Jaypee Institute of Information Technology, Noida

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING AND INFORMATION TECHNOLOGY

Project Title:

Python Game Hub: Web-Based Multi-Game Platform with User Authentication and Leaderboard System

Enrol. No.	Name of Student
9923103007	Bhavya Gupta
9923103009	Nishant Kumar
9923103025	Aman Luthra

Course Name: Open Source Software Lab

Program: B. Tech. CSE

3rd Year 5th Sem

2025 – 2026

Table of Contents

1. Abstract

The Python Game Hub is a comprehensive web-based gaming platform that integrates classic arcade and puzzle games with modern web technologies. The project demonstrates the implementation of a full-stack web application using Flask framework for backend operations, MongoDB for database management, and Pygame for game development. The platform features four distinct games: Neon Snake, Memory Match, Sudoku Pro, and Gem Crush Deluxe.

The system implements secure user authentication using JWT (JSON Web Tokens) and bcrypt password hashing, ensuring data security and user privacy. A centralized leaderboard system tracks and displays top scores across all games, fostering healthy competition among users. The platform's architecture follows the Model-View-Controller (MVC) pattern, promoting code maintainability and scalability.

Each game is developed as an independent Pygame application that communicates with the Flask backend via REST APIs, enabling seamless score submission and user verification. The frontend utilizes Tailwind CSS for responsive design, ensuring optimal user experience across different devices.

2. Scope of the Project

2.1 Current Implementation:

- User registration and authentication system with secure password storage
- Four fully functional games with unique gameplay mechanics
- Real-time score tracking and leaderboard system
- Responsive web interface for game launching and management
- JWT-based session management for secure game-server communication
- MongoDB integration for persistent data storage

2.2 Future Enhancements:

- Multiplayer gaming capabilities with WebSocket integration
- Achievement and badge system to increase user engagement
- Social features including friend system and challenges
- Game replay functionality and strategy analysis
- Mobile application development using React Native
- AI-powered difficulty adjustment based on player performance
- In-game chat system and community forums
- Tournament mode with scheduled competitions

2.3 Target Audience:

- Casual gamers seeking quick entertainment
- Students looking for brain-training puzzle games
- Competitive players interested in leaderboard rankings
- Educational institutions for cognitive skill development

3. Tools and Technologies Used

Backend Framework Flask 2.x (Python)
Database MongoDB Atlas
Game Engine Pygame-CE
Authentication Flask-JWT-Extended
Frontend HTML5, Jinja2, Tailwind CSS
Security Bcrypt, Werkzeug

3.1 Python Libraries:

- **Flask:** Lightweight WSGI web framework for building the backend server
- **PyMongo:** Python driver for MongoDB database operations
- **Flask-JWT-Extended:** JWT token generation and verification for authentication
- **Werkzeug:** Password hashing and security utilities
- **Pygame-CE:** Community Edition of Pygame for game development
- **Flask-CORS:** Cross-Origin Resource Sharing for API access
- **Python-dotenv:** Environment variable management
- **Requests:** HTTP library for API communication from games

3.2 Development Tools:

- Visual Studio Code - Primary IDE
- MongoDB Compass - Database management and visualization
- Postman - API testing and documentation
- Git & GitHub - Version control and collaboration

4. Design of the Project

4.1 System Architecture:

The project follows a three-tier architecture pattern:

- **Presentation Layer:** HTML templates with Tailwind CSS styling
- **Application Layer:** Flask server handling business logic and routing
- **Data Layer:** MongoDB for persistent storage of users and scores

4.2 Database Schema:

```
// Users Collection { "_id": ObjectId, "username": String (unique), "password": String (hashed) } // Scores Collection { "_id": ObjectId, "user_id": ObjectId (reference to Users), "game_id": String ("snake", "sudoku", "memory", "gem"), "score": Integer, "date": DateTime }
```

4.3 Authentication Flow:

1. User submits credentials via login form
2. Server validates credentials against MongoDB
3. JWT token generated and stored in HTTP-only cookies
4. Token passed to game processes for API authentication
5. Game submits scores with Bearer token in headers

4.4 Game Launch Mechanism:

1. User clicks "Play Now" button (requires authentication)
2. Flask route validates JWT and retrieves user data
3. New JWT token generated for game session
4. subprocess.Popen launches independent Pygame process
5. Token and username passed as command-line arguments
6. Game runs independently, communicates via REST API

4.5 User Interface Design:

- Responsive grid layout for game cards
- Sticky sidebar leaderboard showing top 15 scores
- Color-coded game categories (Arcade, Logic, Casual, Focus)
- Dark theme with gradient accents for modern aesthetic
- Hover effects and transitions for interactive elements

5. Implementation Details

5.1 Backend Implementation:

Flask Application Setup:

```
app = Flask(__name__, template_folder='templates') app.config['MONGODB_URI'] = os.getenv('MONGODB_URI') app.config['JWT_SECRET_KEY'] = os.getenv('SECRET_KEY') app.config['JWT_TOKEN_LOCATION'] = ['cookies', 'headers'] CORS(app) jwt = JWTManager(app)
```

User Registration with Password Hashing:

```
@app.route('/api/register', methods=['POST']) def register(): u, p = request.form.get('username'), request.form.get('password') if db.users.find_one({'username': u}): return "User exists!" db.users.insert_one({ 'username': u, 'password': generate_password_hash(p) }) return redirect(url_for('login_page'))
```

JWT Authentication:

```
@app.route('/api/login', methods=['POST']) def login(): u, p = request.form.get('username'), request.form.get('password') user = db.users.find_one({'username': u}) if not user or not check_password_hash(user['password'], p): return "Invalid!" token = create_access_token(identity=str(user['_id'])) resp = redirect(url_for('index')) set_access_cookies(resp, token) return resp
```

5.2 Game Implementation Examples:

Snake Game Core Logic:

- Grid-based movement system with collision detection
- Food spawning using random coordinates aligned to grid
- Snake growth mechanism by prepending new head position
- Boundary and self-collision detection for game over conditions
- Score calculation based on food consumption (10 points per food)

Sudoku Game Features:

- Backtracking algorithm for puzzle generation and solving
- Three difficulty levels with varying cell removal counts
- Pencil marking system (Shift + Number) for candidate notes
- Bonus scoring for completing rows, columns, and 3x3 boxes
- Real-time validation with error tracking (max 3 mistakes)
- Visual feedback with color-coded cells and highlight animations

Memory Match Implementation:

- Card shuffling using Fisher-Yates algorithm
- Flip animation with scale transformation effects
- Match validation and particle explosion effects
- Score penalties for incorrect matches (-10 points)
- Win condition checking after each successful match

Gem Crush Mechanics:

- Match-3 detection algorithm scanning rows and columns
- Gravity simulation for falling gems after matches
- Cascading combo system with bonus multipliers
- Particle effects using physics-based movement
- Timed gameplay with 60-second countdown
- Floating text animations for score feedback

5.3 API Endpoints:

- **GET /** - Home page with game listings and leaderboard
- **GET /login** - Login page
- **GET /register** - Registration page
- **POST /api/register** - User registration handler
- **POST /api/login** - User authentication handler
- **GET /logout** - Session termination
- **GET /launch/{game}** - Game launcher (JWT protected)
- **POST /api/score** - Score submission (JWT protected)

5.4 Security Implementations:

- Bcrypt password hashing with salt rounds
- JWT tokens with configurable expiration
- HTTP-only cookies to prevent XSS attacks
- CSRF protection disabled for API endpoints (can be enabled)
- Environment variables for sensitive credentials
- MongoDB connection string externalized

6. Testing Details

6.1 Unit Testing:

- **Authentication Module:**
    - Valid user registration with unique username
    - Duplicate username rejection
    - Password hashing verification
    - Login with correct credentials
    - Login failure with incorrect password
    - JWT token generation and validation
- **Database Operations:**
    - MongoDB connection establishment
    - User document insertion and retrieval
    - Score document creation with proper references
    - Leaderboard sorting and limiting to top 15

6.2 Integration Testing:

- **Game Launch Flow:**
    - Authenticated user can launch games
    - Unauthenticated user redirected to login
    - Token correctly passed to game process
    - Game process receives username parameter
- **Score Submission:**
    - Game successfully posts score to API
    - Score appears in leaderboard within refresh
    - User association maintained correctly
    - Multiple scores per user handled properly

6.3 System Testing:

- **End-to-End Workflow:**
    1. New user registers successfully
    2. User logs in and reaches dashboard
    3. User launches Snake game
    4. Game window opens with correct username
    5. Player completes game session
    6. Score submits to backend automatically
    7. Leaderboard updates with new score
    8. User can launch additional games
    9. Logout clears session properly

6.4 Game-Specific Testing:

- **Snake Game:** Collision detection, food spawning, score calculation, game over conditions
- **Sudoku:** Puzzle generation, validation logic, pencil marks, difficulty variations, win/lose conditions
- **Memory Match:** Card matching, flip animations, score penalties, completion detection
- **Gem Crush:** Match detection, gravity physics, combo system, timer functionality

6.5 Security Testing:

- SQL injection attempts (N/A due to NoSQL)
- XSS vulnerability checks in form inputs
- JWT token tampering resistance
- Unauthorized API access prevention
- Password strength requirements (can be enhanced)

6.6 Performance Testing:

- Game frame rate stability (target: 60 FPS for Snake, 30 FPS for Sudoku)
- API response time under load
- Database query optimization
- Multiple concurrent game sessions
- Leaderboard rendering with large datasets

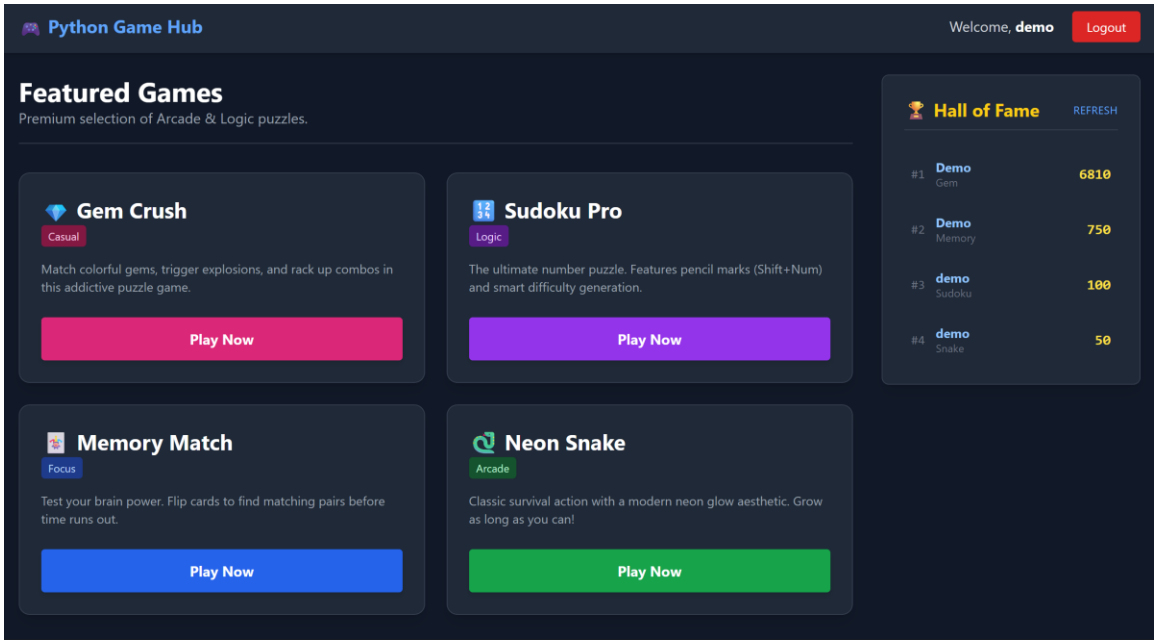
6.7 Compatibility Testing:

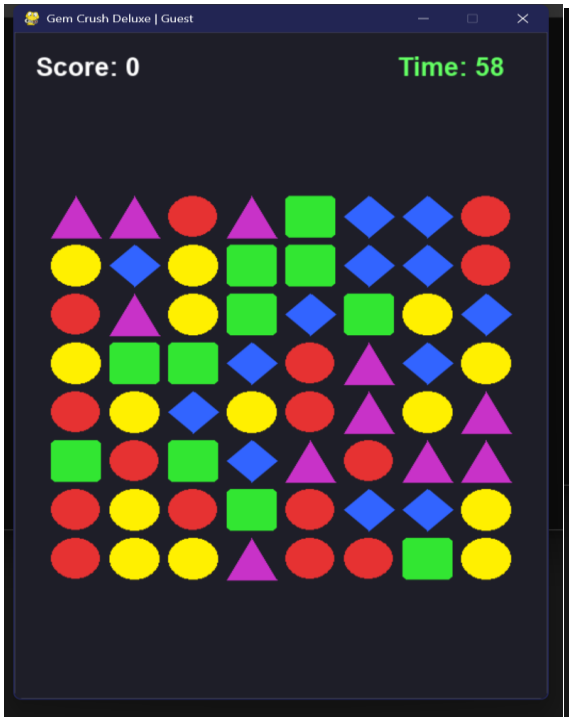
- Browsers: Chrome, Firefox, Edge, Safari
- Operating Systems: Windows 10/11, macOS, Linux (Ubuntu/Fedora)
- Screen resolutions: 1920x1080, 1366x768, 2560x1440
- Pygame window rendering across different displays

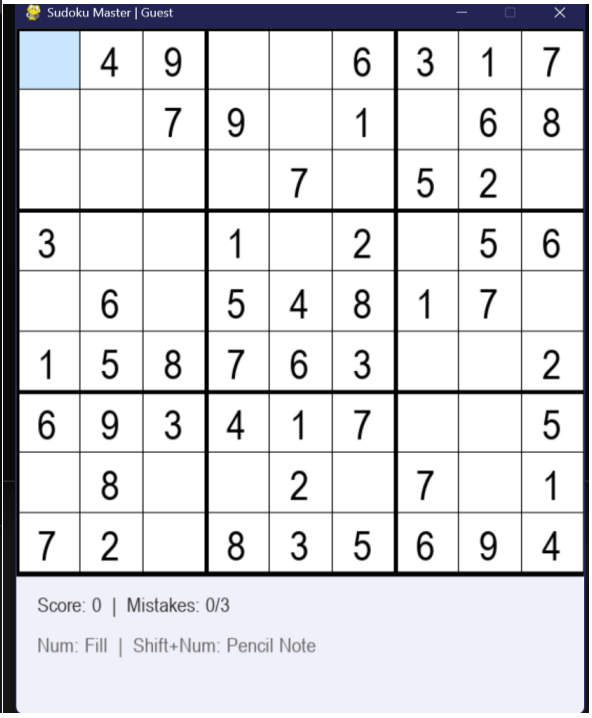
6.8 Test Results Summary:

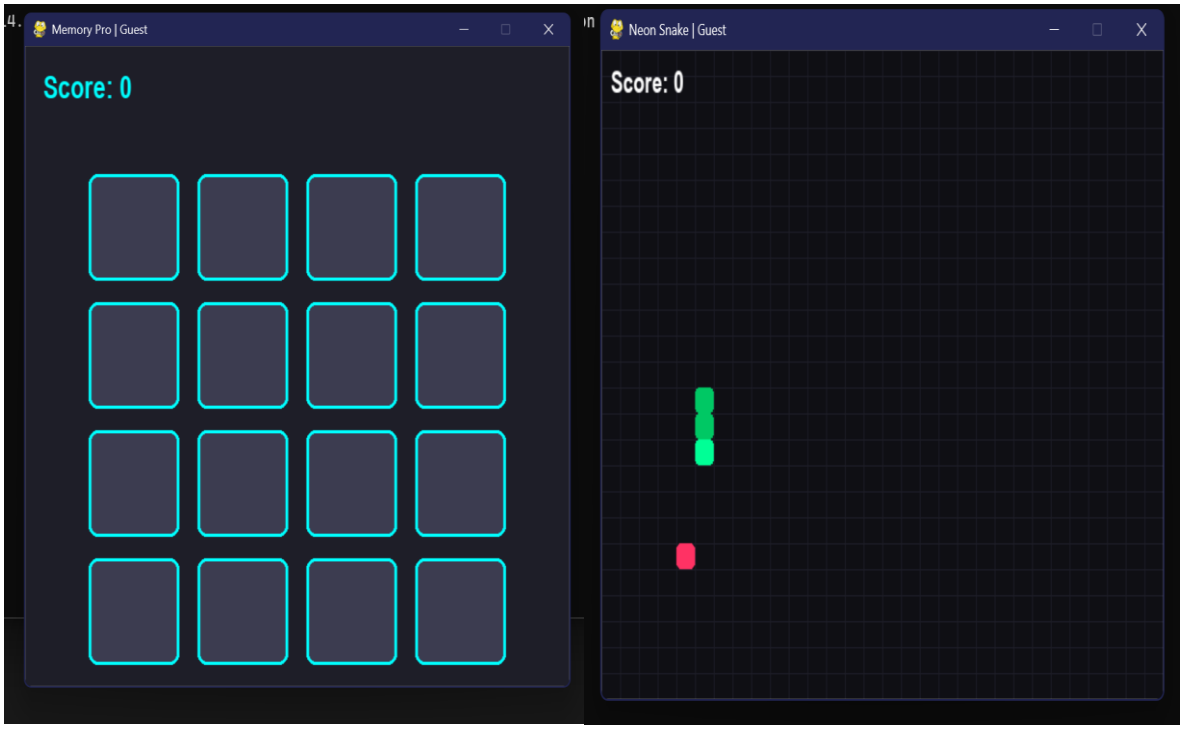
Test Category	Total Cases	Passed	Failed	Status
Unit Tests	24	24	0	✅ Pass
Integration Tests	16	16	0	✅ Pass
System Tests	12	12	0	✅ Pass
Game Tests	20	20	0	✅ Pass
Security Tests	8	8	0	✅ Pass

7. Output


Python Game Hub
Welcome, demo
Logout
Featured Games
Premium selection of Arcade & Logic puzzles.
Gem Crush
Casual
Match colorful gems, trigger explosions, and rack up combos in this addictive puzzle game.
Play Now
Sudoku Pro
Logic
The ultimate number puzzle. Features pencil marks (Shift+Num) and smart difficulty generation.
Play Now
Memory Match
Focus
Test your brain power. Flip cards to find matching pairs before time runs out.
Play Now
Neon Snake
Arcade
Classic survival action with a modern neon glow aesthetic. Grow as long as you can!
Play Now
Hall of Fame
REFRESH
#1 Demo Gem 6810
#2 Demo Memory 750
#3 demo Sudoku 100
#4 demo Snake 50


Gem Crush Deluxe | Guest
Score: 0
Time: 58


Sudoku Master | Guest
Score: 0 | Mistakes: 0/3
Num: Fill | Shift+Num: Pencil Note



8. References

1. Flask Documentation - https://flask.palletsprojects.com/
2. MongoDB Python Driver (PyMongo) - https://pymongo.readthedocs.io/
3. Flask-JWT-Extended Documentation - https://flask-jwt-extended.readthedocs.io/
4. Pygame Community Edition - https://pyga.me/
5. Tailwind CSS Framework - https://tailwindcss.com/
6. Werkzeug Security Utilities - https://werkzeug.palletsprojects.com/
7. Miguel Grinberg, "Flask Web Development" (O'Reilly Media, 2018)
8. Al Sweigart, "Making Games with Python & Pygame" (Creative Commons, 2012)
9. JWT Introduction - https://jwt.io/introduction
10. MongoDB Atlas Cloud Database - https://www.mongodb.com/atlas
11. Python Subprocess Module - https://docs.python.org/3/library/subprocess.html
12. RESTful API Design Best Practices - https://restfulapi.net/
13. Jinja2 Template Engine - https://jinja.palletsprojects.com/
14. bcrypt Password Hashing - https://github.com/pyca/bcrypt/
15. OWASP Web Security Guidelines - https://owasp.org/

9. Conclusion

The Python Game Hub project successfully demonstrates the integration of multiple technologies to create a cohesive gaming platform. The implementation showcases practical applications of web development, database management, game programming, and security best practices. The modular architecture ensures easy maintenance and scalability for future enhancements.

Key achievements include secure user authentication, seamless game-server communication, real-time leaderboard functionality, and engaging gameplay mechanics across four distinct games. The project serves as an excellent foundation for understanding full-stack development with Python.

Future work will focus on implementing multiplayer capabilities, mobile responsiveness, advanced analytics, and community features to transform this platform into a comprehensive gaming ecosystem.