Product Requirements Document (PRD): Exam-Taking Platform - Student Module

Document Information

- **Product Name**: Student Exam Assessment Platform
- **Version**: 1.0
- **Date**: September 2025
- **Product Manager**: [To be assigned]
- **Development Team**: Full-Stack Team (React.js + Backend)
- **Document Type**: Product Requirements Document (PRD)

1. Executive Summary

1.1 Product Vision

To develop a secure, intuitive, and reliable online exam-taking platform that enables students to complete assessments efficiently while maintaining academic integrity and providing a seamless user experience.

1.2 Strategic Objectives

- Create a robust digital assessment solution for educational institutions
- Ensure secure authentication and data protection throughout the examination process
- Provide real-time feedback and immediate score calculation
- Support scalable architecture for multiple concurrent users
- Establish a foundation for future educational technology enhancements

1.3 Success Metrics

- **User Experience**: 95% successful exam completion rate without technical issues
- **Performance**: Page load times under 2 seconds, API response times under 500ms
- **Security**: Zero security breaches, 100% JWT token validation success
- **Reliability**: 99.9% uptime during exam periods
- **User Satisfaction**: 4.5/5 average user rating post-exam

2. Problem Statement

2.1 Current Challenges

Educational institutions face significant challenges in conducting secure, scalable online assessments:

- **Manual Assessment Limitations**: Traditional paper-based exams are time-intensive and prone to human error
- **Security Concerns**: Ensuring academic integrity without physical supervision
- **Scalability Issues**: Difficulty managing large numbers of simultaneous test-takers
- **Real-time Monitoring**: Lack of automated timing and submission systems
- **Data Management**: Manual score calculation and result processing inefficiencies

2.2 Market Opportunity

The global e-learning market is projected to reach $375 billion by 2026, with online assessment tools representing a significant growth segment. Educational institutions increasingly require digital solutions that maintain examination standards while providing flexibility and efficiency.

3. Target Audience & User Personas

3.1 Primary User Persona: Student Test-Taker

Demographics:

- Age: 16-35 years
- Education Level: High school to graduate students
- Technical Proficiency: Intermediate to advanced
- Device Usage: Desktop/laptop primary, mobile secondary

Goals:

- Complete exams efficiently and securely
- Receive immediate feedback on performance
- Navigate the platform intuitively without technical barriers
- Trust that their data and responses are secure

Pain Points:

- Anxiety about technical failures during exams
- Confusion with complex navigation systems
- Concerns about time management during assessments
- Uncertainty about submission confirmation

User Stories:

- "As a student, I want to log in securely so that only I can access my exam"
- "As a student, I want to see a countdown timer so that I can manage my time effectively"
- "As a student, I want to navigate between questions easily so that I can review my answers"
- "As a student, I want automatic submission when time expires so that I don't lose my work"
- "As a student, I want to see my results immediately so that I know my performance"

3.2 Secondary Stakeholders

Educational Institutions:

- Need reliable, secure assessment tools
- Require detailed analytics and reporting capabilities
- Seek cost-effective solutions with minimal training requirements

Technical Administrators:

- Require easy deployment and maintenance
- Need comprehensive security features
- Want scalable architecture for growing user bases

4. Functional Requirements

4.1 User Authentication & Security (Priority: Critical)

4.1.1 User Registration System

- **Registration Interface**: Clean, accessible form with real-time validation
- **Required Fields**: Email address, password, full name, student ID (optional)
- **Password Requirements**: Minimum 8 characters, combination of uppercase/lowercase letters, numbers, and special characters
- **Email Verification**: Optional email confirmation for account activation
- **Duplicate Prevention**: System validation to prevent duplicate email registrations

4.1.2 User Login System

- **Login Interface**: Streamlined form with email/password authentication
- **JWT Token Implementation**: Secure token generation with appropriate expiration (30 minutes)
- **Session Management**: Automatic token refresh for continuous user sessions
- **Security Features**: Rate limiting, account lockout after multiple failed attempts

- **"Remember Me" Functionality**: Optional extended session duration

4.1.3 Security Protocols

- **Password Encryption**: Bcrypt hashing with salt rounds (minimum 12)
- **JWT Security**: RS256 algorithm, secure secret key management
- **HTTPS Enforcement**: All communications encrypted via SSL/TLS
- **Cross-Site Scripting (XSS) Protection**: Input sanitization and validation
- **SQL Injection Prevention**: Parameterized queries and ORM usage

4.2 Exam Interface & Navigation (Priority: Critical)

4.2.1 Exam Initialization

- **"Start Exam" Button**: Prominent, clearly labeled entry point
- **Pre-exam Instructions**: Display exam rules, duration, and navigation tips
- **System Requirements Check**: Browser compatibility and connection verification
- **Question Randomization**: Backend-driven random question selection from question pool

4.2.2 Question Display System

- **Multiple Choice Questions (MCQs)**: Clean presentation with radio button selection
- **Question Numbering**: Clear sequential numbering (e.g., "Question 5 of 20")
- **Option Labeling**: Consistent A, B, C, D labeling system
- **Question Text Formatting**: Support for text formatting, images (future enhancement)
- **Answer Selection Feedback**: Visual confirmation of selected answers

4.2.3 Navigation Controls

- **Next/Previous Buttons**: Intuitive navigation between questions
- **Question Jump Feature**: Sidebar or dropdown for direct question access
- **Progress Indicator**: Visual progress bar showing completion percentage
- **Answer Status Indicators**: Clear marking of answered/unanswered questions
- **Review Mode**: Comprehensive overview of all questions and answers

4.3 Timer & Submission System (Priority: Critical)

4.3.1 Countdown Timer Implementation

- **Display Format**: Clear, prominent timer showing hours:minutes:seconds
- **Timer Position**: Fixed position (top-right or top-center) always visible
- **Color Coding**: Visual warnings as time expires (green > yellow > red)
- **Time Warnings**: Alerts at 10 minutes, 5 minutes, and 1 minute remaining
- **Server Synchronization**: Time validation against server time to prevent manipulation

4.3.2 Auto-Submission Features

- **Automatic Submission**: Immediate submission when timer reaches zero
- **Grace Period**: 30-second grace period for final submissions
- **Connection Loss Handling**: Automatic submission on connection restoration
- **Submission Confirmation**: Visual and textual confirmation of successful submission

4.3.3 Manual Submission

- **Submit Button**: Prominent "Submit Exam" button available throughout exam
- **Confirmation Dialog**: "Are you sure?" confirmation before final submission
- **Review Opportunity**: Final review page before submission confirmation
- **Submission Receipt**: Confirmation page with timestamp and submission ID

4.4 Score Calculation & Results Display (Priority: High)

4.4.1 Scoring System

- **Real-time Calculation**: Immediate score computation upon submission
- **Scoring Algorithm**: Configurable scoring (e.g., +1 correct, 0 incorrect)
- **Percentage Calculation**: Display both raw score and percentage
- **Pass/Fail Determination**: Configurable passing threshold

4.4.2 Results Interface

- **Results Page**: Clean, comprehensive results display
- **Score Breakdown**: Total questions, correct answers, incorrect answers, percentage
- **Question Review**: Optional detailed breakdown by question (configurable)
- **Performance Analytics**: Visual representations (charts/graphs) of performance
- **Certificate Generation**: Downloadable certificate for passing scores (future enhancement)

5. Non-Functional Requirements

5.1 Performance Requirements

- **Page Load Time**: Maximum 2 seconds for initial page load
- **API Response Time**: Maximum 500ms for all API calls
- **Question Navigation**: Instantaneous switching between questions (<100ms)
- **Database Query Performance**: Optimized queries with response times under 200ms
- **Concurrent Users**: Support for minimum 500 simultaneous exam takers

5.2 Security Requirements

- **Data Encryption**: All data encrypted in transit (HTTPS) and at rest
- **Authentication Security**: Secure JWT implementation with appropriate expiration
- **Input Validation**: Comprehensive server-side validation for all user inputs
- **Session Security**: Secure session management with automatic timeout
- **Audit Logging**: Comprehensive logging of all user actions and system events

5.3 Usability Requirements

- **Browser Compatibility**: Support for Chrome, Firefox, Safari, Edge (latest 2 versions)
- **Responsive Design**: Mobile-friendly design (tablet support minimum)
- **Accessibility**: WCAG 2.1 AA compliance for users with disabilities
- **Language Support**: Unicode support for multiple languages (future enhancement)
- **User Interface**: Intuitive, minimal learning curve interface

5.4 Reliability Requirements

- **Uptime**: 99.9% availability during scheduled exam periods
- **Error Handling**: Graceful error handling with user-friendly messages
- **Data Integrity**: Zero data loss during submission process
- **Backup Systems**: Automated backup of all exam data and user responses
- **Recovery Time**: Maximum 15 minutes recovery time from system failures

5.5 Scalability Requirements

- **User Growth**: Architecture supporting 10x user growth without major changes
- **Database Scalability**: Efficient database design supporting large question banks
- **Server Scalability**: Horizontal scaling capabilities for increased load
- **Content Delivery**: CDN integration for static assets (future enhancement)

6. Technical Requirements

6.1 Technology Stack Options

Option 1: MERN Stack

- **Frontend**: React.js 18+ with TypeScript
- **Backend**: Node.js with Express.js framework
- **Database**: MongoDB with Mongoose ODM
- **Authentication**: JWT with refresh token mechanism
- **State Management**: Redux Toolkit or Context API

Option 2: React + Python Stack

- **Frontend**: React.js 18+ with TypeScript
- **Backend**: Python with FastAPI/Django/Flask
- **Database**: PostgreSQL or MySQL
- **Authentication**: JWT with Python libraries
- **ORM**: SQLAlchemy (Flask) or Django ORM

6.2 Frontend Technical Specifications

6.2.1 React.js Implementation

- **Component Architecture**: Functional components with React Hooks
- **State Management**: Context API for global state, useState/useReducer for local state
- **Routing**: React Router v6 for client-side navigation
- **Form Handling**: React Hook Form with validation
- **Styling**: CSS Modules or Styled Components
- **API Integration**: Axios for HTTP requests with interceptors

6.2.2 Key React Hooks Usage

- **useState**: Local component state management
- **useEffect**: Side effects and lifecycle management
- **useContext**: Global state sharing
- **useReducer**: Complex state logic management
- **useMemo/useCallback**: Performance optimization
- **Custom Hooks**: Reusable stateful logic

6.2.3 Timer Implementation

```
// Custom hook for countdown timer
const useCountdown = (targetTime) => {
  const [timeLeft, setTimeLeft] = useState(targetTime);

  useEffect(() => {
    const timer = setInterval(() => {
      setTimeLeft(prev => {
        if (prev <= 1) {
          clearInterval(timer);
          // Auto-submit logic
          return 0;
        }
        return prev - 1;
      });
    }, 1000);

    return () => clearInterval(timer);
  }, []);

  return timeLeft;
};
```

6.3 Backend Technical Specifications

6.3.1 API Design Principles

- **RESTful Architecture**: Standard HTTP methods and status codes
- **API Versioning**: Version control for API endpoints (/api/v1/)
- **Error Handling**: Consistent error response format
- **Documentation**: OpenAPI/Swagger documentation
- **Rate Limiting**: Request throttling to prevent abuse

6.3.2 Database Schema Design

Users Table:

```
CREATE TABLE users (
  id PRIMARY KEY,
  email VARCHAR(255) UNIQUE NOT NULL,
  password_hash VARCHAR(255) NOT NULL,
  full_name VARCHAR(255) NOT NULL,
  student_id VARCHAR(100),
  created_at TIMESTAMP,
  updated_at TIMESTAMP
);
```

Questions Table:

```
CREATE TABLE questions (
  id PRIMARY KEY,
  question_text TEXT NOT NULL,
  option_a VARCHAR(500) NOT NULL,
  option_b VARCHAR(500) NOT NULL,
  option_c VARCHAR(500) NOT NULL,
  option_d VARCHAR(500) NOT NULL,
  correct_answer CHAR(1) NOT NULL,
  difficulty_level ENUM('easy', 'medium', 'hard'),
  subject VARCHAR(100),
  created_at TIMESTAMP
);
```

Exam Sessions Table:

```
CREATE TABLE exam_sessions (
  id PRIMARY KEY,
  user_id FOREIGN KEY,
  start_time TIMESTAMP,
  end_time TIMESTAMP,
  duration_minutes INTEGER NOT NULL,
  status ENUM('active', 'completed', 'submitted'),
  score INTEGER,
  total_questions INTEGER,
  created_at TIMESTAMP
);
```

User Answers Table:

```
CREATE TABLE user_answers (
  id PRIMARY KEY,
  session_id FOREIGN KEY,
  question_id FOREIGN KEY,
  user_answer CHAR(1),
  is_correct BOOLEAN,
  answered_at TIMESTAMP
);
```

6.3.3 Authentication Implementation

```
// JWT Token Structure
const tokenPayload = {
  userId: user.id,
  email: user.email,
  iat: Math.floor(Date.now() / 1000),
  exp: Math.floor(Date.now() / 1000) + (30 * 60) // 30 minutes
};

// Middleware for protected routes
const authenticateToken = (req, res, next) => {
  const authHeader = req.headers['authorization'];
```

```
  const token = authHeader && authHeader.split(' ')[1];

  if (!token) {
    return res.status(401).json({ error: 'Access token required' });
  }

  jwt.verify(token, process.env.JWT_SECRET, (err, user) => {
    if (err) {
      return res.status(403).json({ error: 'Invalid token' });
    }
    req.user = user;
    next();
  });
};
```

6.4 Security Implementation

6.4.1 Password Security

```
const bcrypt = require('bcrypt');
const saltRounds = 12;

// Password hashing
const hashPassword = async (password) => {
  return await bcrypt.hash(password, saltRounds);
};

// Password verification
const verifyPassword = async (password, hash) => {
  return await bcrypt.compare(password, hash);
};
```

6.4.2 Input Validation

- **Frontend Validation**: Real-time validation using React Hook Form
- **Backend Validation**: Server-side validation using Joi or similar
- **SQL Injection Prevention**: Parameterized queries or ORM usage
- **XSS Prevention**: Input sanitization and output encoding

7. User Experience (UX) Requirements

7.1 Design Principles

- **Simplicity**: Clean, uncluttered interface focused on exam-taking
- **Consistency**: Consistent design patterns throughout the application
- **Accessibility**: WCAG 2.1 AA compliance for inclusive design
- **Responsiveness**: Mobile-first design approach

- **Performance**: Optimized for fast loading and smooth interactions

7.2 Visual Design Guidelines

7.2.1 Color Scheme

- **Primary Colors**: Professional blue (#2563eb) for main actions
- **Secondary Colors**: Gray scale for backgrounds and text
- **Success Color**: Green (#10b981) for positive feedback
- **Warning Color**: Amber (#f59e0b) for time warnings
- **Error Color**: Red (#ef4444) for errors and critical alerts

7.2.2 Typography

- **Primary Font**: System fonts (San Francisco, Segoe UI, Roboto)
- **Heading Sizes**: h1: 2.5rem, h2: 2rem, h3: 1.5rem
- **Body Text**: 1rem with 1.5 line height for readability
- **Code/Monospace**: For technical elements if needed

7.2.3 Layout Specifications

- **Container Width**: Maximum 1200px with responsive breakpoints
- **Grid System**: CSS Grid or Flexbox for layouts
- **Spacing**: 8px base unit system (8px, 16px, 24px, 32px)
- **Border Radius**: 8px for cards and buttons
- **Shadows**: Subtle shadows for depth and hierarchy

7.3 User Journey Flow

7.3.1 Registration Flow

1. Landing page with clear registration call-to-action
2. Registration form with real-time validation
3. Email verification (if implemented)
4. Welcome message and dashboard access

7.3.2 Exam Taking Flow

1. Login authentication
2. Dashboard with available exams
3. Exam instructions and requirements

4. Exam interface with timer and navigation
5. Submission confirmation
6. Results display with detailed feedback

7.4 Mobile Responsiveness

- **Breakpoints**: Mobile (320px+), Tablet (768px+), Desktop (1024px+)
- **Touch Targets**: Minimum 44px for touch elements
- **Navigation**: Mobile-optimized navigation patterns
- **Performance**: Optimized images and assets for mobile networks

8. API Specifications

8.1 Authentication Endpoints

POST /api/v1/auth/register

```
{
  "request": {
    "email": "student@example.com",
    "password": "SecurePass123!",
    "full_name": "John Doe",
    "student_id": "ST001"
  },
  "response": {
    "success": true,
    "message": "User registered successfully",
    "user": {
      "id": 1,
      "email": "student@example.com",
      "full_name": "John Doe"
    }
  }
}
```

POST /api/v1/auth/login

```
{
  "request": {
    "email": "student@example.com",
    "password": "SecurePass123!"
  },
  "response": {
    "success": true,
    "token": "eyJhbGciOiJIUzI1NiIs...",
    "user": {
      "id": 1,
```

```
      "email": "student@example.com",
      "full_name": "John Doe"
    },
    "expires_in": 1800
  }
}
```

8.2 Exam Endpoints

GET /api/v1/exams/start

```
{
  "response": {
    "exam_id": "exam_001",
    "duration_minutes": 30,
    "total_questions": 20,
    "questions": [
      {
        "id": 1,
        "question_text": "What is React?",
        "options": {
          "a": "A JavaScript library",
          "b": "A database",
          "c": "A web server",
          "d": "An operating system"
        }
      }
    ]
  }
}
```

POST /api/v1/exams/submit

```
{
  "request": {
    "exam_id": "exam_001",
    "answers": [
      {"question_id": 1, "answer": "a"},
      {"question_id": 2, "answer": "c"}
    ],
    "submission_time": "2025-09-01T10:30:00Z"
  },
  "response": {
    "success": true,
    "score": 18,
    "total_questions": 20,
    "percentage": 90,
    "passed": true,
    "submission_id": "sub_001"
  }
}
```

9. Quality Assurance & Testing

9.1 Testing Strategy

9.1.1 Frontend Testing

- **Unit Testing**: Jest and React Testing Library for component testing
- **Integration Testing**: Testing component interactions and API integration
- **End-to-End Testing**: Cypress or Playwright for complete user journey testing
- **Performance Testing**: Lighthouse and Core Web Vitals optimization
- **Accessibility Testing**: Automated accessibility testing with axe-core

9.1.2 Backend Testing

- **Unit Testing**: Testing individual functions and methods
- **Integration Testing**: API endpoint testing with supertest
- **Database Testing**: Database operation and query testing
- **Security Testing**: Authentication and authorization testing
- **Load Testing**: Performance testing under concurrent user load

9.1.3 Security Testing

- **Authentication Testing**: JWT token security and session management
- **Input Validation Testing**: SQL injection and XSS prevention
- **Authorization Testing**: Role-based access control verification
- **Data Encryption Testing**: HTTPS and data protection verification

9.2 Test Coverage Requirements

- **Code Coverage**: Minimum 80% code coverage for critical paths
- **Feature Coverage**: 100% coverage of core exam-taking functionality
- **Browser Testing**: Cross-browser compatibility testing
- **Device Testing**: Mobile and tablet responsiveness testing

10. Deployment & DevOps

10.1 Deployment Architecture

10.1.1 Development Environment

- **Local Development**: Docker containers for consistent development environment
- **Version Control**: Git with feature branch workflow
- **Code Review**: Pull request reviews before merging
- **Continuous Integration**: GitHub Actions or similar CI/CD pipeline

10.1.2 Production Environment

- **Hosting Options**: AWS, Google Cloud Platform, or Azure
- **Database Hosting**: Managed database services (RDS, MongoDB Atlas)
- **CDN**: CloudFlare or AWS CloudFront for static assets
- **SSL Certificates**: Let's Encrypt or cloud provider SSL
- **Monitoring**: Application performance monitoring and error tracking

10.2 Security & Compliance

- **Environment Variables**: Secure storage of sensitive configuration
- **Database Backups**: Automated daily backups with retention policy
- **Security Updates**: Regular dependency updates and security patches
- **Access Control**: Limited production access with audit logging

11. Success Metrics & KPIs

11.1 User Experience Metrics

- **Task Completion Rate**: >95% successful exam completions
- **User Satisfaction Score**: >4.5/5 average rating
- **Time to Complete Registration**: <2 minutes average
- **Navigation Errors**: <1% users experiencing navigation issues

11.2 Technical Performance Metrics

- **Page Load Time**: <2 seconds for initial load
- **API Response Time**: <500ms average response time
- **Uptime**: >99.9% availability during exam periods
- **Error Rate**: <0.1% application errors

11.3 Security Metrics

- **Authentication Success Rate**: >99% successful login attempts
- **Security Incidents**: Zero security breaches
- **Token Validation Success**: 100% JWT token validation
- **Data Integrity**: Zero data loss incidents

11.4 Business Metrics

- **User Adoption Rate**: Track registration and usage growth
- **System Utilization**: Peak concurrent users and resource usage
- **Cost Efficiency**: Infrastructure cost per user
- **Feature Adoption**: Usage statistics for key features

12. Risk Assessment & Mitigation

12.1 Technical Risks

12.1.1 High-Risk Items

Risk	Impact	Probability	Mitigation Strategy
Server downtime during exams	High	Medium	Redundant servers, load balancing, monitoring
Database corruption	High	Low	Regular backups, database replication
Security breaches	High	Medium	Comprehensive security testing, regular audits
Performance degradation	Medium	Medium	Load testing, performance monitoring, caching

12.1.2 Medium-Risk Items

Risk	Impact	Probability	Mitigation Strategy
Browser compatibility issues	Medium	Medium	Cross-browser testing, progressive enhancement
API rate limiting	Medium	Low	Rate limiting implementation, usage monitoring
Third-party service failures	Medium	Low	Service monitoring, fallback options

12.2 User Experience Risks

- **Timer Accuracy**: Client-server time synchronization issues
- **Network Connectivity**: Connection loss during exams
- **Device Compatibility**: Performance on low-end devices
- **User Error**: Accidental submission or navigation mistakes

12.3 Mitigation Strategies

- **Comprehensive Testing**: Extensive testing across all scenarios
- **Monitoring & Alerting**: Real-time system monitoring
- **Backup Systems**: Redundancy for critical components
- **User Support**: Clear documentation and support channels

13. Timeline & Milestones

13.1 Development Phases

Phase 1: Foundation (Week 1-2)

- **Week 1**: Project setup, development environment, database design
- **Week 2**: Authentication system implementation, basic UI components

Phase 2: Core Features (Week 3-4)

- **Week 3**: Exam interface development, question display system
- **Week 4**: Timer implementation, navigation system

Phase 3: Integration & Testing (Week 5)

- **Week 5**: API integration, score calculation, submission system, testing

Phase 4: Polish & Deployment (Week 6-7)

- **Week 6**: UI/UX refinements, performance optimization
- **Week 7**: Final testing, deployment, documentation

13.2 Key Milestones

- **M1 (End Week 2)**: Authentication system complete
- **M2 (End Week 4)**: Core exam functionality complete
- **M3 (End Week 5)**: Full integration and basic testing complete
- **M4 (End Week 7)**: Production-ready application deployed

13.3 Delivery Timeline

- **MVP Delivery**: End of Week 5 (basic functionality)
- **Beta Release**: End of Week 6 (feature complete)
- **Production Release**: End of Week 7 (fully tested and deployed)

14. Dependencies & Constraints

14.1 Technical Dependencies

- **Frontend Framework**: React.js 18+ availability and stability
- **Backend Framework**: Node.js/Express.js or Python framework selection
- **Database**: MongoDB, PostgreSQL, or MySQL hosting availability
- **Cloud Services**: Reliable cloud hosting provider access
- **Third-party Libraries**: JWT libraries, bcrypt, validation libraries

14.2 Resource Dependencies

- **Development Team**: Skilled full-stack developers
- **Testing Resources**: QA engineers and testing tools
- **Design Resources**: UI/UX design expertise
- **Infrastructure**: Adequate hosting and database resources

14.3 Constraints

- **Time Constraint**: 2-5 working days development window
- **Scope Constraint**: Student-side only (no admin panel)
- **Technology Constraint**: Must use specified technology stack options
- **Security Constraint**: Must implement JWT authentication
- **Performance Constraint**: Must support concurrent users efficiently

15. Future Enhancements & Roadmap

15.1 Phase 2 Features (Post-MVP)

Admin Panel Development

- Question bank management system
- Exam configuration and scheduling
- User management and analytics dashboard
- Advanced reporting and analytics

Enhanced Security Features

- Webcam proctoring integration
- Browser lockdown functionality
- Advanced cheating detection algorithms
- Secure browser requirements

Advanced Functionality

- Multi-language support
- Offline exam capability
- Advanced question types (drag-drop, fill-in-blank)
- Adaptive testing algorithms

15.2 Integration Possibilities

- **Learning Management Systems**: Canvas, Moodle, Blackboard integration
- **Single Sign-On**: SAML, OAuth 2.0 integration
- **Analytics Platforms**: Google Analytics, custom analytics dashboard
- **Communication Systems**: Email notifications, SMS alerts

15.3 Scalability Enhancements

- **Microservices Architecture**: Service decomposition for better scalability
- **Caching Strategies**: Redis/Memcached implementation
- **CDN Integration**: Global content delivery optimization
- **Database Optimization**: Sharding and replication strategies

16. Success Criteria & Acceptance Criteria

16.1 Functional Success Criteria

Must-Have (Critical)

- [ ] User registration and login with JWT authentication
- [ ] Secure exam interface with randomized questions
- [ ] Functional countdown timer with auto-submission
- [ ] Question navigation (next/previous functionality)
- [ ] Automatic score calculation and results display
- [ ] All API endpoints functional and tested

Should-Have (High Priority)

- [ ] Mobile-responsive design
- [ ] Cross-browser compatibility (Chrome, Firefox, Safari, Edge)
- [ ] Input validation and error handling
- [ ] Session management and token refresh
- [ ] Progress indicators and user feedback

Could-Have (Medium Priority)

- [ ] Question review before submission
- [ ] Performance optimizations
- [ ] Advanced UI animations and transitions
- [ ] Detailed analytics and logging

16.2 Technical Success Criteria

- [ ] **Performance**: Page load times under 2 seconds
- [ ] **Security**: JWT implementation with proper validation
- [ ] **Scalability**: Support for 100+ concurrent users
- [ ] **Reliability**: 99% uptime during testing
- [ ] **Code Quality**: Clean, documented, maintainable code

16.3 User Acceptance Criteria

- [ ] **Usability**: Users can complete exam without technical assistance
- [ ] **Accessibility**: WCAG 2.1 AA compliance
- [ ] **Error Recovery**: Graceful handling of network interruptions
- [ ] **Data Integrity**: Zero data loss during normal operation
- [ ] **User Satisfaction**: Positive feedback from test users

17. Appendices

17.1 Glossary of Terms

- **JWT (JSON Web Token)**: Secure token format for authentication
- **MCQ**: Multiple Choice Question
- **API**: Application Programming Interface
- **HTTPS**: HTTP Secure protocol
- **WCAG**: Web Content Accessibility Guidelines

- **CDN**: Content Delivery Network
- **ORM**: Object-Relational Mapping
- **SPA**: Single Page Application

17.2 Technical Standards

- **HTTP Status Codes**: Standard REST API status codes
- **Date Formats**: ISO 8601 format for all timestamps
- **Password Policy**: Minimum 8 characters with complexity requirements
- **Token Expiration**: 30-minute standard expiration time
- **API Rate Limiting**: 100 requests per minute per user

17.3 Compliance Requirements

- **Data Protection**: GDPR compliance for EU users (future consideration)
- **Accessibility Standards**: WCAG 2.1 AA compliance
- **Security Standards**: OWASP security best practices
- **Browser Standards**: W3C web standards compliance

Document Approval

Role	Name	Signature	Date
Product Manager	[To be assigned]		
Technical Lead	[To be assigned]		
UX Designer	[To be assigned]		
QA Lead	[To be assigned]		

Document Status: Draft
Next Review Date: [To be scheduled]
Distribution List: Development Team, Stakeholders, QA Team

This PRD serves as the single source of truth for the Student Exam Assessment Platform development project. All team members should refer to this document for project requirements, scope, and acceptance criteria.