Autonomous Liquid-handling Robotics Scripting for Accessible and Responsible Protein Engineering

Yuan Gao[1,4,#], Yizhou Luo[1,5,#], Wenzhuo Li[1,6,#], Yunquan Lan[1], Han Jiang[1], Yongcan Chen[1], Xiao Yi[1], Boyang Li[7], Hamid Alinejad-Rokny[3], Teng Wang[1], Lihao Fu[1,*], Min Yang[1,2,*], Tong Si[1,4,*]

[1]State Key Laboratory for Quantitative Synthetic Biology, Shenzhen Institute of Synthetic Biology, Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, China

[2]Shenzhen Key Laboratory for High Performance Data Mining, Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, China

[3]UNSW BioMedical Machine Learning Lab, School of Biomedical Engineering, UNSW Sydney, 2052, NSW, Australia

[4]University of Chinese Academy of Sciences, Beijing 100049, China

[5]State Key Laboratory of Pharmaceutical Biotechnology, Institute of Functional Biomolecules, School of Life Sciences, Nanjing University, Nanjing 210023, China

[6]College of Engineering, Southern University of Science and Technology, Shenzhen 518055, China

[7]School of Interdisciplinary Science, Beijing Institute of Technology, Beijing 100081, China

# These authors contributed equally

* To whom correspondence should be addressed

tong.si@siat.ac.cn

min.yang@siat.ac.cn

lh.fu@siat.ac.cn

Abstract

Laboratory automation enhances experimental throughput and reproducibility, yet widespread adoption is constrained by the expertise required for robotic programming. Here, we introduce LabscriptAI, a multi-agent framework that enables large language models to autonomously generate and validate executable Python scripts for protein engineering automation. Across a 55-task benchmark spanning four difficulty levels and multiple liquid-handling platforms, LabscriptAI achieved high success rates and outperformed both direct large language model baselines and a commercial solution. LabscriptAI automated cell-free protein synthesis and characterization of 298 green fluorescent protein (GFP) variants designed by 53 teams from five countries in a student challenge; the top variant achieved functional performance comparable to an extensively optimized benchmark while exploring distinct sequence space. Furthermore, LabscriptAI orchestrated distributed automation across a biofoundry and fume hood-enclosed systems to engineer enzyme variants utilizing formaldehyde, a sustainable but hazardous substrate, and identified a double mutant with sevenfold increase in catalytic efficiency. The platform implements rigorous safety measures, including biosecurity screening, physical containment, and human-in-the-loop oversight, to safeguard autonomous protein engineering. LabscriptAI democratizes laboratory automation by eliminating programming barriers while promoting responsible research practices.

Introduction

Laboratory automation is transforming experimental biology by enhancing throughput, reproducibility, and standardization, particularly in synthetic biology[1-4]. While advanced biofoundries provide centralized high-throughput robotics for large-scale projects, accessible platforms such as Opentrons have democratized automation through cost-effective and open-source design[5], with community projects like AssemblyTron demonstrating DNA assembly automation[6]. Despite these advances, scripting liquid-handling robots remains a persistent bottleneck[7,8]. Protocol generation and adaptation demand substantial programming expertise, compounded by platform heterogeneity across Opentrons, Tecan, Hamilton, and other systems, each with distinct programming interfaces.

Large language models (LLMs) present opportunities to bridge this accessibility gap, albeit with limited success. Inagaki and colleagues demonstrated that GPT-4 could generate functional OT-2 scripts from natural language instructions using simulator feedback[9]; however, this approach struggled with complex

protocols, occasionally circumventing programming errors rather than implementing correct experimental steps. Hardware-agnostic interfaces like PyLabRobot[10] have simplified multi-platform control, yet early integration with GPT-3.5 remains restricted to basic user cases. Coscientist[11] and ChemCrow[12] demonstrated large language model–driven autonomous chemical experimentation but were confined to non-biological tasks. Reliable application of large language models to synthetic biology automation faces several fundamental challenges: general-purpose large language models lack domain-specific knowledge of biological protocols, reagent handling, and safety constraints; they struggle with managing hardware-specific variations in pipetting configurations, timing dependencies, and deck layouts; and they lack structured script validation mechanisms for iterative generation and debugging. Furthermore, reproducible protocols demand bidirectional integration with established databases and standards, a capability that is absent from existing large language model–based systems. These limitations, combined with biosafety and biosecurity concerns[13,14], prevent deployment of large language model–generated scripts in real-world applications, where accuracy, reproducibility, and accountability are essential.

Here we present LabscriptAI (Figure 1, https://labscriptai.cn/), a multi-agent framework that translates natural language descriptions of biological experiments into validated robotic code across heterogeneous liquid-handling platforms while enforcing domain-specific and safety constraints. Building on CodeAct[15], LLM agents iteratively produce executable Python code and analyze execution outcomes to refine protocols rather than generating static scripts. When errors occur, the Precise Refactoring Engine generates targeted corrections instead of regenerating entire scripts, achieving faster convergence on validated protocols. LabscriptAI integrates a dynamic knowledge base that stores validated protocols, hardware specifications, and controlled terminology. Human-in-the-loop oversight maintains alignment between natural language procedures and Python scripts through synchronized updates. We validate LabscriptAI across experiments of increasing complexity, including cross-platform plate reader calibration, cell-free expression and characterization of green fluorescent protein (GFP) variants from a protein engineering competition, and enzyme screening involving hazardous reagents, demonstrating that accessible natural language interfaces can support responsible laboratory automation.

Results

System Architecture

LabscriptAI comprises three integrated components (Figure 1): a web-based user interface, a code-generation and self-correction backend, and a storage layer containing both a long-term knowledge base and dynamic project-specific data. Together, these components translate high-level descriptions of biological experiments into executable automation code (Supplementary Note 1). The multi-agent system architecture is orchestrated using LangGraph[16], with the detailed prompt-engineering strategy provided in Supplementary Note 2.

The web interface enables experimental goal specification for biological workflows through natural language input and hardware configuration (Extended Data Fig. 1). Real-time visualization displays dynamic generation and modification of both standard operating procedures in natural language and executable Python scripts, allowing researchers to monitor the transformation from high-level biological concepts to liquid-handling instructions (Supplementary Note 1). The system also accepts pre-existing standard operating procedures and robotic scripts, supporting both protocol adaptation and code refinement workflows (Extended Data Fig. 1).

The Task Planning Agent coordinates workflow processing (Figure 1), initiating a human-in-the-loop dialogue to first validate and refine natural language SOPs. Once finalized, the agent determines the appropriate code generation strategy: adaptation of existing scripts for new hardware or user requirements, or de novo code generation. This hierarchical processing ensures accurate translation of experimental goals before code generation.

The Code Graph backend manages iterative code refinement through a multi-stage pipeline (Figure 1). Initial Python script generation leverages the knowledge base resources and incorporates project-specific context. Generated scripts undergo validation using platform-appropriate simulators selected based on provided hardware configurations. Successful simulations yield deployment-ready code, while validation failures activate the code-modification pipeline. The Precise Refactoring Engine analyzes error messages and generates targeted SEARCH/REPLACE operations (Extended Data Fig. 2), preserving validated code segments rather than regenerating entire scripts. This iterative cycle continues until validation succeeds or retry limits are reached.

The storage architecture separates persistent domain knowledge from dynamic project data (Figure 1). The

knowledge base maintains validated biological protocols, hardware specifications, application programming interface (API) definitions, and controlled terminology, while workspace storage manages session-specific data, including simulation results and generation logs. Successful validations are promoted from workspace to knowledge base, creating a continuous-learning mechanism that improves system performance over time.

These architectural components provide three key advances: adaptive task planning responsive to varying input scenarios, efficient code refinement through precise patches, and continuous learning through systematic feedback integration. System transparency, from detailed iteration logs to real-time visualization, facilitates interactive debugging and user understanding, enabling researchers without programming expertise to generate robust automation protocols while ensuring accuracy through multi-stage validation.

Benchmark Performance

To evaluate the ability of large language models to generate liquid-handling scripts, we developed a benchmark (Supplementary Dataset 1) adapted and expanded from Inagaki et al.[6] to comprise 55 tasks stratified across four difficulty levels: Easy (n = 13, basic liquid handling), Medium (n = 20, multi-step workflows), Hard (n = 13, complex error-prone operations), and Expert (n = 9, multi-module coordination). Performance was compared against direct LLM implementations (GPT-5, Claude 4, DeepSeek V3.2, Gemini-2.5 Pro) and OpentronsAI, a commercial LLM–based solution (Table 1).

LabscriptAI achieved 89.1% overall success in generating simulation-passing scripts, outperforming all baselines (Table 1). Direct LLM implementations achieved 20–45% success, with performance degrading sharply at higher difficulty levels (zero success on Expert-level tasks for GPT-5 and Claude 4). Utilizing domain-specific fine-tuning, OpentronsAI reached 76.3% overall success, 12.8% below our open-source platform. LabscriptAI achieved 100% success on Easy and Medium tasks and maintained 76.9% and 66.7% success on Hard and Expert tasks, respectively, demonstrating robustness across complexity levels.

Ablation studies quantified the contribution of individual architectural components (Table 2). Replacing the Precise Refactoring Engine with full-script regeneration decreased success to 80.0% while increasing average iterations from 3.07 to 4.38 and token consumption by 40%, demonstrating the efficiency benefits of targeted corrections. Removing domain-specific knowledge retrieval reduced success to 76.3% (12.8% reduction). Disabling iterative refinement reduced success to 18.2% (70.9% reduction), falling below direct LLM implementations (Table 1) despite access to domain knowledge. This dramatic degradation likely resulted from context overload: the combination of extensive knowledge retrieval and detailed agent prompts

increased susceptibility to hardware constraint violations and API version mismatches during initial code generation.

Error-pattern analysis revealed distinct failure modes (Extended Data Fig. 3). Direct LLM implementations exhibited hardware-constraint violations, API syntax errors, and hallucinated hardware capabilities. OpentronsAI showed improved API compliance but encountered logical errors in complex pipetting sequences. LabscriptAI failures involved edge cases in deck configuration and multi-dispense logic, representing refinement opportunities rather than architectural limitations.

During benchmark evaluation, LLM occasionally refused to generate protocols and scripts for certain workflows, citing biosafety and biosecurity concerns (Table 1). These refusals underscore the importance of balancing risk mitigation with legitimate use of large language models in scientific automation. LabscriptAI implements a multi-layered safety framework to prevent inappropriate refusals: (1) protocol generation with explicit human oversight and approval, (2) biosecurity screening for DNA synthesis, (3) physical containment for hazardous workflows, and (4) procedural safety instructions for research personnel. These safety-rail approaches (Figure 1) enable responsible robotic code generation and execution without sacrificing access to high-risk, high-value experiments.

Cross-platform Validation of Community Plate Reader Calibration Protocol

For experimental validation of cross-platform compatibility of LabscriptAI-generated scripts, the iGEM Plate Reader Fluorescence Calibration protocol[17] was selected as a proof-of-concept challenge. This community standard, hosted on the protocols.io repository, enables conversion of arbitrary fluorescence units to standardized fluorescein equivalents for reproducible comparison across laboratories[17]. A collaboration agent was implemented (Extended Data Fig. 4), interfacing with protocols.io to retrieve the latest protocol version and deposit generated scripts via API. The natural language input to LabscriptAI consisted of the request "Prepare a fluorescein standard curve following the iGEM protocol" (Figure 2A). The Task Planning Agent generated a structured procedure comprising 10 μM fluorescein stock preparation, serial 2-fold dilutions across 7 concentrations, and triplicate plating in 96-well black plates with 200 μL final volumes (Figure 2B).

Platform-specific scripts were generated for four liquid-handling systems (Figure 2C): Opentrons OT-2 and Flex (native Python API[18], Supplementary Fig. 1), Hamilton Vantage (PyLabRobot[10]), and Tecan Fluent. For Tecan compatibility, pyFluent was developed in this study to provide a standardized Python API (Extended

Data Fig. 5). The Precise Refactoring Engine generated targeted modifications for hardware-specific constraints, including pipette tip compatibility, deck layout differences, and liquid-handling protocols. Experimental validation produced coefficients of variation below 5% for triplicates, with linear regression R^2 values of standard curves exceeding 0.999 across all systems (Figure 2C). These results demonstrate LabscriptAI's capability to translate natural language protocols into executable, platform-optimized scripts.

Robotic Screening of GFP Variants for Student Competition

To validate LabscriptAI in a biological context, a cell-free protein synthesis (CFPS) workflow was automated to express GFP using the pET28 vector on Opentrons Flex (Figure 3A), enabling rapid, transformation-free fluorescence profiling. The natural language input "Set up cell-free GFP expression reactions with varying template concentrations" was coupled with the commercial kit protocol, generating automated scripts for master mix preparation, template dilution series, and 37°C incubation (Figure 3B, Supplementary Fig. 2). The parent avGFP from the jellyfish *Aequorea victoria*[19] (Supplementary Table 1) demonstrated robust and reproducible expression: CFPS reactions in 20 μL volume yielded strong fluorescence signals across template concentrations from 0.25-7.50 ng μL^{-1}, with 5.0 ng μL^{-1} producing 0.130 ± 0.001 μM fluorescein equivalents after 3-hour incubation (Figure 3C).

LabscriptAI was subsequently applied to the Critical Assessment of Protein Engineering (CAPE) competition[20]. Modeled after the Critical Assessment of Structure Prediction (CASP) competition for protein structure prediction[21], student teams develop computational models, submit designs for centralized characterization, and receive standardized performance data (Figure 3D). This framework promotes protein engineering by providing experimental feedback on cloud labs[22] for model benchmarking and refinement. The 2025 CAPE challenge focused on engineering GFP variants with enhanced fluorescence (Figure 3D): Teams received 147,950 sequence-function data points from previous studies[23,24] for model training, then submitted six designs each containing up to six mutations in avGFP, resulting in 298 dereplicated sequences from 53 participating teams across five countries. When compared with avGFP variants in FPbase[25], a community-curated database cataloging photophysical properties of over 800 fluorescent proteins, student designs explored distinct sequence space (Figure 3E). Design strategies ranged from structure-guided methods to multi-stage screening (Extended Data Fig. 6A), and were implemented using diverse machine learning frameworks with prevalent transformer-based architectures including ESM[26] and Saprot[27] (Extended Data Fig. 6B). All submitted designs, computational strategies, and trained models were made openly accessible to promote community learning and method development.

For experimental validation, all sequences underwent biosecurity screening using the open-source IBBIS Common Mechanism[28] prior to synthesis (Figure 3D), ensuring submitted variants posed no dual-use concerns while supporting open collaboration. To accommodate competition scale, LabscriptAI adapted the CFPS protocols for multiple plates, optimizing master mix volumes and scheduling sequential measurements (Supplementary Fig. 2). GFP variants were characterized under standardized conditions, with fluorescence values normalized against batch-specific fluorescein calibration curves. Among 298 characterized variants, 53 demonstrated improved fluorescence values relative to avGFP under CFPS conditions (Supplementary Table 2). Single and double mutants showed limited improvements, while improved variant proportions increased substantially at three or more mutations (Figure 3F, Supplementary Table 2). This distribution pattern suggested that multiple coordinated mutations were required for substantial fluorescence enhancement, necessitating algorithm-assisted exploration of expansive sequence space. The top-performing variant Var1, containing five mutations (Supplementary Table 1) and exhibiting 8.11-fold improvement over avGFP in CFPS reactions (Figure 3F), was designed by multi-model ensemble predictions from six independent models with hierarchical filtering to converge on final candidates (Extended Data Fig. 6C). This provided a biologically meaningful demonstration that community-designed sequences can yield substantially enhanced fluorescent proteins suitable for further biophysical characterization.

Benchmark Community-Designed GFP Variants with Public Database

To rigorously evaluate CAPE-winning variants, quantitative comparison was performed against FPbase benchmarks. While CFPS provides screening data, fluorescence measurement conflates expression level and molecular brightness[29], so that protein purification is necessary for accurate photophysical assessment. LabscriptAI was therefore challenged with the request "Quantitatively compare CAPE GFP variants with FPbase bright GFP variants in purified form". The collaboration agent was extended to interface with FPbase (Extended Data Fig. 4), retrieving database entries via API and identifying superfolder GFP (sfGFP, Supplementary Table 1) as a high-performance benchmark variant of avGFP[30]. Metric specifications were extracted by the agent from the FPbase glossary and relevant literature[31] and transmitted to LabscriptAI for protocol development.

LabscriptAI first generated updated CFPS scripts with magnetic bead purification incorporating stepwise imidazole washes (Supplementary Fig. 2), Bradford assay for protein concentration normalization (Supplementary Fig. 3), and spectral collection for photophysical characterization. However, SDS-PAGE

analysis revealed insufficient purity from microtiter cultures using bead-based purification (Extended Data Fig. 7A). Therefore, conventional two-step column chromatography was employed to manually purify selected variants from flask cultures (Extended Data Fig. 7B).

The workflow concluded with automated data deposition back to FPbase through the collaboration agent (Extended Data Fig. 4). The system formatted characterization data according to FPbase standards, including excitation and emission spectra (Extended Data Fig. 7B), quantum yields, brightness, and extinction coefficients (Figure 3G). Measured values for avGFP and sfGFP differed numerically from those reported in FPbase, reflecting different assay conditions. Under our conditions, Var1 exhibited slightly higher quantum yield (0.80 vs 0.65) and molecular brightness (49.50 vs 41.33) than sfGFP, indicating that a crowd-designed variant can match or exceed the performance of an extensively optimized benchmark.

Enzyme Variant Screening Involving Hazardous Reagents

To extend LabscriptAI beyond fluorescent proteins, enzyme engineering workflows involving hazardous reagents were evaluated. This application targets one-carbon biomanufacturing using a glycolaldehyde synthase (GALS, Supplementary Table 1) from *Pseudomonas putida*, which catalyzes formaldehyde (FALD) condensation into glycolaldehyde (GALD) and dihydroxyacetone (DHA)[32] (Figure 4A). Robotic automation mitigates researcher exposure to this carcinogenic chemical while enabling high-throughput screening.

Rapid spectrophotometric assays have previously been developed for product quantification of GALD and DHA from whole-cell biotransformation[33]. LabscriptAI was challenged with two key advances: first, distributing the workflow across the biofoundry (DNA and strain construction, Extended Data Fig. 8) and a fume hood-enclosed Opentrons OT-2 (FALD reaction and analysis, Figure 4B), connected by human-mediated transfers; and second, performing combinatorial mutagenesis at the V281 and F282 residues (Figure 4A), where single mutations had previously failed to yield improvements[33]. The natural language request was "Create combinatorial double-site mutagenesis library at V281 and F282, screen for enhanced FALD biotransformation activity using distributed automation".

LabscriptAI orchestrated a multi-platform workflow (Figure 4B). The biofoundry workflow included: (1) automated library construction using Tecan Fluent with primers encoding 19 amino acid substitutions per position (38 single mutants and 361 double mutants); (2) clone picking and sequence verification; (3) automated cell cultivation, induction, and washing. Subsequently, induced cells were transferred to the fume hood-enclosed OT-2, and physically contained workflows included: (4) automated preparation of FALD-

containing assay buffer (1.5 g L□¹ FALD in 50 mM PBS with cofactors); (5) static biotransformation at room temperature for 24 h; and (6) supernatant collection and splitting with separate derivatization for GALD and DHA quantification.

LabscriptAI generated both robotic scripts and human operator instructions delineating safety-critical steps. FALD biotransformation (Supplementary Fig. 4) and product detection (Supplementary Figs. 5, 6) were confined to the fume hood-enclosed system, minimizing researcher exposure while maintaining throughput. For inter-station transfers, explicit safety instructions specified personal protective equipment, sealed plate handling procedures, and designated transfer routes (Supplementary Figs. 4-7). This approach demonstrates how strategic workflow partitioning enhances laboratory safety and automation efficiency.

Automated screening for whole-cell FALD biotransformation (50 mM substrate, 24-hour incubation) identified several double mutants achieving improvements unattainable through single-site mutations (Figure 4C), validating the combinatorial mutagenesis strategy. Top variants were manually purified and evaluated for *in vitro* FALD conversion (66.7 mM substrate, 1 mg mL^{-1} enzyme, 2-hour incubation). Parent GALS yielded 5.3% conversion (1.52 mM GALD, 0.17 mM DHA), while W/L and W/T variants achieved 31.5% (8.39 mM GALD, 1.41 mM DHA) and 14.9% (3.30 mM GALD, 1.11 mM DHA) conversion, respectively (Figure 4D).

LabscriptAI developed automated workflows for enzyme kinetics characterization (Supplementary Fig. 7). The W/L variant achieved 3.2-fold and 7.0-fold enhancement in catalytic efficiency k_{cat}/K_m for GALD and DHA, respectively, driven primarily by increased turnover (5.3-fold and 6.1-fold k_{cat}) (Extended Data Fig. 9). Conversely, W/T variant improvements (2.2-fold and 3.1-fold catalytic efficiency) resulted from enhanced substrate binding (0.64-fold and 0.60-fold K_m) with moderate turnover increases (1.4-fold and 1.8-fold) (Extended Data Fig. 9). The W/L variant also showed superior tolerance with little inhibition at 1,500 mM FALD (45 g L□¹), whereas parent GALS and W/T variant productivity decreased above 500 mM (15 g L□¹) (Figure 4E). This substrate tolerance is particularly notable given FALD's well-characterized protein-denaturing properties, representing a critical advance for industrial biocatalysis applications requiring high substrate concentrations.

The collaboration agent processed enzyme data and metadata to address heterogeneous formatting requirements and access interfaces across repositories (Extended Data Fig. 4), ranging from programmatic API submissions in SABIO-RK[34] to manual web form entries in STRENDA[35], and from standardized XML-

based SBML documents in EnzymeML[36] to spreadsheet templates in BioCatNet[37]. The framework adapted experimental data to each database's metadata schemas, controlled vocabularies, and submission protocols, eliminating laborious manual reformatting and ensuring cross-laboratory standardization.

Discussion

Laboratory automation is central to synthetic biology and protein engineering, where high-throughput, reproducible workflows are required to explore large sequence spaces and systematically optimize biological function[1-4]. Liquid-handling stations, biofoundries, and distributed robotic platforms can scale experiments far beyond manual capacity, but this potential is constrained by the difficulty of translating complex experimental designs into robust code; scripting and maintaining automation protocols remain major bottlenecks, concentrating automation in groups with substantial engineering expertise and limiting broader community participation in large-scale, data-driven biology. Existing approaches only partially address this gap. LLM-based tools can generate scripts for specific platforms and tasks[9-12], but often struggle with complex multi-step protocols, hardware heterogeneity, and reproducible debugging. Other autonomous protein engineering platforms[38-41] focus on iterative sequence optimization guided by active learning algorithms, yet are typically tied to pre-defined protocols, fixed hardware configurations, and limited safety and data-integration support. Consequently, there is still no widely accessible framework that unifies cross-platform automation, robust error handling, built-in biosafety safeguards, and deep integration with community data resources.

In this study, we introduced LabscriptAI, a multi-agent framework that integrates domain knowledge with iterative self-correction and provide natural language interfaces for programming-intensive laboratory automation. Across a 55-task benchmark stratified by difficulty, LabscriptAI achieved 89.1% success, outperforming direct LLM implementations (20–45%) and a commercial solution (76.3%). Experimental validation progressed through increasing biological and operational complexity. First, iGEM fluorescence calibration across multiple platforms demonstrated hardware-agnostic implementation. Second, characterization of 298 community-designed avGFP variants identified a leading candidate that, with five mutations, matched the strong fluorescence of sfGFP, which contains 11 mutations (Supplementary Table 2) from extensive directed evolution[30]. Finally, screening of FALD-utilizing enzyme variants yielded seven-fold enhanced catalytic efficiency and substantially improved tolerance from 500 mM to 1,500 mM substrate.

Notably, integration of LabscriptAI with the CAPE competition exemplifies democratized protein engineering, enabling students to contribute diverse computational models, collectively explore expansive sequence spaces, and benchmark designs against both peer submissions and state-of-the-art benchmark. These implementations confirmed that natural language interfaces can preserve automation precision and efficacy across diverse synthetic biology applications.

Beyond empirical performance, LabscriptAI highlights general design principles for LLM-driven automation and its safe deployment. Three architectural features were particularly important for reliable code generation: hierarchical task decomposition, which separates high-level experimental logic from platform-specific implementation so that biological procedures can be validated and refined before any code is produced; the Precise Refactoring Engine, which shows that targeted modifications are more effective than wholesale regeneration by preserving validated code segments and making debugging more interpretable; and knowledge retrieval from curated resources, which supplies domain expertise that generic LLMs lack, while separation of long-term knowledge from short-term project data enables continuous learning without compromising data integrity. In addition, structured prompts that emphasize biological intent rather than programming syntax further improved performance (Supplementary Note 2), underscoring that system architecture and interaction design must be co-optimized for robust and scalable automation. In parallel, a multi-layered safety framework establishes a template for responsible use of such systems in the wet lab. Biosecurity screening of all community-submitted sequences before synthesis, including a student-accessible Google Colab implementation (Supplementary Fig. 8), embeds safety at the design stage. Workflow partitioning confines hazardous operations, such as formaldehyde handling, to controlled environments, complemented by automated execution of high-risk steps, explicit safety instructions for manual interventions, and human-in-the-loop oversight at critical decision points. Progressive validation from computational simulation through physical execution further reduces the likelihood that unsafe or erroneous protocols reach the bench. Together, these architectural and safety layers illustrate how LLM-based automation can be engineered to strengthen, rather than erode, existing biosafety and biosecurity norms.

Bidirectional integration with community databases advanced reproducible protein engineering in line with FAIR (Findable, Accessible, Interoperable, and Reusable) principles[42-45] (Extended Data Fig. 4). By retrieving protocols from open repositories such as protocols.io, adapting them to specific hardware, and depositing validated implementations, LabscriptAI contributed to a self-reinforcing cycle of shared methods and data. Automated reconciliation of heterogeneous database formats, via API calls, file format conversions,

and structured data exports, reduced barriers to multi-database deposition that typically limit dissemination. More broadly, dynamic retrieval-augmented generation from curated community resources enabled accurate translation of biological requirements into robotic instructions, helping to establish collaborative ecosystems in which each successful protocol enhances future experimental and computational capabilities.

Several limitations indicate development priorities. First, the current implementation addresses individual liquid-handling robots; extending integration to additional processing, conditioning, and analytical devices would enable end-to-end automated workflows. Second, real-time decision-making based on intermediate results remains challenging, as the framework operates on pre-defined protocols rather than adaptive experimentation. Hardware-in-the-loop feedback systems could enable dynamic error detection and correction beyond pre-defined patterns. Third, incorporating experimental outcome data into the learning loop would facilitate protocol optimization based on historical performance metrics. Integration with electronic laboratory notebooks would provide comprehensive experimental tracking and enhance reproducibility documentation.

Looking forward, the convergence of more capable language models, broader hardware integration, and expanding community knowledge bases suggests that natural language-guided automation will increasingly become standard practice in biological laboratories. We envisage extending LabscriptAI from liquid handling to full biofoundry orchestration, coupling it with adaptive experimentation engines that update protocols in response to interim measurements, and embedding it more deeply within FAIR-compliant data infrastructures. In parallel, open-source release of LabscriptAI with built-in biosafety and biosecurity safeguards will enable community-driven extension of its capabilities while maintaining responsible use, helping to democratize protein engineering and synthetic biology for the benefit of society.

In conclusion, our results position LabscriptAI as both a practical tool for current laboratories and a conceptual blueprint for future LLM-driven automation. By coupling natural language interfaces with cross-platform code generation, multi-layered safety, and deep integration into community data ecosystems, LabscriptAI shows that automation can be made more accessible without compromising rigor or responsibility. As the underlying models, instruments, and datasets continue to evolve, we anticipate that similar frameworks will help transform laboratory practice from bespoke scripting to reproducible, shareable, and collaboratively optimized automated workflows.

References

1. Zhang, X.E. et al. Enabling technology and core theory of synthetic biology. *Sci China Life Sci* **66**, 1742–1785 (2023).
2. Hillson, N. et al. Building a global alliance of biofoundries. *Nat Commun* **10**, 2040 (2019).
3. Zhang, J. et al. Accelerating strain engineering in biofuel research via build and test automation of synthetic biology. *Curr Opin Biotechnol* **67**, 88–98 (2021).
4. Chao, R., Mishra, S., Si, T. & Zhao, H. Engineering biological systems using automated biofoundries. *Metab Eng* **42**, 98–108 (2017).
5. Chory, E.J., Gretton, D.W., DeBenedictis, E.A. & Esvelt, K.M. Enabling high-throughput biology with flexible open-source automation. *Mol Syst Biol* **17**, e9942 (2021).
6. Bryant, J.A., Jr., Kellinger, M., Longmire, C., Miller, R. & Wright, R.C. AssemblyTron: flexible automation of DNA assembly with Opentrons OT-2 lab robots. *Synth Biol (Oxf)* **8**, ysac032 (2023).
7. Laverick, A. et al. OT-Mation: an open-source code for parsing CSV files into Python scripts for control of OT-2 liquid-handling robotics. *Synth Biol (Oxf)* **10**, ysaf009 (2025).
8. Gao, Y., Fu, L. & Si, T. Large language model agents as experimental orchestrators in data-driven enzyme engineering. *Curr Opin Biotechnol* **96**, 103376 (2025).
9. Inagaki, T., Kato, A., Takahashi, K., Ozaki, H. & Kanda, G.N. LLMs can generate robotic scripts from goal-oriented instructions in biological laboratory automation. *arXiv* (2023).
10. Wierenga, R.P., Golas, S.M., Ho, W., Coley, C.W. & Esvelt, K.M. PyLabRobot: An open-source, hardware-agnostic interface for liquid-handling robots and accessories. *Device* **1**, 100111 (2023).
11. Boiko, D.A., MacKnight, R., Kline, B. & Gomes, G. Autonomous chemical research with large language models. *Nature* **624**, 570–578 (2023).
12. A, M.B. et al. Augmenting large language models with chemistry tools. *Nat Mach Intell* **6**, 525–535 (2024).
13. Wang, M. et al. A call for built-in biosecurity safeguards for generative AI tools. *Nat Biotechnol* **43**, 845–847 (2025).
14. Wittmann, B.J. et al. Strengthening nucleic acid biosecurity screening against generative protein design tools. *Science* **390**, 82–87 (2025).
15. Wang, X. et al. Executable Code Actions Elicit Better LLM Agents. in *Proceedings of the 41st International Conference on Machine Learning* (2024).
16. LangGraph, T. LangGraph: Build resilient language agents as graphs. (2025).
17. Fedorec, A.J.H., Robinson, C.M., Wen, K.Y. & Barnes, C.P. FlopR: An Open Source Software Package for Calibration and Normalization of Plate Reader and Flow Cytometry Data. *ACS Synth Biol* **9**, 2258–2266 (2020).
18. opentrons. Opentrons Protocol API v2 Documentation. (2025).
19. Prasher, D.C., Eckenrode, V.K., Ward, W.W., Prendergast, F.G. & Cormier, M.J. Primary structure of the Aequorea victoria green-fluorescent protein. *Gene* **111**, 229–33 (1992).
20. Fu, L. et al. Critical Assessment of Protein Engineering (CAPE): A Student Challenge on the Cloud. *ACS Synth Biol* **13**, 3782–3787 (2024).
21. Moult, J., Pedersen, J.T., Judson, R. & Fidelis, K. A large-scale experiment to assess protein structure prediction methods. *Proteins* **23**, ii–v (1995).
22. Armer, C., Letronne, F. & DeBenedictis, E. Support academic access to automated cloud labs to improve reproducibility. *PLoS Biol* **21**, e3001919 (2023).
23. Gonzalez Somermeyer, L. et al. Heterogeneity of the GFP fitness landscape and data-driven protein design. *Elife* **11**, e75842 (2022).

24. Sarkisyan, K.S. et al. Local fitness landscape of the green fluorescent protein. *Nature* **533**, 397–401 (2016).
25. Lambert, T.J. FPbase: a community-editable fluorescent protein database. *Nat Methods* **16**, 277–278 (2019).
26. Lin, Z. et al. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science* **379**, 1123–1130 (2023).
27. Su, J. et al. Democratizing protein language model training, sharing and collaboration. *Nat Biotechnol* (2025).
28. Wheeler, N.E. et al. Developing a Common Global Baseline for Nucleic Acid Synthesis Screening. *Appl Biosaf* **29**, 71–78 (2024).
29. Thornton, E.L., Boyle, J.T., Laohakunakorn, N. & Regan, L. Cell-Free Protein Synthesis as a Method to Rapidly Screen Machine Learning-Generated Protease Variants. *ACS Synth Biol* **14**, 1710–1718 (2025).
30. Pedelacq, J.D., Cabantous, S., Tran, T., Terwilliger, T.C. & Waldo, G.S. Engineering and characterization of a superfolder green fluorescent protein. *Nat Biotechnol* **24**, 79–88 (2006).
31. Cranfill, P.J. et al. Quantitative assessment of fluorescent proteins. *Nat Methods* **13**, 557–62 (2016).
32. Lu, X. et al. Constructing a synthetic pathway for acetyl-coenzyme A from one-carbon through enzyme design. *Nat Commun* **10**, 1378 (2019).
33. Luo, Y. et al. Mass Spectrometric Screening for Improving Enzymatic Conversion of Formaldehyde into C2 and C3 Products. *ACS Synth Biol* **14**, 2718–2728 (2025).
34. Wittig, U. et al. SABIO-RK--database for biochemical reaction kinetics. *Nucleic Acids Res* **40**, D790–6 (2012).
35. Swainston, N. et al. STRENDA DB: enabling the validation and sharing of enzyme kinetics data. *FEBS J* **285**, 2193–2204 (2018).
36. Lauterbach, S. et al. EnzymeML: seamless data flow and modeling of enzymatic data. *Nat Methods* **20**, 400–402 (2023).
37. Buchholz, P.C. et al. BioCatNet: A Database System for the Integration of Enzyme Sequences and Biocatalytic Experiments. *Chembiochem* **17**, 2093–2098 (2016).
38. Rapp, J.T., Bremer, B.J. & Romero, P.A. Self-driving laboratories to autonomously navigate the protein fitness landscape. *Nat Chem Eng* **1**, 97–107 (2024).
39. Singh, N. et al. A generalized platform for artificial intelligence-powered autonomous enzyme engineering. *Nat Commun* **16**, 5648 (2025).
40. Zhang, Q. et al. Integrating protein language models and automatic biofoundry for enhanced protein evolution. *Nat Commun* **16**, 1553 (2025).
41. Hu, R. et al. Protein engineering via Bayesian optimization-guided evolutionary algorithm and robotic experiments. *Brief Bioinform* **24**, bbac570 (2023).
42. Torbiak, F. The Laboratory Automation Protocol (LAP) Repository. *ACS Synthetic Biology* (2023).
43. Morrell, W.C. et al. The Experiment Data Depot: A Web-Based Software Tool for Biological Experimental Data Storage, Sharing, and Visualization. *ACS Synth Biol* **6**, 2248–2259 (2017).
44. Lister, A. Experimental Data Connector (XDC): Integrating the Capture of Experimental Data and Metadata Using Standard Formats and Digital Repositories. *ACS Synthetic Biology* **12**, 1364–1370 (2023).
45. Wilkinson, M.D. et al. The FAIR Guiding Principles for scientific data management and stewardship. *Sci Data* **3**, 160018 (2016).
46. Maaten, L.v.d. & Hinton, G. Visualizing data using t-SNE. *J Mach Learn Res* **9**, 2579–2605 (2008).

Methods

Software Framework

LabscriptAI was implemented as a multi-agent system orchestrated using LangGraph (v0.4.8)[16]. The web-based user interface (Extended Data Fig. 1, Supplementary Note 1) was developed using React (v18.3.1) and deployed at https://labscriptai.cn/. The system architecture (Figure 1) comprised a Task Planning Agent that coordinated workflow processing and initiated human-in-the-loop dialogue to validate natural language protocols. The Code Graph backend contained two primary agents: a **Coder** agent for generating Python scripts using platform-specific APIs, and a **Reviewer** agent for code validation using appropriate simulators. When validation errors were detected, the **Reviewer** passed error messages back to the **Coder** agent, which employed the Precise Refactoring Engine (Extended Data Fig. 2) to generate SEARCH and REPLACE instructions targeting specific error locations rather than regenerating entire scripts. The **Reviewer** agent executed these modifications and re-validated the updated code. This Coder-Reviewer interaction cycle continued iteratively, with a maximum limit of nine iterations per script generation request.

Hardware platform-specific simulators were integrated for validation: opentrons_simulate (v8.5.1) for Opentrons systems, the PyHamilton module of PyLabRobot (v0.1.6) for Hamilton Vantage, and the newly developed pyFluent (Extended Data Fig. 5) for Tecan Fluent to provide a standardized Python interface compatible with the proprietary Tecan software environment.

A **Collaboration** Agent interfaced with external databases and repositories (Extended Data Fig. 4). For protocols.io, the agent inter-converted between natural language SOPs and API-compatible JSON structures for versioned storage and retrieval. For protein databases, *ad hoc* Python code was implemented to accommodate heterogeneous access methods, formatting requirements, metadata schemas, and submission protocols, enabling automated deposition in compliance with platform-specific requirements and FAIR data principles.

LLM Selection and Prompt Engineering

The framework integrated Gemini 2.5 Pro through the OpenRouter API (v1.0) for all LLM operations. Prompts were structured with three hierarchical components (Supplementary Note 2): (1) system-level instructions defining agent roles and operational constraints, (2) hardware constraints from the static configuration files VALID_LABWARE_NAMES and VALID_INSTRUMENT_NAMES, and (3) user-provided experimental descriptions and parameters. To ground the LLM output and improve accuracy,

dynamic retrieval and injection were performed to enrich prompts based on a growing knowledge base of CODE_EXAMPLES, which was constructed using validated scripts, embedded using Qwen3-Embedding-8B, and stored in Chroma DB vector database.

Hardware Configuration

For liquid handling robots, the Opentrons OT-2 model was equipped with 1-channel and 8-channel pipettes, along with a Temperature Module and a Magnetic Module. The OT-Flex model was configured with 1-channel and 8-channel pipettes, a Thermocycler Module GEN2, a Heater-Shaker Module, a Magnetic Block, and a Temperature Module, along with the integrated Flex Gripper for labware movement (Figure 3A). The Hamilton Vantage was configured with an 8-channel pipetting arm, a Heater-Shaker Module for incubation and agitation, an integrated Gripper for labware movement, and dedicated carriers for pipette tip racks and reagent reservoirs. The biofoundry work cell configuration (Extended Data Fig. 8) was described previously[33], with a Tecan Fluent station with an 8-channel pipette and a 96/384-channel pipette, a Heater-Shaker Module for plate incubation and agitation, a Temperature Module for microcentrifuge tube incubation, an integrated Gripper for labware movement.

Benchmark Design and Evaluation

A benchmark for evaluating LLM-based liquid-handling script generation was developed by expanding the task set from Inagaki et al.[6] from 27 to 55 tasks, with each task annotated with fine-grained labels for precise error categorization and ablation analysis (Supplementary Dataset 1). Tasks were stratified across four difficulty tiers: Easy (n=13, basic liquid handling), Medium (n=20, multi-step workflows), Hard (n=13, complex error-prone operations), and Expert (n=9, multi-module coordination), with a mean difficulty index of 2.33 on a 4.0 scale. Prompts were kept concise (mean 91 tokens) to rigorously test instruction adherence. The benchmark incorporated 31% safety-critical scenarios requiring appropriate refusal or cautionary responses, 11% multi-module orchestration tasks, and 15% parameterized templates enabling systematic variation of experimental parameters. This structure enabled quantitative assessment of script generation robustness across realistic operational constraints and hardware configurations. Performance was evaluated by simulation pass rate, with generated scripts validated using platform-specific simulators. Ablation studies systematically removed individual architectural components to quantify their contributions to overall performance.

Strains and Cultivation Conditions

E. coli DH5α (Sangon Biotech, Shanghai, China) was used for plasmid construction and amplification. *E. coli* BL21(DE3) (Sangon Biotech) served as the recombinant host for protein expression. *E. coli* BL21(DE3)-Δ6 (Δ*frmA*, Δ*dhaK*, Δ*fsaA*, Δ*fsaB*, Δ*gldA*, Δ*glpK*)[1] was employed for GALS expression and whole-cell FALD biotransformation. *E. coli* cells were cultured at 37°C with 220 rpm shaking in Luria-broth (LB) media supplemented with appropriate antibiotics for plasmid maintenance: 50 μg mL^{-1} kanamycin (Kan) and/or 50 μg mL^{-1} spectinomycin (Spec). Solid media were prepared by adding 1.5% (w/v) agar. All chemicals were purchased from Sigma□Aldrich (St. Louis, MO, USA) or Thermo Fisher Scientific (Pittsburgh, PA, USA) unless otherwise noted.

DNA and Strain Construction

All strains and plasmids used in this study are listed in Supplementary Tables 3, 4. Primers and genes (Supplementary Dataset 2) were synthesized by GenScript Biotech (Nanjing, China) or RootPath (Guangzhou, China). Enzymes for recombinant DNA cloning, including Q5 PCR polymerase, restriction endonucleases, and T4 ligase, were purchased from New England Biolabs (Ipswich, MA, USA). QIAprep Spin Plasmid Mini-prep Kits (Qiagen, Valencia, CA, USA) were utilized to isolate plasmid DNA from *E. coli*. PCR, digestion, and ligation products were purified using QIAquick PCR Purification and Gel Extraction Kits (Qiagen). Plasmid construction and mutagenesis were performed via Gibson assembly using NEBuilder HiFi DNA Assembly Master Mix. Chemically competent *E. coli* was transformed using heat shock at 42°C. Sanger DNA sequencing was performed at Sangon Biotech.

Cell-free GFP expression

CFPS reactions were assembled using the NEBExpress Cell-free *E. coli* Protein Synthesis System (New England Biolabs), containing 6 μL of S30 Synthesis Extract, 12.5 μL of 2× Protein Synthesis Buffer, 0.5 μL of T7 RNA polymerase, 0.5 μL of RNase Inhibitor (Murine), and 125 ng of template DNA. Reactions were brought to a final volume of 25 μL with nuclease-free water in 96-well plates and incubated at 37°C. Fluorescence measurements were performed using an Agilent Microplate Reader with excitation at 480 nm and emission at 520 nm.

Biosecurity Screening and Landscape Visualization of GFP Sequences

Biosecurity screening of student-submitted sequences was performed using a local implementation of the Common Mechanism for DNA Synthesis Screening as previously described[28] (https://github.com/ibbis-

bio/common-mechanism). The pipeline comprised four steps[28]: (1) biorisk search using HMM-based matching against curated sequence profiles; (2) taxonomy search to identify matches to regulated pathogens; (3) protein search using BLASTX/DIAMOND against the NCBI nr database; and (4) nucleotide search using BLASTN against the NCBI core_nt database. To promote responsible research awareness among CAPE participants, a cloud-based implementation was provided as a Google Colab notebook for self-screening prior to design submission (available at colab.research.google.com/drive/1JvIxU67nSD5IMSvRqvCYnGl8ERaVLHCT).

To visualize the sequence landscape, CAPE submissions (n = 298) and FPbase references (n = 140) were encoded with the esm2_t12_35M_UR50D[26] model and dimensionality was reduced via t-SNE[46] (perplexity=30, PCA initialization). The resulting 2D projection incorporated kernel density estimation (KDE) to display distribution densities, with the top 10 performing CAPE variants, wild-type avGFP, and benchmark sfGFP sequences overlaid as distinct data points.

Magnetic bead-based protein purification in microtiter plates

His-tagged proteins from CFPS reactions were purified using Ni-NTA Magnetic Beads (Smart-Lifesciences, Changzhou, China). Bead slurry was resuspended in five bead volumes of Binding Buffer by vortexing and transferred to 96-well plates with 100 μL aliquot per well. Beads were equilibrated by magnetic separation on the magnetic block, supernatant removal, and two washes with 100 μL of Lysis/Binding Buffer, resuspending by pipetting 10 times between washes. CFPS reactions (50 μL) were added to equilibrated beads and incubated for 30 minutes at room temperature with 1,000 rpm shaking on the heater-shaker module. Beads were magnetically separated, and supernatant was removed. Non-specifically bound proteins were removed through two sequential washes with 150 μL of Wash Buffer containing 0 mM and 20 mM imidazole, respectively. Target proteins were eluted by resuspending beads in 150 μL of Elution Buffer containing 250 mM imidazole and incubating for 10 minutes.

Bradford assay for protein quantification

Protein concentration was determined using the Bradford Protein Assay Kit (P0006, Beyotime Biotechnology, Shanghai, China). A standard curve was generated using bovine serum albumin (BSA) diluted in PBS to concentrations ranging from 0 to 1.5 mg mL^{-1}. For the assay, 5 μL of each standard or sample was mixed with 250 μL of G250 dye reagent in a 96-well plate. Following a 10-minute incubation at room temperature,

absorbance was measured at 595 nm using a microplate reader, and protein concentrations were calculated by comparison to the BSA standard curve.

Manual GFP expression and purification

Transformed BL21(DE3) cells were cultured in 1 L of LB medium at 37°C until OD_{600} reached 0.6-0.8, followed by induction with 0.5 mM IPTG at 18°C for 16-20 hours. Cells were harvested by centrifugation and resuspended in lysis buffer (50 mM Tris-HCl, 500 mM NaCl, 10 mM imidazole, pH 8.0). Cell lysis was performed by sonication on ice, cleared lysate obtained by centrifugation (4,000 × g, 30 min, 4°C) was applied to Ni-NTA agarose (Qiagen), and proteins were eluted with 250 mM imidazole. Following concentration using Amicon Ultra-15 centrifugal filters (30 kDa MWCO, Millipore), proteins were further purified by size exclusion chromatography on a Superdex 200 pg 16/600 column using an ÄKTA prime PLUS system (GE Healthcare) equilibrated with 25 mM Tris-HCl, 150 mM NaCl, pH 8.0, with elution monitored at 280 nm absorbance (A_{280}).

GFP characterization

For spectroscopic characterization of purified proteins, excitation spectra were recorded from 300-520 nm (λ_{em} = 550 nm) and emission spectra from 460-650 nm (λ_{ex} = 430 nm), both with 2 nm bandwidth. Quantum yields were determined using sfGFP as a reference standard (QY = 0.65) with integrated emission normalized to absorbance at 450 nm, according to the following equation,

$$QY_i = \frac{\int_{480}^{650} F_i d\lambda}{\int_{480}^{650} F_{sfGFP} d\lambda} \frac{A_{sfGFP}}{A_i} QY_{sfGFP}$$

where QY denotes quantum yield, F denotes fluorescence emission intensity at wavelength λ nm, A denotes absorbance at 450 nm, and i denotes a particular GFP. Extinction Coefficient (ε) was calculated using the formula:

$$\varepsilon = \frac{A_{\lambda_{max}}}{c \times I}$$

where ε denotes extinction coefficient, $A_{\lambda max}$ is absorbance of proteins at maximum absorption wavelength, c is molar concentration, I is path length (0.7 cm). Molecular brightness was calculated as the product of extinction coefficient and quantum yield.

Construction of GALS Mutants

GALS mutagenesis libraries were created by introducing specific mutations at selected positions via PCR using designed primers. Gibson assembly was used to construct plasmid libraries, which were then used to transform *E. coli* DH5α on an automatic thermocycler. For each construct, three random colonies were picked using a QPix 450 colony picker for sequence verification. A representative collection of mutant strains covering all 19 single mutants and 361 double mutants were then selected by hit picking for plasmid DNA extraction using a Tecan Fluent equipped with a magnetic stand and a centrifuge using the 96 Mag-MK Plasmid DNA Mini-Preps Kit (TianGen Biotech Co., Ltd., Beijing, China). Plasmid DNA was then used to transform BL21(DE3)-Δ6, and two independent clones were selected for subsequent phenotypic screening.

Whole-cell FALD Biotransformation for Mutant Screening

Mutant colonies were picked into 96 deep-well microplates (Jetbiofil DMP221196) containing 200 μL of LB+Kan and incubated for 24 h at 37°C. Cultures were diluted 1:100 into 1 mL of fresh LB+Kan containing 0.1 mM isopropyl β-D-1-thiogalactopyranoside (IPTG) and incubated at 37°C for 4 h followed by 20 h at 30°C. Cells were harvested by centrifugation at 3,300×*g* for 5 min, washed twice using phosphate-buffered saline (PBS, 50 mM, pH 7.4), and resuspended in assay buffer (50 mM PBS, 5 mM $MgSO_4$, 0.5 mM thiamine pyrophosphate (TPP), 1.5 g L^{-1} FALD, pH 7.4). After 30°C incubation for 24 h, supernatants were collected after centrifugation at 3,300×*g* for 5 min for product analysis.

Spectrometric Assays for GA and DHA Detection

For spectrometric GALD detection, diphenylamine (DPA) reagent was prepared by dissolving 1.5 g diphenylamine in 100 mL of glacial acetic acid followed by addition of 1.5 mL of concentrated sulfuric acid. Sample and standard solutions (30 μL) were mixed with 150 μL of DPA reagent in 96-well PCR plates (Jetkeen PC-0200-9B), incubated at 90°C for 30 min, and endpoint absorbance was measured at 650 nm following transfer to 96-well microtiter plates (Corning 3599).

For spectrometric DHA detection, 90 μL of sample or standard solutions were dried at room temperature in 96-well microtiter plates in a chemical fume hood. Samples were reconstituted in 90 μL of 50 mM PBS (pH 7.4), followed by addition of 60 μL of Buffer A (0.2 mg mL^{-1} galactose oxidase, 24 U mL^{-1} horse radish peroxidase, 50 mM PBS, 5 mM $MgSO_4$, pH 7.4) and then 50 μL of Buffer B (3.2 mM ABTS, pH 7.4). Absorbance at 410 nm was monitored for 20 min.

Manual GALS Protein Expression and Purification

Genes encoding the parent and mutant GALS were assembled into pET28m expression vector via Gibson Assembly. *E. coli* BL21(DE3) cells carrying recombinant plasmids were cultivated in 5 mL of LB+Kan at 37 °C for 16-20 hrs and subsequently scaled to 1 L cultures. When OD_{600} reached 0.6~0.8, IPTG was added to a final concentration of 0.5 mM, followed by 18 °C induction for 20 h. Cells were harvested by centrifugation at 8,000×*g* and resuspended in 50 mL of lysis buffer (50 mM Tris-HCl, 5 mM $MgSO_4$, pH 7.5).

Cell lysis was performed using a high-pressure homogenizer (JNBIO, China), and debris was removed by centrifugation at 10,000×*g* for 30 min at 4 °C. The clarified lysate was loaded onto a nickel affinity column (GE Healthcare), washed with 50 mL of wash buffer (25 mM Tris-HCl buffer, 5 mM $MgSO_4$, and 20 mM imidazole, pH 7.5,) and eluted with 20 mL of elution buffer (25 mM Tris-HCl, 5 mM $MgSO_4$, and 250 mM imidazole, pH 7.5). Purified protein was concentrated and buffer-exchanged into 50 mM PBS with 5 mM $MgSO_4$ (pH 7.4) using an Amicon Ultra centrifugal filter device (Millipore, USA, 30 kDa MWCO). Protein concentration was determined using the Bradford Protein Assay Kit (Beyotime).

In vitro Enzyme Kinetics Characterization

Initial reaction rates were measured at 37°C in 50 mM PBS (pH 7.4) containing 5 mM $MgSO_4$ and 0.5 mM TPP upon addition of 1 mg mL^{-1} purified enzyme. FALD substrate concentrations ranged from 2 to 1,500 mM. Kinetic parameters k_{cat} and K_m were estimated by measuring initial velocities and curve-fitting to the Michaelis-Menten equation or allosteric sigmoidal equation, as appropriate, using GraphPad Prism 9 software. All measurements were performed in triplicate.

Data and Code Availability

All data, code, and protocols generated in this study are publicly available. The LabscriptAI source code is available at https://github.com/KRATSZ/LabScript-AI. A web implementation can be accessed at https://labscriptai.cn/. CAPE competition datasets and participant repositories are archived at https://github.com/KRATSZ/CAPE-CriticalAssessmentofProteinEngineering. Benchmark tasks, raw experimental data, generated scripts, and analysis files have been deposited in Zenodo (DOI: 10.5281/zenodo.17697326). Automated protocols are available on protocols.io (https://protocols.io/workspaces/labscriptai). GFP variant data have been deposited in FPbase (https://www.fpbase.org/protein/var1). Enzyme kinetic data have been deposited in STRENDA DB: parent GALS (DOI: 10.22011/strenda_db.NAEKEK), V281W/F282L (DOI: 10.22011/strenda_db.32PJ7F), and

V281W/F282T (DOI: 10.22011/strenda_db.OQPSZP). Standardized files formatted for EnzymeML, SABIO-RK, and BioCatNet are included in the Zenodo repository.

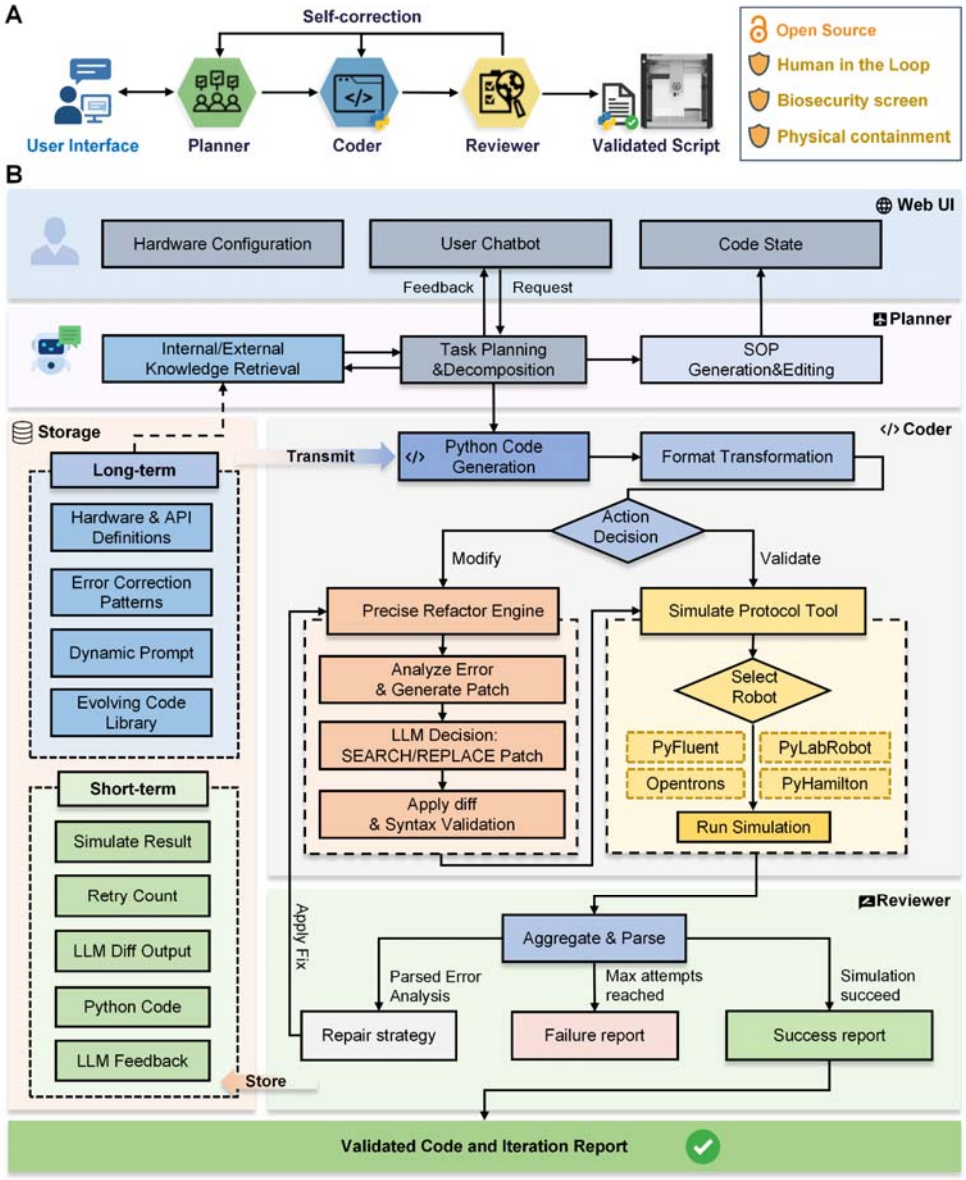


Figure 1. LabscriptAI system architecture. A. Conceptual workflow of the LabscriptAI multi-agent system with accessible and responsible features. **B.** Detailed system architecture supporting workflow from user input to validated script. The Planner translates natural language requests into structured standard operation procedures (SOPs). The Coder generates executable Python code informed by domain knowledge and project context from the Storage layers. The Reviewer validates generated code using platform-specific simulators. Upon error detection, the Precise Refactor Engine performs targeted code patches. This iterative refinement cycle continues until validation succeeds or retry limits are reached, with intermediate outputs stored in the short-term storage layer and validated code promoted to the long-term knowledge base.

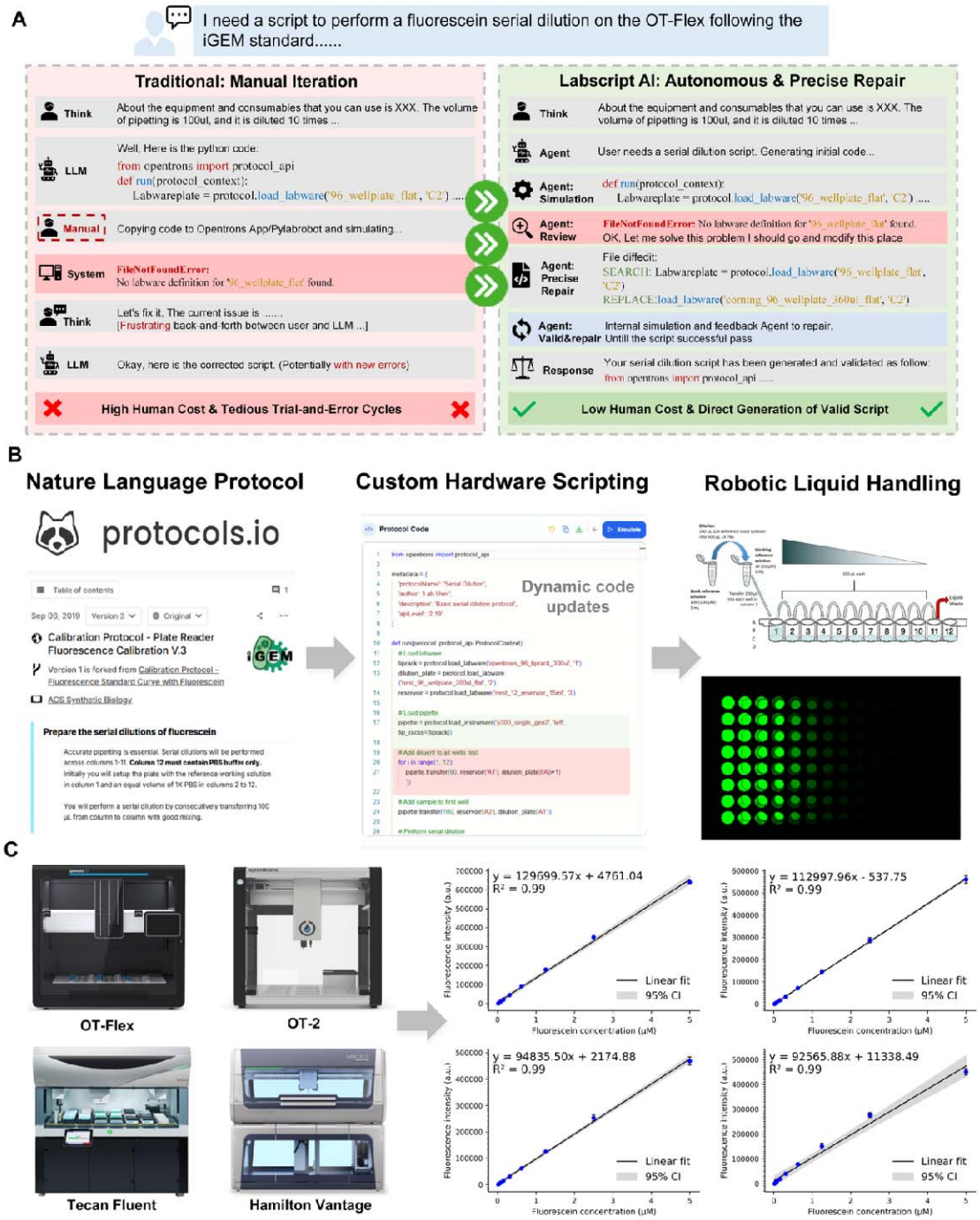


Figure 2. Cross-platform robotic automation of iGEM plate reader calibration protocol. A. Workflow comparison between LLM-assisted manual scripting (left) and LabscriptAI autonomous generation (right). Traditional approaches require multiple debugging cycles with extensive human intervention, whereas LabscriptAI autonomously identifies and corrects simulation errors. **B.** End-to-end workflow from protocol retrieval via external repository to validated script execution for fluorescein serial dilution. **C.** Experimental validation across four liquid handling platforms, producing standard calibration curves with high linear regression fits.

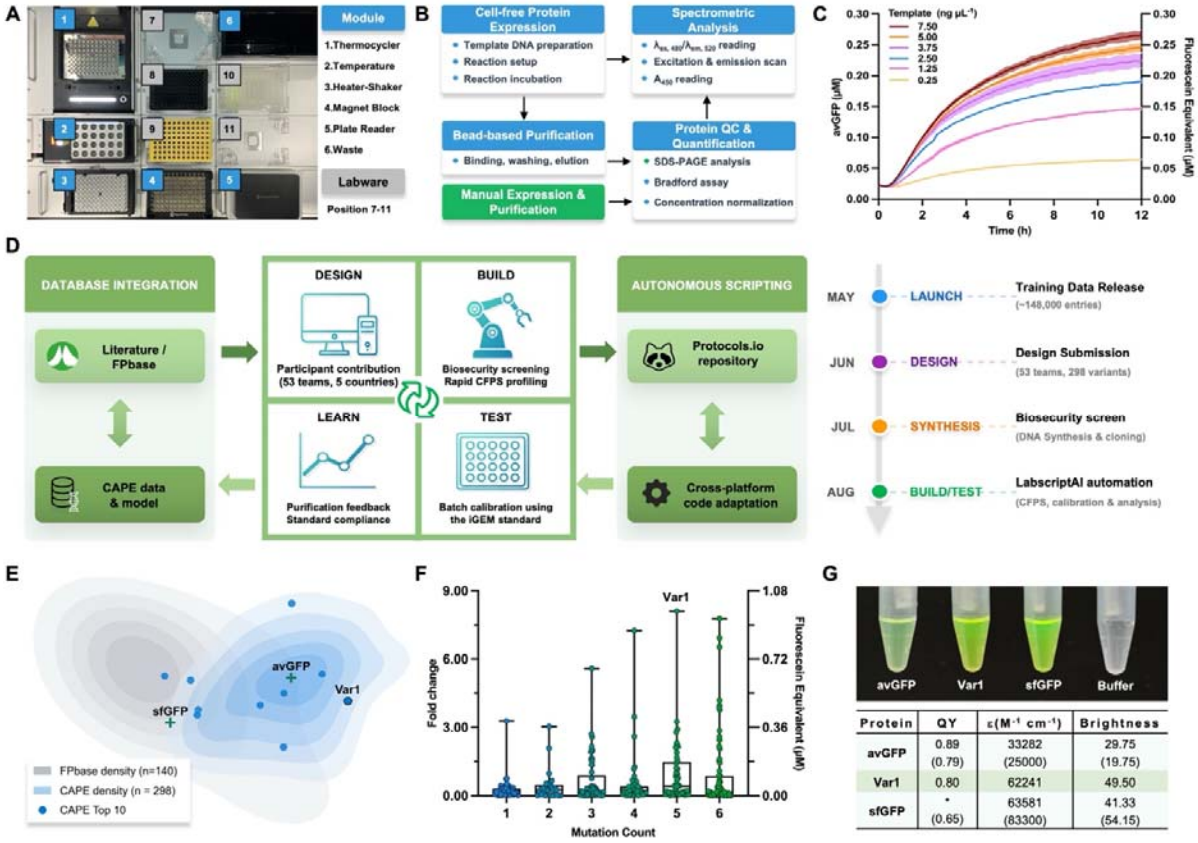


Figure 3. Robotic characterization of community-designed GFP variants from CAPE challenge. A. Opentrons Flex deck layout with integrated modules. **B.** Automated workflow (blue) for cell-free protein synthesis (CFPS), purification, and characterization. Manual processes (green) for protein expression and purification were necessary for quantitative photophysical characterization. **C.** Time-course GFP fluorescence measurements across template DNA concentrations in CFPS reactions. **D.** Schematic of the CAPE challenge integrating community-driven Design-Build-Test-Learn cycle with bidirectional connectivity to protein databases and protocol repositories. All protocols, data, and models were open access with integrated biosecurity screening. Timeline shows the 2025 challenge stages. **E.** Sequence landscape of variants designed by CAPE participants and variants deposited in FPbase for avGFP. Sequences were encoded using ESM2[26], and dimensionality reduction for 2D visualization was performed using t-SNE[46]. **F.** Fluorescence of CAPE variants in CFPS reactions stratified by mutation count. Fold change was normalized to avGFP under identical conditions. **G.** Photophysical characterization of purified CAPE top variant, avGFP parent, and sfGFP benchmark. Photograph shows protein solutions (0.1 mg mL^{-1}) illuminated by UV. Quantum yield (QY) was calculated using sfGFP as reference denoted by asterisk (QY_{sfGFP} = 0.65). Values in parentheses represent FPbase records for avGFP and sfGFP. ε denotes extinction coefficient.

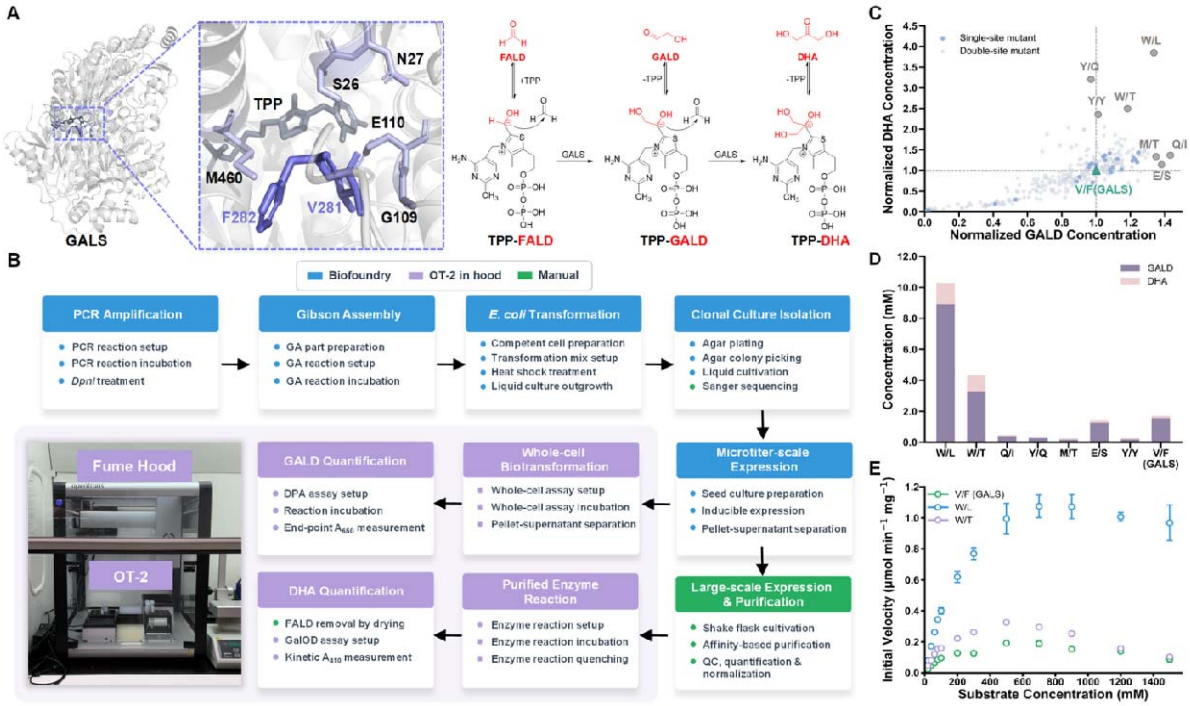


Figure 4. Robotic GALS enzyme engineering using distributed automation. A. Left: structural model of GALS with active site residues V281 and F282 highlighted; Right: thiamine pyrophosphate (TPP)-dependent catalytic mechanism for FALD condensation into GALD and DHA. **B.** Distributed workflow partitioning tasks across biofoundry automation (Tecan Fluent for library construction and cell cultivation, blue), fume hood-enclosed automation (Opentrons OT-2 for FALD assays and analysis, purple), and manual operations (for inter-station transfers and protein purification, green). **C.** Whole-cell biotransformation screening results for double-site mutants (gray) and single-site mutants (blue), with product concentrations normalized to parent GALS (green). **D.** In vitro GALD and DHA product formation from purified enzyme variants using 66.7 mM FALD substrate. **E.** Enzyme kinetics analysis of parent GALS and top variants (V281W/F282L, V281W/F282T), showing initial velocity measurements for combined GALD and DHA production. See Extended Data Fig. 9 for kinetics curves and parameter fittings for individual products.

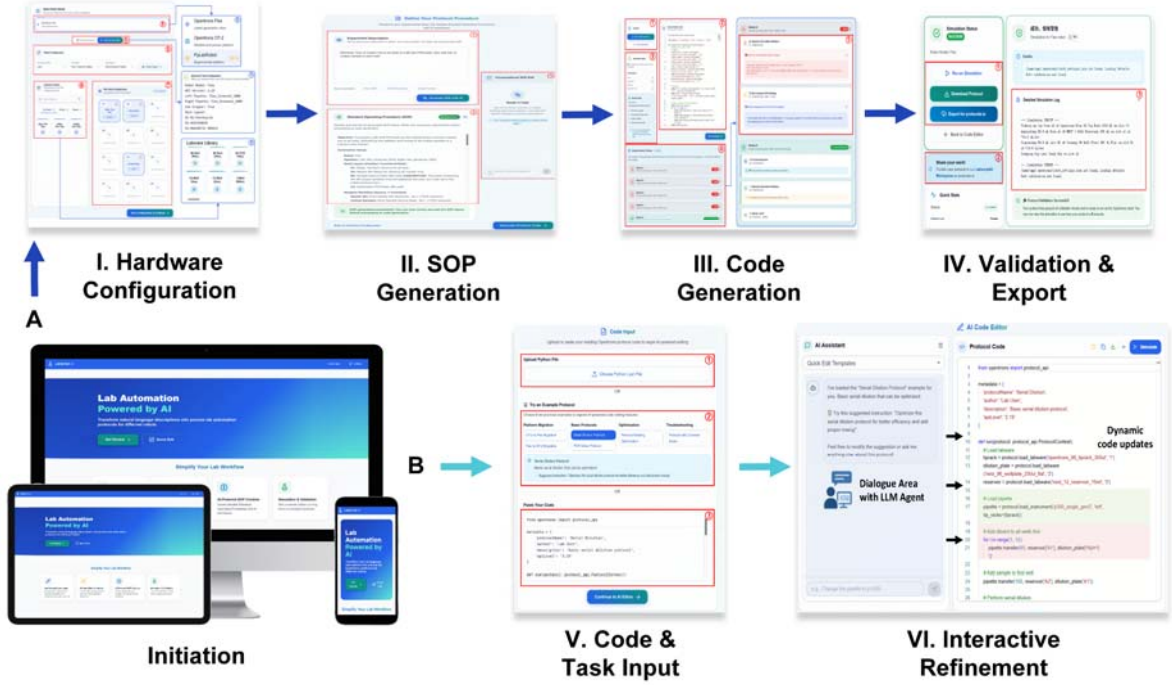
Table 1. Benchmark performance of LLM systems on liquid-handling script generation. Success rates across 55 benchmark tasks stratified by complexity: Easy (n=13, basic liquid handling), Medium (n=20, multi-step workflows), Hard (n=13, complex error-prone operations), and Expert (n=9, multi-module coordination).

LLM system	Success Rate (%)				
	Easy (13)	**Medium (20)**	**Hard (13)**	**Expert (9)**	**Overall (55)**
GPT-5	46.2	20.0	7.7	0.0	20.0
(refusal*)	(7.0)	(15.6)	(38.5)	(44.4)	(23.6)
DeepSeek V3.2	53.8	45.0	46.2	11.1	41.8
Claude 4	61.5	50.0	53.8	0.0	45.4
Gemini-2.5 Pro	46.2	20.0	38.5	33.3	32.7
OpentronsAI	76.9	100	61.5	44.4	76.3
LabScriptAI	**100**	**100**	**76.9**	**66.7**	**89.1**

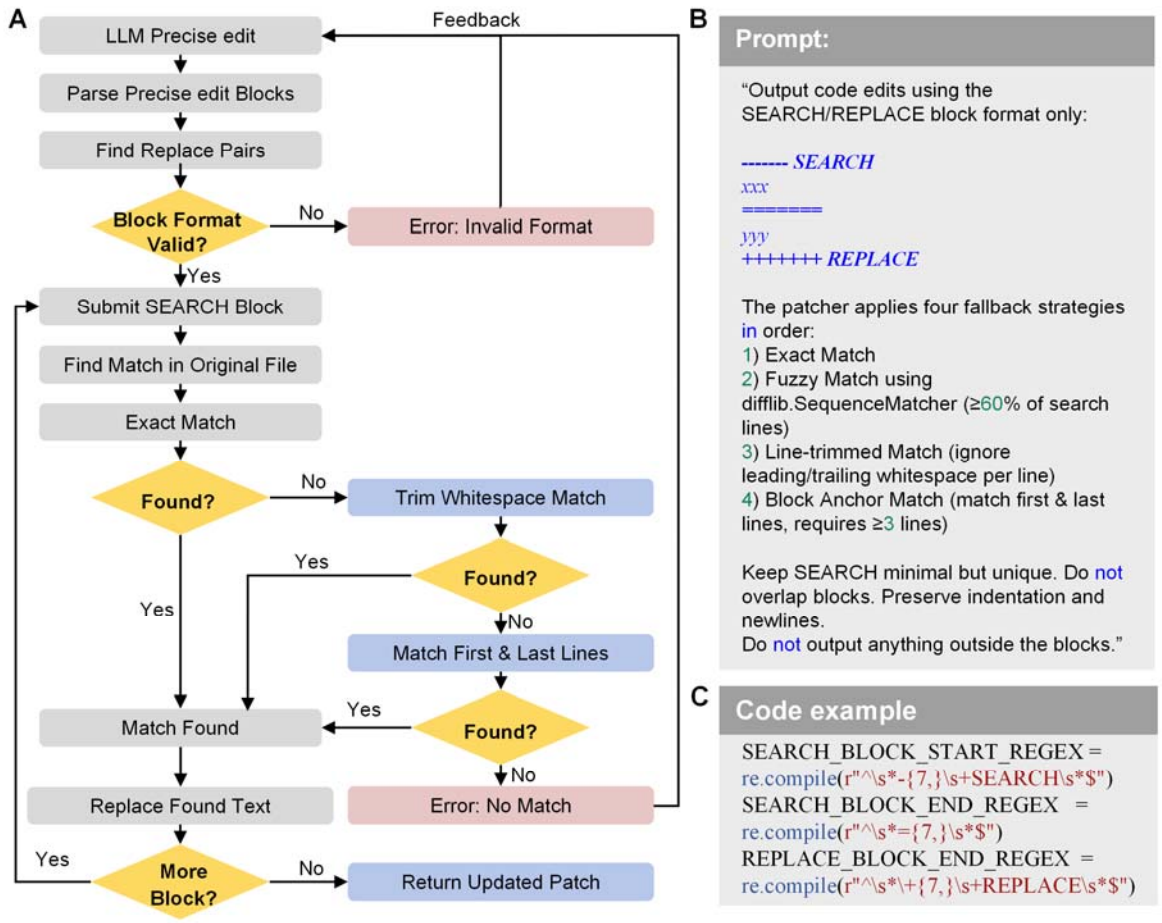
* GPT-5 refused to generate scripts for certain tasks due to violation of its safety policy; refusal rates shown in parentheses are counted as failures in success rate calculations.

Table 2. Ablation study quantifying contributions of architectural components.

Configuration	Success Rate and drop (%)	Average Iterations	Average Token Cost
Full model			
Complete LabscriptAI with Precise Refactoring Engine (PRE), Retrieval-Augmented Generation (RAG), and iterative feedback loops.	**89.1**	**3.07**	**19,549**
Without PRE			
Replaces targeted code modifications with complete script regeneration upon error detection	80.0 (-9.1)	4.38	27,309
Without Knowledge			
Removes domain knowledge retrieval (RAG), relying solely on base LLM capabilities.	76.3 (-12.8)	5.27	23,560
Without Iteration			
Removes iterative feedback loops, constraining the system to single-pass generation.	18.2 (-70.9)	1.00	N/A



Extended Data Figure 1 | LabscriptAI web interface and user workflow modules. The platform supports two primary operational modes initiated from the dashboard (https://labscriptai.cn/). **A. Generation Pipeline (Top Row, I-IV)**, where users define hardware constraints (I), generate Standard Operating Procedures (SOPs) from natural language (II), convert SOPs into executable Python scripts (III), and validate protocols through simulation (IV). **B. Refinement Pipeline (Bottom Row, V-VI)**, allowing users to upload existing scripts or define specific tasks (V) and engage in Interactive Refinement (VI), where an LLM-driven agent assists in debugging and optimizing code through conversational interface, enabling dynamic code updates based on user feedback. See Supplementary Note 1 for detailed description of LabscriptAI frontend architecture and user workflows.



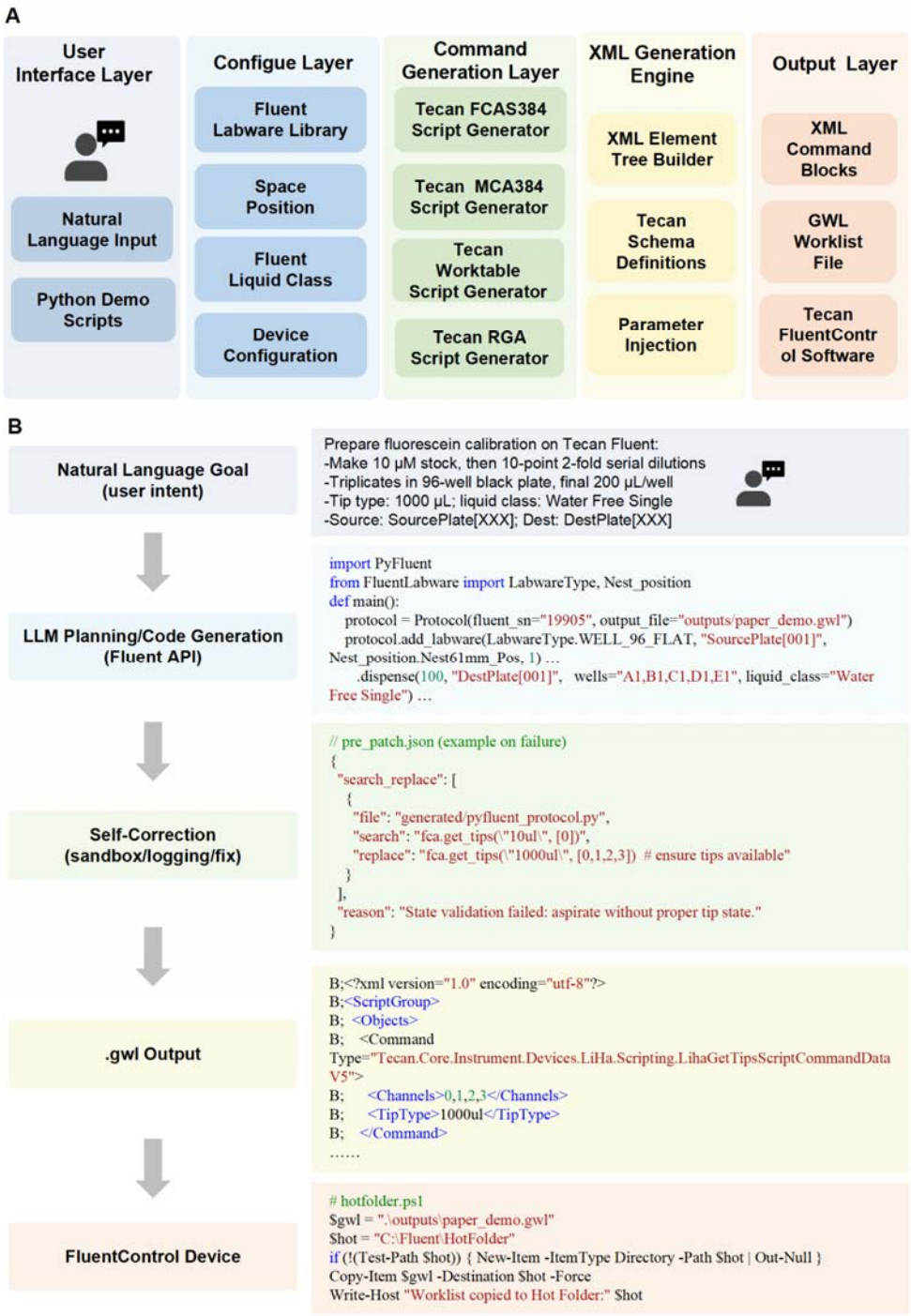
Extended Data Figure 2 | Precise Refactor Engine architecture and workflow. A. Flowchart illustrating patch application logic. The engine uses a multi-step fallback strategy to locate target code block for modification, starting with exact matching and progressing to more flexible matching of whitespace and anchor lines. **B.** Structured prompt constraining LLM output to the required SEARCH/REPLACE format. **C.** Regular expressions for parsing LLM-generated patch blocks. The implementation was inspired by the diff-edit workflow from Cline (https://github.com/cline/cline), re-implemented and enhanced in Python and LangGraph to address laboratory automation requirements.

LLM System	Difficulty Level	Primary Error Type	Root Cause Analysis
LLM (general)	Easy	OutOfTipsError; SyntaxError	Inefficient Tip Management: Used a new tip for every well in a 96-well plate wash without reuse, rapidly exhausting tips. Also exhibited logical flaws (e.g., repeated aspirations) and missing operational details.
	Medium	SyntaxError; APIVersionError	API/Syntax Contract Violation: Included stray Markdown characters, used API features (protocol.params) with an incompatible apiLevel, and declared metadata incorrectly. Range checks ignored physical instrument and labware constraints.
	Hard	Parser/Loader Errors; API SyntaxError	Module/API and Logistical Failures: Attempted to load invalid or non-existent modules, misused API calls (e.g., protocol.transfer for plates), and implemented unrealistic on-deck delays for long off-deck incubations.
	Expert	Loader Errors	Hallucinated Capabilities and API Violations: Contained syntax errors from code fences, used invalid module names, and invented non-existent hardware APIs (e.g., environmental sensors), rendering code un-runnable or operationally meaningless.
OpentronsAI	Easy	InvalidAspirateVolumeError	Pipetting Logic Error: The air_gap() command was incorrectly called on an already empty tip, violating the instrument's physical operating logic.
	Hard	InvalidAspirateVolumeError	Hardware Constraint Violation: Attempted to aspirate a volume (400 µL) that exceeded the physical capacity of the specified pipette (P300, max 300 µL).
	Expert	OutOfTipsError	Insufficient Labware: The protocol required more tips (480) for its 96-channel operations than were loaded onto the deck (288), causing run failure. Conceptual flaws were also present.
LabScriptAI	Hard	LocationIsOccupiedError	Deck Configuration Error: Two different labware items were assigned to the same physical deck slot (using aliases "1" and "D1"), causing a load-time conflict. The script also contained missing experimental steps.
	Expert	InvalidDispenseVolumeError	Pipetting Logic Error: Attempted to dispense liquid from an empty tip after a multi-dispense sequence, causing an error. The protocol also failed to meet core requirements (wrong plate type, missing steps).

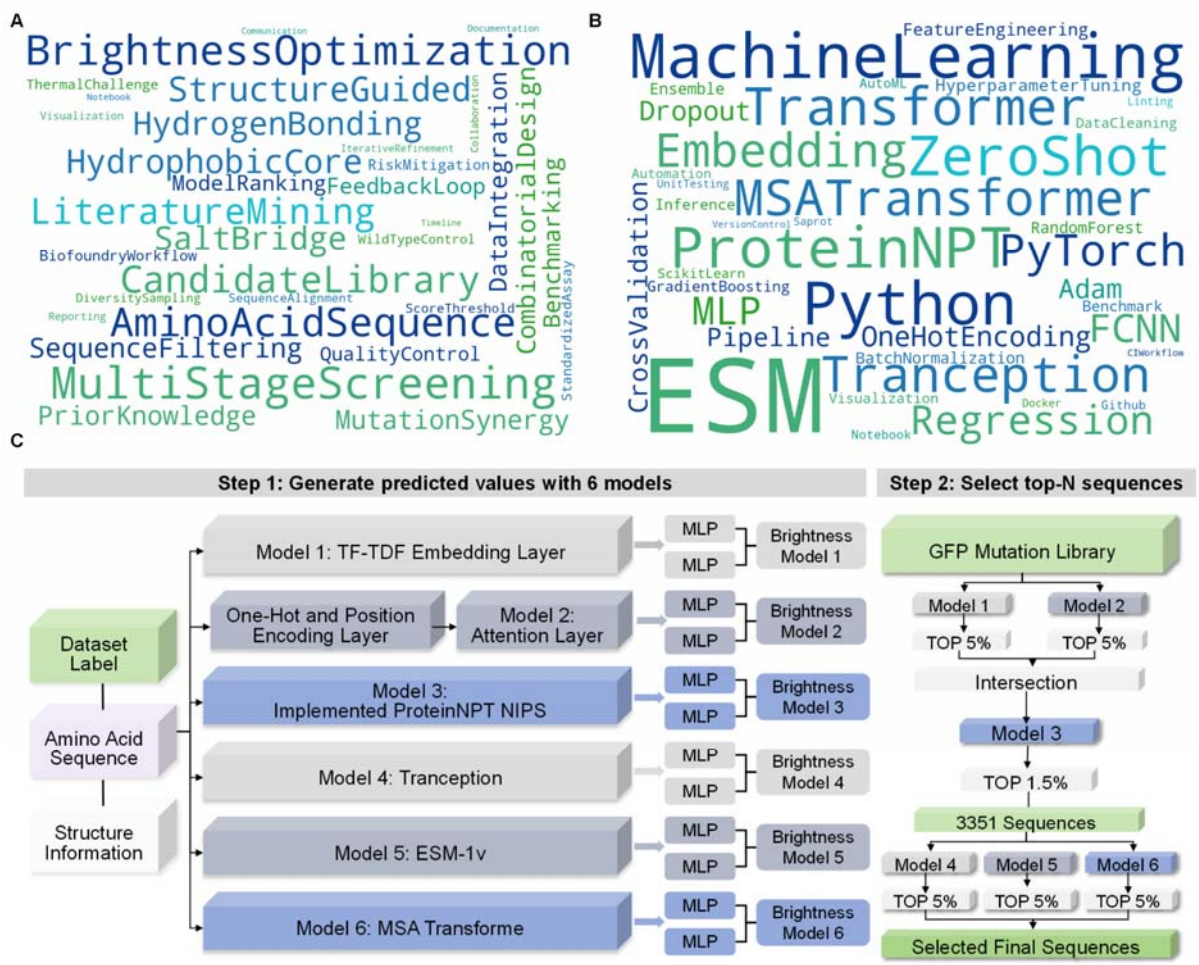
Extended Data Figure 3 | Error patterns across LLM systems on benchmark tasks.

Content	Database / Repository	Upload Route	Data Format	Conversion Method
Protocol	**Protocols.io**	API	JSON	Python Script
GFP data	**FPbase**	Web	Web Table	Python Script + Manual
GALS data	**STRENDA**	Web	Web Table	Python Script + Manual
	BioCatNet	Email	Excel	Python Script
	SABIO-RK	API	JSON	Python Script
	EnzymeML	N/A	JSON-LD	PyEnzyme[36]

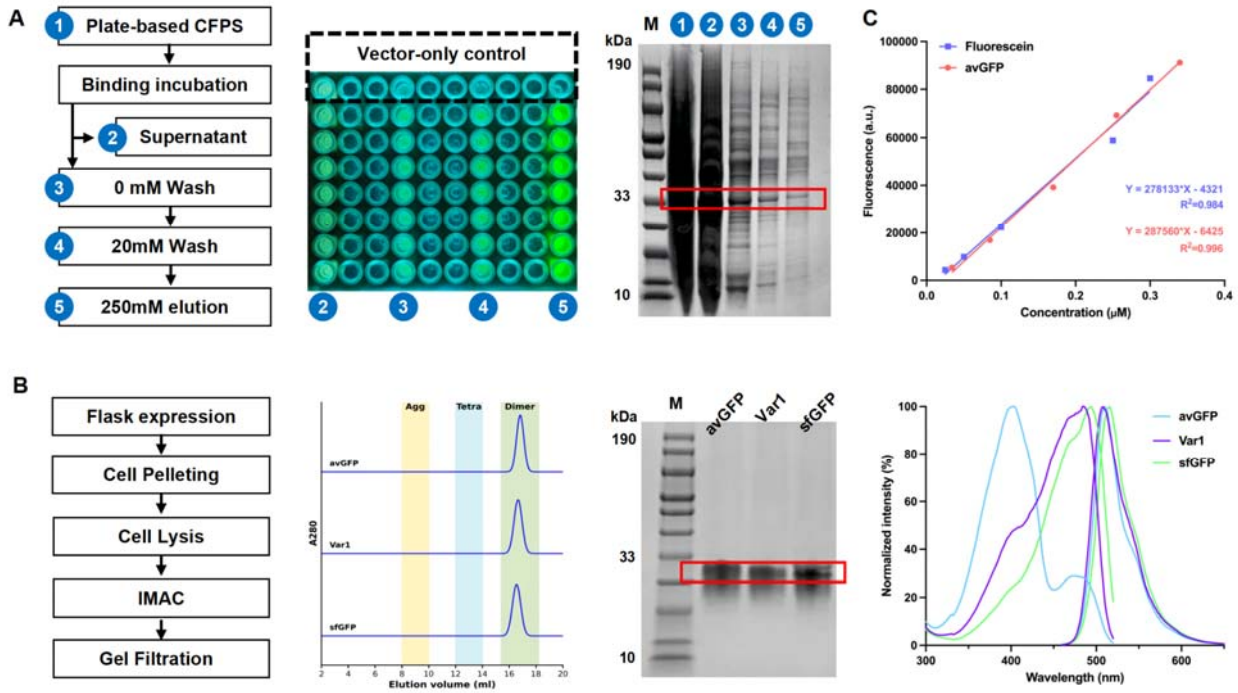
Extended Data Figure 4 | Collaboration agent integration with external databases and repositories.



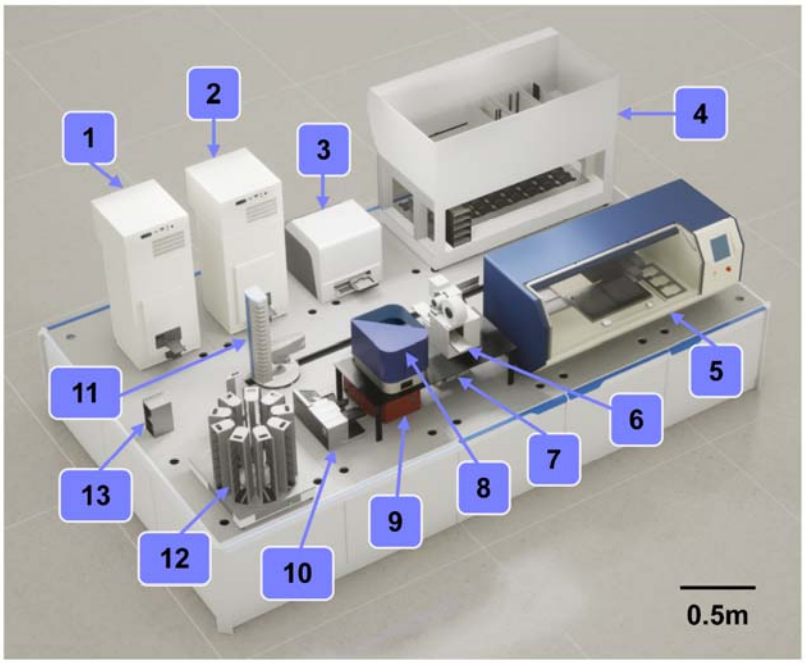
Extended Data Figure 5 | pyFluent architecture and implementation workflow. A. Layered architecture of the pyFluent framework providing a Python API for controlling Tecan Fluent robots. The framework abstracts low-level XML command generation from high-level user inputs. **B.** End-to-end workflow demonstrating conversion of natural language instructions into Tecan-executable code through LLM translation to Python using pyFluent API, self-correction, and compilation into .gwl worklist files.



Extended Data Figure 6 | Computational strategy summary and a showcase of CAPE teams. A. Word cloud of protein design strategies used by teams. **B.** Word cloud of machine learning methods and computational tools employed. **C.** The two-step computational pipeline of the top-performing team. Step 1 generated brightness predictions using an ensemble of six different models. Step 2 intersected the top-ranked sequences from these models to select a final high-confidence set of candidates for experimental validation.

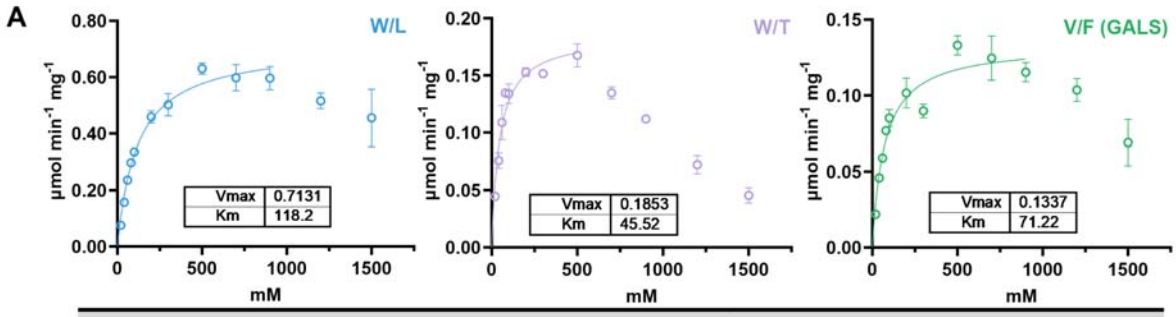


Extended Data Figure 7 | Comparison of plate-based and manual GFP purification workflows. A. Automated plate-based workflow with magnetic bead purification of CFPS reactions containing sfGFP template and vector-only control. Left: purification steps and numbered fractions. Middle: 96-well plate containing numbered fractions illuminated by UV. Right: SDS-PAGE analysis. **B.** Manual purification. Left: workflow schematic employing flask-scale expression and two chromatographic steps. Middle: size exclusion chromatography elution profiles with dimer peaks for avGFP, Var1, and sfGFP, and corresponding SDS-PAGE analysis. Right: fluorescence excitation and emission spectra (300-600 nm). **C.** Fluorescein and purified avGFP calibration curves. M, protein molecular weight marker. Red boxes indicate target GFP proteins. IMAC, Immobilized Metal Affinity Chromatography.

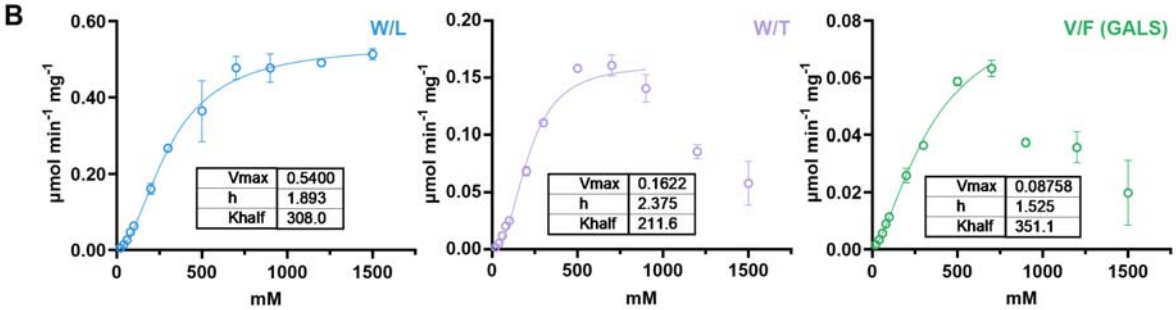


1. Shaking incubator I
2. Shaking incubator II
3. Plate reader
4. Liquid handler
5. Colony picker
6. Seal peeler
7. Thermocycler
8. Plate sealer
9. Automated centrifuge
10. Reagent dispenser
11. Robotic arm
12. Plate hotel I
13. Plate hotel II (interfacing w/ AGV)

Extended Data Figure 8 | Automated biofoundry for GALS library construction and strain cultivation. The biofoundry platform consists of: (1, 2) two STX44 incubators (LiCONiC, Mauren, Liechtenstein); (3) a Spark microplate reader (Tecan, Männedorf, Switzerland); (4) a Fluent 780 liquid handling robot (Tecan); (5) a QPix HT Colony Picker (Molecular Devices, San Jose, CA, USA); (6) an Xpeel seal peeler (Brooks Life Sciences, Manchester, UK); (7) four ATC thermocyclers (Thermo Scientific, Waltham, MA); (8) a WASP plate sealer (Kbiosystems, Essex, UK); (9) a HiG 3 robotic plate centrifuge (BioNex, San Jose, CA); (10) a Multidrop Combi reagent dispenser (Thermo Scientific); (11) a Spinnaker robotic arm on a 3.6-m track (Thermo Scientific); (12) an LPX440 storage carousel (LiCONiC); (13) a plate hotel interfacing with Automated Guided Vehicle (AGV). Fluent 780 was equipped with an FCA 8-channel independent pipetter, an RGA robotic manipulation arm, an MCA384 with EVA adapter for 96-channel pipetting, a CPAC Ultraflat temperature-controlled block (Inheco, Munich, Germany), and two Te-Shake Silver shakers (Tecan). FluentControl (Tecan, Version 3.6) was used to control the liquid handling robot and program pipetting, labware transportation, and temperature control. Momentum (Thermo Scientific, Version 7.1.2) was used to communicate with peripheral devices, control the central robotic arm, and program unit operations and sample transportation routes.



GALD	K_M [mM^{-1}]	V_{max} [μmol min^{-1} mg^{-1}]	k_{cat} [S^{-1}]	k_{cat}/K_M [S^{-1} M^{-1}]
W/L	118.2	0.713	0.660	5.587
W/T	45.52	0.185	0.172	3.770
V/F(GALS)	71.2	0.134	0.124	1.738



DHA	K_{half} [mM^{-1}]	h	V_{max} [μmol min^{-1} mg^{-1}]	k_{cat} [S^{-1}]	k_{cat}/K_{half} [S^{-1} M^{-1}]
W/L	308.0	1.893	0.540	0.497	1.614
W/T	211.6	2.375	0.162	0.149	0.705
V/F(GALS)	351.1	1.525	0.088	0.081	0.231

Extended Data Figure 9 | Enzyme kinetics of parent GALS and engineered variants. A. Kinetic characterization of GALD formation fitted to Michaelis-Menten model ($v = V_{max} \times [S]/ (K_m + [S])$), where v is initial enzyme velocity, S is substrate concentration, V_{max} is the maximum enzyme velocity, and K_m is the substrate concentration needed to achieve half of the V_{max}). **B.** Kinetic characterization of DHA formation fitted to allosteric sigmoidal model ($v = V_{max} \times S^h/ (K_{half}^h + S^h)$, where v is initial enzyme velocity, S is substrate concentration, V_{max} is the maximum enzyme velocity, K_{half} is the ligand concentration at which half of the active sites are occupied (concentration of half saturation), and h is the Hill coefficient providing a measure of the cooperativity of substrate binding to protein). Both panels compare parent GALS (V281/F282) with top-performing variants (W/L, V281W/F282L; W/T, V281W/F282T), showing kinetic plots (top) and summary tables of derived parameters (bottom). Data represent means of three independent experiments; error bars indicate standard deviation.