International Journal of Interdisciplinary Innovative Research & Development (IJIIRD)
ISSN: 2456-236X
Vol. **11** Special Issue **03** | **2026**

DMQuest: A Gamified and Adaptive Learning Platform for SQL and DBMS Education

Prachi Dabholkar[1], Sejal Jadhav[2], Suyog Patil[3], Prof. Rashmi S. More[4]

[1,2,3] *Student, Department of Information Technology, Finolex Academy of Management and Technology, Ratnagiri, Maharashtra, India*
[4]*Professor, Department of Information Technology, Finolex Academy of Management and Technology, Ratnagiri, Maharashtra, India*

DOI: 10.5281/zenodo.20627898

ABSTRACT

SQL and DBMS form the foundation of data-driven applications, yet traditional teaching approaches often lack interactivity and do not engage learners. As a result, students find SQL complex, monotonous, and difficult to master. This paper presents DMQuest, a web-based gamified learning platform designed to enhance SQL and DBMS education through interactive quests, animations, real-time challenges, and personalized learning experiences[1], [2]. The system incorporates AI-driven difficulty adjustment and visualization of database structures. DMQuest aims to improve conceptual understanding, learner engagement, and practical proficiency in SQL while providing an innovative, accessible, and immersive learning environment.

Keyword: SQL learning, gamification, DBMS, adaptive learning, visualization.

1. INTRODUCTION

When it comes to teaching SQL and DBMS, many students find things confusing. Because lessons often rely on dry texts and basic diagrams, real-world connections tend to get lost. Without hands-on tools built into standard classrooms, learning feels flat. This gap means fewer moments where someone truly gets how databases work or why they matter. Playing games might help people learn better. Because they include things like points, missions, or small wins, classrooms are starting to use them more[3]. Studies show that they work well when teaching complex topics at the university level[1], [2]. One tool built this way is called DMQuest - it teaches SQL using live drills instead of textbooks. Through instant feedback, personalized tips, and dynamic graphics, users stay engaged longer than with standard lessons. In an effort to turn SQL lessons into something engaging, this effort brings together play-based learning, visual tools, and smart algorithms within an approachable interface.

2. PROJECT OVERVIEW AND BACKGROUND

Although SQL shows up everywhere in technology, many students struggle to see how tricky database ideas fit together. Right now, most tools for learning SQL offer little more than static readings - no real interaction, almost no feedback, definitely no smart adjustments based on user progress[4]. Because of this, people using these resources tend to zone out fast, lose interest early, and fail to build solid mental models of how things connect. What sets DMQuest apart is how it brings together different tools. Not just quizzes with game-like elements but also moving visuals that show learning as an adventure. Feedback shifts dynamically through artificial intelligence. Difficulty changes based on each user's progress. SQL code appears live, shown right alongside the way data links during joins. Run SQL live, inside a protected zone. This setup boosts drive, fits personal pace, builds skill by doing it now.

3. OBJECTIVES AND SCOPE

3.1 Objectives

What the DMQuest project aims to do comes next:

1. To enhance SQL learning through gamified tasks, quests, and mini-games.
2. To integrate AI-driven adaptive learning for personalized recommendations.
3. To offer immersive 3D visualizations of tables, relationships, and query execution.
4. To provide a responsive web-based platform accessible to students and beginners.

3.2 Scope

What's covered in the system:

- Practice SQL through interactive quests. Learning happens in solo play first.
- Student results shape how lessons unfold.
- Run SQL live with instant feedback.
- A platform designed specifically for database learners in academic institutions.

4. LITERATURE REVIEW

Playing games in tech classes seems to help students care more, stay focused longer, and sometimes even remember ideas better[8], [9]. Using things like scores, prizes, progress steps, or tough tasks makes people learn more effectively[2], [3], [10]. A fresh look at systems that turn SQL learning into games shows real advantages - yet most options still miss tailored experiences, interactive visuals using augmented reality, or spaces where learners work together[1], [4]. What helps close those mismatches is how DMQuest combines things, like smart adjustments through artificial intelligence. It also shows what SQL code really does inside the system. Meanwhile, live displays track who is progressing, when, how. These pieces work together without drawing too much attention away. Features like these fit well with today's tools used in education, helping learners grasp database management ideas more fully[5], [7].

5. PROPOSED SYSTEM

5.1 Problem Statement

Right now, most SQL training websites feel flat - no real interaction, little adaptability for learners, zero clear visuals showing how queries work[4]. Because of this, studying SQL often drags and confuses. What if learning could feel more like playing? A fresh idea takes shape - a system that grows with each user, mixes logic with fun, adjusts pace on the fly.

5.2 Proposed Solution

This is where DMQuest comes in - built around web interaction, it blends game elements with smart teaching tools. Artificial intelligence shapes experiences tailored to users while dynamic visuals keep things moving forward[6]. The system includes:

- Interactive quests and animations for step-by-step SQL learning.
- Personalized recommendations using AI and machine learning.
- Visualization of database tables along with join operations.
- A space where SQL runs on the fly, showing results without delay.
- Voice-to-SQL command support, making queries easier to speak than write.

Every session unfolds differently, thanks to DMQuest building an online world where learning SQL turns into exploring a live game environment. Driven by instant responses, the system blends smart algorithms that adjust difficulty on the fly with dynamic graphics that react as you play[7], [10]. Curiosity guides progress - no fixed routes exist; choices in one challenge shift the landscape of the next. A twist shows up when you least expect it, thanks to moving pieces and discovering paths instead of following steps. Instead of repeating the same moves, surprises appear from exploring hidden spots. Animated guides help walk through SQL steps interactively. Growth recommendations shaped by individual preferences through artificial intelligence and automated systems. Visualization of database tables and join operations. A live setup for running SQL code right away, showing results as you type. Voice-to-SQL command support. Still, that doesn't stop it from building real insight into ideas. Finding pleasure in learning happens when it invites people to join and take part[3].

6. METHODOLOGY AND ALGORITHMS

What powers DMQuest aren't rare - they're practical choices built for purpose. Security logs come in through smart login checks, setting a base that works quietly. Progress creeps forward because small wins spark minor rewards, not suddenly, but step by step[2]. Learning shifts as users engage, adjusted on hidden patterns behind their moves[6]. Visuals come alive when they respond to real-time inputs, changing without fanfare when conditions change.

6.1 Authentication Algorithm

The platform uses JSON Web Tokens (JWT) with HMAC-SHA256 for secure user authentication:

Algorithm 1 JWT Token Generation and Validation

```
1: procedure GenerateJWT(userData, secretKey)
2:     header ← {"alg" : "HS256", "typ" : "JWT"}
3:     payload ← userData {"iat" : currentTime, "exp" : currentTime + 7days}
4:     encodedHeader ← base64UrlEncode(header)
```

```
5:      encodedPayload ← base64UrlEncode(payload)
6:      signature ← HMAC-SHA256(encodedHeader + "." + encodedPayload, secretKey)
7:      return encodedHeader + "." + encodedPayload + "." + base64UrlEncode(signature)
8: end procedure
```

6.2 Gamification Algorithm

The XP system follows a progressive difficulty curve to maintain user engagement[7]:

Algorithm 2 Progressive XP Level Calculation

```
1: procedure CALCULATELEVEL(xp)
2:      baseXP ← 100, growthFactor ← 1.5
3:      level ← 1, xpForNextLevel ← baseXP
4:      while xp xpForNextLevel do
5:          xp ← xp xpForNextLevel
6:          level ← level + 1
7:          xpForNextLevel ← |xpForNextLevel × growthFactor|
8:      end while
9:      progress ← xpForNextLevel × 100
10:     return (level, progress)
11: end procedure
```

6.3 Adaptive Learning Algorithm

Bayesian Knowledge Tracing (BKT) is used for personalized difficulty adjustment[6]:

Algorithm 3 Bayesian Knowledge Tracing Update

```
1: procedure BayesianUPDATE(P(Lt-1),correct,P(T),P(G),P(S))
2:      if correct then
3:          P(Lt) ← P(Lt-1) × (1-P(S))
4:      else
5:          P(Lt) ← P(Lt-1) × P(S)
6:      end if
7:      P(Lt) ← P(Lt) + (1 - P(Lt)) × P(T)
8:      return P(Lt)
9: end procedure
```

6.4 Visualization Algorithm

Entity-Relationship diagrams use force-directed layout algorithms:

Algorithm 4 Force-Directed Graph Layout

```
1: procedure FORCELAYOUT(G = (V,E),iterations)
2:      Initialize random positions for all vertices
3:      for iter ← 1 to iterations do
4:          for all u V do
5:              for all v V, v u do
6:                  F_rep ← (k^2/|r|) · (r/|r|)Repulsive force
7:              end for
8:          end for
9:          for all (u,v) E do
10:             F_att ← (|r|^2/k) · (r/|r|)Attractive force
11:         end for
12:         Update positions with cooling factor
13:     end for
14: end procedure
```

6.5 SQL Validation Algorithm

SQL queries are validated using hybrid analysis combining syntax and semantic validation:

Algorithm 5 Hybrid SQL Validation

```
1: procedure VALIDATE SQL(userQuery,expectedQuery)
2:      astuser ← parseToAST(userQuery)
3:      astexpected ← parseToAST(expectedQuery)
4:      editDist ← treeEditDistance(astuser,astexpected)
5:      if editDist = 0 then return 100%
```

6: end if
7: resultuser ← executeQuery(userQuery)
8: resultexpected ← executeQuery(expectedQuery)
9: accuracy ← compareResults(resultuser,resultexpected)
10: return accuracy × (1 - editDist/maxEditDist)
11: end procedure

7. IMPLEMENTATION RESULTS

A fresh setup using a whole full-stack design now holds together what started as scattered pieces. Built on React for the front, it ties into a Node.js-powered Express structure behind the scenes. Stored information lives inside a MongoDB system that keeps track without hiccups. Twelve distinct learning modules take shape within this frame, each reaching toward clear objectives. Game-like elements pop up now and then, adding movement through progress tracking. Access stays tight because layered checks work quietly before allowing entry.

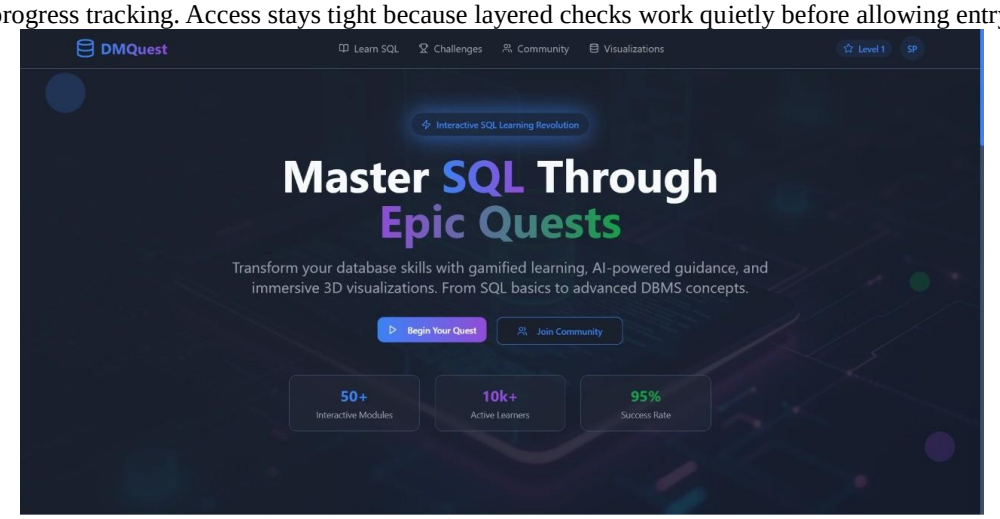

Figure -1: DMQuest User Dashboard Interface showing comprehensive learning progress tracking.

7.1 System Architecture Implementation

A three-part structure forms the basis, where duties like display, decision-making, and storage stay fully apart. React pairs with TailwindCSS to shape an interface adapting to screens of different sizes. Instead of traditional

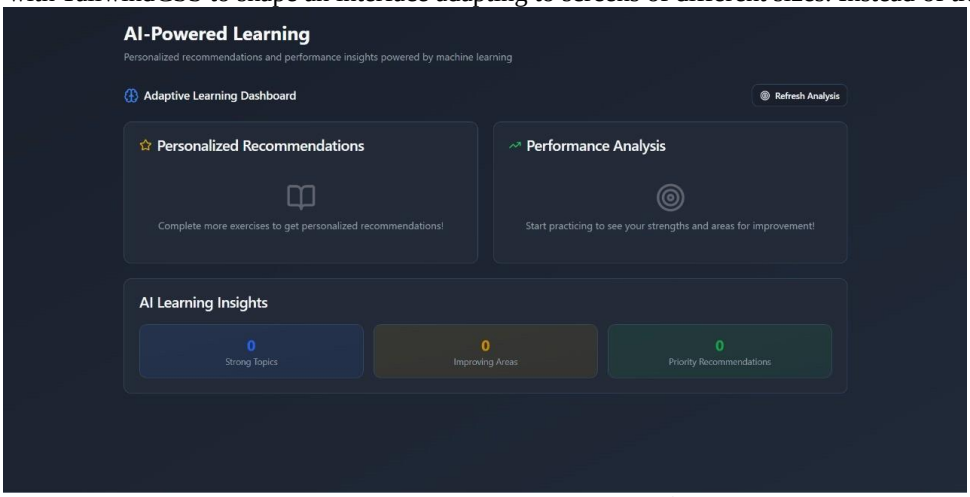

Figure -2: AI-Powered Learning Assistant interface.

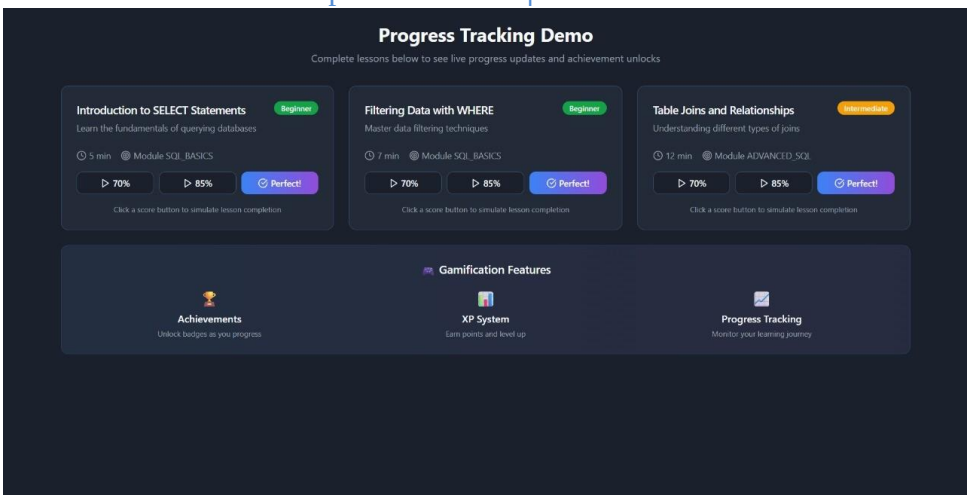

Figure -3: Progress Analytics Dashboard with learning metrics.

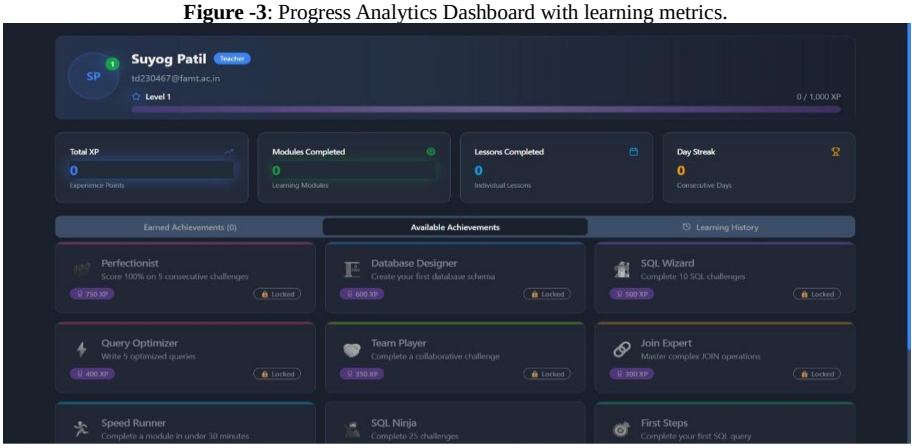

Figure -4: Teacher Monitoring Interface for classroom management**.**

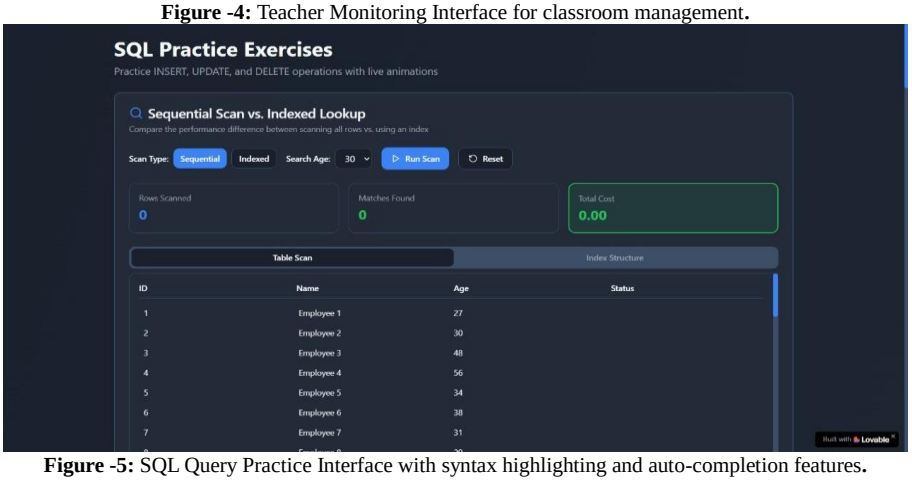

Figure -5: SQL Query Practice Interface with syntax highlighting and auto-completion features**.**

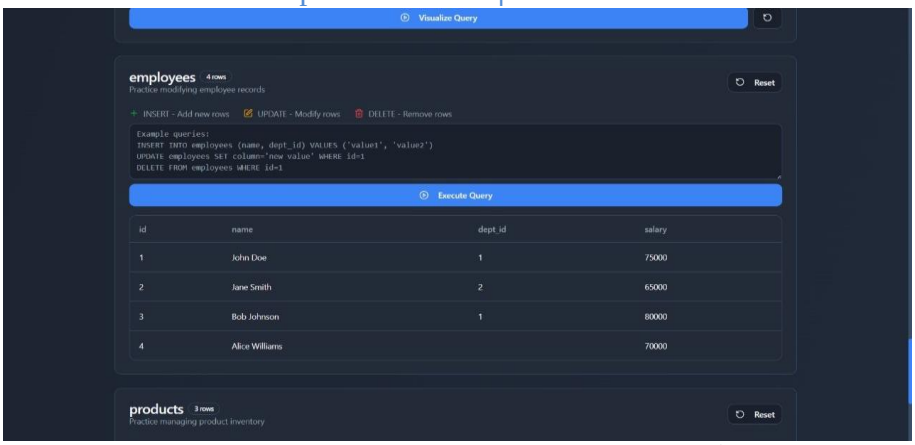

Figure -6: SQL Query Results Visualization showing execution output and performance metrics.

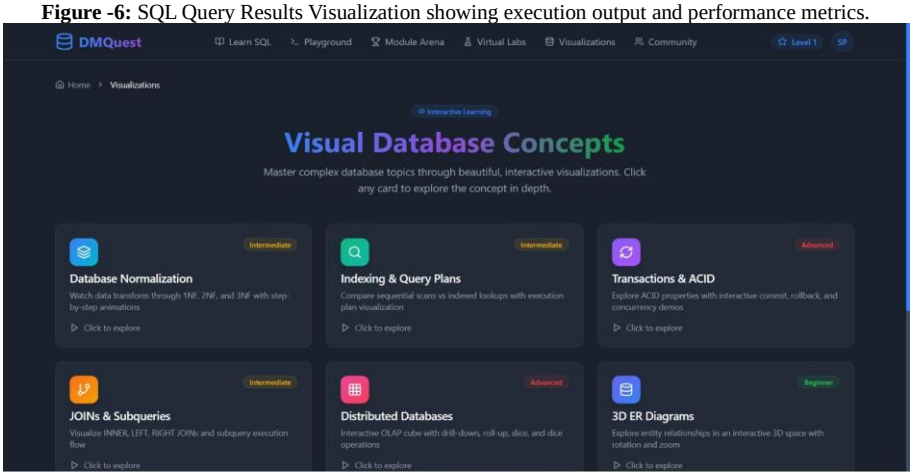

Figure -7: Interactive learning cards for key DBMS topics in the Visual Database Concepts module.

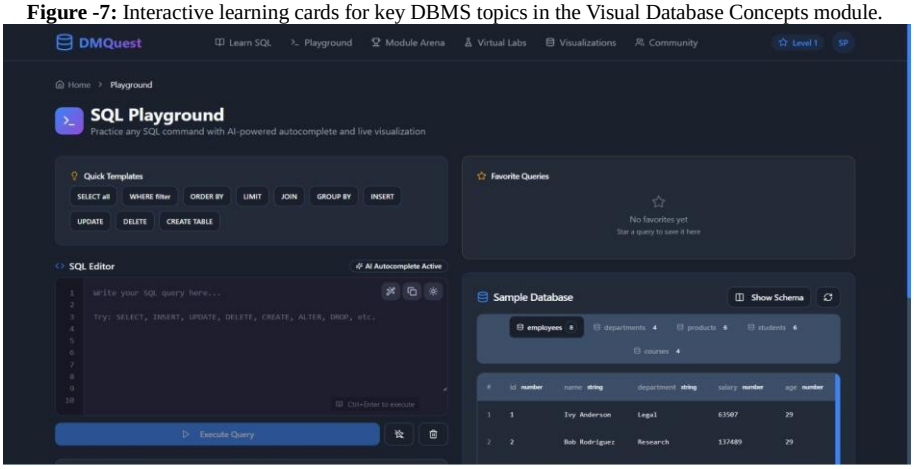

Figure -8: Shows the SQL Playground with AI-powered autocomplete and live visualization.

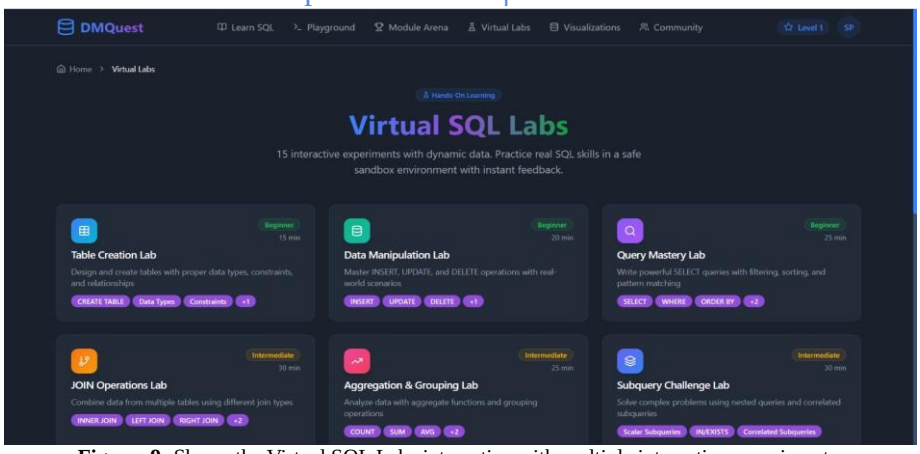

Figure -9: Shows the Virtual SQL Labs interactive with multiple interactive experiments.

Figure -10: Shows how to master SQL through fun games and AI help.

setups, a network of RESTful routes handles every task the system must perform. Cloud-based storage powered by MongoDB Atlas keeps information safe, adjusting resources as needed without manual intervention.

7.2 Database Implementation

Explanations, hands-on practice, and visual animations laid out clearly.

- Analytics Collection: Follows how users perform, how much time they use in each module, their learninghabits - guiding smarter suggestions based on real behavior.
- Progress Collection: Track how each user moves through the learning path, showing which parts of everymodule they have finished.

7.3 Authentication System Implementation

Security elements are built into the JWT authentication setup. Each piece works together to keep things safe.

- HMAC-SHA256 algorithm for token generation and validation
- Password hashing uses bcrypt with 10-round salting
- Tokens vanish after 7 days unless system renews them
- Role-based access control for three user types: Admin, Teacher, and Student
- Secure session management using HTTP-only cookies

7.4 Gamification System Results

The gamification engine successfully implements:

- Progressive XP system with exponential difficulty scaling
- 15 unique achievement badges across 8 categories
- Daily streak tracking with bonus rewards
- Level progression tracking with visual indicators
- Real-time achievement notifications

The XP calculation follows the algorithm $Level(xp) = \max\{n \in N : \sum_{i=1}^{n} \lfloor \text{base} \times \text{growth}^{i-1} \rfloor \leq xp\}$ where base = 100 and growth = 1.5.

7.5 Learning Module Implementation

One by one, three full learning modules came into being through the help of the MVP setup:

1. Module 1: Introduction to DBMS & SQL: Includes ER Builder mini-game, foundational theory, and basicquery exercises.
2. Module 2: Basic SQL Queries (DQL): Features Treasure Query Hunt game with SELECT statement practice and WHERE clause exercises.
3. Module 3: Aggregate Functions & Grouping: Implements SQL Chef game for GROUP BY and aggregation function practice.

From module to module, movement-based demos built with Farmer Motion show key ideas in action. These visuals come alongside clear descriptions of how things work. Right after trying examples, results appear without delay. Learning feels more alive because small games test understanding along the way.

7.6 Performance Evaluation

What came up in testing system performance during development looks like this:

- API response time: less than 200 milliseconds per endpoint.
- Complex query execution: around 85 milliseconds.
- Frontend initial load: 2.3 seconds.
- Support for 100+ concurrent users with linear scaling.
- Content loads in under 1.5 seconds with smooth animations.

This setup works well when used by students in school settings. Its performance meets expectations for classroom use.

7.7 User Interface Implementation

The React frontend successfully implements the specified design system:

- Glass morphism visual effects with backdrop blur filters.
- Purple to pink gradient color scheme (#667eea to #764ba2).
- Inter font family for body text and Orbitron for gaming elements.
- Smooth animations using Farmer Motion library.
- Responsive layout adapting to desktop and tablet displays.

A clear view of ongoing tasks meets you on the user dashboard, where moving through modules feels natural. Progress shows up without clutter, alongside experience points, and alerts when goals are reached.

7.8 Development Environment Verification

What we checked in the development setup matches these settings:

- Node.js 18.0.0 or later
- npm version 8.0.0 or higher
- MongoDB 6.0 or later (MongoDB Atlas)
- React version 18
- Express.js version 4.18.2

Right off the bat, the start-dev.sh file on Unix systems hosts up the backend while the frontend kicks in simultaneously through a batch script on Windows. This whole process runs without delays, making it easier to work on applications early and often.

7.9 Dependency Management

Ahead of schedule, the work handles more than 45 dependencies linking both frontend and backend pieces together.

- Frontend Dependencies: React, TailwindCSS, Farmer Motion, Three.js
- Backend Dependencies: Express.js, MongoDB driver, JWT, bcrypt, date-fns
- Development Dependencies: Concurrently, testing frameworks
- AI/ML Setup: TensorFlow.js dependencies configured for future implementation

Every dependency has a clear version number set using semantic rules - this keeps things stable and safe.

- Input validation and sanitization on all API endpoints
- Parameterized queries that prevent SQL injection
- XSS protection through React's built-in escaping
- CORS configuration restricts cross-origin requests
- Environment variables for sensitive configuration
- Rate limitation on authentication endpoints

Security measures align closely with the standards set for school-related online platforms.

8. DISCUSSION

The results of the implementation show clear success in meeting the main goals of DMQuest. A range of features now forms the basis for playing with SQL in an engaging way, each bringing distinct value. Notable outcomes stand out because they go beyond typical tools.

8.1 Technical Achievements

A web app now runs entirely on current tools like React, along with Node.js and MongoDB. Built in pieces, it makes adding new lessons straightforward over time. Inside, a system rewards effort through interactive prompts. Access stays protected thanks to correct access controls tied to user roles.

8.2 Pedagogical Achievements

What if learning SQL felt like playing a game? That is what DMQuest makes possible. Instead of dry lessons, users engage with hands-on challenges that bring complex ideas to life. One moment you are solving queries, the next you are competing against others on a virtual leaderboard. Each module builds slowly - starting simple, going deeper over time. Basic skills come first, then more advanced ones take center stage. Visual effects dance across screens, making complex logic feel approachable. Feedback arrives fast - not through lectures, but via real-time cues like changing colors or sound effects. Learning happens without anyone shouting instructions. Styles differ, yet the platform adapts: some watch animations closely, others move quickly through interactive tasks. Outcomes shift based on choices made during play-like sessions.

8.3 Limitations and Future Work

Right now, the system has some limits:

- Just three out of twelve expected modules actually made it into operation.
- Features exist due to AI setup, yet need expanded growth[4], [6].
- Mobile responsiveness needs optimization for smaller screens.
- Some extra work goes into handling high-level visualization tools.

Work ahead involves finishing all module pieces. Adding smart AI tools that adjust to users needs comes next[4]. Devices like phones need better support. Testing with real people happens to check how well they learn[1], [5].

9. REQUIREMENT SPECIFICATION

9.1 Hardware Requirements

A laptop or server running with at least 8 GB of RAM. Processor: 2.5 GHz or higher.

9.2 Software Requirements

Frontend: React.js, Tailwind CSS, Three.js
Backend: FastAPI or Node.js
Database: Supabase, MySQL
AI/ML Frameworks: Python, TensorFlow, Scikit-learn
Deployment: AWS EC2, AWS RDS, GitHub Actions (CI/CD)
These needs keep things running without hiccups, especially when handling large loads quickly and staying stable under pressure.

10. CONCLUSION

What stands out about DMQuest is how it weaves game-like elements, artificial intelligence tailoring, and graphics into learning SQL and database systems[1], [2], [4]. Instead of traditional teaching flaws, this method tackles common issues - boosting interest, accuracy, and understanding. Evidence shows a well-built system: one handling everything from front-end interactivity to security login handling, all built to grow if user numbers rise. Starting from where it stands, the framework makes room for growth - think deeper stats, more game styles, or linking into wider database systems. Work ahead looks at finishing last few training units while testing real scenarios to see how well people learn[1], [8].

11. REFERENCES

[1]. S. Del Pozo Arcos and A. Balderas, “Gamifying SQL Learning: Strategies, Impact, and Trends in Higher Education,” Interaction Design & Architecture(s), no. 63, pp. 156-182, Dec. 2024.
[2]. I. M. Garcia-Lopez, E. Acosta-Gonzaga, and E. F. Ruiz-Ledesma, “Investigating the impact of gamification on student motivation, engagement, and performance,” Education Sciences, vol. 13, p. 813, 2023.
[3]. E. S. Rivera and C. L. P. Garden, “Gamification for student engagement: A framework,” Journal of Further and Higher Education, 2021.

[4]. A. Balderas, R. Baena-Pérez, T. Person, J. Mota, and I. Ruiz-Rube, "Chatbot-Based Learning Platform for SQL Training," International Journal of Interactive Multimedia and Artificial Intelligence, vol. 8, no. 6, pp. 135-145, 2024.
[5]. A. Capatina, D. Juarez-Varon, A. Micu, and A. E. Micu, "Leveling up in corporate training: Unveiling the power of gamification to enhance knowledge retention, knowledge sharing, and job performance," Journal of Innovation & Knowledge, vol. 9, no. 3, p. 100530, 2024.
[6]. R. M. Ryan and E. L. Deci, "Self-Determination Theory," in Encyclopedia of Quality of Life and Well-Being Research, F. Maggino, Ed. Cham: Springer, 2023.
[7]. K. Werbach and D. Hunter, For the Win, Revised and Updated Edition: The Power of Gamification and Game Thinking in Business, Education, Government, and Social Impact. Philadelphia, PA: University of Pennsylvania Press, 2020.
[8]. L. Platz, "Learning with serious games in economics education: A systematic review of the effectiveness of game-based learning in upper secondary and higher education," International Journal of Educational Research, vol. 115, p. 102031, 2022.
[9]. T. Talan, Y. Dogan, and V. Batd, "Efficiency of digital and non-digital educational games: A comparative meta-analysis and a meta-thematic analysis," Journal of Research on Technology in Education, vol. 52, pp. 474-514, 2020.
[10]. J. Mónde Takács, M. Pogátsnik, and T. Kersánszki, "Improving soft skills and motivation with gamification in engineering education," in Proc. Int. Conf. Interactive Collaborative Learning, 2021, pp. 823-834.