levels:

- **Critical:** System crashes, data loss, security vulnerabilities
- **High:** Major functionality broken, significant user impact
- **Medium:** Moderate functionality issues, workarounds available
- **Low:** Minor issues, cosmetic problems, enhancement suggestions

User Roles:

- **Tester:** Student user who performs testing activities
- **Maintainer:** Project owner who submits projects for testing
- **Admin:** System administrator with full access privileges

Appendix B: Use Case Diagrams

```
[Use Case Diagram would be inserted here showing:]
- Tester use cases (Register, Browse Projects, Submit Bug Reports, View XP)
- Maintainer use cases (Submit Projects, Review Bug Reports, Manage Projects)
- Admin use cases (User Management, System Analytics, Content Moderation)
```

Appendix C: Data Flow Diagrams

```
[Data Flow Diagrams would be inserted here showing:]
- User registration and authentication flow
- Bug report submission and approval flow
- XP calculation and leaderboard update flow
- Project submission and discovery flow
```

Appendix D: Entity Relationship Diagram

```
[ERD would be inserted here showing relationships between:]
- Users, Projects, Bug Reports, XP Transactions, Badges, Comments
```

Appendix E: Wireframes and Mockups

```
[Wireframes would be inserted here for:]
- Login/Registration pages
- Dashboard layouts
- Project listing and detail pages
- Bug report forms
- Profile pages
```

Appendix F: API Specifications

```
[API documentation would be inserted here including:]
- Authentication endpoints
- User management endpoints
- Project management endpoints
- Bug report endpoints
- File upload endpoints
```

Document Control:

- **Version:** 1.0
- **Last Updated:** July 2025
- **Next Review:** Upon milestone completion
- **Approved By:** Mam Wajiha
- **Distribution:** Development team, project supervisor, external examiner

---

This document serves as the primary specification for the QAVerse FYP project and will be updated as requirements evolve during development.