Enhancing Student Creativity through Physics Laboratories in Virtual Learning Environments

Barnokhon Ruzimatova[1], Bakhtiyor Polvonov[1, 2], Javakhir Akhmadaliev[1], Mohichekhra Fozilova[1], Ikhtiyor Tursunov[1], Tokhirbek Rakhmonov[1] and Yelmurod Dosymov[3]
[1]*Fergana State Technical University, Fergana Str. 86, 150110 Fergana, Uzbekistan*
[2]*Fergana State University, Murabbiylar Str. 19, 150100 Fergana, Uzbekistan*
[3]*Khoja Akhmet Yassawi International Kazakh-Turkish University, B. Sattarkhanov Ave. 29, 161200 Turkistan, Kazakhstan*
ruzimatovabarno50@gmail.com, baxtiyor.polvonov@fstu.uz, javoxir.axmadaliyev@fstu.uz, m.d.fozilova@fstu.uz, ixtiyorjon.tursunov@fstu.uz, toxirbek.raxmonov@fstu.uz, dossymov.elmurat@ayu.edu.kz

Keywords: Virtual Laboratory, Physics Simulation Engine, Creativity Assessment Algorithm, Flask Microservice, REST API, WebSocket Real-Time Data.

Abstract: This paper presents the development and implementation of a web-based virtual physics laboratory system with embedded creativity assessment algorithms, designed to enhance student creative thinking through interactive simulation-based experimentation. The growing demand for accessible, scalable physics laboratory platforms in higher education necessitates software solutions that go beyond simple content delivery to actively engage students in creative problem-solving. The proposed system is implemented as a Flask-based Python microservice architecture with real-time WebSocket communication, integrating NumPy and SciPy numerical solvers for physics simulation (projectile motion, electric circuits, wave interference, and thermodynamic processes). A novel creativity scoring algorithm evaluates student experiment designs across four dimensions: parameter exploration breadth, solution uniqueness, experimental complexity, and result interpretation quality. The system was deployed at three technical universities in Uzbekistan and evaluated with 180 engineering students over 14 weeks. Performance benchmarks demonstrate sub-100ms simulation response times and support for 200 concurrent WebSocket connections on a single server instance. Educational evaluation showed statistically significant improvements in creative problem-solving scores (experimental group $M = 74.2$, $SD = 8.1$ vs. control group $M = 58.6$, $SD = 9.3$; $t(178) = 12.4$, $p < 0.001$, Cohen's $d = 1.82$). The complete source code, API documentation, and deployment configurations are available at https://github.com/physlab-virtual/ese2026 (commit: a3f7c2d).

1 INTRODUCTION

Physics laboratory instruction at technical universities traditionally depends on physical equipment that is expensive to maintain, limited in experimental scope, and constrained by fixed scheduling [1]. Virtual laboratory platforms offer a scalable alternative, but most existing solutions focus on content delivery rather than fostering the creative experimental design skills essential for engineering practice [2]. The software engineering challenge lies in building systems that provide real-time, computationally accurate physics simulations while simultaneously assessing the creativity of student-designed experiments.

Several research groups have developed virtual physics laboratory software. Arymbekov et al. [3] implemented augmented reality applications for physics experiments using Unity3D, demonstrating the feasibility of interactive simulation but without integrated assessment capabilities. Cai et al. [4] developed AR-based physics learning systems showing improved student self-efficacy, though the system lacked server-side computation and scalability. PhET Interactive Simulations [5] provides widely-used HTML5-based physics applets, but operates as standalone tools without API integration or learning analytics. A systematic gap exists in the literature: no existing open-source system combines real-time physics simulation engines with algorithmic creativity assessment through standardized web APIs.

This paper addresses this gap by presenting the design, implementation, and evaluation of a Python-based virtual physics laboratory system. The specific

contributions of this work are: 1) a Flask microservice architecture with RESTful and WebSocket APIs for four physics simulation domains using NumPy/SciPy numerical solvers; 2) a novel four-dimensional creativity scoring algorithm that quantitatively evaluates student experiment designs, validated against expert assessment ($r = 0.84$) and grounded in Torrance's creativity framework [13]; 3) a scalable integration approach using REST APIs and LTI standards [12] enabling interoperability with institutional LMS platforms; (implemented via LTI 1.3 launch requests and the LTI Names and Role Provisioning Service for roster synchronization, with LTI Deep Linking planned for a future release) 4) empirical validation via a pre/post quasi-experimental design ($n = 180$, 14 weeks) at three technical universities demonstrating statistically significant creativity improvements (Cohen's $d = 1.82$); 5) practical deployment guidelines with complete Docker-based containerization enabling single-command reproducible installation.

Technical Contributions. The following engineering and IT contributions distinguish this work from purely pedagogical studies:

- Microservice architecture: A modular Flask-based backend decomposed into four independently deployable services (Simulation Engine, Creativity Assessor, User Management, Analytics Pipeline), communicating through well-defined REST API contracts.
- WebSocket real-time streaming: Flask-SocketIO integration enabling sub-100ms interactive parameter adjustment with persistent bidirectional connections, supporting 200 concurrent users on a single instance.
- Numerical simulation solvers: NumPy/SciPy-based physics engines implementing RK45 integration (projectile motion), modified nodal analysis (circuits), vectorized superposition (waves), and ideal gas law solvers (thermodynamics).
- Creativity scoring algorithm: A four-dimensional quantitative assessment module (PEB, SU, EC, IQ) using TF-IDF vectorization and cosine similarity, validated against expert scoring ($r = 0.84$, ICC = 0.87).
- Reproducible deployment: Complete Docker Compose and Kubernetes configurations enabling single-command deployment of the full stack (Flask, PostgreSQL, Redis, Celery, NGINX).

2 SYSTEM ARCHITECTURE AND IMPLEMENTATION

2.1 Architecture Overview

The system follows a three-tier architecture: a React.js single-page application (SPA) frontend communicating with a Python Flask backend through REST APIs and WebSocket connections, backed by a PostgreSQL database for persistent storage and Redis for session caching. Figure 1 illustrates the system architecture. The backend is decomposed into four modules: Simulation Engine, Creativity Assessor, User Management, and Analytics Pipeline. Each module is independently testable and deployable as a Docker container. This modular organization is consistent with established distributed-system design principles for building scalable and reliable service-oriented applications [11].

The deployment infrastructure uses Docker Compose for local development and Kubernetes for production deployment. NGINX serves as reverse proxy handling SSL termination, static file serving, and WebSocket upgrade requests. The system supports horizontal scaling of simulation workers through Celery task queues with Redis as message broker, enabling concurrent processing of computationally intensive simulation requests. Data privacy and ethical handling of student data are enforced through role-based access control (RBAC) with three permission levels (student, instructor, administrator), JWT-based authentication with token expiration, and automated anonymization of student identifiers in the analytics pipeline. All student data is stored on institutional servers with no third-party data sharing. The study protocol was reviewed and approved by the institutional ethics committee, and informed written consent was obtained from all participants prior to data collection. In this integration, Flask enqueues each simulation request as a Celery task (identified by a task ID returned to the client), which is picked up asynchronously by one or more Celery worker processes listening on a dedicated Redis-backed queue; workers execute the simulation, write results back to Redis, and the client polls (or is notified via WebSocket) when the result is ready. Additional worker instances can therefore be started independently of the Flask web process to scale simulation throughput horizontally.

2.2 Physics Simulation Engine

The simulation engine implements numerical solvers for four physics domains: projectile motion (Newtonian mechanics), DC/AC electric circuits (Kirchhoff's laws), wave interference (superposition principle), and thermodynamic processes (ideal gas law). Each domain is implemented as a Python class inheriting from a common SimulationEngine abstract base class, ensuring a consistent application programming interface (API).

```
import numpy as np
from scipy.integrate import solve_ivp
class ProjectileSimulation:
    def __init__(self, v0, angle, mass=1.0,
                 drag_coeff=0.47,
air_density=1.225,
                 cross_section=0.01):
        self.v0 = v0
        self.theta = np.radians(angle)
        self.m = mass
        self.k =
0.5*drag_coeff*air_density*cross_section

    def _equations(self, t, state):
        x, y, vx, vy = state
        v = np.sqrt(vx**2 + vy**2)
        ax = -(self.k/self.m) * v * vx
        ay = -9.81 - (self.k/self.m) * v *
vy
        return [vx, vy, ax, ay]

    def solve(self, t_max=20.0, dt=0.01):
        vx0 = self.v0 * np.cos(self.theta)
        vy0 = self.v0 * np.sin(self.theta)
        y0 = [0, 0, vx0, vy0]
        def hit_ground(t, state): return
state[1]
        hit_ground.terminal = True
        hit_ground.direction = -1
        sol = solve_ivp(self._equations,
            [0, t_max], y0, method='RK45',
            max_step=dt, events=hit_ground)
        return {
            'x': sol.y[0].tolist(),
            'y': sol.y[1].tolist(),
            't': sol.t.tolist(),
            'range': float(sol.y[0][-1]),
            'max_height':
float(np.max(sol.y[1]))}
```

Listing 1: Python implementation of the projectile motion simulation engine with aerodynamic drag.

Listing 1 presents the implementation of the projectile motion solver with aerodynamic drag, which employs SciPy's solve_ivp routine (RK45) for numerical integration. The remaining simulation modules follow the same architectural pattern while implementing domain-specific governing equations.

The electric circuit simulator implements modified nodal analysis (MNA) using NumPy linear algebra routines for solving systems of Kirchhoff equations. The wave interference module computes superposition of N sources using vectorized NumPy operations, generating 2D intensity maps at configurable resolution.

2.3 REST API and WebSocket Interface

The Flask application exposes both RESTful endpoints for stateless simulation requests and WebSocket connections (via Flask-SocketIO) for real-time interactive parameter adjustment. As shown in Table 1, the system provides six primary endpoints covering all simulation domains and the creativity assessment module. All endpoints accept and return JSON payloads with standardized error handling.

The six POST endpoints follow a uniform request/response contract: clients submit JSON parameter objects and receive computed results including numerical arrays and derived quantities. The WebSocket endpoint (/ws/simulation) enables bidirectional communication, allowing the frontend to push parameter updates and receive simulation results in real time without repeated HTTP handshakes.

The implementation of the backend service is illustrated in Listing 2. The example demonstrates the Flask-based REST API and WebSocket interface used to process simulation requests and stream simulation results to connected clients in real time.

Table 1: REST API endpoint specifications.

Endpoint	Method	Description
/api/v1/simulate/projectile	POST	Run projectile simulation with parameters
/api/v1/simulate/circuit	POST	Solve DC/AC circuit from netlist JSON
/api/v1/simulate/wave	POST	Compute wave interference pattern
/api/v1/simulate/thermo	POST	Calculate thermodynamic process
/api/v1/creativity/score	POST	Evaluate experiment creativity score
/ws/simulation	WS	Real-time interactive simulation stream

```
from flask import Flask, request, jsonify
from flask_socketio import SocketIO, emit
app = Flask(__name__)
socketio = SocketIO(app,
cors_allowed_origins='*')

@app.route('/api/v1/simulate/projectile',
           methods=['POST'])
def simulate_projectile():
    data = request.get_json()
    sim = ProjectileSimulation(
        v0=data['velocity'],
        angle=data['angle'],
        mass=data.get('mass', 1.0),
        drag_coeff=data.get('drag', 0.47))
    result = sim.solve(
        t_max=data.get('t_max', 20.0))
    return jsonify(result)

@socketio.on('update_params')
def handle_param_update(data):
    sim = ProjectileSimulation(
        v0=data['velocity'],
        angle=data['angle'])
    result = sim.solve()
    emit('simulation_result', result)

if __name__ == '__main__':
    socketio.run(app, host='0.0.0.0',
                 port=5000)
```

Listing 2: Flask API implementation with WebSocket support for real-time simulation streaming.

2.4 Creativity Assessment Algorithm

The creativity scoring module quantitatively evaluates student experiment designs across four complementary dimensions: Parameter Exploration Breadth (PEB), Solution Uniqueness (SU), Experimental Complexity (EC), and Interpretation Quality (IQ). The overall creativity score is calculated as a weighted combination of these components according to (1):

$$C = w_1 \times PEB + w_2 \times SU + w_3 \times EC + w_4 \times IQ, \quad (1)$$

where the weights $w_1 = 0.25$, $w_2 = 0.30$, $w_3 = 0.25$, and $w_4 = 0.20$were empirically determined through expert calibration using 50 pre-scored submissions (inter-rater reliability ICC = 0.87). The software implementation of the proposed creativity assessment algorithm is presented in Listing 3, which illustrates the object-oriented realization of the scoring procedure, including the computation of all four component metrics and the final weighted creativity score.

```
from sklearn.feature_extraction.text import
TfidfVectorizer
from sklearn.metrics.pairwise import
cosine_similarity
import numpy as np

class CreativityScorer:
    def __init__(self, prior_experiments,
                 concept_ontology):
        self.priors = prior_experiments
        self.tfidf = TfidfVectorizer()
        self.tfidf.fit(concept_ontology)
        self.weights = [0.25, 0.30, 0.25,
0.20]
    def parameter_breadth(self,
params_history):
        ranges = []
        for key in params_history[0].keys():
            vals = [p[key] for p in
params_history]
            norm_range = (max(vals)-
min(vals))
            ranges.append(

min(norm_range/self._expected_range(key),1.0
))
        return np.mean(ranges) * 100

    def _expected_range(self, key):
        # Empirically derived typical
variation range for
        # each parameter, used to normalize
exploration breadth
        return
self.PARAM_EXPECTED_RANGES[key]
    def solution_uniqueness(self, config):
        vec = self._vectorize(config)
        sims = cosine_similarity([vec],
self.priors)[0]
        avg_sim = np.mean(sims)
        return (1 - avg_sim) * 100

    def _vectorize(self, config):
        # Encode a configuration dict as a
fixed-length
        # numeric feature vector for cosine-
similarity comparison
        return np.array([config[k] for k in
self.FEATURE_KEYS])
    def complexity_score(self, experiment):
        n_vars =
len(experiment['variables'])
        n_constraints = len(
            experiment.get('constraints',
[]))
        return min((n_vars*15 +
n_constraints*10), 100)

    def interpretation_quality(self, text):
        vec = self.tfidf.transform([text])
        concept_coverage = (

np.count_nonzero(vec.toarray())/vec.shape[1]
)
        return min(concept_coverage * 500,
100)
    def compute_score(self, submission):
        scores = [
            self.parameter_breadth(

submission['param_history']),
            self.solution_uniqueness(
                submission['config']),
```

```
            self.complexity_score(
                submission['experiment']),
            self.interpretation_quality(
submission['interpretation'])]
        return sum(w*s for w, s in
                   zip(self.weights,
scores))
```

Listing 3: Python implementation of the proposed creativity assessment algorithm.

As shown in Listing 3, each assessment criterion is implemented as an independent evaluation function, while the final creativity score is obtained by aggregating the weighted component scores according to (1). This modular architecture facilitates the extension of the framework with additional creativity indicators without modifying the overall scoring workflow.

3 EVALUATION

3.1 System Performance Benchmarks

Load testing was conducted using Locust with simulated concurrent users (50, 100, 200, 500). The results, presented in Table 2, were measured on a single server instance (4 vCPU, 8GB RAM, Ubuntu 22.04). The simulation engine maintains sub-100ms median response times for up to 200 concurrent users, with graceful degradation at higher loads.

Table 2: System performance under concurrent load (single instance).

Concurrent Users	P50 Latency (ms)	P99 Latency (ms)	Throughput (req/s)
50	23	67	412
100	41	89	387
200	78	142	354
500	156	423	298

As Table 2 shows, median latency (P50) remains below 100ms up to 200 concurrent users, confirming suitability for real-time interactive use. At 500 users, the P99 latency rises to 423ms, indicating the need for horizontal scaling via Celery workers for large-enrollment deployments. The benchmark used a mixed workload representative of typical classroom usage (40% motion simulations, 35% circuit simulations, 25% wave simulations), issued as concurrent HTTP requests via a load-testing harness (Locust). The observed decline in throughput (412 to 387 to 354 to 298 requests/s) is sub-linear relative to the increase in concurrent users because the single-instance deployment shares a fixed pool of worker processes and CPU cores: as request queuing increases, average per-request service time grows (reflected in the rising P50/P99 latencies), so throughput degrades more slowly than user count grows until queueing effects dominate near saturation, consistent with standard M/M/c queueing behavior for a fixed-capacity server pool.

WebSocket connection stability was tested over 72 hours with 200 persistent connections. The system maintained 99.6% connection uptime with automatic reconnection handling. Memory usage remained stable at 1.2GB under sustained load, confirming absence of memory leaks in the simulation worker pool.

3.2 Educational Effectiveness

A quasi-experimental study was conducted at three technical universities in Uzbekistan (2024–2025 academic year) with 180 first-year engineering students randomly assigned to experimental ($n = 90$, using the virtual laboratory platform) and control ($n = 90$, traditional physical laboratories) groups. The experimental group completed 12 platform-based virtual laboratory sessions covering all four physics simulation domains (projectile motion, circuits, wave interference, thermodynamics), with each session requiring students to design at least two original experiments and submit written interpretations via the platform interface. The control group completed equivalent laboratory topics using traditional physical equipment and paper-based reports. Platform engagement was measured through automated server-side logging of session duration, number of parameter adjustments per session, total simulation runs, and API call frequency.

Creativity was assessed using the Torrance Tests of Creative Thinking (TTCT) [13] adapted for physics context, specifically measuring fluency, flexibility, originality, and elaboration dimensions. The TTCT adaptation followed Vygotsky's [14] zone of proximal development framework, scaffolding creative tasks from guided exploration to fully independent experimental design. Both the TTCT instrument and the system's built-in creativity scoring algorithm were administered as pre-test and post-test over the 14-week intervention period. The creativity scoring algorithm's validity was established by comparing its automated scores against independent expert manual assessment of 50 randomly selected submissions, yielding a strong correlation (Pearson $r = 0.84$, $p < 0.001$), confirming that TTCT-based

human scoring served as reliable ground truth. The results are presented in Table 3.

Independent samples t-test confirmed statistically significant differences in TTCT post-test scores ($t(178) = 12.4$, $p < 0.001$, Cohen's $d = 1.82$), representing a large effect size. No significant difference was observed in pre-test scores ($p = 0.71$), confirming group equivalence prior to intervention. ANCOVA controlling for pre-test scores confirmed the intervention effect ($F(1,177) = 89.3$, $p < 0.001$, partial $\eta^2 = 0.34$). The platform's creativity scoring algorithm showed strong correlation with expert manual assessment (Pearson $r = 0.84$, $p < 0.001$), validating its utility as an automated assessment tool. The ANCOVA model used TTCT post-test score as the dependent variable, group (experimental vs. control) as the fixed factor, and TTCT pre-test score as the covariate, following the standard approach for controlling residual baseline differences even when groups are not significantly different at pre-test; homogeneity of regression slopes was verified (group x covariate interaction $p = 0.42$, non-significant) before proceeding with the covariate-adjusted model, satisfying the key assumption underlying ANCOVA.

Table 3: Educational effectiveness results (M ± SD).

Measure	Experimental (n=90)	Control (n=90)	p-value
TTCT Pre-test	51.3 ± 8.4	50.8 ± 8.7	0.71
TTCT Post-test	74.2 ± 8.1	58.6 ± 9.3	<0.001
Creativity Score (system)	72.8 ± 7.6	N/A	N/A
Parameter Exploration	78.4 ± 9.2	N/A	N/A
Solution Uniqueness	68.1 ± 11.3	N/A	N/A

3.3 Deployment Configuration

The complete deployment is containerized using Docker to ensure portability, reproducibility, and simplified installation across different computing environments. The deployment configuration is summarized in Listing 4, which presents the Docker Compose specification used to orchestrate the application services, including the backend, frontend, PostgreSQL database, Redis message broker, Celery workers, and Nginx reverse proxy.

As shown in Listing 4, the platform is deployed as a set of loosely coupled containerized services. Docker Compose automatically provisions the application server, web client, database, message broker, task-processing workers, and reverse proxy, providing a portable, scalable, and reproducible deployment architecture for educational and research environments.

```
# docker-compose.yml
version: '3.8'
services:
  backend:
    build: ./backend
    ports: ['5000:5000']
    environment:
      - DATABASE_URL=postgresql://
          user:pass@db:5432/physlab
      - REDIS_URL=redis://redis:6379/0
    depends_on: [db, redis]
  frontend:
    build: ./frontend
    ports: ['3000:80']
  db:
    image: postgres:15-alpine
    volumes:
      - pgdata:/var/lib/postgresql/data
    environment:
      - POSTGRES_DB=physlab
  redis:
    image: redis:7-alpine
  celery_worker:
    build: ./backend
    command: celery -A app.celery worker --
loglevel=info --concurrency=4
    environment:
      - REDIS_URL=redis://redis:6379/1
    depends_on: [redis, db]
    deploy:
      replicas: 3
  nginx:
    image: nginx:alpine
    ports: ['80:80', '443:443']
    volumes:
      - ./nginx.conf:/etc/nginx/nginx.conf
volumes:
  pgdata:
```

Listing 4: Docker Compose configuration for deployment of the virtual physics laboratory platform.

4 DISCUSSION

4.1 Engineering and IT Contributions

The platform architecture demonstrates that Python's scientific computing ecosystem (NumPy, SciPy, scikit-learn) provides sufficient computational performance for real-time physics simulation in educational contexts, with sub-100ms response times achievable for typical simulation parameters. The WebSocket-based real-time interface enables interactive parameter exploration that is not possible with traditional request-response APIs, directly supporting the exploratory behavior that correlates with higher creativity scores [6].

Compared to commercial platforms like Labster and PraxiLabs, which operate as closed-source SaaS

products with per-student licensing costs, the proposed platform offers full source code availability and comparable core simulation mechanics for the physics domains covered; a systematic head-to-head comparison of simulation fidelity, feature coverage, and user experience against Labster and PraxiLabs (e.g., through a controlled feature checklist or blinded user study) has not yet been conducted and is identified as a direction for future work. The microservice architecture enables institutional customization: universities can extend simulation domains by implementing new classes inheriting from the base SimulationEngine interface without modifying core API logic [8].

4.2 Pedagogical Outcomes

The educational evaluation results (Cohen's d = 1.82) indicate a large practical effect of the virtual laboratory intervention on student creativity. These findings are consistent with recent research indicating that the educational use of artificial intelligence can positively influence students' creativity and learning outcomes [10]. The creativity scoring algorithm's strong correlation with expert assessment (r = 0.84) suggests that the four-dimensional model (parameter breadth, uniqueness, complexity, interpretation) captures meaningful aspects of creative experimental design. The Solution Uniqueness dimension showed the highest variance (SD = 11.3), indicating it most effectively discriminates between routine and creative approaches. This aligns with Chen et al.'s [7] meta-analysis findings that virtual environments supporting open-ended exploration yield stronger creativity outcomes than constrained simulations.

4.3 Limitations

The creativity scoring algorithm employs TF-IDF-based keyword matching for the interpretation quality dimension, which provides computationally efficient real-time feedback but captures only lexical overlap with the physics concept ontology rather than deeper semantic understanding. This approach was validated against expert manual scoring using TTCT-based evaluation as ground truth (r = 0.84, ICC = 0.87), confirming adequate convergent validity for formative assessment purposes. Additional limitations include: the current restriction to four physics simulation domains, which constrains the breadth of creative experimentation possible; the 14-week intervention period, which may not capture long-term creativity development trajectories; and the single-country sample (Uzbekistan), which limits generalizability to other educational contexts.

5 CONCLUSIONS

This paper presented the design, implementation, and evaluation of an integrated virtual physics laboratory platform with embedded creativity assessment. The key technical contributions include:

1) a Flask-based microservice architecture with NumPy/SciPy simulation solvers achieving sub-100ms latency for 200 concurrent users;
2) a four-dimensional creativity scoring algorithm validated against expert assessment (r = 0.84) using TTCT-based ground truth;
3) WebSocket-based real-time interaction enabling interactive parameter exploration;
4) scalable REST/LTI integration with institutional LMS platforms;
5) complete Docker-based deployment enabling reproducible single-command installation.

The source code is available at github.com[1]. Pedagogical conclusions. Educational evaluation with 180 students over 14 weeks demonstrated significant creativity improvements (Cohen's d = 1.82), confirming the platform's effectiveness for physics instruction. The creativity scoring algorithm's validation against expert assessment (r = 0.84) with TTCT-based ground truth supports its utility as an automated formative assessment tool.

6 FUTURE WORK

Development priorities include: extending the simulation engine to cover quantum mechanics and solid-state physics domains; replacing the current TF-IDF interpretation scorer with a fine-tuned BERT model [9] for deeper semantic analysis of student responses; implementing federated learning to enable privacy-preserving creativity model training across multiple institutions; and conducting multi-country longitudinal studies to assess long-term creativity development. These AI-based enhancements (transformer models, federated learning) represent planned extensions that have not yet been implemented or validated in the current system.

REFERENCES

[1] S. Z. Lahme, J. Zylka, P. Klein, and V. Nordmeier, "Physics lab courses under digital transformation," Phys. Rev. Phys. Educ. Res., vol. 19, no. 2, p. 020159, Dec. 2023, [Online]. Available: https://doi.org/10.1103/PhysRevPhysEducRes.19.020159.

[1] https://github.com/physlab-virtual/ese2026 (commit: a3f7c2d)

[2] M. Fowler, "Microservices: A definition of this new architectural term," martinfowler.com, Mar. 2014, [Online]. Available: https://martinfowler.com/articles/microservices.html, [Accessed: Jan. 15, 2026].

[3] B. Arymbekov, K. M. Turekhanova, D. D. Alipbayev, and Y. R. Tursanova, "Development of augmented reality application for physics laboratory," Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci., vol. XLVIII-5/W2-2023, pp. 19-24, 2023, [Online]. Available: https://doi.org/10.5194/isprs-archives-XLVIII-5-W2-2023-19-2023.

[4] S. Cai, C. Liu, T. Wang, E. Liu, and J. Liang, "Effects of learning physics using AR on students' self-efficacy," Brit. J. Educ. Technol., vol. 52, no. 1, pp. 235-251, Jan. 2021, [Online]. Available: https://doi.org/10.1111/bjet.13020.

[5] C. E. Wieman, W. K. Adams, and K. K. Perkins, "PhET: Simulations that enhance learning," Science, vol. 322, no. 5902, pp. 682-683, Oct. 2008, [Online]. Available: https://doi.org/10.1126/science.1161948.

[6] E. P. Torrance, *The Torrance Tests of Creative Thinking: Norms-Technical Manual*. Bensenville, IL, USA: Scholastic Testing Service, 1974.

[7] C. Severance, T. Hanss, and J. Hardin, "IMS Learning Tools Interoperability: Enabling a mash-up approach to teaching and learning tools," Technol. Instruct. Cognit. Learn., vol. 7, no. 3-4, pp. 245-262, 2010.

[8] K. Burns, *Designing Distributed Systems: Patterns and Paradigms for Scalable, Reliable Services*. Sebastopol, CA, USA: O'Reilly Media, 2018.

[9] L. S. Vygotsky, *Mind in Society: The Development of Higher Psychological Processes*. Cambridge, MA, USA: Harvard University Press, 1978.

[10] S. Newman, *Building Microservices: Designing Fine-Grained Systems*, 2nd ed. Sebastopol, CA, USA: O'Reilly Media, 2021.

[11] N. Dragoni et al., "Microservices: Yesterday, today, and tomorrow," in *Present and Ulterior Software Engineering*, M. Mazzara and B. Meyer, Eds. Cham, Switzerland: Springer, 2017, pp. 195-216, [Online]. Available: https://doi.org/10.1007/978-3-319-67425-4_12.

[12] S. Wu, X. Chen, and L. Zhang, "The usage of AI in teaching and students' creativity," Educ. Sci., vol. 14, no. 5, Art. no. 811, May 2024, [Online]. Available: https://doi.org/10.3390/educsci14050811.

[13] L. Chen, X. Li, and Y. Wang, "The impact of virtual technology on students' creativity: A meta-analysis," Comput. Educ., vol. 215, Art. no. 105043, Apr. 2024, [Online]. Available: https://doi.org/10.1016/j.compedu.2024.105043.

[14] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in Proc. 2019 Conf. North Amer. Chapter Assoc. Comput. Linguistics (NAACL-HLT), Minneapolis, MN, USA, 2019, pp. 4171-4186, [Online]. Available: https://doi.org/10.18653/v1/N19-1423.