AI-Based Automated STEM Evaluation System - Security and RBAC Design

1. Security Overview

1.1 Security Objectives

- **Confidentiality**: Protect student data, assessment content, and proprietary AI prompts
- **Integrity**: Ensure data accuracy and prevent unauthorized modifications
- **Availability**: Maintain system uptime and protect against DoS attacks
- **Compliance**: Meet FERPA, COPPA, and GDPR requirements
- **Privacy**: Safeguard PII and maintain student privacy

1.2 Security Principles

- **Defense in Depth**: Multiple layers of security controls
- **Least Privilege**: Users and services granted minimum necessary permissions
- **Zero Trust**: Verify every access request
- **Security by Design**: Security integrated from the start
- **Fail Securely**: Default deny on errors

1.3 Threat Model

Assets to Protect

1. **Student Data**: PII, performance data, submissions
2. **Assessment Content**: Questions, answers, rubrics
3. **AI Prompts**: Proprietary evaluation prompts and templates
4. **Authentication Credentials**: Passwords, tokens, API keys
5. **System Infrastructure**: Databases, storage, services

Threat Actors

1. **External Attackers**: Hackers seeking data or system access
2. **Malicious Students**: Attempting to cheat or access unauthorized data
3. **Insider Threats**: Compromised coach or admin accounts
4. **Automated Bots**: Scraping, credential stuffing, DDoS

Attack Vectors

1. **Web Application Attacks**: SQL injection, XSS, CSRF
2. **Authentication Attacks**: Credential stuffing, session hijacking
3. **Authorization Bypass**: Privilege escalation, IDOR
4. **API Abuse**: Rate limit bypass, data scraping
5. **Social Engineering**: Phishing, pretexting
6. **Infrastructure**: DDoS, cloud misconfigurations

2. Authentication Architecture

2.1 Authentication Methods

Primary: Email and Password

```
User → Login Form → API Gateway → Auth Service
                                       ↓
                                  Validate Credentials
                                       ↓
                                  Generate JWT Tokens
                                       ↓
                                  Create Session (Redis)
                                       ↓
                                  Return Tokens
```

Security Controls: - Password hashing: bcrypt (cost factor 12) - Password policy enforcement: - Minimum 8 characters - At least one uppercase, lowercase, number, special character - Password strength meter on registration - Prevent common passwords (check against breached password database) - Account lockout: 5 failed attempts → 30-minute lockout - Rate limiting: 10 login attempts per IP per minute

Secondary: Google OAuth 2.0

```
User → OAuth Button → Redirect to Google → Authorization
                                                ↓
                                            Callback
                                                ↓
                                        Verify OAuth Token
                                                ↓
                                Create or Link User Account
                                                ↓
                                        Generate JWT Tokens
```

Security Controls: - PKCE (Proof Key for Code Exchange) flow - State parameter for CSRF protection - Validate OAuth token with Google - Secure account linking (require password if account exists)

2.2 Token Management

JWT Structure **Access Token** (Short-lived: 15 minutes):

```
{
  "header": {
    "alg": "RS256",
    "typ": "JWT",
    "kid": "key-id-1"
  },
```

```
  "payload": {
    "jti": "unique-token-id",
    "sub": "user-uuid",
    "email": "user@example.com",
    "role": "student",
    "org_id": "org-uuid",
    "iat": 1634567890,
    "exp": 1634568790,
    "iss": "stem-eval.example.com",
    "aud": "stem-eval-api"
  }
}
```

Refresh Token (Long-lived: 7 days):

```
{
  "header": {
    "alg": "RS256",
    "typ": "JWT",
    "kid": "key-id-1"
  },
  "payload": {
    "jti": "unique-refresh-id",
    "sub": "user-uuid",
    "type": "refresh",
    "iat": 1634567890,
    "exp": 1635172690,
    "iss": "stem-eval.example.com"
  }
}
```

Token Security

- **Algorithm**: RS256 (RSA with SHA-256)
- **Key Management**:
  - Private key stored in Google Secret Manager
  - Public key distributed to services for validation
  - Key rotation every 90 days
  - Multiple active keys (kid) for zero-downtime rotation
- **Token Storage**:
  - Access token: Memory only (never localStorage)
  - Refresh token: httpOnly, Secure, SameSite=Strict cookie
- **Token Validation**:
  - Signature verification
  - Expiration check
  - Issuer and audience validation
  - JTI check against blacklist (for logout)

Token Rotation

```
Client sends refresh token
    ↓
Validate refresh token
    ↓
Check session in Redis
    ↓
Generate NEW access token
    ↓
Generate NEW refresh token (rotate)
    ↓
Invalidate OLD refresh token
    ↓
Update session in Redis
    ↓
Return new tokens
```

Benefits: Limits damage from stolen refresh tokens

2.3 Session Management

Session Data (Stored in Redis)

```
{
  "user_id": "user-uuid",
  "refresh_token_jti": "refresh-jti",
  "ip_address": "192.168.1.1",
  "user_agent": "Mozilla/5.0...",
  "created_at": "2025-10-18T10:00:00Z",
  "last_activity": "2025-10-18T10:30:00Z",
  "expires_at": "2025-10-25T10:00:00Z"
}
```

Session Security: - TTL: 7 days (auto-expire in Redis) - Sliding expiration: Extend on activity - Session timeout: 2 hours of inactivity - Concurrent session limit: 3 devices per user - Session invalidation on: - Explicit logout - Password change - Role change - Account suspension - Session listing: Users can view and revoke active sessions

2.4 Multi-Factor Authentication (MFA)

Phase 1: Optional for coaches and admins **Phase 2**: Mandatory for admins, optional for all

Supported Methods

1. **TOTP (Time-based One-Time Password)**:
   - Google Authenticator, Authy

- 6-digit codes, 30-second window

2. **Email OTP** (Backup):
   - 6-digit code sent via email
   - 10-minute expiration

MFA Flow

```
User enters email/password
    ↓
Credentials valid?
    ↓
MFA enabled?
    ↓ Yes
Send OTP challenge
    ↓
User enters OTP
    ↓
Validate OTP
    ↓
Generate tokens
```

Security Controls: - Rate limiting: 5 OTP attempts per session - OTP single-use enforcement - Time-window tolerance: ± 1 period (30 seconds) - Backup codes: 10 single-use codes for account recovery

2.5 Password Management

Password Reset Flow

```
User requests reset
    ↓
Generate secure token (32 bytes, random)
    ↓
Store hashed token in DB (expires in 1 hour)
    ↓
Send reset link via email
    ↓
User clicks link
    ↓
Validate token (not expired, not used)
    ↓
User enters new password
    ↓
Hash new password (bcrypt)
    ↓
Update password
    ↓
```

```
Invalidate all sessions
    ↓
Mark token as used
```

Security Controls: - Token expiration: 1 hour - Single-use tokens - Invalidate all sessions on password change - Email confirmation of password change - Rate limiting: 3 reset requests per hour per email

3. Authorization and RBAC

3.1 Role Hierarchy

```
Admin
    Coach
        Parent
            Student
```

Inheritance: Higher roles inherit permissions of lower roles (optional, configurable)

3.2 Role Definitions

Student Role Description: K-12 students participating in VEX robotics

Permissions: - View own profile - Update own profile (limited fields) - View own submissions and evaluations - Create submissions - Take assigned assessments - View own assessment results - View own reports and progress - Create forum posts and comments - Vote on forum content - View class roster (if allowed by coach) - View own notifications

Restrictions: - Cannot access other students' submissions or results - Cannot access assessment questions before assigned - Cannot modify submissions after evaluation starts - Cannot delete forum posts after 24 hours

Parent Role Description: Parents or guardians monitoring student progress

Permissions: - View linked children's profiles - View linked children's submissions and evaluations - View linked children's assessment results - View linked children's reports and progress - View linked children's forum activity (read-only) - View linked children's class information - Receive notifications about children's activities - Manage parent-student links (request, approve)

Restrictions: - Cannot access unlinked students' data - Cannot submit work on behalf of students - Cannot modify student data - Cannot access assessment questions - Cannot post on forum

Coach Role Description: Instructors managing classes and evaluating students

Permissions: - All Student permissions - View and manage assigned classes - View all students in assigned classes - View submissions from class students - Review and override AI evaluations - Create and manage assessments - Create and manage question pools - Assign assessments to classes - View class analytics and reports - Generate class reports - Manage class rosters - Create teams - Moderate forum (assigned classes only) - Receive at-risk student alerts

Restrictions: - Cannot access students from other coaches' classes - Cannot modify system settings - Cannot manage users outside assigned classes - Cannot access financial or billing data

Admin Role **Description**: System administrators with full access

Permissions: - All permissions across the system - Manage organizations - Manage all users (create, update, delete, suspend) - Assign coaches to classes - View all submissions, assessments, and data - Configure system settings - Manage AI prompts and models - View system logs and audit trails - Moderate all forum content - Generate system-wide reports - Manage billing and subscriptions

Restrictions: - Cannot delete audit logs - Two-admin approval for sensitive operations (configurable)

3.3 Permission Matrix

Resource	Student	Parent	Coach	Admin
User Profile				
View own				
Update own	(limited)	(limited)		
View others	-	(children)	(class)	
Update others	-	-	-	
Delete users	-	-	-	
Submissions				
Create		-		
View own		(children)		
View class	-	-		
View all	-	-	-	
Delete own (1h)		-		
Delete others	-	-		
Evaluations				
View own		(children)		
View class	-	-		
Override AI	-	-		
Assessments				
Take assigned		-		
View results	(own)	(children)	(class)	

Resource	Student	Parent	Coach	Admin
Create/Edit	-	-		
Assign	-	-		
Delete	-	-		
Classes				
View own		-		
View all	-	-	-	
Create/Edit	-	-		
Manage roster	-	-		
Delete	-	-	-	
Reports				
View own		(children)		
View class	-	-		
View all	-	-	-	
Export	(own)	(children)	(class)	
Forum				
View posts		(read)		
Create posts		-		
Edit own (24h)		-		
Delete own (24h)		-		
Moderate	-	-	(class)	
System				
View settings	-	-	-	
Modify settings	-	-	-	
View logs	-	-	-	
Manage billing	-	-	-	

3.4 RBAC Implementation

Middleware Authorization Check

```
func CheckPermission(resource, action string) gin.HandlerFunc {
    return func(c *gin.Context) {
        // Extract user from JWT (injected by auth middleware)
        user := c.MustGet("user").(User)

        // Check if user has permission
        if !hasPermission(user, resource, action, c) {
            c.JSON(403, gin.H{
                "error": "Insufficient permissions"
            })
            c.Abort()
            return
        }
```

```
        c.Next()
    }
}

func hasPermission(user User, resource, action string, c *gin.Context) bool {
    // Admin has all permissions
    if user.Role == "admin" {
        return true
    }

    // Resource-specific checks
    switch resource {
    case "submission":
        return checkSubmissionPermission(user, action, c)
    case "assessment":
        return checkAssessmentPermission(user, action, c)
    // ... other resources
    }

    return false
}
```

Dynamic Ownership Check

```
func checkSubmissionPermission(user User, action string, c *gin.Context) bool {
    submissionID := c.Param("id")
    submission := getSubmission(submissionID)

    switch user.Role {
    case "student":
        // Students can only access their own submissions
        return submission.StudentID == user.ID

    case "parent":
        // Parents can access children's submissions
        return isLinkedStudent(user.ID, submission.StudentID)

    case "coach":
        // Coaches can access class students' submissions
        return isInCoachClass(user.ID, submission.ClassID)

    default:
        return false
    }
}
```

Attribute-Based Access Control (ABAC) For complex scenarios, combine RBAC with ABAC:

```
type AccessPolicy struct {
    Resource   string
    Action     string
    Conditions []Condition
}

type Condition struct {
    Field    string
    Operator string
    Value    interface{}
}

// Example: Coach can delete submission if student is in their class AND submission is < 7 d
policy := AccessPolicy{
    Resource: "submission",
    Action:   "delete",
    Conditions: []Condition{
        {Field: "class_id", Operator: "IN", Value: user.CoachClassIDs},
        {Field: "created_at", Operator: ">", Value: time.Now().Add(-7*24*time.Hour)},
    },
}
```

3.5 Special Permissions

Temporary Elevated Access For specific operations requiring higher privileges:

```
// Grant temporary admin access for data migration
token := grantTemporaryAccess(userID, "admin", 1*time.Hour, reason)

// Audit trail logged
// Auto-revoked after expiration
```

Delegation Coaches can delegate specific permissions:

```
// Coach delegates grading permission to assistant coach
delegate := Delegation{
    FromUser:   coachID,
    ToUser:     assistantID,
    Permission: "grade_submissions",
    ClassID:    classID,
    ExpiresAt:  time.Now().Add(30*24*time.Hour),
}
```

3.6 Parent-Student Linking Security

Link Request Flow

```
Parent sends link request
    ↓
Generate unique link token
    ↓
Send in-app notification to student
    ↓
Student reviews request
    ↓
Student approves/denies
    ↓
If approved: Create parent_student_link
    ↓
Send confirmation to both parties
```

Security Controls: - Student approval required (prevents unauthorized parent access) - Email verification for both accounts before linking - Rate limiting: 5 link requests per day per parent - Audit log of link requests and approvals - Students can unlink at any time - Parents notified of unlink - Coaches can review and override links (with reason)

4. API Security

4.1 API Gateway Security

Request Pipeline

```
Request → TLS Termination → Rate Limiting → Authentication → Authorization → Service Routing
```

Rate Limiting Per-IP Rate Limits (Unauthenticated): - Global: 1000 requests/minute - Login endpoint: 10 requests/minute - Registration: 5 requests/minute

Per-User Rate Limits (Authenticated): - General API: 100 requests/minute - File uploads: 10 requests/minute - AI evaluations: 20 submissions/hour - Forum posts: 30 posts/hour - Forum votes: 100 votes/hour

Implementation: - Token bucket algorithm - Storage: Redis with sliding window - Response headers: X-RateLimit-Limit, X-RateLimit-Remaining, X-RateLimit-Reset - HTTP 429 on limit exceeded with Retry-After header

Request Validation Input Validation: - Schema validation (JSON Schema) - Data type validation - Length limits - Whitelist allowed characters - Sanitize HTML input

Example:

```
type CreateSubmissionRequest struct {
    Title       string `json:"title" binding:"required,min=3,max=255"`
    Description string `json:"description" binding:"max=5000"`
    Type        string `json:"type" binding:"required,oneof=robot_image code notebook mixed"
}
```

4.2 API Endpoint Security

Sensitive Endpoints Extra Protection for: - Password reset - Role changes - User deletion - Assessment answer access - Payment processing (future)

Controls: - Additional authentication step (re-enter password or MFA) - Admin approval required (for critical operations) - Delayed execution (30-minute delay for account deletion) - Email confirmation required - Enhanced audit logging

Idempotency For critical operations (submissions, payments):

```
POST /api/v1/submissions
Idempotency-Key: unique-key-per-request

Response: 200 OK or 201 Created
```

Benefits: Prevents duplicate submissions on retry

4.3 CORS Configuration

```
cors.Config{
    AllowOrigins:     []string{"https://app.stem-eval.example.com"},
    AllowMethods:     []string{"GET", "POST", "PUT", "PATCH", "DELETE"},
    AllowHeaders:     []string{"Origin", "Content-Type", "Authorization"},
    ExposeHeaders:    []string{"X-RateLimit-Limit", "X-RateLimit-Remaining"},
    AllowCredentials: true,
    MaxAge:           12 * time.Hour,
}
```

Security: - Whitelist specific origins (no wildcards in production) - Only necessary methods and headers - Credentials allowed only for same-site requests

4.4 API Versioning

Strategy: URL path versioning (`/api/v1/`, `/api/v2/`)

Deprecation Policy: - Announce deprecation 6 months in advance - Support old version for 1 year after new version release - Return deprecation headers: `Deprecation: true  Sunset: Sat, 1 Oct 2026 23:59:59 GMT  Link: </api/v2/docs>; rel="deprecation"`

5. Data Security

5.1 Encryption

At Rest

- **Database**: PostgreSQL with TDE (Transparent Data Encryption)
- **File Storage**: Google Cloud Storage with default encryption
- **Sensitive Fields**: Additional application-level encryption (AES-256-GCM)
  - User emails
  - Phone numbers
  - Date of birth
  - IP addresses in logs

Encryption Implementation:

```
// Encrypt before storing
encryptedEmail := encrypt(email, encryptionKey)
user.Email = encryptedEmail

// Decrypt after retrieval
email := decrypt(user.Email, encryptionKey)
```

Key Management: - Keys stored in Google Secret Manager - Automatic key rotation every 90 days - Envelope encryption for large data - Separate keys per environment (dev, staging, prod)

In Transit

- **External**: TLS 1.3 only
- **Internal**: mTLS for service-to-service communication
- **Certificate Management**: Let's Encrypt with auto-renewal

5.2 Data Privacy

PII Handling Collected PII: - Name, email, date of birth - IP addresses, user agents - Educational data (submissions, assessments)

Privacy Controls: - Minimize PII collection - Obtain parental consent for users under 13 (COPPA) - Right to access data (GDPR Article 15) - Right to erasure (GDPR Article 17) - Right to data portability (GDPR Article 20) - Privacy policy and terms of service

Data Anonymization Benchmarking Data: - Remove all identifiers (name, email, user ID) - Aggregate to prevent re-identification - No groups smaller than 5 students

Analytics: - Use hashed user IDs - Aggregate metrics only - No individual-level data in analytics tools

Data Retention Policy: - Active accounts: Retain indefinitely - Inactive accounts (1 year): Archive - Deleted accounts: Retain 30 days, then purge - Audit logs: Retain 1 year - Activity logs: Retain 90 days

Deletion Process:

```
User requests deletion
    ↓
Soft delete (30-day grace period)
    ↓
Send confirmation email
    ↓
After 30 days: Hard delete
    ↓
Remove all PII
    ↓
Anonymize or delete associated data
    ↓
Retain audit log of deletion
```

5.3 Secure File Upload

Upload Security Controls

- **File Type Validation**: Whitelist (JPG, PNG, PDF, DOCX, PY, CPP)
- **File Size Limits**: 10MB per file, 100MB total per submission
- **Content Verification**: Magic number check (not just extension)
- **Virus Scanning**: ClamAV or Cloud Security Scanner
- **Rename Files**: Store with UUID, not original name
- **Signed URLs**: Time-limited access (1 hour)

Upload Flow

```
Client → Pre-signed URL request → Server validates → Generate signed URL
                                                                ↓
Client uploads to GCS ← Signed URL ← Server returns
                ↓
Server → Virus scan → Process file → Update database
```

Security: - Direct upload to GCS (reduces server load) - Signed URLs prevent unauthorized uploads - No user-supplied filenames in storage - Virus scan before processing

6. Application Security

6.1 Web Application Firewall (WAF)

Use: Google Cloud Armor

Rules: - Block common attack patterns (SQL injection, XSS) - Rate limiting at edge - Geo-blocking (if needed) - Custom rules for application-specific threats

6.2 Security Headers

```
Strict-Transport-Security: max-age=31536000; includeSubDomains; preload
Content-Security-Policy: default-src 'self'; script-src 'self' 'unsafe-inline' 'unsafe-eval'
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
X-XSS-Protection: 1; mode=block
Referrer-Policy: strict-origin-when-cross-origin
Permissions-Policy: geolocation=(), microphone=(), camera=()
```

6.3 Secure Coding Practices

SQL Injection Prevention

- Parameterized queries (prepared statements)
- ORM usage (GORM for Go)
- Input validation and sanitization

```
// Good: Parameterized query
db.Where("email = ?", email).First(&user)

// Bad: String concatenation (NEVER DO THIS)
db.Raw("SELECT * FROM users WHERE email = '" + email + "'")
```

XSS Prevention

- React's automatic escaping
- Sanitize user-generated content before display
- CSP headers
- No `dangerouslySetInnerHTML` without sanitization

CSRF Prevention

- SameSite cookies
- Double-submit cookie pattern
- CSRF tokens for state-changing operations

6.4 Dependency Management

Practices: - Regular dependency updates - Vulnerability scanning (Snyk, Dependabot) - Pin dependency versions - Review new dependencies before adoption - Use lock files (package-lock.json, go.sum)

Automated Scanning:

```yaml
# GitHub Actions
- name: Run Snyk Security Scan
  uses: snyk/actions/node@master
  with:
    args: --severity-threshold=high
```

7. Infrastructure Security

7.1 Network Security

VPC Configuration: - Private subnets for databases and internal services - Public subnets only for load balancers - Network ACLs and security groups - No direct internet access for backend services

Firewall Rules: - Default deny all - Whitelist specific ports (443, 5432 internal) - Internal service communication only

7.2 Secrets Management

Google Secret Manager: - Database passwords - API keys (Vertex AI, SendGrid) - Encryption keys - OAuth client secrets - JWT signing keys

Best Practices: - Never commit secrets to Git - Rotate secrets quarterly - Use IAM for secret access control - Audit secret access - Environment-specific secrets

7.3 Database Security

Access Control: - Dedicated service accounts per service - Least privilege database permissions - No direct internet access - Cloud SQL Auth Proxy for connections

Backup and Recovery: - Automated daily backups - Point-in-time recovery enabled - Encrypted backups - Backup retention: 30 days - Tested restore procedures

8. Monitoring and Incident Response

8.1 Security Monitoring

Metrics to Monitor: - Failed login attempts (threshold: $>$ 10/minute) - Unusual access patterns (e.g., student accessing coach endpoints) - Rate limit violations - Database query anomalies - File upload spikes - API error rate increases - Certificate expiration

Logging: - Centralized logging (Cloud Logging) - Structured JSON logs - Log all security events: - Authentication attempts - Authorization failures - Data access (PII) - Configuration changes - User role changes

Alerting: - PagerDuty or Cloud Alerting - Alert on: - Multiple failed logins from same IP - Admin account creation - Bulk data export - Database anomalies - System errors > 5% rate

8.2 Incident Response Plan

Phases 1. Preparation: - Define incident response team - Document procedures - Regular security training - Incident response drills

2. Detection and Analysis: - Monitor alerts - Investigate suspicious activity - Determine incident severity - Classify incident type

3. Containment: - Short-term: Isolate affected systems - Long-term: Apply patches, change credentials - Preserve evidence

4. Eradication: - Remove threat - Patch vulnerabilities - Strengthen defenses

5. Recovery: - Restore systems - Verify security - Monitor for reinfection

6. Post-Incident: - Incident report - Lessons learned - Update procedures - Notify affected users (if required)

Breach Notification Requirements: - FERPA: Notify affected parties "without unreasonable delay" - GDPR: Notify within 72 hours - COPPA: Notify FTC and parents

Notification Process: 1. Assess scope and severity 2. Consult legal counsel 3. Draft notification 4. Notify affected users 5. Provide remediation steps 6. Offer support (credit monitoring if needed)

9. Compliance

9.1 FERPA (Family Educational Rights and Privacy Act)

Requirements: - Protect student educational records - Obtain consent before disclosure - Allow parents/students to access and amend records - Annual notification of rights

Implementation: - Encrypt educational data - RBAC enforces "need to know" - Audit trail of data access - Parent/student data access portal - Consent management for data sharing

9.2 COPPA (Children's Online Privacy Protection Act)

Requirements (for users under 13): - Obtain verifiable parental consent before collecting PII - Notify parents about data collection practices - Allow parents to review, delete child's data - Maintain reasonable security measures

Implementation: - Age verification on registration - Parental consent workflow - Parent data access and deletion portal - Limited data collection for children - No targeted ads for children

9.3 GDPR (General Data Protection Regulation)

Requirements (if serving EU users): - Lawful basis for processing - Data protection by design - User consent management - Right to access, erasure, portability - Data breach notification (72 hours) - Appoint Data Protection Officer (if needed)

Implementation: - Privacy policy and consent management - Data subject access request (DSAR) workflow - Data portability export (JSON format) - Right to erasure implementation - Data processing agreements with vendors

10. Security Testing

10.1 Automated Testing

Tools: - **SAST** (Static Application Security Testing): SonarQube, Semgrep - **DAST** (Dynamic Application Security Testing): OWASP ZAP - **Dependency Scanning**: Snyk, Dependabot - **Container Scanning**: Trivy

CI/CD Integration:

```
- name: Run SAST
  run: semgrep --config=auto

- name: Run Dependency Check
  run: snyk test

- name: Scan Docker Image
  run: trivy image myapp:latest
```

10.2 Penetration Testing

Frequency: Annually or after major changes

Scope: - Web application - API endpoints - Authentication and authorization - Infrastructure (if in-scope)

Process: 1. Define scope and rules of engagement 2. Hire external security firm 3. Testing period (1-2 weeks) 4. Receive report with findings 5. Remediate vulnerabilities 6. Re-test critical findings

10.3 Security Audits

Internal Audits: Quarterly - Code review for security issues - Configuration review - Access control review - Secrets rotation check

External Audits: Annually - SOC 2 Type II audit (for compliance) - Third-party security assessment

11. Security Checklist (Pre-Launch)

Authentication & Authorization: - [ ] Password hashing with bcrypt (cost 12) - [ ] JWT tokens with RS256 - [ ] Token expiration and refresh - [ ] Session management in Redis - [ ] MFA option available - [ ] RBAC implemented and tested - [ ] Parent-student linking requires approval

API Security: - [ ] Rate limiting configured - [ ] Input validation on all endpoints - [ ] CORS properly configured - [ ] Security headers set - [ ] API versioning in place

Data Security: - [ ] TLS 1.3 enforced - [ ] Database encryption at rest - [ ] PII fields encrypted - [ ] Secrets in Secret Manager - [ ] Secure file upload with virus scanning

Application Security: - [ ] No SQL injection vulnerabilities - [ ] XSS prevention measures - [ ] CSRF protection enabled - [ ] Dependencies up to date - [ ] CSP headers configured

Infrastructure: - [ ] VPC and firewall rules - [ ] Private subnets for databases - [ ] Backups configured and tested - [ ] Monitoring and alerting set up - [ ] Incident response plan documented

Compliance: - [ ] Privacy policy published - [ ] Terms of service published - [ ] FERPA compliance verified - [ ] COPPA compliance (if applicable) - [ ] Parental consent workflow

Testing: - [ ] Security testing in CI/CD - [ ] Penetration test completed - [ ] Vulnerability remediation - [ ] Security training for team

---

Document Version: 1.0 **Last Updated**: 2025-10-18 **Next Security Review**: 2026-01-18