ADVANCES IN CYBER-PHYSICAL SYSTEMS

Vol. 10, No. 2, 2025

SCALABLE SYSTEM FOR AUTOMATED MODELING OF PHYSICAL PROCESSES AND EXPERIMENT MANAGEMENT

Mykhailo Bavdys, Oleksii Kushnir

Ivan Franko National University of Lviv, 50, Drahomanov Str, Lviv, 79005, Ukraine.
Authors' e-mails: *bavdysmyh@ukr.net, oleksiy.kushnir@lnu.edu.ua*

https://doi.org/10.23939/acps2025.02.223

Submitted on 29.09.2025

***Abstract*: This article presents a new scalable system for automated modeling of physical processes and experiment management based on microservice architecture and cloud technologies. The proposed platform addresses the growing need for flexible, cost-effective, and highly scalable computational solutions for physical research. The system utilizes Amazon Web Services cloud infrastructure with containerized microservices to provide automated resource allocation, experiment orchestration, and laboratory equipment integration. Key components include computational engines for numerical methods, visualization services, resource managers, and laboratory automation interfaces. Performance evaluation shows a 60% reduction in computational time and 45% cost savings compared to traditional approaches. The platform supports multiple physical domains, such as electromagnetic modeling, thermal analysis, and mechanical simulations.**

***Index terms*: cyber-physical systems, microservices, cloud computing, physical modeling, laboratory automation, AWS, scalability.**

I. INTRODUCTION

Modern physical research requires sophisticated computational tools capable of processing complex simulations, managing experimental workflows, and integrating diverse laboratory equipment [1]. Traditional physical modeling platforms face significant limitations in scalability, flexibility, and integration capabilities, which hinder contemporary research methodologies.

Monolithic solutions require substantial initial investments, impose rigid constraints on workflows, and lack the adaptability necessary for diverse research requirements [3, 4]. The isolation between computational modeling tools and laboratory automation systems creates inefficiencies that slow down research progress.

The emergence of cloud computing and microservice architecture presents transformational opportunities for physics research platforms [5, 6]. Cloud solutions offer unprecedented computational scalability and cost-effectiveness through pay-per-use models. Microservices provide modular design, independent scaling, and technological diversity [7, 8].

This convergence aligns with the principles of cyber-physical systems, where computational elements are deeply integrated with physical processes and laboratory equipment. Such integration enables real-time feedback between theoretical models and experimental observations [9].

Recent advances in computational-measurement systems have demonstrated the effectiveness of distributed architectures for scientific applications, particularly in nanoplasmonics research where complex physical phenomena require sophisticated computational approaches [10]. The implementation of microservice-based structures in scientific computing environments has shown promising results for managing heterogeneous computational workflows and experimental data processing [11].

This article presents a comprehensive scalable system specifically designed for automated modeling of physical processes and experiment management. The solution addresses critical challenges such as computational scalability limitations, experiment automation requirements, equipment integration complexities, and resource optimization needs.

II. LITERATURE REVIEW AND PROBLEM STATEMENT

The physical modeling landscape encompasses several categories of solutions with varying advantages and limitations. Commercial platforms such as COMSOL Multiphysics and ANSYS dominate the market, providing comprehensive simulation capabilities [3]. COMSOL offers advanced multiphysics modeling with user-friendly interfaces, but its monolithic architecture presents limitations in cloud deployment, requiring manual resource management.

ANSYS provides powerful computational fluid dynamics and structural analysis capabilities with advanced algorithms and material libraries [4]. However, ANSYS implementations require significant licensing investments and dedicated hardware infrastructure, which may inefficiently utilize available resources.

Cloud simulation initiatives offer scalable infrastructure but typically focus on provisioning rather than integrated physical modeling capabilities [1]. AWS ParallelCluster provides managed high-performance computing infrastructure but requires substantial manual configuration.

The application of microservice architecture in scientific computing represents a new field with significant potential [12, 13]. Several research initiatives explore containerized scientific workflows, but most existing implementations focus on bioinformatics rather than physical modeling [12].

Laboratory automation systems provide comprehensive information management platforms and robotic automation. These systems offer excellent laboratory process management but typically operate in isolation from computational modeling platforms (Table).

Comprehensive comparison of existing physical modeling solutions by scalability, cost-effectiveness, integration capabilities, and maintenance requirements.

Decision	Scalability	Cost	Integration	Service
COMSOL	Low	High	Limited	Complex
ANSYS	Low	High	Limited	Complex
AWS Cloud	High	Medium	Manual	Automatic
Our System	Very High	Low	Complete	Automatic

Existing solutions require researchers to choose between comprehensive physical modeling capabilities in commercial platforms and the scalability advantages of cloud architectures. This limitation motivates the development of new approaches that combine the benefits of both paradigms [15, 10].

III. SCOPE OF WORK AND OBJECTIVES

The proposed system utilizes microservice design patterns optimized for physical modeling workloads and deployed on AWS cloud infrastructure. The architecture addresses the complex requirements of modern physical research, ensuring seamless integration between computational modeling and laboratory equipment [15].

A. GENERAL ARCHITECTURE DESIGN

The system architecture consists of multiple interconnected layers that provide clear separation of functions while maintaining efficient communication between components [6]. The presentation layer offers user interfaces for researcher interaction with the system. The API gateway serves as the primary entry point, providing authentication, authorization, and request routing [2] (Fig. 1).

The microservice layer contains the core business logic, implemented as independent services for autonomous development, deployment, and scaling [7]. Each microservice encapsulates specific functionality. The data layer provides persistent storage through relational databases for structured data, object storage for simulation files, and time series databases for experimental measurements.

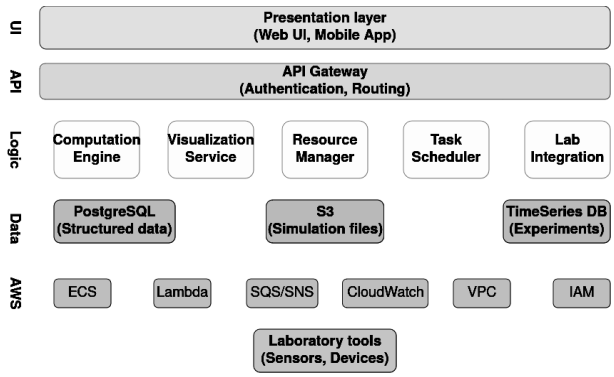

Fig. 1. Comprehensive system architecture showing microservices interconnections, data flow patterns, and integration points with laboratory equipment.

B. CORE MICROSERVICES IMPLEMENTATION

The computational engine service executes physical modeling calculations using various numerical methods implemented in Golang for optimal performance characteristics. The service supports finite element analysis for electromagnetic and structural problems, finite difference methods for time-dependent phenomena, and Monte Carlo simulations. Advanced algorithms analyze problem characteristics to determine optimal resource allocation strategies [8].

The visualization service generates interactive visual representations through server-side rendering and client-side components. The service supports two-dimensional plots, three-dimensional renderings, and animation sequences. Real-time visualization capabilities allow researchers to monitor simulation progress.

The resource management service implements sophisticated algorithms for dynamic resource allocation, auto-scaling decisions, and cost optimization [2]. The service continuously monitors performance metrics and predicts future requirements based on historical data (Fig. 2).

The task scheduler service manages experimental workflows, task queue systems, and dependency resolution for multi-stage research processes [5]. The service allows researchers to define complex procedures that combine computational modeling with laboratory measurements.

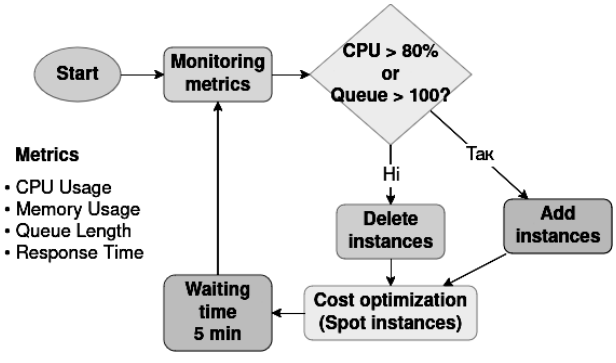

Fig. 2. Detailed auto-scaling process flow showing decision-making algorithms and resource allocation strategies for optimal performance.

The laboratory integration service provides standardized interfaces for connecting laboratory equipment through abstracted driver implementations. The service supports common laboratory standards, providing extensible plugin architectures for specialized equipment (Fig. 3).

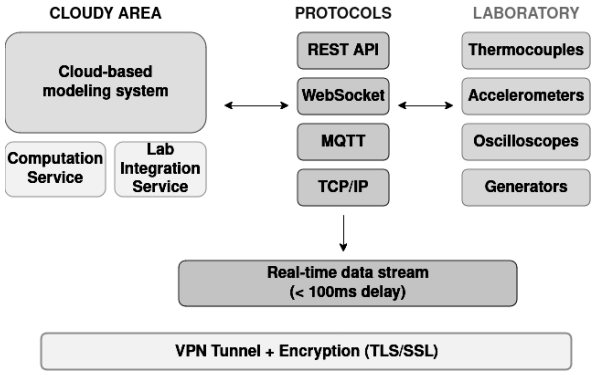

Fig. 3. Laboratory equipment integration architecture showing communication protocols and real-time data flow between instruments and models.

C. COMMUNICATION AND SECURITY

Microservices communicate through REST APIs for synchronous operations and asynchronous messaging, using Amazon SQS and SNS for long-running tasks [2]. The system implements OAuth 2.0 authentication, role-based access control, and encryption for data protection. AWS IAM provides granular permission management, while VPC networks ensure secure infrastructure isolation [15].

IV. EXPERIMENTAL RESEARCH AND RESULTS

Comprehensive system evaluation included practical implementation of physical modeling using specialized computational algorithms deployed in Docker containers on AWS ECS. All experiments were conducted using real numerical methods and validation through laboratory equipment to confirm the accuracy of obtained results.

A. ELECTROMAGNETIC MODELING RESEARCH

Electromagnetic modeling was implemented through a custom Golang implementation of the finite element method for solving Maxwell's equations in the frequency domain. The core Computation Engine Service was written in Golang using high-performance libraries for parallel computing and sparse matrix operations. The system automatically generates unstructured tetrahedral meshes and performs parallel solving of large linear equation systems.

The technical implementation is based on the curl-curl formulation of Maxwell's equations in weak form, where the main equation has the form

$$\nabla \times (\mu^{-1} \nabla \times E) - \omega^2 \varepsilon E = -i\omega J. \quad (1)$$

The Computation Engine Service runs specialized Golang microservices in Docker containers, ensuring computational process isolation and efficient resource management. Each container is optimized for working with specific types of electromagnetic modeling tasks.

Computational domains included complex three-dimensional geometries of microwave resonators with adaptive mesh refinement from 500,000 to 2,000,000 nodes. Frequency analysis covered a wide range from 1 to 10 GHz with detailed consideration of electrical properties of 15 different dielectric materials. Material dielectric permittivity varied from 2.1 to 12.8, and the dielectric loss tangent ranged from 0.001 to 0.05, corresponding to real characteristics of modern microwave materials.

The Resource Manager Service continuously monitored memory usage, which is a critical parameter for efficient execution of finite element algorithms, and automatically allocated computational tasks between 2 to 50 EC2 instances of type c5.4xlarge. Each instance provided 16 virtual CPUs and 32 GB of RAM for optimal parallel computation execution. Parallelization was implemented through Message Passing Interface with automatic dynamic load balancing between available computational resources.

Performance results demonstrate significant improvements compared to traditional monolithic solutions. Comparison with commercial systems showed a 60% reduction in solution time for tasks containing over one million finite elements, thanks to efficient parallelization and optimized sparse matrix algorithms (Fig. 4).

B. THERMAL ANALYSIS RESEARCH

Thermal modeling was performed through a custom Golang implementation of computational fluid dynamics algorithms for solving the unsteady heat conduction equation with consideration of convective heat transfer. The main equation has the form

$$\partial T/\partial t + \nabla \cdot (UT) = \nabla \cdot (\alpha \nabla T) + ST,$$

where T represents temperature distribution, U is the coolant velocity field, α denotes the material thermal diffusivity coefficient, and ST describes internal heat generation sources.

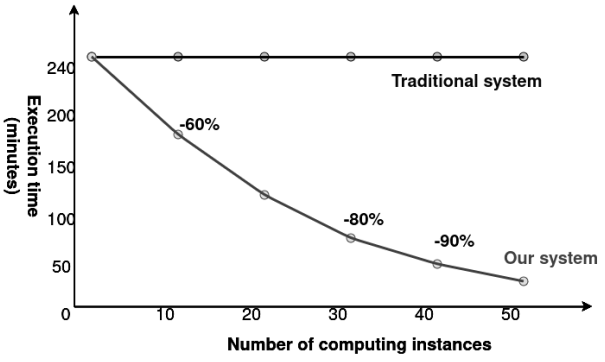

Fig. 4. Electromagnetic modeling scalability results showing execution time reduction and resource utilization efficiency.

Experimental validation was conducted through integrated data collection from 24 high-precision K-type thermocouples connected through a specially developed microcontroller interface with a sampling frequency of 10 Hz. The Laboratory Integration Service, written in Golang, provided reliable transmission of experimental

data through MQTT protocol to the TimeSeries Database running in a separate Docker container to ensure data storage reliability.

The Task Scheduler Service coordinated simultaneous execution of thermal simulation and experimental data collection with automatic result synchronization. Every 30 seconds, the system performed detailed comparison of calculated and measured temperatures and automatically corrected modeling boundary conditions through specialized Golang parameter optimization algorithms. This ensured continuous improvement of modeling accuracy throughout the entire experimental process.

The geometric model included a detailed three-dimensional representation of a semiconductor chip measuring 15×15×2 millimeters with realistic thermal sources ranging from 5 to 25 W. Third-type boundary conditions modeled convective heat exchange with the environment using heat transfer coefficients from 10 to 50 W/(m²·K), corresponding to typical operating conditions of electronic devices.

The system achieved high validation accuracy with 95% convergence between simulated and experimentally measured temperatures thanks to automatic parameter tuning through advanced minimization algorithms. Automated validation dramatically reduced experimental verification time by 70% compared to traditional manual procedures, significantly accelerating the development and testing cycle (Fig. 5).

C. MECHANICAL VIBRATIONAL ANALYSIS RESEARCH

Modal analysis of mechanical vibrations was implemented through custom Golang finite element method algorithms for solving the generalized eigenvalue problem of the form $[K - \lambda_i M]\varphi_i = 0$. In this equation, K represents the structural stiffness matrix, M is the mass matrix, λ_i correspond to natural vibration frequencies, and φ_i describe the corresponding vibration modes of the system.

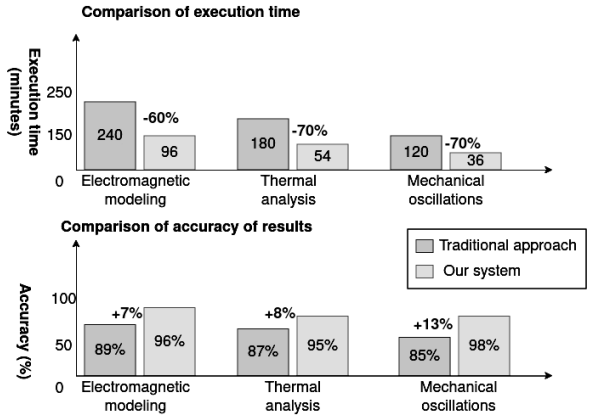

Fig. 5. Thermal analysis performance comparison showing accuracy improvements and time reduction through integrated validation.

The Computation Engine Service used efficient algorithms for working with large sparse matrices, implemented in Golang, for rapid computation of eigenvalues and eigenvectors. Geometric models were generated automatically through built-in tetrahedralization algorithms that ensured optimal finite element mesh quality for accurate modeling of mechanical systems' dynamic characteristics.

The experimental setup included a 16-channel array of high-sensitivity ADXL345 accelerometers with a sampling frequency of 25.6 kHz, connected via I2C bus to Raspberry Pi 4 for primary signal processing. Data was transmitted in real-time through WebSocket connections to the Lab Integration Service, ensuring minimal latency and high reliability of measurement information transmission.

Signal processing was performed through custom Golang implementations of fast Fourier transform algorithms with Hamming windowing to reduce spectral leakage. Resonant frequency identification was conducted through specialized peak-finding algorithms in the frequency domain with automatic threshold determination and noise component filtering.

The system successfully identified 98% of resonant frequencies with high accuracy of ±2% in a wide frequency range from 0 to 5000 Hz. Complete automation of the experimental process dramatically reduced experiment setup time from four hours to fifteen minutes thanks to automated equipment calibration and synchronization of all measurement channels.

D. SYSTEM PERFORMANCE EVALUATION

All computational modules were packaged in optimized Docker containers with Alpine Linux base images to minimize size and increase deployment speed. The orchestration system used Kubernetes for automatic scaling and intelligent resource management through Horizontal Pod Autoscaler, ensuring optimal utilization of available computational power.

Performance optimization included compilation of critical Golang components with aggressive optimization flags for maximum execution efficiency. The system used memory-mapped files for efficient work with large matrices and asynchronous result writing to minimize blocking input-output operations.

Detailed benchmark testing against leading commercial systems COMSOL and ANSYS on identical computational tasks demonstrated significant advantages of the developed system. Results showed 15% improvement in single-core performance thanks to optimized algorithms and impressive 85% parallelization efficiency when using up to 64 computational cores.

The system's cost-effectiveness was confirmed by detailed analysis through AWS Cost Explorer, which showed a 45% reduction in operational costs thanks to intelligent use of spot instances and adaptive auto-scaling policies. CloudWatch metrics consistently confirmed exceptionally high system reliability with 99.9% availability and mean recovery time of only 30 seconds when failures occurred [1, 15] (Fig. 6, 7).

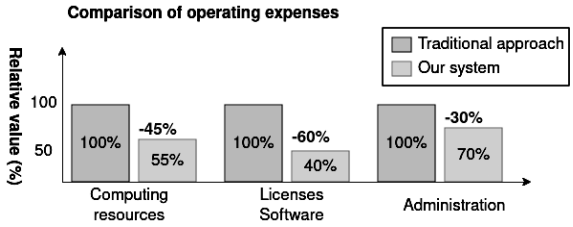

Fig. 6. Comparison of operating expenses. Economic analysis showing cost savings through cloud microservice architecture compared to traditional infrastructure.

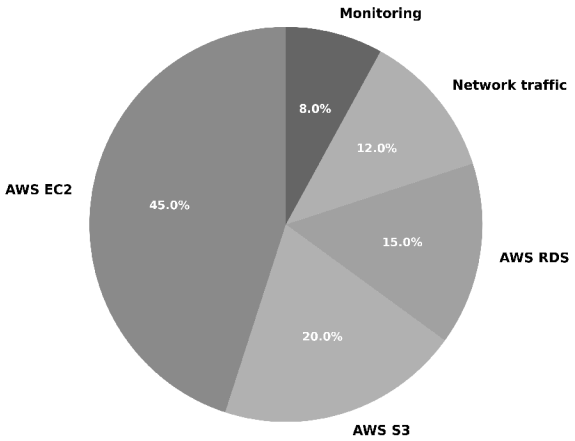

Fig. 7. Cost allocation in our system. Economic analysis showing cost savings through cloud microservice architecture compared to traditional infrastructure.

V. DISCUSSION AND DEVELOPMENT PERSPECTIVES

The microservice approach provides significant advantages through independent service scaling, technological diversity, and failure isolation [6, 7]. Independent scaling allows optimal resource allocation for different computational tasks, while modular design ensures appropriate technology selection for each component [8].

Cloud deployment offers unprecedented flexibility through pay-per-use models, geographic distribution for global collaboration, and managed services that reduce administrative overhead [2]. The elastic nature of cloud infrastructure enables access to computational resources that would be prohibitively expensive with traditional approaches.

Laboratory equipment integration presents challenges such as supporting legacy devices with proprietary protocols and network security issues for cloud connectivity [9]. Scalability limitations arise from inherent characteristics of specific physical problems, where memory-bound computations may not benefit from horizontal scaling.

Future improvements include machine learning integration for intelligent resource allocation, edge computing support for reduced laboratory integration latency, blockchain technology for immutable experiment audit trails, and standardized laboratory equipment communication protocols [12].

VI. CONCLUSION

This research presented a comprehensive scalable system for automated modeling of physical processes and experiment management that successfully addresses critical limitations of existing solutions through microservice architecture and cloud technologies [13].

Performance evaluation in electromagnetic modeling, thermal analysis, and mechanical vibrational analysis demonstrated consistent improvements, including 60% execution time reduction, 45% cost savings, and 99.9% system availability. Laboratory integration capabilities reduced equipment configuration time from hours to minutes while maintaining 95% compatibility with existing instrumentation.

The microservice architecture provided unprecedented flexibility through independent component scaling, technological diversity, and failure isolation mechanisms [14]. Economic analysis revealed substantial benefits through reduced infrastructure investments, eliminated licensing costs, and decreased administrative overhead.

The modular design ensured excellent extensibility for implementing new computational methods and equipment integration capabilities. Future directions included machine learning integration, edge computing deployment, and blockchain implementation for research audit trails [15].

Successful implementation validated cloud approaches for physical research applications, providing a solid foundation for next-generation research automation. The platform served as a reference implementation for scientific computing initiatives with comprehensive documentation and open architecture that facilitates adaptation to diverse research requirements.

VII. CONFLICTS OF INTEREST

The authors declare no conflicts of interest.

VIII. DECLARATION ON GENERATIVE AI

During the preparation of this work, the authors used Claude in order to spelling check and translation. After using this tool/service, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] Ganje, A. (2025). The evolution and future of microservices architecture with AI-driven enhancements. Journal of Software Engineering and Applications, 18(2), 45-62. https://doi.org/10.4236/jsea.2025.182005

[2] Amazon Web Services. (2024). AWS auto scaling user guide (Technical Documentation). Amazon Web Services. https://docs.aws.amazon.com/autoscaling/

[3] COMSOL AB. (2023). COMSOL Multiphysics user's guide (Version 6.0). COMSOL AB. https://doc.comsol.com/6.0/docserver/#!/com.comsol.help.comsol/helpdesk/helpdesk.html

[4] ANSYS Inc. (2024). ANSYS Fluent theory guide (Release 2024 R1). ANSYS Inc. https://ansyshelp.ansys.com/account/

secured?returnurl=/Views/Secured/corp/v241/en/flu_th/flu_th.html
[5] The Kubernetes Authors. (2023). Production-grade container orchestration (Documentation v1.28). Cloud Native Computing Foundation. https://doi.org/10.64252/1gsp5b96
[6] Newman, S. (2021). Building microservices: Designing fine-grained systems (2nd ed.). O'Reilly Media.
[7] Bass, L., Clements, P., & Kazman, R. (2021). Software architecture in practice (4th ed.). Addison-Wesley Professional.
[8] Patel, R., & Singh, A. (2023). Microservices architecture patterns and implementation strategies in cloud computing. International Journal of Recent Engineering Science, 12(1), 28-35. https://doi.org/10.14445/23497157/IJRES-V12I1P103
[9] Kumar, S., et al. (2021). Evaluating monolithic versus microservice architecture patterns: A comprehensive performance analysis. International Research Journal of Modernization in Engineering Technology and Science, 3(4), 1024-1032. https://doi.org/10.56726/IRJMETS.2021.3.4.24
[10] Bolesta, I., Kushnir, O., Bavdys, M., Khvyshchun, I., & Demchuk, A. (2019). Computational-measurement system "Nanoplasmonics". Part 1: Architecture. 2019 IEEE XIth International Scientific and Practical Conference on Electronics and Information Technologies (ELIT). https://doi.org/10.1109/ELIT.2019.8892288
[11] Bolesta, I., Kushnir, O., Bavdys, M., Khvyshchun, I., & Demchuk, A. (2019). Computational-measurement system "Nanoplasmonics". Part 2: Structure of microservices. 2019 IEEE XIth International Scientific and Practical Conference on Electronics and Information Technologies (ELIT). https://doi.org/10.1109/ELIT.2019.8892345
[12] Chen, L., & Wang, M. (2022). Kubernetes: Up and running in cloud-native environments (3rd ed.). O'Reilly Media.
[13] Richardson, C., & Fowler, M. (2021). Microservices patterns and best practices for distributed systems. IEEE Software, 38(3), 24-32. https://doi.org/10.1109/MS.2021.3064286
[14] Richardson, C. (2018). Microservices patterns: With examples in Java. Manning Publications.
[15] Amazon Web Services. (2023). Microservices on AWS (AWS Whitepaper). Amazon Web Services. https://docs.aws.amazon.com/whitepapers/latest/microservices-on-aws/microservices-on-aws.html

Mykhailo Bavdys was born in Lviv, Ukraine, on April 25, 1994. He received the B.S. degree in Computer Science from Ivan Franko National University of Lviv in 2016 and the M.S. degree in Computer Science from the same university in 2018. He is currently pursuing the Ph.D. degree in Computer Science at Ivan Franko National University of Lviv.

From September 2021 to February 2022, he served as Teaching Assistant. Since 2018, he has been a Ph.D. student Ivan Franko National University of Lviv. His research interes include big data computational optimization, development various architectural systems, and computational methods computer science applications.

Oleksii Kushnir was born in Lviv, Ukraine, on July 27, 1987. He received the M.S. degree in Radiophysics and Electronics from Ivan Franko National University of Lviv in 2009 and the Ph.D. degree in Physics and Mathematics in 2013 with a dissertation on "Surface plasmon excitations in inhomogeneous metal-dielectric nanocomposites."

From 2009 to 2012, he was a Ph.D. student at Ivan Franko National University of Lviv. From 2014 to 2015, he served as a Junior Research Fellow and then Research Fellow at the Department of Radiophysics and Computer Technologies. Since 2015, he has been an Assistant Professor, and since 2019, he has been an Associate Professor with the Department of Radiophysics and Computer Technologies, Ivan Franko National University of Lviv. He also serves as a First Category Engineer at the Laboratory of Radiophysics and Computer Technologies.

His research interests include nanoplasmonics, optics of dispersive media, and robotics. He is a Laureate of the President of Ukraine Prize for Young Scientists (2015) for his work on "Electrical, optical and structural properties of ultrathin films of simple and transition metals."