Intel Unnati Industrial Training Program 2025-2026

Project Report

Problem Statement:

GenAI Interactive Learning Games

Team Name: Karunya_Nexus

Team Members:

THIRMALREDDY MARY SHALINI

JENULIN MAKROS G

GOPU RACHEL REJOICE REDDY

TABLE OF CONTENT

I. Introduction:

In today's fast-paced world, access to essential services such as healthcare, finance, education, and utilities remains a significant challenge for many individuals, particularly those in underserved and rural communities. These challenges are often exacerbated by the fragmented nature of service delivery systems, where each service operates in isolation, requiring individuals to navigate multiple platforms and interfaces. This fragmentation not only makes it difficult for people to access the services they need but also leads to inefficiencies and increased costs for service providers.

Problem Statement

1. Lack of Personalization in Physics Education

- Traditional teaching methods follow a one-size-fits-all approach, failing to adapt to students' individual learning speeds.
- Many students struggle with abstract concepts due to the absence of customized explanations based on their understanding level.

2. Limited Interactive Learning Resources

- Textbooks and static online tutorials lack engagement, making it difficult for students to visualize physics concepts.
- Conventional materials do not provide hands-on experiences or allow experimentation with different scenarios.

3. Difficulty in Understanding Abstract Concepts

- Topics like Newton’s Laws, friction, and electricity often require real-world analogies or interactive demonstrations for clarity.
- Without dynamic visualizations, students fail to grasp fundamental physics principles intuitively.

4. Absence of Real-Time Feedback and Guidance

- Existing physics learning tools do not offer immediate corrective feedback, leaving students uncertain about their mistakes.
- Lack of real-time hints and explanations results in frustration and disengagement.

5. Need for Gamified and AI-Driven Learning

- Gamification has been proven to enhance student engagement, yet it is underutilized in physics education.
- AI-powered tools can personalize learning experiences but are rarely integrated into educational games.

Project Objectives

- Develop an interactive web-based game that simplifies physics concepts through engaging puzzles and an AI-powered tutor.

- Integrate adaptive difficulty levels (easy, medium, hard) to accommodate diverse learning abilities and ensure gradual knowledge progression.
- Provide real-time hints and feedback using the Groq API, ensuring learners receive context-aware assistance without feeling overwhelmed.
- Enhance user engagement with visuals by implementing Phaser.js animations and space-themed graphics to make learning enjoyable.
- Ensure accessibility and scalability by designing a database-free architecture that allows students and educators to use the platform effortlessly.
- Explore AI's potential in education by integrating AI-generated lessons and responses, paving the way for future enhancements in learning tools.

Team members & Contributions:

i. Thirmalreddy Mary Shalini:

- Team Lead.
- Front end Development.
- Implementing Phaser.js scenes (HomeScene, StoryScene, PuzzleScene).
- User Interface with HTML5 and CSS in index.html and style.css.

ii. Jenulin Makros:

- Backend development
- Integrated Flask with the Groq API in app.py.
- AI Content

iii. Gopu Rachel Rejoice:

- Handled the AI tutor module (avatar.js).
- Feature Sync
- Testing and debugging

II. Literature Review

Previous Work and Existing Solutions

1. Traditional Educational Games and Platforms

- Platforms like Khan Academy and PhET Interactive Simulations provide physics tutorials but lack AI-driven adaptability.
- Simulations improve engagement but do not adjust difficulty based on the learner's needs.

2. Gamified Learning Tools

- Kahoot and Quizizz gamify learning but focus more on quizzes than deep conceptual understanding.
- These platforms encourage competition but do not facilitate interactive problem-solving.

3. AI in Education

- Research by Gee (2003) emphasizes that game-based learning improves retention by embedding problem-solving in engaging contexts.
- AI-driven learning tools are still in their infancy, with limited real-time feedback mechanisms.

Analysis of Existing Web Applications Providing Similar Information

1. Strengths of Current Learning Platforms

- Brilliant.org offers structured physics problems with explanations, but it requires a paid subscription.
- NASA's educational site provides physics-related space content but lacks interactivity.

2. Limitations and Gaps in Current Platforms

- Most existing platforms rely on server-side databases, whereas our approach ensures portability by using a client-side model.
- The absence of personalized, adaptive difficulty adjustments in existing resources highlights the need for our solution.

Unlike these platforms, it integrates AI, game-based challenges, and adaptive physics learning without requiring a server-side database.

Theoretical Framework and Key Concepts

1. Constructivist Learning Theory:

- Constructivism asserts that learners actively construct their knowledge rather than passively absorbing information.
- Learning is most effective when students engage with concepts through hands-on activities, exploration, and problem-solving.
- Story-driven learning aligns with constructivist principles, making abstract physics concepts easier to understand by embedding them in real-world scenarios.

2. Adaptive Learning Model:

- If a student struggles with a concept, the AI tutor provides simpler examples and more hints before progressing.
- If a student excels, the game introduces more complex challenges to deepen understanding.
- Adaptive learning adjusts difficulty levels based on the learner's competence, ensuring continuous improvement without frustration.

3. AI-Driven Personalized Learning Mechanisms:

- The AI generates customized physics questions based on the student's past performance.
- It adjusts difficulty dynamically by modifying numerical values and complexity levels in each problem.

III. Requirement Analysis

Functional requirements.

1. Topic Selection

- Users must be able to browse and select physics topics (e.g., Newton's Laws, Friction, Gravity) through an intuitive interface.
- The modal interface should visually categorize topics and highlight key subtopics, ensuring ease of navigation.

2. Game Progression

- The game should start with introductory concepts before gradually introducing laws and problem-solving elements.
- Progression should be nonlinear, allowing users to revisit past topics or skip ahead based on AI recommendations.

3. AI Tutor Assistance

- The AI tutor provides real-time hints through speech synthesis and text bubbles, adapting to user responses.
- Assistance is context-aware, meaning the AI tailors feedback based on past mistakes and knowledge gaps.

4. Puzzle Mechanics

- The game includes multiple-choice questions, interactive physics simulations, and real-world problem-solving tasks.
- Dynamic difficulty adjustment ensures that puzzles get harder or easier based on the player's performance.

5. Client-Side Score Tracking

- User progress is stored in local storage, ensuring session continuity without requiring a database.
- The system should allow users to reset progress, view past achievements, and track improvement over time.

6. Seamless Navigation

- Users should be able to restart sessions, switch topics, or return to the main menu without losing progress.
- The navigation should be smooth and intuitive, using UI elements like breadcrumbs and quick-access buttons.

Non-Functional Requirements

1. Performance and Speed

- Application must load within 5 seconds for an optimal user experience.
- Phaser.js animations must maintain a smooth 30 FPS.

2. Accessibility and Portability

- Must function without a backend database to ensure lightweight deployment.
- Should work on modern browsers without additional installations.

System Requirements

1. Hardware

- **Client Devices:** Any modern desktop, laptop, or tablet with a web browser that supports HTML5 and JavaScript.
- At least 4GB RAM to handle browser-based game execution efficiently.

2. Software

- **Frontend**: Phaser.js for game development, HTML5 for structure, JavaScript ES6 for logic, and CSS for styling.
- **Backend**: Flask framework in Python 3.8+, integrating with Groq API for AI-generated responses.

3. Network

- **Internet Connectivity:** A stable internet connection is required for Groq API calls; fallback data ensures offline functionality.

Stakeholder Requirements

1. Middle School Students

- Require an engaging, interactive learning experience.

2. Teachers and Educators

- Need a reliable, free resource for classroom use.

IV. Implementation

Technologies and Tools Used

- Phaser.js for animations and gameplay.
- HTML5 and CSS for structure and styling.
- Flask for API development.
- Groq API for AI-generated content.
- Version Control: Git
- IDE: VS Code

Development Environment Setup

Install Python:

Ensure Python 3.x is installed on your system. Verify by running

```
python –version
```

Install Required Libraries:

```
npm install phaser
```

```
node -v

npm -v

pip install flask

pip install groq

pip install flask-cors
```

Implementation Details of Major Modules/Components

1. Game Scene Module

Scenes:

- **HomeScene:** Displays the initial space background and sets up the game environment.
- **StoryScene:** Introduces the selected physics topic with narrative content and laws.
- **PuzzleScene:** Presents interactive puzzles with progress tracking.

Code:

```
export class PuzzleScene extends Phaser.Scene {
  constructor() {
    super('PuzzleScene');
    this.topic = 'newton_laws';
  }
  preload() {
    this.load.image('puzzle_bg', 'space_background3.jpeg');
    console.log('PuzzleScene: Preloading puzzle_bg');
  }
  init(data) {
    this.difficulty = data.difficulty || 'easy';
    this.topic = data.topic || 'newton_laws';
    console.log(`PuzzleScene: Init with topic ${this.topic}, difficulty ${this.difficulty}`);
  }
  create() {
    const { width, height } = this.scale;
    this.add.image(width / 2, height / 2, 'puzzle_bg').setAlpha(0.9);
    console.log('PuzzleScene: Background created');
  }
}
```

Views/Functions:

- preload(): Loads assets like images (space_background.jpeg, astronaut.png).
- create(): Renders backgrounds and initializes scene elements with scaling logic.

2. AI Tutor Module

Classes:

- **AITutor: Manages avatar creation, speech, and animations.**

Code:

```
speak(message, emotion = 'neutral') {
        this.avatarElement.className = `avatar-${emotion}`;
        this.speechBubble.textContent = message;
        this.speechBubble.style.display = 'block';
        if ('speechSynthesis' in window) {
            const utterance = new SpeechSynthesisUtterance(message);
            utterance.rate = 0.9;
            speechSynthesis.speak(utterance);
        }
giveHint(hint) {
        this.speak("Let me give you a hint...", 'thinking');
        setTimeout(() => {
            this.speak(hint, 'helpful');
        }, 1500);
    }
async narrateLaws(lawsArray) {
        this.avatarElement.classList.remove('avatar-puzzle-mode');
        this.avatarElement.classList.add('avatar-lesson-mode');
        this.avatarElement.style.left = '50px';
        this.avatarElement.style.right = 'auto';
        this.avatarElement.style.bottom = '20px';
        this.avatarElement.style.transform = 'scaleX(1)';
        this.speechBubble.style.left = '30px';
        this.speechBubble.style.right = 'auto';
        this.speechBubble.style.bottom = '230px';
        await this.speak("Now, let's learn the key concepts:", 'teaching');
        for (const law of lawsArray) {
            await this.speak(law, 'explaining');
            await new Promise(resolve => setTimeout(resolve, 1500));
        }
    }
```

Views/Functions:

- speak(message, emotion): Displays text and triggers speech synthesis with emotion-based styling.
- giveHint(hint): Delivers hints with timed delays for student reflection.

- narrateLaws(lawsArray): Sequentially explains physics laws with pauses for comprehension.

Templates/Styles:
Relies on avatar.css for positioning (e.g., lesson vs. puzzle mode) and animations (e.g., celebrate).

3. Content Generation Module

Endpoints:

- **/api/lesson:** Generates lesson content (intro, laws, puzzles) based on topic and difficulty.
- **/api/verify:** Validates user answers and adjusts difficulty dynamically.

Code:

```
def query_groq(prompt, max_retries=5, backoff_factor=1.5):
    for attempt in range(max_retries):
        try:
            logger.info(f"Calling Groq API - Attempt {attempt+1}/{max_retries}")
            completion = client.chat.completions.create(
                model="llama-3.3-70b-versatile",
                messages=[
                    {"role": "system", "content": "You are an AI tutor helping students understand
physics."},
                    {"role": "user", "content": prompt}
                ],
                temperature=0.7,
                max_completion_tokens=1000,
                top_p=1,
                timeout=10  # Add timeout parameter
            )
            response_text = completion.choices[0].message.content.strip()
                logger.info(f"Groq API response: {response_text[:50]}... ({len(response_text)}
chars)")
            return clean_response(response_text, prompt)


def adjust_difficulty(current_difficulty, user_answer_correct):
    """Adjusts difficulty based on user's answer correctness."""
    levels = ['easy', 'medium', 'hard']
    index = levels.index(current_difficulty)
    if user_answer_correct.lower() == 'correct' and index < len(levels) - 1:
        return levels[index + 1]
    elif user_answer_correct.lower() == 'incorrect' and index > 0:
        return levels[index - 1]
    return current_difficulty

@app.route('/api/lesson', methods=['POST'])
def generate_lesson():
    data = request.get_json()
    if not data:
        return jsonify({"error": "No data provided"}), 400
    current_difficulty = data.get('difficulty', 'easy')
    user_answer_correct = data.get('user_answer_correct')
    topic = data.get('topic', 'newton_laws')
        next_difficulty = adjust_difficulty(current_difficulty, user_answer_correct) if
user_answer_correct else current_difficulty
    logger.info(f"Difficulty: {current_difficulty} -> {next_difficulty}, Topic: {topic}")
    story_text = query_groq(story_prompt(topic))
    laws_text = query_groq(laws_prompt(topic))
    formatted_laws = format_laws(laws_text, topic)
    cache_key = f'{topic}_{next_difficulty}'
```

Functions:

- query_groq(prompt): Fetches AI-generated content with retry logic for robustness.
- adjust_difficulty(current_difficulty, user_answer_correct): Adapts difficulty (easy, medium, hard).
- format_laws(laws_text, topic): Structures laws into a student-friendly array.

4. User Interaction Module

State Management:

Tracks variables like currentTopic, currentPuzzleIndex, score, and lives.

Code:

```
function fetchLessonData() {
    introContent.innerHTML = '<div class="loading">Loading...</div>';
    console.log(`Fetching lesson data for ${currentTopic}, difficulty: ${currentDifficulty}`);
   aiTutor.speak(`Loading ${currentDifficulty} level content about ${currentTopic.replace('_',
' ')}...`);
    fetch('http://127.0.0.1:5000/api/lesson', {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify({
            difficulty: currentDifficulty,
            user_answer_correct: null,
            topic: currentTopic
        })
    })
```

```
function handleAnswer(data, puzzle, blockPositions, optionButtons, selectedIndex) {
    optionButtons.forEach(b => {
        b.disabled = true;
        const idx = parseInt(b.getAttribute('data-index'));
        if (idx === data.correctIndex) b.classList.add('correct');
        else if (idx === selectedIndex) b.classList.add('incorrect');
    });
    const astronaut = puzzleContent.querySelector('.astronaut-character');
    if (data.correct) {
        aiTutor.celebrate();
        correctAnswers++;
        score += 10;
        const targetLeft = blockPositions[currentRockIndex];
        astronaut.classList.add('jump');
function displayPuzzleScene() {
    if (currentPuzzleIndex >= currentPuzzles.length || lives <= 0) {
        showResults();
        return;
    }
 nextPuzzleBtn.style.display = 'none';
    setupPuzzle(puzzle, blockPositions);
    puzzleContent.querySelector('.hint-btn').addEventListener('click', function() {
        aiTutor.giveHint(puzzle.hint || "Think about the key concept.");
    });
```

Views/Functions:

- fetchLessonData(): Retrieves lesson content via API calls with fallback handling.
- displayPuzzleScene(): Renders puzzles with user progression and UI elements.
- handleAnswer(data, puzzle): Processes user responses, updating state and triggering tutor feedback

V. Testing

Testing Methodologies Used

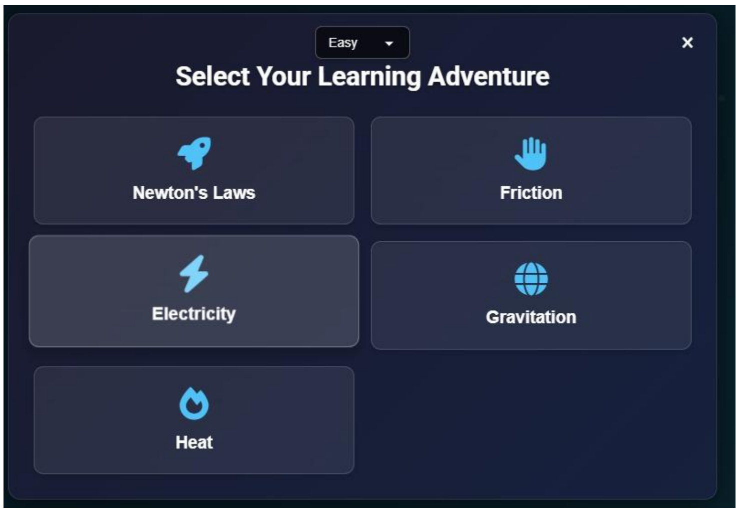
- **Unit Testing:** Verified AI tutor functionality and Flask API response.
- **Integration Testing:** Ensured frontend and backend components communicated correctly.
- **System Testing**: Testing the entire system to verify that it meets the specified requirements.

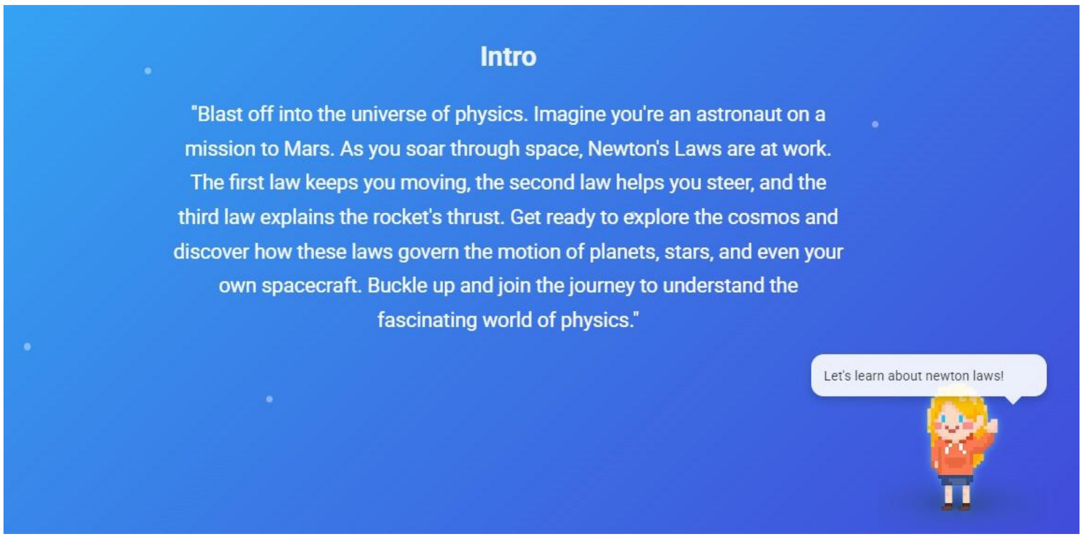
Test Cases and Test Scenarios

1. Topic Selection Test

- **Scenario:** A user selects a physics topic and difficulty level to begin the learning adventure.
- **Steps:**
    1. Open the application in a browser (e.g., Chrome).
    2. Click the "Choose Your Physics Topic" button on the topic screen.
    3. Select "Newton's Laws" from the topic modal.
    4. Choose "Easy" from the difficulty dropdown.
    5. Click the topic card to confirm selection.
- **Expected Result:** The intro screen loads with a space-themed introduction about Newton's Laws, and the AI tutor speaks a welcome message.
- **Actual Result:** Pass

Screenshot:




Intro

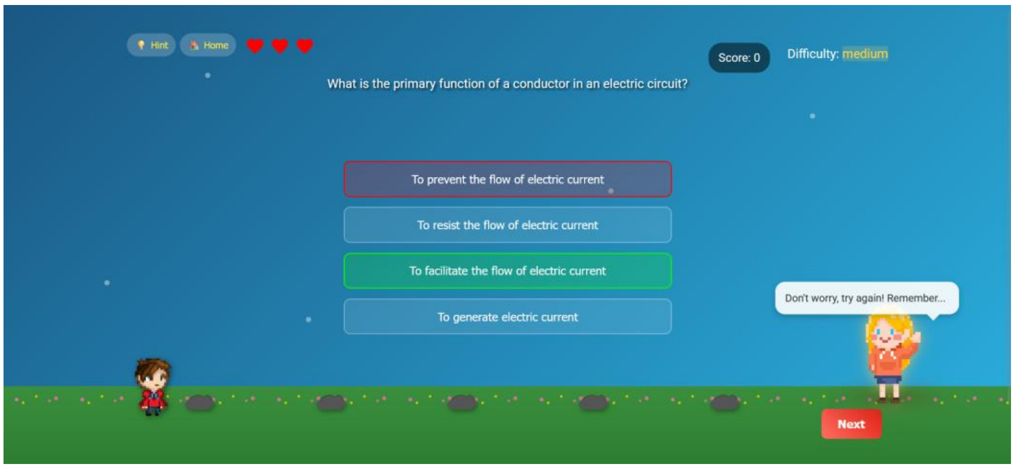
"Blast off into the universe of physics. Imagine you're an astronaut on a mission to Mars. As you soar through space, Newton's Laws are at work. The first law keeps you moving, the second law helps you steer, and the third law explains the rocket's thrust. Get ready to explore the cosmos and discover how these laws govern the motion of planets, stars, and even your own spacecraft. Buckle up and join the journey to understand the fascinating world of physics."

2. Difficulty Adjustment Test

- **Scenario**: The system adjusts difficulty after an incorrect puzzle answer.

- **Steps:**
    1. Start on the puzzle screen with "Medium" difficulty selected initially.
    2. Answer the first puzzle question incorrectly.
    3. Submit the answer and proceed to the next puzzle.
- **Expected Result**: The difficulty drops to "Easy" (reflected in the next puzzle), the AI tutor encourages the user, and lives decrease by 1.
- **Actual Result:** Pass

Screenshot:



What is the primary function of a conductor in an electric circuit?

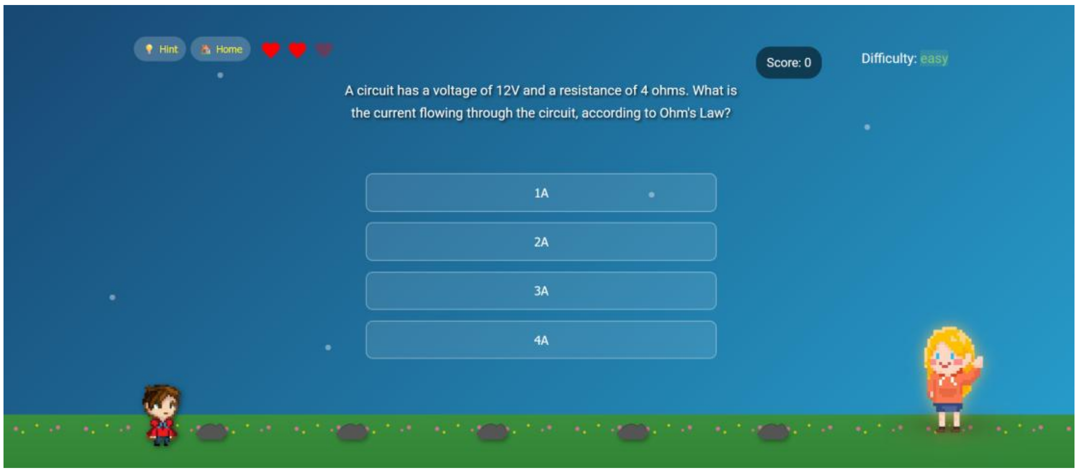
To prevent the flow of electric current

To resist the flow of electric current

To facilitate the flow of electric current

To generate electric current

Don't worry, try again! Remember...



A circuit has a voltage of 12V and a resistance of 4 ohms. What is the current flowing through the circuit, according to Ohm's Law?

1A

2A

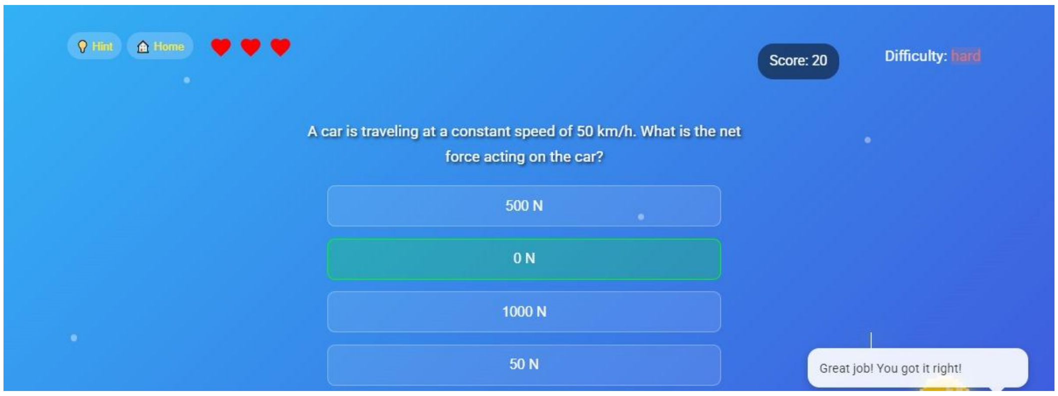
3A

4A

3. Puzzle Solving Test

- **Scenario:** A user answers a puzzle question correctly and progresses in the game.
- **Steps:**
    1. Navigate to the puzzle screen after completing the laws section (e.g., via "Next" from laws screen).
    2. Read the first puzzle question.
    3. Select the correct option ("No friction") from the available choices.
    4. Submit the answer by clicking the option button.
- **Expected Result:** The astronaut jumps to the next rock, the score increases by 10, and the AI tutor celebrates with a "Great job!" message.
- **Actual Result:** Pass

Screenshot:



A car is traveling at a constant speed of 50 km/h. What is the net force acting on the car?

500 N

0 N

1000 N

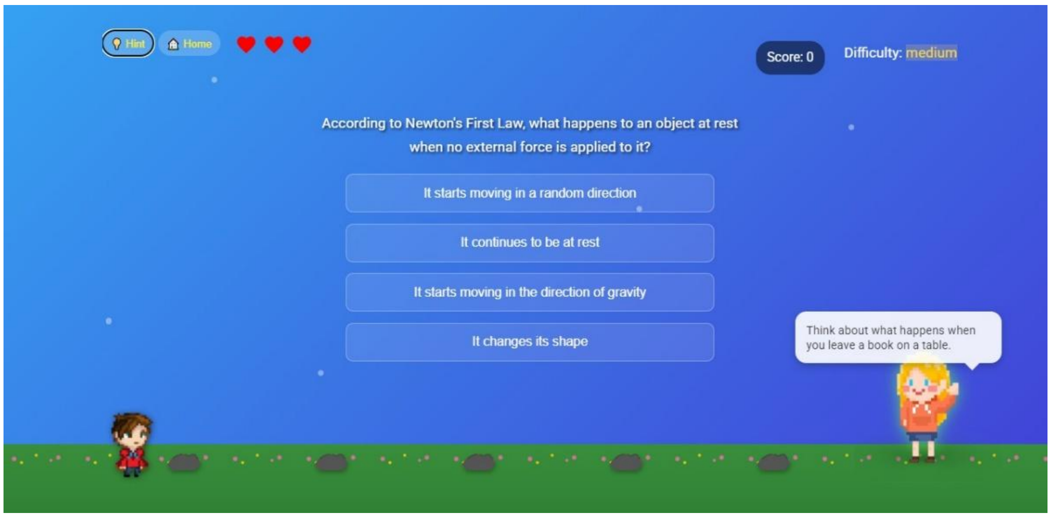
50 N

Great job! You got it right!

4. Hint Request Test

- **Scenario:** A user requests a hint for a puzzle question.
- **Steps:**
    1. Navigate to the puzzle screen as in the Puzzle Solving Test.
    2. Click the "Hint" button before selecting an answer.
    3. Wait for the AI tutor to provide a hint.
- **Expected Result:** The AI tutor displays a speech bubble with a helpful hint within 2 seconds.
- **Actual Result:** Pass

Screenshot:



According to Newton's First Law, what happens to an object at rest when no external force is applied to it?

It starts moving in a random direction

It continues to be at rest

It starts moving in the direction of gravity

It changes its shape

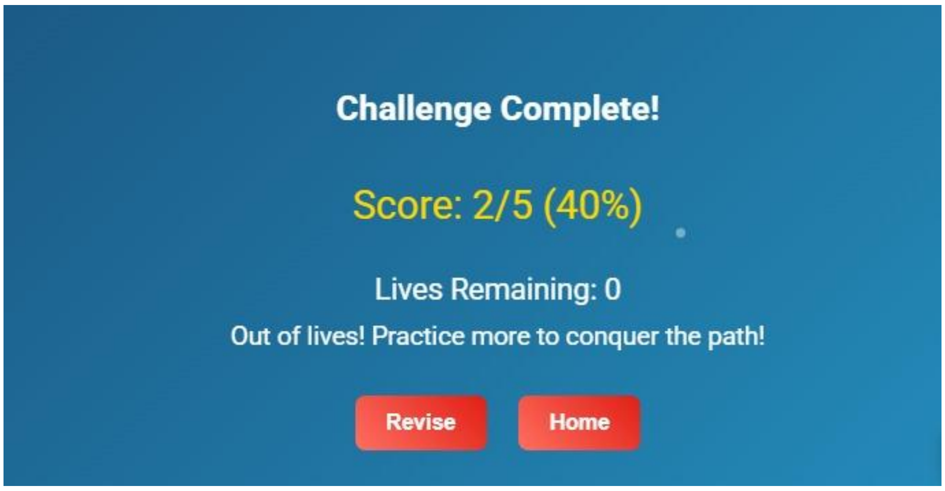
Think about what happens when you leave a book on a table.

5. Results Display Test

- **Scenario:** A user completes all puzzles or runs out of lives, triggering the results screen.
- **Steps:**
    1. Navigate to the puzzle screen and answer all 5 puzzles correctly (or lose all 3 lives).
    2. Wait for the results screen to appear after the final puzzle or lives depletion.

- **Expected Result:** The results screen displays the score (e.g., "4/5 (80%)"), lives remaining, and a message (e.g., "Good job! You're on the right track!"), with "Revise" and "Home" buttons.
- **Actual Result:** Pass

Screenshot:


Challenge Complete!

Score: 2/5 (40%)

Lives Remaining: 0

Out of lives! Practice more to conquer the path!

VI. Conclusion and Future Work

Conclusion

The " Gen Ai Learning Based Games " project has successfully delivered an innovative, AI-powered web application that transforms how middle school students learn physics concepts like Newton's Laws, friction, and electricity. By integrating interactive puzzles, a dynamic AI tutor, and adaptive difficulty levels into a unified, space-themed platform, the project addresses the challenge of making abstract physics engaging and accessible. Key achievements include the seamless use of Phaser.js for game dynamics, the Groq API for real-time content generation, and a database-free design that ensures scalability and ease of deployment. The application enhances user experience through intuitive navigation, visually appealing animations, and personalized feedback from the AI tutor, fostering both learning and retention. Its ability to operate with fallback data during API failures further demonstrates its robustness and reliability. This project not only proves the potential of gamified education but also sets a precedent for leveraging AI in STEM learning, significantly improving comprehension and enjoyment for students of varying skill levels.

- **Interactive Gamified Learning:** The project transforms physics education into an engaging game using Phaser.js, featuring scenes like HomeScene, StoryScene, and PuzzleScene
- **AI-Powered Tutor Assistance:** An intelligent AI tutor, implemented in avatar.js, provides real-time guidance with speech synthesis and dynamic hints. This feature personalizes learning, offering immediate feedback and support without overwhelming students, making complex physics more approachable and interactive.

- **Adaptive Difficulty Adjustment:** Correct answers increase difficulty (e.g., from easy to medium), while incorrect ones lower it, ensuring challenges match skill levels

Future Work

While "Gen Ai Learning Based Games" establishes a strong foundation for interactive physics education, several opportunities exist to enhance its functionality and reach in future iterations:

1. **Expanded Topic Coverage:** Add more physics topics such as magnetism, optics, or thermodynamics to broaden the curriculum and appeal to a wider audience.
2. **Mobile Optimization:** Develop a mobile-friendly version or app using responsive design or a framework like React Native, enabling on-the-go learning for students.
3. **Advanced AI Features:** Integrate more sophisticated AI models (e.g., for natural language Q&A) to provide deeper explanations or interactive dialogues beyond hints and narration.
4. **Multilingual Support:** Implement language options to support non-English-speaking students, increasing global accessibility and inclusivity.
5. **Progress Persistence:** Add local storage or a lightweight database to save user progress, scores, and preferences across sessions, enhancing the learning journey.
6. **Collaborative Learning:** Introduce multiplayer or classroom modes where students can compete or collaborate on puzzles, fostering peer engagement.
7. **Teacher Integration:** Create a teacher dashboard to customize lessons, track student performance, and integrate the tool into classroom settings.
8. **Continuous Content Updates:** Establish a mechanism to periodically refresh AI-generated puzzles and intros, ensuring variety and preventing repetition over time.

By pursuing these enhancements, the project can evolve into a more versatile and impactful educational tool, maintaining its commitment to innovation, engagement, and student-centered learning in the ever-growing field of educational technology.