IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.151130

NEOARCHITECT: A LOCAL AI-ASSISTED ARCHITECTURAL DESIGN AND SYNTHESIS PLATFORM WITH PRIVACY-PRESERVING STRUCTURAL INTELLIGENCE

Afsal S[1], Aravind A[2], Salini Lalu[3], Abhishek Babu[4], Merlin K Thomas[5], Jinu L[6]

Department of Computer Science and Engineering, College of Engineering Kottarakkara, Kerala, India[1-6]

Abstract: Architectural design requires coordinated decisions about room relationships, circulation, geometry, structural safety, regional aesthetics, materials, cost, and visualization. Conventional computer-aided design workflows often involve repeated manual drafting, while image-oriented generative systems may produce visually plausible plans without preserving editable geometry or engineering context. This paper presents NeoArchitect, a proposed standalone and offline AI-assisted platform that connects natural-language requirement parsing, architectural-style classification, adjacency-constrained two-dimensional vector floor-plan generation, preliminary structural and seismic screening, material and cost estimation, and parametric three-dimensional synthesis. The system is designed to keep project data and model inference on the local machine, reducing dependence on cloud services for core processing. The Phase-I work establishes the problem context, reviews 16 core research works as represented in the project survey, identifies privacy, engineering integration, and regional-style representation as research gaps, and defines a modular architecture and validation strategy. The paper describes the proposed workflow and its key modules. Since the project is at the Phase-I design and planning stage, no measured model accuracy, end-to-end runtime, or validated structural performance is claimed. Engineering outputs are intended only for preliminary decision support and require independent professional verification.

Keywords: Local AI, architectural design, floor-plan generation, natural-language processing, graph-based layout, structural intelligence, privacy-preserving AI, 3D visualization.

I. INTRODUCTION

Architectural planning is a constraint-rich process. A residential plan must accommodate room functions, adjacency, circulation, access, approximate dimensions, daylight, privacy, and structural feasibility. A change in one requirement can affect several others: adding a bedroom may alter corridor geometry, room connections, structural spans, and the external form. Conventional CAD tools provide precision, but they do not eliminate the repeated manual effort involved in coordinating these changes.

Generative artificial intelligence offers a prompt-driven alternative in which users describe a design and receive candidate layouts. However, a generated image is not necessarily an engineering-ready artifact. Raster outputs may not encode exact room dimensions, wall connectivity, openings, or machine-readable geometry. The gap between visual generation and engineering-oriented design motivates a workflow that preserves structured information from the initial requirement through downstream analysis.

NeoArchitect addresses this need through a proposed local platform that combines natural-language processing (NLP), style classification, room-adjacency graphs, vector geometry, preliminary engineering intelligence, and parametric 3D visualization. Its offline-first principle is intended to reduce the need to transmit sensitive prompts and blueprints to external inference services. The project is positioned as an integration-oriented design direction; it does not claim that each constituent technique is individually novel.

The contributions of this paper are: (1) a unified architecture linking user intent to structured floor-plan and engineering outputs; (2) an adjacency- and vector-based representation intended to support editability and traceability; (3) a local-processing design for privacy-conscious operation; and (4) a Phase-II validation plan that separates proposed capabilities from results not yet measured.

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.151130

II. RELATED WORK

Research on controllable vector floor-plan generation emphasizes editable geometric outputs rather than raster-only images [1]. Generative functional layout methods similarly focus on functional organization [2]. Hybrid approaches such as HouseGanDi investigate the balance between sampling time and diversity in floor-plan generation [3]. Text-controlled residential design using diffusion models, ControlNet, and low-rank adaptation demonstrates how semantic input can guide layout generation, although native CAD compatibility and integrated structural checks remain important concerns [4].

Adjacency-driven Pix2Pix methods use symbolic room relationships to guide layout synthesis [5]. Graph-based generation treats rooms as nodes and spatial relationships as edges, providing an explicit representation for functional constraints [6]. Graph2Plan learns floor-plan generation from layout graphs [7], while graph-augmented text generation explores a connection between textual requirements and graph representations [12]. The RPLAN dataset provides a large-scale benchmark for floor-plan learning [13]. These works support the use of structured room relationships, but graph quality and dataset coverage remain important factors.

Other research addresses analysis and perception rather than end-to-end synthesis. LLM-integrated semantic frameworks combine segmentation, object recognition, reasoning, area estimation, and compliance-oriented analysis [8]. Activity-guided floor-plan design incorporates patterns of human activity [9], and multiscale object detection supports recognition of objects in complex architectural plans [10]. Text-to-building studies explore AI-generated 3D geometry [11], while algorithmic space optimization uses explicit rules and parameters [14]. KAIROS investigates cost-efficient machine-learning inference across heterogeneous resources [15].

The project survey identifies three connected gaps: privacy risks associated with cloud-dependent workflows; insufficient integration of structural and seismic considerations into generated layouts; and limited representation of regional architectural styles. NeoArchitect therefore combines local inference, graph-constrained vector generation, preliminary engineering screening, and a style-classification component in one proposed pipeline.

III. PROBLEM STATEMENT AND OBJECTIVES

Existing AI-assisted architectural workflows can exhibit both a deployment disconnect and an engineering disconnect. Cloud-based systems may require private design information to leave the user's environment, while generated layouts are often treated as visual outputs without integrated structural screening, cost estimation, or machine-readable geometry. The problem addressed is: how can a local AI platform convert natural-language architectural requirements into editable vector floor plans while preserving spatial relationships and connecting the plan to preliminary engineering intelligence, material and cost estimation, and 3D visualization?

The project objectives are to develop a local architectural-style classifier, provide an offline desktop workflow, generate CAD-oriented two-dimensional vector plans from textual requirements, and connect those plans to preliminary safety and cost checks and interactive 3D models. Functional objectives include extracting room types and area preferences, constructing adjacency constraints, generating and validating geometry, supporting user review, mapping structural elements, estimating quantities, and exporting design information.

Non-functional objectives include local data handling, modularity, reproducibility, practical performance on local hardware, usability for early design exploration, and exportability. These objectives describe the intended system; they should not be interpreted as evidence that every module has already been implemented or experimentally validated.

IV. PROPOSED SYSTEM ARCHITECTURE

NeoArchitect is organized as a set of cooperating modules connected through structured data interfaces. The user interaction layer accepts design requirements and manages project preferences. The AI processing layer performs language parsing and style classification. The layout layer converts normalized requirements into an adjacency graph and then into vector geometry. After user review and approval, the engineering layer derives preliminary structural, seismic, quantity, and cost information. The visualization layer converts the approved two-dimensional plan into a parametric three-dimensional representation.

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.151130

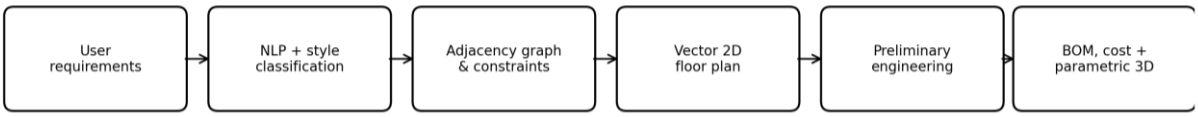


Fig. 1. Proposed end-to-end NeoArchitect workflow.

The database module is intended to store project metadata, requirements, room and adjacency records, plan versions, analysis results, material quantities, and model-export references. A local SQLite database is proposed for structured project records, while larger datasets and model weights may remain in the local file system. The architecture follows five principles: locality of core processing, separation of concerns, structured intermediate representations, human approval before engineering analysis, and traceability of inputs, model versions, and outputs.

The main data flow is: textual requirement → NLP parsing → style classification and constraint normalization → adjacency graph → vector floor plan → user review → engineering analysis → bill of materials (BOM), cost estimate, and 3D synthesis. A common geometry representation is retained across stages so that downstream modules do not need to reconstruct the plan from an image.

V. METHODOLOGY

A. Requirement Parsing and Style Classification

The NLP module is intended to extract room categories, room counts, approximate area preferences, and user preferences from a natural-language prompt. The extracted information is normalized into a structured requirement object. Ambiguous or conflicting constraints should be exposed for user confirmation rather than silently discarded. A style-classification model based on a locally executed PyTorch convolutional neural network is proposed to identify the requested architectural motif. The Phase-I material reports the curation of a multi-class architectural image collection covering six global and regional styles; the complete class taxonomy and measured classifier performance are not specified in the report.

B. Adjacency Graph and Vector Layout Generation

Rooms are represented as graph nodes and required spatial relationships as edges. Edge attributes can encode the type or strength of an adjacency requirement. This graph preserves functional intent independently of the final room coordinates. A layout generator uses the graph, site boundary, approximate dimensions, and area constraints to place rooms and construct a two-dimensional vector plan. The intended output is a structured collection of room polygons, walls, doors, windows, and associated metadata rather than a raster-only image.

Geometry validation is designed to detect overlapping rooms, disconnected spaces, insufficient dimensions, unclosed polygons, missing entrances, and violated adjacency constraints. Deterministic checks should be applied wherever possible. Minor coordinate inconsistencies may be repaired using explicit rules, whereas fundamental conflicts should trigger regeneration or user intervention. A circulation graph can additionally represent movement paths between room entrances and the main access point.

C. Preliminary Structural, Seismic, Material, and Cost Analysis

The engineering pipeline consumes an approved two-dimensional plan. It maps walls and openings and places conceptual structural elements such as columns, beams, and slabs according to documented project assumptions. For preliminary load reasoning, total load is represented as $W = W_D + W_L$, where W_D denotes dead load and W_L denotes live or imposed load. Exact design values must be drawn from validated standards and the assumptions of the implementation. The Phase-I report identifies a seismic index and IS-code-oriented quantity calculations as intended components, but it does not provide a final seismic-index equation or a complete set of verified calculation clauses. Accordingly, this paper does not assign numerical seismic results or claim code compliance. Material quantities are intended to feed a bill of materials, while a locally maintained rate table can support deterministic cost estimates. Every estimate should retain its input dimensions, assumptions, unit rates, and calculation trace.

D. Parametric 3D Synthesis and Local Execution

The approved two-dimensional geometry is the authoritative input to the 3D stage. Room boundaries and wall segments can be extruded into parametric meshes, with openings and structural metadata mapped to corresponding 3D components. This approach is intended to preserve consistency between the plan and the model. An interactive viewer and export panel are proposed for inspection and reporting.

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.151130

Core model inference, project storage, and geometry processing are intended to operate locally. Local execution can reduce the need to send architectural data to cloud services, but it is not an absolute security guarantee. The privacy claim must be demonstrated through network-disabled operation and controlled network-monitoring tests.

VI. PHASE-I STUDY AND EVALUATION PLAN

The Phase-I work established the project context, surveyed related research, identified the research gaps, and defined the modular architecture and implementation direction. The supplied report records a survey of 16 core papers and a multi-class architectural image collection spanning six global and regional styles. It also describes room-adjacency algorithms, modular connections between NLP, layout generation, and analytics, and quantity-formula work aligned with the project's IS-code-oriented direction. These are reported Phase-I activities, not evidence of completed end-to-end system performance.

The next phase is intended to implement and evaluate the individual modules before full integration. NLP tests should measure whether required rooms and relationships are extracted consistently. Style classification should be evaluated on held-out data using documented metrics. Layout testing should measure constraint satisfaction, geometric validity, adjacency accuracy, and circulation reachability. Structural and cost calculations should be compared with trusted reference cases and reported with assumptions and tolerances. The 2D-to-3D pipeline should be checked to ensure that major geometry is preserved after extrusion.

Offline reliability should be tested with network access disabled. Export tests should confirm that generated DXF, SVG, and 3D files can be reopened correctly. Performance measurements should report NLP parsing, style classification, layout generation, engineering analysis, mesh generation, and end-to-end response times separately for CPU-only and GPU-enabled configurations where supported. User evaluation should assess usability and whether the integrated workflow reduces repeated early-stage drafting effort.

The Phase-I schedule covered 1 July to 30 October 2026. The recorded activities included topic selection, literature study and survey, objective formulation, feasibility study, hypothesis and methodology, work planning and task allocation, and documentation. Phase-II work is planned around dataset refinement, NLP and style-model development, 2D layout generation, structural and cost modules, 3D/UI integration, and end-to-end testing.

VII. DISCUSSION

The main design choice in NeoArchitect is to treat architectural generation as an engineering workflow rather than a picture-generation task. Retaining requirement objects, adjacency graphs, and vector geometry provides a traceable path from user intent to derived quantities and visualization. This also creates clear interfaces between independently developed modules and allows deterministic validation to complement learned generation.

The proposed integration has several practical challenges. Model quality depends on dataset diversity and annotation quality; regional styles may be underrepresented in available benchmarks. Graph errors can propagate into geometry, and invalid geometry can undermine structural and 3D processing. Structural screening is particularly sensitive to assumptions, load definitions, code clauses, and building context. Cost estimates depend on the accuracy and freshness of local unit rates. Local inference may also be constrained by GPU memory and hardware availability.

The system must therefore distinguish model predictions, deterministic geometry checks, preliminary engineering calculations, and professional approvals. Its outputs are not substitutes for licensed architectural or structural review. The Phase-I report does not provide measured model accuracy, a fully enumerated six-style taxonomy, exact seismic equations, validated structural results, or a regional price database. These limitations define the evidence required in the next development phase.

VIII. CONCLUSION

This paper presented NeoArchitect, a proposed offline AI-assisted platform for architectural design and preliminary engineering intelligence. The platform connects natural-language requirement extraction, architectural-style classification, graph-constrained floor-plan generation, vector geometry, preliminary structural and seismic screening, material and cost estimation, and parametric 3D synthesis. Its central design contribution is the integration of these stages while retaining structured information and keeping core processing local.



International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026
DOI: 10.17148/IJARCCE.2026.151130

At the current Phase-I stage, the project has established its research foundation, identified gaps in privacy, engineering integration, and regional-style representation, and defined a modular architecture and validation plan. No measured end-to-end performance or certified engineering outcome is claimed. Future work must implement the modules, validate geometry and calculations against reference cases, demonstrate offline behavior, and evaluate usability and performance on representative projects. All engineering outputs must remain preliminary until independently verified by qualified professionals.

ACKNOWLEDGMENT

The authors gratefully acknowledge the guidance and support provided by the Department of Computer Science and Engineering, College of Engineering Kottarakkara, and the project guide, Ms. Merlin K Thomas, throughout the Phase-I study.

REFERENCES

[1]. S. Wang and R. Pajor, "Directly from alpha to omega: Controllable end-to-end vector floor plan generation," IEEE Transactions on Visualization and Computer Graphics, 2026.
[2]. M. Abouagour and E. Garyfallidis, "Gflan: Generative functional layouts," arXiv preprint arXiv:2512.16275, 2026.
[3]. A. B. Yenew, B. G. Assefa, and E. G. Belay, "HouseGanDi: A hybrid approach to strike a balance of sampling time and diversity in floorplan generation," 2025.
[4]. Z. Qiu, J. Liu, Y. Xia, H. Qi, and P. Liu, "Text semantics to controllable design: A residential layout generation method based on stable diffusion model," 2025.
[5]. M. F. H. Rezve and S. Chowdhury, "Adjacency-driven floor plan generation with pix2pix: A hybrid user-centered workflow," 2025.
[6]. Y. Takeuchi, Q. An, and A. Yamashita, "Single-stage graph-based floorplan generation," 2025.
[7]. N. Nauata, K.-H. Chang, C.-Y. Cheng, G. Mori, and Y. Furukawa, "Graph2Plan: Learning floorplan generation from layout graphs," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2021.
[8]. Y. Guo, X. Zhou, and S.-K. Tang, "LLM-integrated semantic deep learning framework for automated floor plan analysis, area estimation, and compliance assessment of existing buildings," 2025.
[9]. S. Wang, W. Zeng, X. Chen, Y. Ye, Y. Qiao, and C.-W. Fu, "ActFloor-GAN: Activity-guided adversarial networks for human-centric floorplan design," 2021.
[10]. Z. Xu, N. Jha, S. Mehadi, and M. Mandal, "Multiscale object detection on complex architectural floor plans," 2024.
[11]. G. Bono, "Text-to-building: Experiments with AI-generated 3D geometry for building design and structure generation," 2024.
[12]. Y. Wei and X. Li, "Graph-augmented text-based floorplan generation," 2025.
[13]. W. Wu, X.-M. Fu, R. Tang, Y. Wang, Y.-H. Qi, and L. Liu, "A large-scale data set and benchmark for learning to design floorplans," IEEE Transactions on Pattern Analysis and Machine Intelligence, 2021.
[14]. J. Cudzic and M. Nessel, "Algorithmic space optimization in building adaptations: Generative design comparison," 2024.
[15]. B. Li, S. Samsi, V. Gadepally, and D. Tiwari, "KAIROS: Building cost-efficient machine learning inference systems with heterogeneous cloud resources," in Proceedings of the ACM International Symposium on High-Performance Parallel and Distributed Computing, 2023.

 This work is licensed under a Creative Commons Attribution 4.0 International License