ClassQuest System Design

1. System Overview

ClassQuest is a web-based classroom gamification platform that turns regular school activities into an RPG-style experience.

Teachers can create classes and assign quests, while students complete activities to earn XP, gold, and rewards for their characters.

At the current stage of development:

- Teacher signup and authentication are working.
- Teacher accounts are being created in the backend.
- Confirmation emails are being sent through Cognito.
- We are currently working on:
    - Connecting classes to the backend.
    - Implementing quest delivery and completion for students.

The system supports teachers managing multiple classes, each with a unique class code that students use to join.

2. Current Project State

The backend infrastructure is now operational, and authentication is functional.

Completed

- Teacher signup/login works.
- Cognito handles authentication and confirmation emails.
- Teacher profiles are being stored.
- Frontend dashboard and navigation pages exist.
- Teachers can access dashboard features after login.

In Progress

- Backend integration for classes.
- Backend integration for quests and student progress.
- Linking student accounts to classes.
- Quest assignment and completion workflows.

Planned

- Boss fight system integration.
- Rewards and shop system backend connection.
- Class analytics and teacher insights.

3. Architecture Overview

The system uses a modern web architecture with a React frontend and a serverless AWS backend.

Frontend

- Built with React + TypeScript using Vite.
- Handles UI rendering and user interaction.
- Calls backend APIs for data operations.
- Uses Cognito authentication tokens for secure requests.

Main teacher pages:

- Dashboard
- Classes
- Quests/Subjects
- Activity
- Profile

Main student pages:

- Character page
- Quests
- Rewards/shop
- Boss fights
- Guild/leaderboards

Backend (AWS)

The backend is serverless and built using AWS managed services:

- Cognito handles user authentication.
- API Gateway routes frontend requests.
- Lambda functions handle business logic.
- DynamoDB stores application data.
- CloudWatch handles logs and monitoring.

This architecture reduces server maintenance and scales automatically with usage.

4. Core System Concepts

The system revolves around several main entities:

Teacher

Creates classes and assigns quests.

Student

Joins classes and completes quests.

Class

Owned by a teacher. Has a unique join code.

Class Membership

Represents students belonging to classes.

Quest

Tasks or assignments students complete.

Quest Assignment

Connects quests to classes.

Progress

Tracks student completion.

Boss Fight

Optional group activity mode.

5. Database Design (DynamoDB)

Multiple tables are used to keep the design simple and maintainable.

TeacherProfiles

Stores teacher information.

Fields:

- teacherId (PK)
- displayName
- email
- createdAt

StudentProfiles

Stores student information and stats.

Fields:

- studentId (PK)
- displayName
- xp
- level
- gold
- createdAt

Classes

Represents teacher-created classes.

Fields:

- classId (PK)
- teacherId
- className
- classCode
- createdAt
- status

Indexes:

- classCode index (used when students join)
- teacherId index (list teacher classes)

ClassMemberships

Connects students to classes.

Fields:

- classId (PK)
- studentId (SK)
- joinedAt

Optional index allows listing classes for a student.

Quests
Stores teacher-created quests.
Fields:

- questId (PK)
- teacherId
- title
- subject
- difficulty
- reward
- createdAt
- status

ClassQuestAssignments
Links quests to classes.
Fields:

- classId (PK)
- questId (SK)
- publishedAt

QuestProgress
Tracks student progress.
Fields:

- classId (PK)
- studentId#questId (SK)
- status
- xpAwarded
- goldAwarded
- completedAt

6. API Structure
All endpoints require authenticated users via Cognito.
Class APIs

- POST /classes to teacher creates class
- GET /teachers/{teacherId}/classes to list teacher classes
- POST /classes/join to student joins using class code
- PATCH /classes/{classId} to rename/archive

Quest APIs

- POST /quests to create quest
- POST /classes/{classId}/quests/{questId}/publish
- GET /classes/{classId}/quests
- POST /classes/{classId}/quests/{questId}/complete

Profile APIs

- POST /teacher-profiles
- POST /student-profiles
- GET profile endpoints

7. Main System Flows

Teacher creates class

1. Teacher enters class name.
2. Backend generates unique class code.
3. Class is stored.
4. Code displayed for teacher.

Student joins class

1. Student enters class code.
2. Backend finds matching class.
3. Membership entry created.
4. Student now sees class quests.

Teacher publishes quest

1. Teacher creates quest.
2. Teacher assigns quest to class.
3. Students receive quest.

Student completes quest

1. Completion submitted.
2. Progress stored.
3. XP/gold updated.

8. Authorization Model

Cognito manages authentication.
Backend checks roles:
Teachers:

- Can manage their classes and quests.

Students:

- Can access only classes they belong to.

This prevents cross-class data access.

9. Scalability Considerations

- DynamoDB scales automatically.
- Class code index ensures fast join operations.
- Serverless Lambda functions scale per request.
- Boss fights can use event-style logging for performance.

10. Monitoring and Reliability
CloudWatch logs and metrics are used to monitor:

- API failures
- Lambda errors
- Signup failures
- Quest completion metrics

Alarms can be added later.

11. Development Strategy Going Forward
The current focus is backend integration:
Next steps:

1. Connect class creation fully to backend.
2. Implement student class joining.
3. Connect quests to backend.
4. Implement progress tracking.
5. Add boss fight backend logic.

Frontend UI is already largely built, so future work focuses on backend connections rather than new UI.