FLOORPLAN RETRIEVAL AND GENERATOR

UIT2718 PROJECT WORK PHASE II

PROJECT REPORT

Submitted by

Arshat Parvaes B (3122215002014)
Arunkumar A (3122215002016)

Guided by
Dr. D. Preetha Evangeline

in partial fulfilment for the award of the degree of

BACHELOR OF TECHNOLOGY IN INFORMATION TECHNOLOGY

DEPARTMENT OF INFORMATION TECHNOLOGY

Sri Sivasubramaniya Nadar College of Engineering
(An Autonomous Institution, Affiliated to Anna University)

MAY 2025

Sri Sivasubramaniya Nadar College of Engineering
(An Autonomous Institution, Affiliated to Anna University)

BONAFIDE CERTIFICATE

Certified that this Report titled **FLOORPLAN RETRIEVAL AND GENERATOR** is the bonafide work of **Arshat Parvaes B (3122215002014)**, **Arunkumar A (3122215002016)** who carried out the work under my supervision. Certified further that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

Dr. A. SHAHINA
Professor and Head of Department
Department of Information Technology
SSN College of Engineering
Kalavakkam – 603110

Dr. D. PREETHA EVANGELINE &
Dr. K. NEPOLEON
Assistant Professor
Department of Information Technology
SSN College of Engineering
Kalavakkam – 603 110

Submitted for project viva-voce examination held on .....

EXTERNAL EXAMINER INTERNAL EXAMINER

ABSTRACT

This work introduces an AI-driven solution for floor plan retrieval and generation, addressing the dual need for rapid search and flexible creation. To enable efficient floor plan retrieval, we leverage BERT embeddings to transform user queries and floor plan metadata into high-dimensional vector representations that preserve both semantic and spatial information. These embeddings are indexed in a FAISS-based vector database, ensuring rapid and precise similarity searches. While this retrieval method significantly enhances efficiency, its accuracy is contingent on data quality. To improve generalization and diversity, we employ data augmentation strategies that expand the system's capacity to match retrieved plans to user requirements effectively.

For a more adaptive and precise floor plan generation, we extend our approach by integrating simulated annealing with Voronoi diagrams. The floor plan interior is partitioned into six directional regions, with dynamic expansion rates controlling room growth. To correct circular artifacts from Voronoi cells, we apply minimum enclosing rectangles, ensuring accurately shaped rooms. Additionally, a flood-fill algorithm eliminates overflow beyond the designated boundary, maintaining spatial integrity. While this generation technique allows for the creation of realistic and complex floor plans, it comes with increased computational costs compared to retrieval-based methods.

By combining these complementary retrieval and generation techniques, our system provides a comprehensive solution for floor plan exploration and synthesis. The retrieval process enables users to quickly find existing plans that match their needs, while the generation approach ensures the creation of unique and adaptable layouts. This integrated AI-based framework enhances both accessibility and customization, making it a powerful tool for architects, designers, and homeowners seeking efficient floor plan solutions.

ACKNOWLEDGEMENT

I thank **ALMIGHTY GOD** who gave me the wisdom to complete this Project. My sincere thanks to our beloved founder **Dr. SHIV NADAR, Chairman, HCL Technologies.** I also express my sincere thanks to **Ms. KALA VIJAYAKUMAR,** President, SSN Institution and our Principal **Dr. S. RADHA**, for all the help he has rendered during this course of study.

We are highly indebted to **Dr. A. Shahina, Head of the Department** for providing us with the opportunity and facilities to take up this project.

I am deeply obliged and indebted to the timeless help and guidance provided by **Dr. D. Preetha Evangeline, Assistant Professor**, Department of Information Technology and also expresses my heartfelt thanks for making this project a great success.

I also thank all the faculty of the Department of Information Technology for their kind advice, support and encouragement and last but not the least I thank my parents and my friend for their moral support and valuable help.

Arshat Parvaes B **Arunkumar A**

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVATIONS

NO.	ACRONOYM	ABBREVATIONS
1.	AI	Artificial Intelligence
2.	BFS	Breadth First Search
3.	NLP	Natural Language Processing
4.	CSV	Comma Separated Values
5.	BERT	Bidirectional Encoder Representations from Transformers
6.	FAISS	Facebook AI Similarity Search
7.	VQ-VAE	Vector Quantized Variational Autoencoder
8.	YOLO	You Only Look Once
9.	CAD	Computer Aided Design
10.	CNN	Convolutional Neural Network
11.	GNN	Graph Neural Network
12.	GAN	Generative Adversarial Network
13.	DCGAN	Deep Convolutional Generative Adversarial Network
14.	LLM	Large Language Model
15.	COT	Chain Of Thought
16.	CLIP	Contrastive Language-Image Pretraining
17.	IOU	Intersection Of Union
18.	BLP	Boundary Leakage Percentage

CHAPTER 1
INTRODUCTION

1.1 OVERVIEW:

Over the past few years, the architectural profession has adopted recent advancements in technology to meet the increasing demand for effective design processes and tailored user experiences. The conventional method of making architectural design, where floor plans are hand-drawn by considering the verbal or written descriptions, is bound to consume much time, skills, and resources. With the increasing passion for automation and artificial intelligence, there has been a trend of creating systems that simplify the process of floor plan selection and design using natural language processing (NLP) and machine learning to interpret text-based descriptions into architectural plans and allow automatic generation of functional floor plans.

Our work targets this newly evolving field with a goal of developing an intelligent floor plan system that not only looks for architectural designs from descriptive queries but also creates personalized designs from user input. Through the embedding of floor plan descriptions as well as user queries into a vector space using BERT embeddings, we enable smooth similarity search in a FAISS-based vector database, providing efficient retrieval of matching layouts in terms of semantic similarity to user queries. In pursuit of robustness, we implement data augmentation strategies to amplify the diversity of potential descriptions so that the system becomes more accommodating to diverse input styles. Subsequently, on top of this foundation, we expand the ability of the system by adding a generative model with simulated annealing to generate optimized floor plans. Through integration of Voronoi diagrams to enable spatial partitioning, the optimization of floor plans using minimum enclosing rectangles, and boundary conformity ensured through the application of a flood-fill algorithm, our process creates well-organized, functional floor plans meeting architectural constraints without loss of aesthetic consistency.

Our approach aims to overcome the weaknesses of conventional, human-labor-intensive floor plan design processes and keep pace with the architectural profession's shift towards data-driven, AI-facilitated solutions. Through an intuitive and user-friendly interface, this system enables users to make design decisions that respond to given spatial needs and aesthetic tastes by recovering existing designs or creating new ones. The desired result is a tool not only that streamlines the floor plan recovery and generation process but also that pushes

the overall discourse on AI utilization in architectural planning and provides insights in line with the prevailing trend in automation and intelligent design.

1.2 MOTIVATION:

Floor plan designing and visualizing is an important phase in designing useful and attractive residential and commercial buildings. The conventional processes of choosing or creating floor plans tend to be highly problematic. Current software packages are normally complicated, involving specific architectural know-how and a lot of manual effort. For people who lack professional designing experience, such software can be daunting, labor-intensive, and not very intuitive, restraining access and creativity.

Furthermore, the procedure of selecting or designing an appropriate floor plan is iterative, revised, and refined several times according to changing spatial needs and individual taste. Traditional tools do not have seamless interaction capabilities, which impede users from easily exploring and refining alternatives. The disconnect between user intention and system outcome also causes difficulties, resulting in frustration and undesirable design decisions.

Our project overcomes these shortcomings with the introduction of an intelligent, AI-based system that merges retrieval and generation to automate the process of designing floor plans. By using a chat-based interface, users are able to describe their needs in natural language easily, enabling the AI to look up existing floor plans that fit their needs or create new ones that are specifically tailored to their requirements. The system takes advantage of BERT embeddings for semantic comprehension, FAISS-based similarity search for fast retrieval, and simulated annealing with Voronoi-based segmentation to dynamically create structured and optimized layouts. Through data augmentation techniques incorporated in our system, adaptability is increased so that a broad variety of user input can be handled and translated effectively.

With the integration of cutting-edge natural language processing, AI-based retrieval, and computational design methods, our system enables users to:

Effortlessly Fetch and Create Floor Plans: From finding a pre-existing design to designing a new one, users can do it all with straightforward conversational input.
Iterate and Refine Designs Effectively: The dynamic interactive AI enables users to dynamically modify layouts, minimizing the hassle of manual changes.
Make Data-Driven and Personalized Choices: With AI-driven suggestions and personalized design generation, users can discover a range of spatial configurations specific to their needs.

This project capitalizes on innovation in AI-based retrieval-augmented generation, machine learning spatial optimization, and conversational interfaces to revolutionize how users think about and engage with architectural floor plans. By filling the gap between professional design knowledge and human-readable AI-powered interfaces, our system increases accessibility, efficiency, and fun in floor plan selection and creation. Ultimately, this solution democratizes architectural design, allowing users to envision, personalize, and bring their perfect spaces to life without technical hindrances.

1.3 BACKGROUND:

Floor plan designing has historically been a painstaking task involving expert skills and professional equipment in the form of Computer-Aided Design (CAD) software. Though such equipment offers feature-rich packages for experts like architects and designers, they are out of reach for non-professionals by virtue of their steep learning curve and technical nature. For individuals without formal training in architecture or design, navigating such software can be time-consuming and discouraging, ultimately limiting creativity and experimentation. Additionally, even experienced professionals must undergo repetitive trial-and-error iterations to refine layouts based on evolving constraints and preferences, making the process inefficient.

Recent developments in Artificial Intelligence (AI) and Machine Learning (ML) across Natural Language Processing (NLP), retrieval-based models, and generative algorithms offer new possibilities to automate and augment the process of floor plan design. AI-based methods facilitate semantic interpretation of user input, permitting users to explain their spatial needs using natural language instead of using sophisticated design interfaces. In addition, simulated annealing-based generative models, Voronoi-based segmentation, and spatial optimization open up the possibility of generating structured and aesthetically sound layouts dynamically.

This project takes these developments and makes a bridge of sorts between manual architectural design procedures and AI-led automation. The system combines the use of a chat-based UI, BERT embeddings for semantically retrieving results, and generative algorithms to create personalized floor plans with it. The system converts user feedback into usable, aesthetically pleasing layouts in this manner. This not only makes the designing process more open to a mass audience but is also more efficient and decision-led for both novices and professionals.

Through the integration of retrieval-based floor plan matching and AI-facilitated generation, our system provides a new, intuitive, and user-friendly mechanism for engaging with architectural layouts. This technology conforms to current AI-enabled automation and smart design paradigms by providing a scalable and adaptive answer to the drawbacks of conventional floor plan design.

1.4 PROBLEM DEFINITION:

To create a chat-based interface for facilitating real-time collaboration between clients and AI to create and tailor architectural floor plans. Users can specify or draw the boundary of the layout using an easy-to-use map interface or input boundary specifications in chat. The AI produces several floor plan designs, which the user can shape and adjust using conversational interaction. Following iterative refinements, the user settles on a satisfactory design directly via the interface.

1.5 OBJECTIVES:

The goal of this research is to create an AI-based system for floor plan retrieval and generation, utilizing natural language processing (NLP) and machine learning to simplify architectural design. Our main goal is to allow users to retrieve existing architectural layouts efficiently and generate optimized floor plans from textual descriptions, simplifying the complexity and expertise involved in conventional design tools.

We will do this by:

1. Build a floor plan retrieval system employing BERT embeddings and FAISS-based similarity search, where users can search for architectural floor plans that best describe their inputs in words.
2. Deploy a generative model based on simulated annealing and Voronoi-based spatial partitioning to generate new, functionally optimal floor plans with custom specifications provided by the users.
3. Increase flexibility and precision through data augmentation processes, enhancing both retrieval and generation models' capabilities to deal with various input forms.
4. Design a natural language-based conversation interface, permitting users to directly engage with the system without necessitating complicated design tools.
5. Measure system performance based on retrieval accuracy, generative page coherence, and user satisfaction as compared to current design practices.

By means of this research, we hope to fill the gap between architectural design and AI by simplifying the selection and generation of floor plans, making them

more accessible, efficient, and friendly to users. Through the combination of semantic retrieval and generative design, our system provides a scalable and flexible solution that follows existing trends in AI-driven automation and smart spatial planning

1.6 ORGANIZATION OF THE REPORT:

The organization of this report is written to present an overall and orderly picture of the project, starting from its motivation, methodology, experiments, to its results. Chapter 1 is used as the introductory chapter, wherein the objectives of the project, motivation, background, and good problem statement are presented. This sets the priority of automating floorplan extraction and generation with its applicability in architectural design and real applications.

Chapter 2 gives an in-depth survey of the literature, summarizing current approaches to floorplan retrieval and creation, such as graph-based models, generative adversarial networks (GANs), and optimization techniques. It identifies research gaps and forms the basis for our suggested technique.

Chapter 3 is dedicated to the description of the dataset, that is, the Tell2Design (T2D) dataset, which is a collection of floorplan images along with textual descriptions. It covers dataset analysis, distribution, and preprocessing, guaranteeing an organized method to prepare the data.

Chapter 4 outlines the system architecture and design, discussing the retrieval and generation pipeline. It addresses the important features like FAISS for similarity search, optimization using Simulated Annealing, and incorporating Gemini LLM for user interaction. The block diagram gives an idea of the overall workflow of the system.

Chapter 5 describes experiments and their results for comparing techniques tried out for floorplan retrieval and generation. CLIP embeddings, FAISS vector search, deep generative models such as DCGAN and StackGAN, bin packing for room placement, HouseGAN++ for structured layout synthesis, and Voronoi-based segmentation methods are explained. The weaknesses and issues encountered in each experiment are also noted.

Chapter 6 compares the outcomes and system performance with predetermined measures like Micro IoU and Macro IoU to provide a quantitative measure of retrieval accuracy and generative efficiency.

Chapter 7 presents the social relevance and sustainability of the project, which can be used in smart architecture, AI-assisted design automation, and urban planning.

Chapter 8 wraps up the report by providing an overview of significant contributions, challenges, and findings, as well as potential future enhancements. Recommendations include combining diffusion models, enhancing user-guided generation, and optimizing techniques for improved architectural realism.
This organized approach guarantees readability and ease of understanding, rendering the report accessible to both technical and non-technical audiences.

CHAPTER 2

LITERATURE SURVEY

1. Architectural Layout Design Optimization:

The paper explores automated optimization of architectural floor plans using gradient-based and evolutionary algorithms, focusing on design exploration, scalability, and integrating human decision-making. Two algorithms are employed: geometry optimization, effective for large problems but dependent on initial conditions, and topology optimization, which identifies optimal layouts for the best geometry. Future improvements include incorporating new shapes, constraints, and refining topology definitions to enhance results, balancing efficiency, aesthetics, and usability in complex scenarios.Two optimization algorithms, geometry and topology, are used for design layouts. The geometry algorithm is effective for large problems but relies on initial conditions. A topology algorithm enhances this by finding optimal topologies for the best geometry. Improvements include adding new shapes, constraints, and refining topology definitions to improve results.

2. Data-driven Interior Plan Generation for Residential Building:

The study explores AI-driven automated floor plan generation, aiming to streamline design processes and improve accessibility for non-professionals.It introduces a hybrid model combining sequential deep learning for room creation and graph-based techniques for room adjacencies.The proposed approach outperforms isolated and average-based models in generating optimized layouts. Future enhancements will focus on addressing limitations in room shapes to further refine the system.This method efficiently generates residential floor plans using a two-stage approach and a dataset of over 80K plans. It mimics human design and outperforms existing methods. Future improvements include handling

additional constraints and expanding to multi-story and varied building types, such as offices and malls.

3. House-GAN: Relational Generative Adversarial Networks for Graph-constrained House Layout Generation:

This research presents House-GAN, a graph-constrained generative adversarial network for generating diverse and realistic house layouts from bubble diagrams. It encodes architectural constraints into a relational graph structure and demonstrates superior performance in realism, diversity, and compatibility compared to existing methods, enhancing automated architectural design processes. This paper proposes a house layout generation problem and a graph-constrained relational GAN solution. It defines metrics (realism, diversity, compatibility) and shows superior performance over existing methods. The work advances computer-aided design of house layouts, with plans to share code and data.

4. Graph2Plan: Learning Floorplan Generation from Layout Graphs:

The paper presents Graph2Plan, a deep learning framework for automated floorplan generation that incorporates user-defined constraints through layout graphs, enabling customizable designs. It demonstrates effective qualitative and quantitative evaluations, showcasing the framework's ability to generate diverse and plausible floorplans based on user inputs.This deep learning framework for floorplan generation allows user input to guide the design through layout graphs. It enables room placement adjustments but faces limitations in constraints, accessibility, and boundary matching. Future improvements include adding functionality for furniture design and structural refinements.

5. FLNET: Graph constrained Floor Plan Generator:

The research presents FLNet, a generative model for automated floor plan design that incorporates user constraints like boundaries and room types. Evaluated on 80,000 professional layouts, FLNet outperforms existing methods in accuracy, realism, and quality, streamlining the design process without requiring extensive expertise.FLNet is a generative model that creates floor layouts based on user constraints like boundary, room types, and spatial relationships. It outperforms previous methods in accuracy, realism, and quality, removing the need for post-processing. Future work includes generating 3D models with furniture placement.

6. FloorGAN: Generative Network for Automated Floor Layout Generation:

FloorGAN, a generative adversarial network, synthesizes realistic floor plans

guided by user constraints on room types and spatial relationships. It outperforms existing methods in generating diverse and compatible layouts, as evaluated on a large dataset of 80,000 residential floor plans.This research paper proposes FloorGAN, a model that generates floor plans based on user constraints like room types and spatial relationships. Evaluated on 80,000 RPLAN data, it outperforms existing methods in realism, diversity, and compatibility. Future work will include generating 3D models and furniture placements for enhanced realism.

7. Tell2Design: A Dataset for Language-Guided Floor Plan Generation:

The research paper introduces Tell2Design (T2D), a dataset for generating floor plans from natural language instructions. It comprises over 80,000 designs and emphasizes the challenges of incorporating spatial and relational constraints inherent in architectural layouts. The authors propose a Sequence-to-Sequence model as a benchmark for this task and conduct evaluations comparing their approach with existing text-conditional image generation models, highlighting the complexity of aligning generated designs with user specifications.This paper introduces a novel language-guided design generation task focused on floor plans, using the Tell2Design (T2D) dataset with natural language instructions. We propose a Seq2Seq model, compare it with text-conditional image generation models, and highlight challenges in aligning text with design.

8. Zero-shot Sequential Neuro-symbolic Reasoning for Automatically Generating Architecture Schematic Designs:

This paper presents a novel automated system for generating architectural schematic designs using a sequential neuro-symbolic reasoning approach. By integrating generative AI and mathematical solvers, it streamlines complex decision-making in multifamily real estate development, enabling the rapid creation of diverse building designs tailored to neighborhood characteristics without manual cost function crafting. The method's effectiveness is validated through comparative studies with real-world buildings.This paper introduces a sequential neuro-symbolic approach for generating apartment floor plans, combining foundational models with symbolic techniques for valid designs. Our method shows promise in transforming real- estate workflows and sets the stage for future research in generative design, addressing challenges in traditional methods.

9. Generative Floor Plan Design Using Deep Learning: An AI-Powered Approach:

This research paper explores the use of AI in architectural design, focusing on automated floor plan generation. It proposes a hybrid model combining sequential

and graph-based approaches to create floor plans. The system processes user inputs to produce tailored designs, considering factors like room layout and space optimization. The study aims to streamline architectural processes and make design more accessible to non-professionals.This research paper proposes a hybrid model combining deep learning and graph techniques for floor plan generation. This approach integrates a sequential machine learning model for room creation with a graph model for room adjacencies, outperforming isolated components and average-based models. Future improvements will address limitations in room shapes.

10. MSD: A Benchmark Dataset for Floor Plan Generation of Building Complexes:

This research introduces the Modified Swiss Dwellings (MSD) dataset, a benchmark for floor plan generation in building complexes. It highlights the challenges faced by existing graph-based models when applied to intricate multi-apartment layouts. Evaluations of Modified HouseDiffusion (MHD) and Graph-informed U-Net reveal their limitations in handling large-scale complexity. The study emphasizes the need for advanced graph-based approaches to improve realism and adaptability in automated architectural design, contributing to the evolution of AI-driven floor plan generation.

11. Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding:

This research presents Imagen, a state-of-the-art text-to-image generation model that leverages diffusion techniques and large language models. Imagen surpasses existing models like DALL-E 2 and GLIDE in both image fidelity and text alignment, as confirmed by human evaluations. The study demonstrates that scaling the text encoder and optimizing model architecture significantly enhance output quality. By setting a new benchmark in text-to-image synthesis, Imagen highlights the potential of combining advanced language models with innovative training methodologies to improve generative performance.

12. Learning Transferable Visual Models From Natural Language Supervision:

This research introduces CLIP (Contrastive Language-Image Pretraining), a vision model trained on 400 million image-text pairs using a contrastive learning approach. Unlike traditional models that rely on labeled datasets, CLIP learns directly from natural language supervision, enabling zero-shot learning for diverse vision tasks without task-specific training. The model demonstrates

competitive performance across 30+ benchmarks, excelling in object recognition, OCR, and action recognition. CLIP's robustness to distribution shifts and strong generalization capabilities mark a major advancement in vision-language learning. However, its limitations include lower performance in specialized domains like medical imaging. Future research aims to enhance zero-shot learning and improve fine-tuning integration for broader applications.

13. GENPLAN:

This research introduces GenPlan, a deep learning framework for automated architectural floor plan generation, integrating CNNs and GNNs to enhance room placement flexibility and precision. Trained on a diverse dataset of real-world floor plans, GenPlan improves design efficiency while minimizing common flaws such as trapped rooms and windowless spaces. This advancement provides architects and developers with a powerful tool for exploring creative yet functional layouts. Beyond architecture, GenPlan's applications extend to urban planning, robotics, gaming, and film, highlighting its broad impact across multiple industries.

14. HouseDiffusion: Vector Floorplan Generation via a Diffusion Model with Discrete and Continuous Denoising:

This research introduces HouseDiffusion, a diffusion-based model for generating high-quality vector floor plans from input bubble diagrams. Addressing limitations in previous models like House-GAN++, HouseDiffusion incorporates advanced attention mechanisms and a unique denoising process to enhance room configurations and wall structures. The model excels in generating both Manhattan and non-Manhattan layouts, significantly improving diversity, compatibility, and realism. Its effectiveness establishes it as a major advancement in architectural design automation, paving the way for more precise and flexible floor plan generation.

15. HouseLLM: LLM-Assisted Two-Phase Text-to-Floorplan Generation:

This research introduces HouseLLM, a two-phase text-to-floorplan generation framework that leverages Large Language Models (LLMs) and conditional diffusion models. Utilizing Chain of Thought (CoT) prompting, HouseLLM enhances LLM reasoning, enabling the generation of initial layouts based on user specifications, which are later refined into final designs. The model outperforms existing methods like HouseDiffusion, demonstrating superior diversity and performance. By integrating LLMs with diffusion models, HouseLLM represents a significant advancement in automated house layout generation, improving both creativity and functional design precision.

CHAPTER 3

DATASET DESCRIPTION

3.1 DATA INTRODUCTION:

Our work draws upon the Tell2Design (T2D) dataset, a large dataset designed for use in language-conditioned floor plan retrieval and generation. The dataset consists of more than 80,000 floor plan designs along with natural language descriptions, representing a rich dataset with which to train models to recognize and produce architectural layouts given text input.

3.2 DATASET ANALYSIS:

3.2.1 DATASET COMPOSITION:

The T2D dataset offers a structured correspondence between natural language descriptions and floor plan layouts. Every floor plan is associated with a textual description that includes:

- **Semantics:** The functional use of each room (e.g.”Kitchen for cooking”, ”Bathroom with attached shower”).
- **Geometry:** Room size, aspect ratio, and orientation (e.g.”A bedroom of size 10×12 feet on the east side”).
- **Topology:** Spatial relationship between rooms (e.g.”The kitchen is next to the dining room”).

3.2.2 FLOORPLAN REPRESENTATION:

Each floor plan is represented by a 256×256 image, where different pixel values represent different room types. The dataset includes 8 primary room categories: Living room, kitchen, master bedroom, bathroom, balcony, dining room, storage, and common room. The room-type names and bounding boxes are derived from these images to facilitate both retrieval and generation tasks.

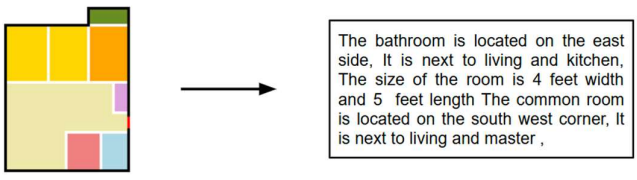

Fig 3.1 A Sample Data from Tell2Design Dataset

Fig 3.2 Color Code

3.2.3 LANGUAGE ANNOTATIONS:

The data set contains two forms of text-based descriptions:

- **Human-Annotated Instructions (5,051 samples):** Collected using Amazon Mechanical Turk (MTurk), where annotators wrote free-text descriptions of floor plans, such as room semantics, geometry, and topology.
- **Synthetic Instructions (75,737 samples):** Constructed based on pre-existing templates to render descriptions consistent and comprehensive.

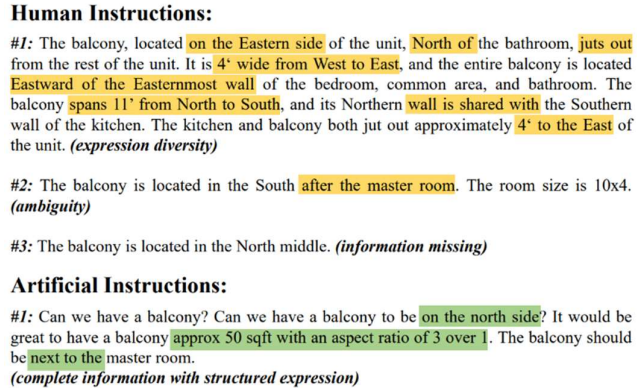

Fig 3.3 Human instructions vs Artificial instructions

3.2.4 LANGUAGE INSTRUCTION ANALYSIS:

To get a better grasp of the Tell2Design dataset, we study some of the aspects of its language instructions.

Every floor plan in the dataset has descriptive textual instructions, which are organized into about **11 sentences** per example, with an average of **200 words**. Per room, descriptions have about 30 words spread over more than two sentences. The dataset also contains **human-written and artificially created instructions**, with the latter tending to have a slightly higher word count.

	Human	Artificial
Avg. # words per instance	200.30	260.47
Avg. # sent. per instance	11.89	23.46
Avg. # words per room	29.48	38.44
Avg. # sent. per room	1.75	3.46

Table 1: Language instruction statistics.

3.2.4 DATASET COMPARISION:

To place our T2D dataset in context, we observe its primary differences
In relation to other concerned datasets for other similar generation purposes T2D is distinguished from other datasets from the following different aspects: (1) T2D is the earliest mega-scale dataset proposed to generate designs (i.e., floor plans) directly based on users' natural language inputs; (2) Text annotation lengths of T2D are significantly longer than those in other text-conditional generation benchmarks, i.e., 256 words per instance; (3) All texts within T2D are typed and created artificially by humans or programmed artificially, as opposed to those crawled on the internet.

3.3 DATASET DISTRIBUTION:

Below are the plots representing the T2D dataset distributions:

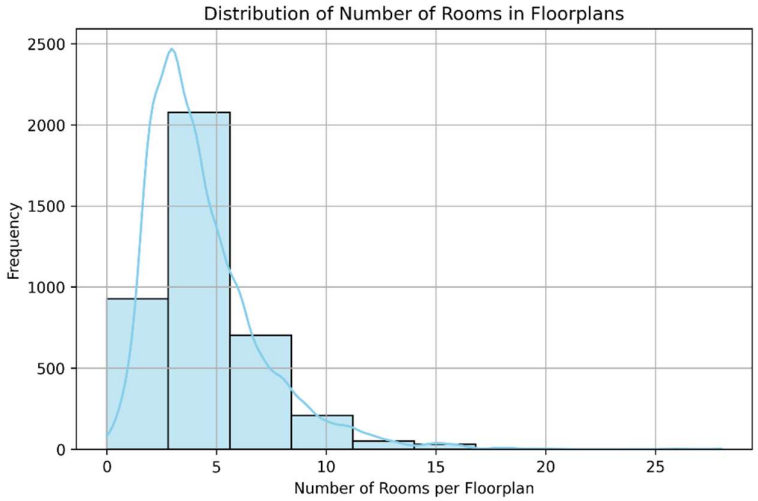

Fig:3.4: Histogram of Room Counts in Floorplans

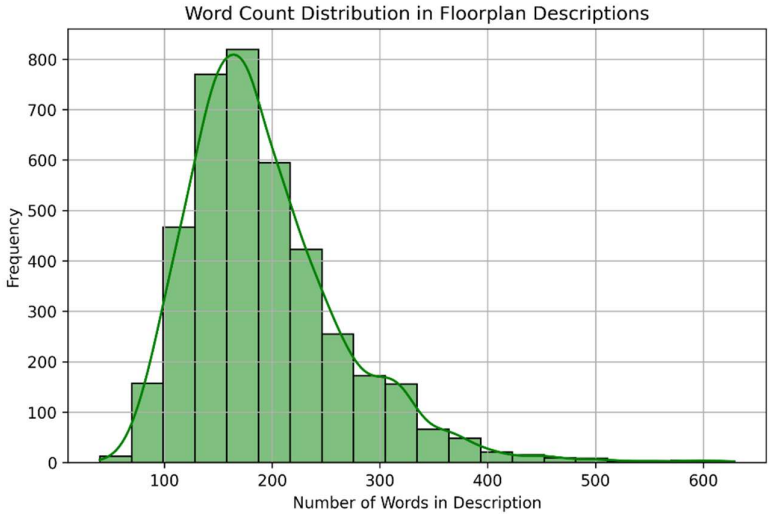

Fig:3.5: Word Count Distribution in Floorplan Descriptions

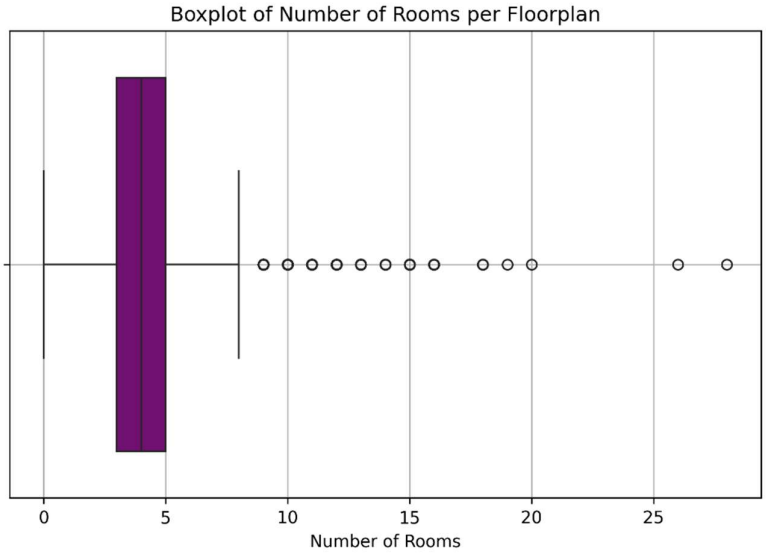

Fig:3.6: Boxplot of Room Counts in Floorplans

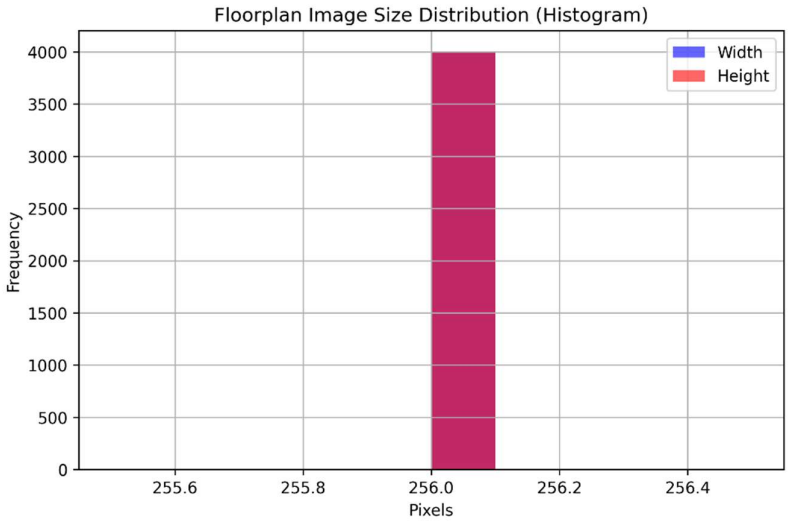

Fig:3.7: KDE Plot of Floorplan Image Sizes

3.4 DATA COLLECTION AND PREPARATION:

The dataset for this project was created using the **Tell2Design** dataset as the source, followed by processing steps to format it for fine-tuning. The steps involved are outlined below:

3.4.1. Loading the Dataset:

The Tell2Design dataset, provided in JSON format, was loaded into a Pandas DataFrame for further manipulation.

3.4.2. Preparing Input Data:

- A function was created to prepare input strings by concatenating key fields: `annotated_strings`, `boundary`, and `boundary_boxs`.
- The `boundary_boxs` field was flattened into a readable format, where each item included room type, dimensions (x, y, height, and width), and additional metadata, ensuring a structured and descriptive input string.

3.4.3. Formatting Output Data:

- Each floor plan's `rooms` field was parsed into a detailed dictionary.
- This dictionary captured essential room attributes such as room type, dimensions, bounding box coordinates, spatial relations, and other metadata like size, aspect ratio, and privacy attributes.

3.4.4. Combining Inputs and Outputs:

- The processed inputs and outputs were combined into a structured format suitable for fine-tuning.
- Input strings summarized the design context, while output dictionaries provided precise room details.

3.4.5. Saving the Dataset:

The final dataset, containing the prepared inputs and outputs, was exported to a CSV file. This CSV was specifically formatted for use in fine-tuning tasks for transformer-based models.

The result was a clean, well-structured dataset that effectively bridges the gap between natural language inputs and room-level floor plan descriptions, enabling robust training of generative models for this project.

CHAPTER 4

SYSTEM DESIGN AND ARCHITECTURE

4.1 OVERVIEW:

The floorplan generation and retrieval system is intended to work perfectly in conjunction with user-provided descriptions and produce corresponding architectural plans. The system uses natural language processing (NLP) retrieval and generation optimization methods to facilitate efficient and accurate floorplan generation.

The pipeline starts with user input, maybe natural language floorplan description or interactive chatbot discussion. The text is embedded in BERT text-embeddedness, extracting semantic meaning and expressing it as a numerical vector. The embeddings are stored in an indexed FAISS-based vector database, which provides fast similarity searching. Upon arrival of a query, the topmost similar floorplans are retrieved by semantic similarity.

To generate floorplans, the system takes an interior design-oriented method where boundary layout is first defined. Randomly allocated directional seed points are based on user commands that specify room locations. The system takes a Simulated Annealing + Voronoi strategy, iteratively optimizing room locations for optimal space allocation. For maintaining a disciplined layout, minimum rectangularization is used, optimizing room shapes while maintaining spatial restrictions. The boundary fitting process corrects any inaccuracy in the layout to make sure every room is accommodated within the prescribed limits.

Either a retrieved floorplan (if retrieval is chosen) or a generated layout (if generation is indicated) is generated as output. Such design allows for flexible floorplan assembly with preservation of user intention and spatial arrangement optimization.

4.2 WORKFLOW DIAGRAM:

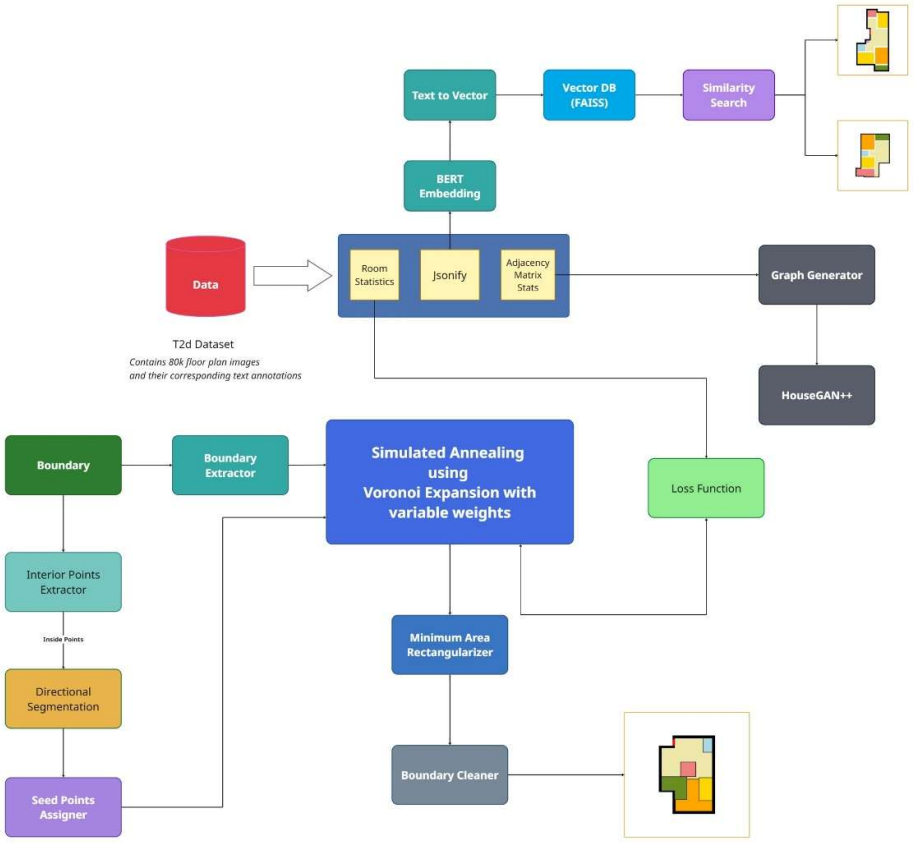

Fig: 4.1: Workflow for Floor Plan

4.3 BLOCK DIAGRAM:

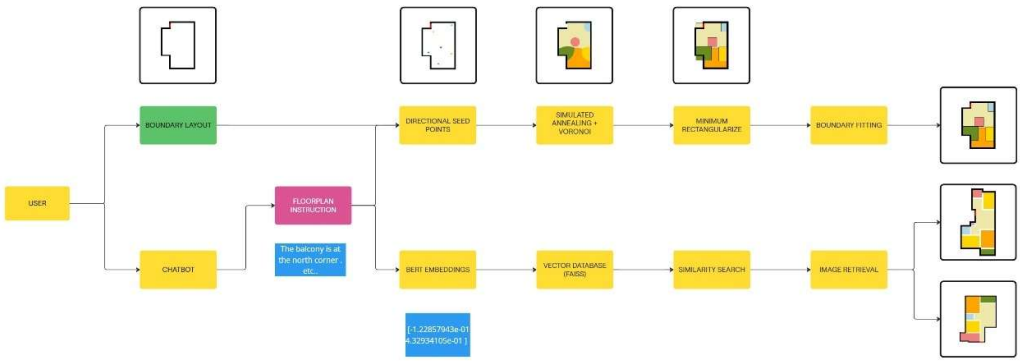

Fig 4.2: Block Diagram of Proposed System

4.4 FAISS IMAGE RETRIEVAL:

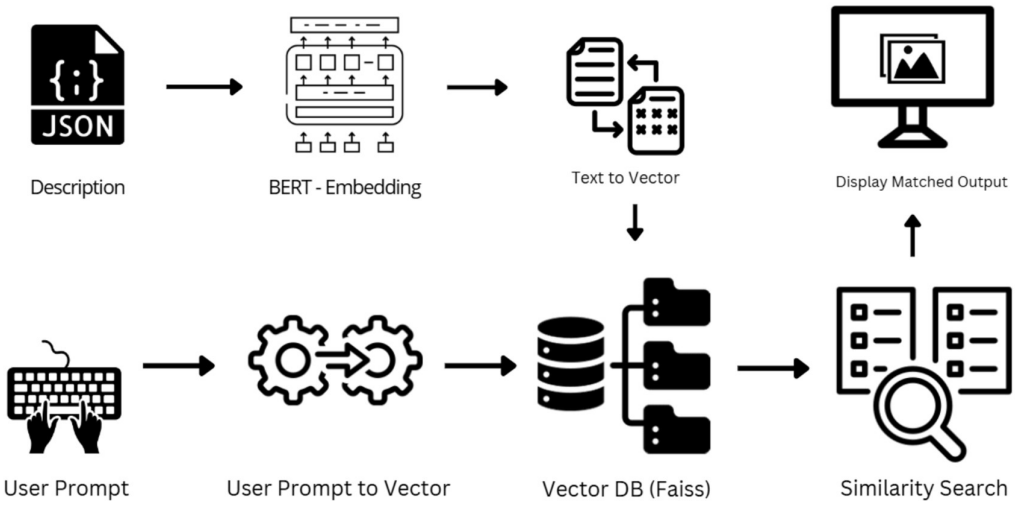

Fig 4.3: Architecture of Retrieval System

4.4.1. Dataset Preparation:

Floor Plan Dataset: We employ a pre-collected set of floor plan images and accompanying text descriptions. The images consist of high-resolution, labeled layouts for varying architectural designs. The floor plan images are each matched with a corresponding text description covering the important attributes, including the type of room, size, and spatial connection

4.4.2. Text Embedding Using BERT:

Embedding Text Descriptions: We employ a pre-trained BERT (Bidirectional Encoder Representations from Transformers) model to encode both the descriptions of floor plans and the user query searches into vector representations (embeddings). We select BERT due to its semantic sensitivity to language, which is essential in reading the spatial and functional relations in architectural descriptions.

Injecting User Queries: When a user enters a string description of the intended floor plan, this query is injected as well with the same BERT model. The outcome is a vector form of the user's input that can then directly be compared with the floor plan embeddings in our database.

Vector Space Alignment: Through the placement of both the floor plans and the user queries within the same vector space, we make sure that similar descriptions (semantically based) are located closer to one another in the vector space. This

alignment allows for effective similarity matching between floor plan descriptions and user inputs.

4.4.3. Similarity Search Using FAISS:

Vector Database with FAISS: To carry out fast similarity search, we employ Facebook AI Similarity Search (FAISS), a high-performance library for FAISS and dense vector clustering. All the BERT-generated floor plan embeddings are indexed in a FAISS vector index, so efficient retrieval based on similarity can be performed.

Top-K Similarity Retrieval: Upon the submission of a query by a user, we employ FAISS to fetch the top K floor plans with the most similar scores to the user's description. The similarity score is determined based on cosine similarity between the stored floor plan embeddings and the user query embedding.

4.5 SIMULATED ANNEALING FOR FLOORPLAN GENERATION:

Simulated Annealing (SA) is a **probabilistic optimization method** based on the **metallurgy process of annealing**, which slowly cools metals to achieve a stable crystalline form with very little energy. For floorplan generation, this method is used to repeatedly improve room arrangements within a specified boundary.

The algorithm begins with a **preliminary floorplan arrangement**, in which rooms are placed according to **directional seed points** provided by user input. Space is assigned to rooms through a **Voronoi-based segmentation** technique. The **simulated annealing** algorithm then refines this arrangement by constantly changing room locations, sizes, and adjacencies in a bid to reduce spatial conflicts and increase compliance with design constraints.

The simulated annealing also has the merit of being capable of escaping **the local minima** and searching more in the solution space. It does this through accepting worse solutions every now and then with decreasing probability as time elapses, hence enabling the system to explore different configurations for the layout prior to settling into an optimized form. As the **temperature parameter** drops, the chance of accepting worse solutions also declines, and the system approaches a **globally optimized floorplan** structure.

Fig 4.4: Architecture of Generation Process

4.5.1 INSPIRATION:

In traditional interior design processes, the designers first set the exterior perimeter of the space and collect information on:

1. Number and room type
2. Estimate room sizes
3. Neighbour constraint (e.g., bedroom near bathroom, kitchen near dining)

Once all these facts are collected, designers assign one color to each type of room, then plot coarse rectangular room positions inside the boundary. Wherever rooms cross over the boundary or overlap each other, they update it iteratively. Our AI-based generation process is similar in an organized fashion as well.

4.5.2 GENERATION:

Step 1: Direction-Based Segmentation:

The floor plan is separated into regions of space by direction constraints that are inputted by the user in a chat interface. One room is allotted to each divided region based on user choice.

Step 2: Voronoi-Based Room Expansion with Dynamic Weights:

After initializing seed points for each room, Voronoi diagrams are used to define room boundaries. Unlike traditional Voronoi partitions with uniform growth, our approach assigns dynamic growth rates based on real-world room proportions from the T2D dataset:

- **Living Room:** 43%–48% of total floor area → **Weight: 0.48**
- **Bedroom:** 18%–22% of total floor area → **Weight: 0.22**

Rooms with higher weights expand at a faster rate, ensuring that the floor plan maintains realistic proportions.

Step 3: Simulated Annealing for Room Placement Optimization:

To refine the floor plan, simulated annealing is applied to adjust seed points iteratively. The optimization minimizes a composite loss function consisting of:

- **Overlap Loss (L_{ol})**: Penalizes overlapping rooms.
- **Total Pixels Occupied (L_{po})**: Ensures the combined room areas match the total floor plan.
- **Boundary Violation Loss (L_{β})**: Penalizes rooms exceeding the floor boundary.
- **Merging Loss (L_m)**: Prevents functionally distinct rooms from merging.

The total loss function is formulated as:

$$L_{total} = w_1 L_{ol} + w_2 L_{po} + w_3 L_{\beta} + w_4 L_m$$

Where w_1, w_2, w_3, w_4 are hyperparamters controlling each term's contribution.

Step 4: Spatial Refinement and Connectivity:

To maintain spatial coherence and accessibility:

- A **flood-fill algorithm** ensures all rooms are connected and accessible from entrances/hallways.
- **Boundary rectification** eliminates any excess space overflow, ensuring the final layout adheres to user constraints.

This approach ensures **realistic, optimized floor plan generation** with spatial adaptability and functional coherence.

4.6 User Interface and Interaction:

Search Results Display: The system presents retrieved or generated floor plans in an intuitive UI, showcasing both visual layouts and key attributes (e.g., room dimensions, connectivity). Users can compare layouts side by side.

Customization & Refinement: Users can interact with the Gemini LLM through a chat-based interface to refine layouts by adjusting spatial constraints,

modifying design elements, or requesting alternative configurations. The AI dynamically updates the floor plan based on user inputs, ensuring a more personalized design experience.

Feedback Mechanism: To continuously improve the system, a feedback mechanism allows users to rate the relevance of retrieved or generated floor plans. This feedback helps fine-tune the model in future iterations, enhancing both retrieval accuracy and generative performance.

CHAPTER 5

EXPERIMENTS AND OUTCOMES

5.1 Initial Experiment with CLIP Model:

Our first strategy was to employ the CLIP (Contrastive Language–Image Pretraining) model to fetch floor plans from user-supplied text descriptions. The CLIP model operates by embedding both images and text into vectors, enabling it to calculate similarity between them. In our configuration, every floor plan image was embedded into a vector representation at runtime, and then matched against the vector representation of the user's text input. The system would then fetch the 3 most similar images based on the similarity scores..

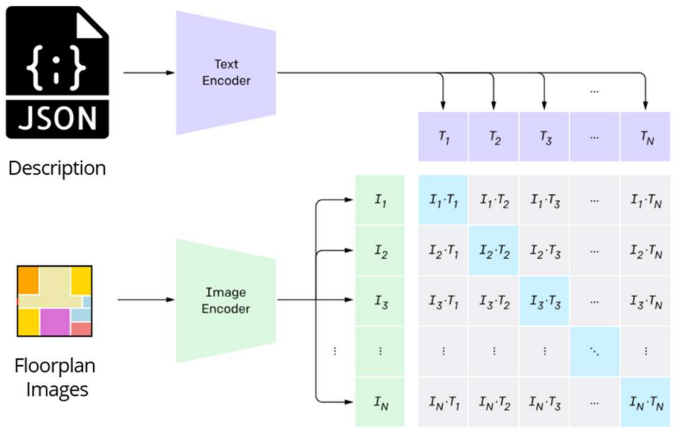

Fig 5.1: CLIP Model Architecture

But we faced a major limitation with this method. Because every image in the whole dataset was being transformed into a vector in runtime, the retrieval process was very slow, particularly with a large dataset. This undermined the effectiveness and practicability of the system, rendering it infeasible for real-time or near-instant retrieval of floor plans.

5.2 Transition to FAISS for Improved Performance:

To overcome the performance bottleneck of the CLIP-based method, we incorporated Facebook AI Similarity Search (FAISS) into our framework. FAISS is an optimized library for speedy similarity search and clustering of dense vectors that suits the requirement of large-scale vector-based retrieval applications. Rather than encoding images to vectors at runtime, we computed all floor plan image embeddings beforehand and indexed them using a FAISS index.

This enabled us to retrieve comparable images from the precomputed embeddings, which took a much shorter time.

With FAISS, we obtained more efficient and accelerated similarity search by searching the FAISS index for the 3 most similar images to the input text vector from the user. This pre-indexing solution negated on-the-fly vector conversion per image, leading to a significant boost in processing time. Our ultimate results proved that the system could now retrieve applicable floor plans in real time with negligible delay, and thus was applicable for real-world use.

COMPARISON:

The outcome of our experiment indicated a significant performance edge of the FAISS-based approach compared to the original CLIP-based approach in both speed and usability. Although the two approaches delivered comparable accuracy in retrieving applicable floor plans, FAISS considerably saved retrieval time, which is instrumental for user experience in real-world scenarios. This enhancement underscores the significance of effective similarity search techniques when dealing with large datasets in image retrieval applications.

5.3 Generation: VQ-VAE for Floor Plan Image Generation:

The Vector Quantized Variational Autoencoder (VQ-VAE) is a strong generative model capable of efficient compression, reconstruction, and generation of images. For our project, VQ-VAE is used to produce floor plan images from natural language descriptions given by the user. The process consists of encoding floor plan images as discrete latent codes, training a Transformer model for text-to-latent code, and generating new floor plans via the VQ-VAE decoder.

Below is a detailed breakdown of the process:

Core Components of VQ-VAE:

Encoder:

The encoder compresses input floor plan images into a lower-dimensional latent representation. This representation encodes important spatial and visual features, including room configurations and spatial relationships, and reduces the dimensionality of the input data substantially.

Vector Quantizer:

The latent representation is then discretized with a discrete codebook. This process converts the continuous latent space into discrete codes, which facilitates efficient reconstruction and generation. The codebook learns

essentially a collection of discrete representations that capture the visual and structural patterns in the floor plan dataset.

Decoder:

The decoder reconstructs the original floor plan image from quantized latent codes. It is trained to transform these discrete codes into detailed and precise visual forms.

Training Objectives

Reconstruction Loss:

Is assured of producing close-match output of similar high-fidelity to that input floor plan originally.

Quantization Loss:

Optimizes the discrete codebook to capture the most relevant patterns for efficient and accurate reconstruction.

Application of VQ-VAE in Image Generation:

In our project, the VQ-VAE model is learned on a dataset of 80,000 floor plan images and their text descriptions. Training the system in this way enables it to construct a solid "codebook" of visual and spatial patterns from the dataset, allowing it to generate precise floor plan images based on user inputs.

Training Phase:

The VQ-VAE model is trained to compress and reconstruct the images of the dataset so that it retains the visual and structural features of floor plans. This step generates a latent space representation of the dataset in which each image is described using discrete codes.

Latent Space Representation:

After training, the encoder maps the textual descriptions into latent representations that are further quantized into discrete codes. These codes correspond to the patterns and spatial relationships in floor plans.

Image Generation:

A Transformer model is also trained in conjunction with the VQ-VAE to predict the latent codes from the input textual descriptions. When a new description is input by a user, the Transformer predicts the respective latent codes, and the VQ-VAE decoder outputs a new image based on these codes.

End-to-End Workflow for Image Generation:

Input:

The user provides a natural language description of the desired floor plan. For example:

"The balcony juts out on the south side, with the master room to the north. The kitchen is in the southwest corner, and the living room is at the northwest corner."

Process:

The textual description is transformed into latent codes using the Transformer model.

The VQ-VAE decoder takes these latent codes and reconstructs a new floor plan image that matches the description.

Output:

The system generates a new floor plan image adhering to the spatial relationships, room configurations, and other details specified in the input description.

Input Text annotation:

"The balcony juts out on the south side, with the master to the north. It is approximately 12 feet wide by 5 feet deep, for a total square footage of 60. The bathroom is in the northeast corner with the common room to the south and the living room to the west. It is approximately 12 feet wide by 5 feet deep, for a total square footage of 60. The common room is on the east side with the bathroom to the north, the master room to the south and the living room to the west. Half of the south wall is exterior. It is approximately 12 feet wide by 10 feet deep, for a total square footage of 120. The kitchen is in the southwest corner with the living room to the north and the master room to the east. It is approximately 10 feet wide by 10 feet deep, for a total square footage of 100. The living room is at the northwest corner, with the bathroom and common room to the east, the master room inset at the south west corner, and the kitchen to the south. It is approximately 15 feet wide by 30 feet deep, minus the knock out for the master room, for a total square footage of 400. The master room is on the east side with the living room at the northwest corner, the common room to the north, balcony to the south, and kitchen to the west. It is approximately 12 feet wide by 20 feet deep, for a total square footage of 240."

Output Produced:

when trained with 4000 human annotated strings/text descriptions and images we get this image as output:

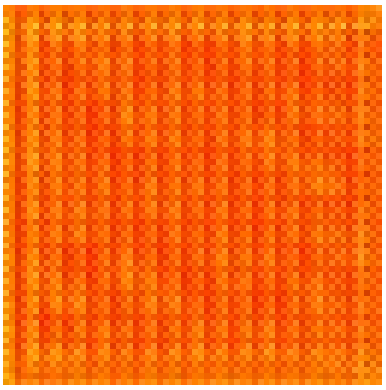

Fig 5.2: Output of VQ-VAE Transformer

5.4 Using Gemini:

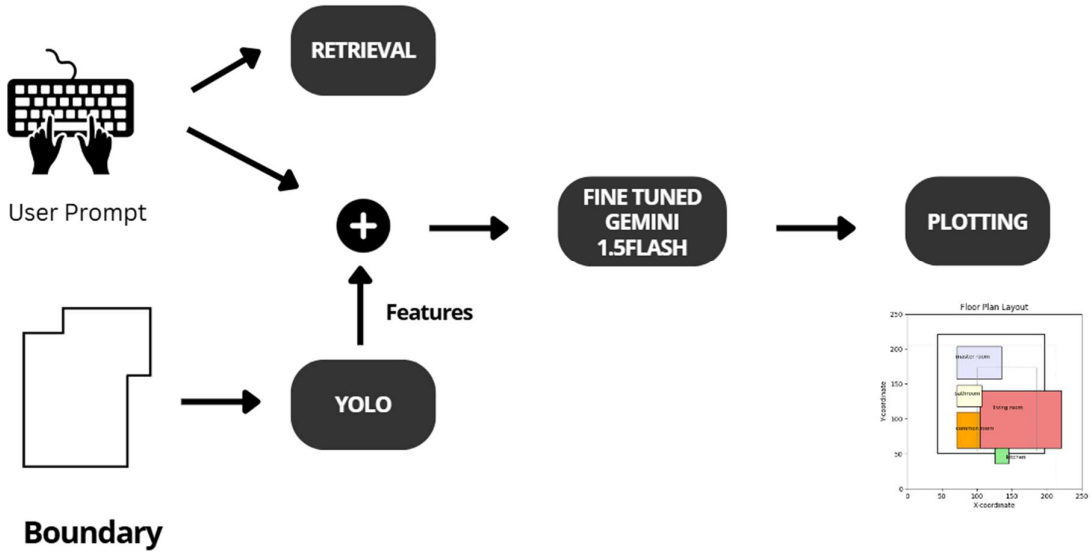

Fig 5.3: YOLO Model

Input: Annotated Strings + Boundary Image

Output: Plotted Floorplan

Step 1: Input Boundary Detection with YOLO v11:

The process begins by feeding a boundary image of a building's floor plan into the YOLO v11 model. YOLO (You Only Look Once) is a powerful object

detection algorithm capable of identifying spatial information quickly and accurately.

Objective:

YOLO v11 identifies the minimum bounding box that can encompass the entire building area and detects empty spaces within this boundary.

Output:

The YOLO model produces a structured list of dictionaries containing boundary information. Each dictionary specifies the following details:

{class: positive/negative, x_min, y_min, x_max, y_max}

class: Indicates whether the detected space is usable (positive) or not (negative).

x_min, y_min: Coordinates of the top-left corner of the bounding box.

x_max, y_max: Coordinates of the bottom-right corner of the bounding box.

This output provides a foundation for defining the layout and identifying usable spaces within the given boundary.

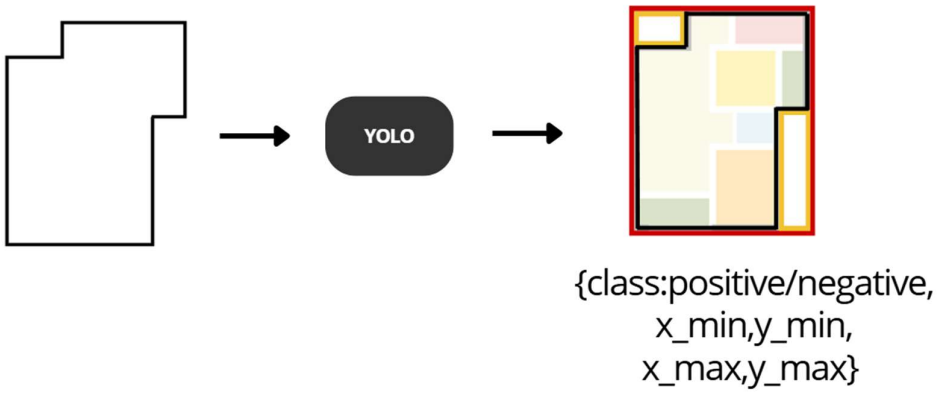

Fig 5.4: Output of YOLO Model

Step 2: Concatenation with Annotated Strings:

The boundary data generated in Step 1 is concatenated with user-provided textual annotations (referred to as "Annotated Strings"). These strings describe additional specifications, such as room types, relationships, or locations.

Purpose:

The concatenation of the spatial data from YOLO with the user's detailed descriptions enriches the input for further processing. This combination ensures that the next model receives a comprehensive dataset that integrates visual boundaries with user-defined requirements.

Step 3: Fine-Tuned Model for Room Specifications:

The concatenated data (boundary details + annotated strings) is then passed into a fine-tuned model, Gemini-1.5 Flash. This advanced language model has been trained specifically to interpret and process architectural data.

Input:

A combination of YOLO output and user-provided textual descriptions.

Process:

The model analyzes the input and generates detailed specifications for each room in the floor plan. It provides a structured output in the form of a list of dictionaries.

Output Format:

Each dictionary specifies:

{room_type, x, y, w, h, x_min, x_max, y_min, y_max, x_near, y_near, relation, size, location}

room_type: Type of the room (e.g., kitchen, living room).

x, y: Coordinates for plotting the room.

w, h: Width and height of the room.

x_min, x_max, y_min, y_max: Bounding box dimensions.

x_near, y_near: Proximity relations with other rooms or features.

relation: Positional relationship with other entities in the plan.

size: Area of the room.

location: Specific placement in the layout (e.g., northeast corner).

This output serves as a blueprint for creating a structured and coherent layout based on the user input and detected boundaries.

Data:

	input	output
0	Annotated String: Balcony 1 is about 8 foot X ...	[{'room_type': 'master room', 'x': 102, 'y': 7...
1	Annotated String: balcony is situated at south...	[{'room_type': 'common room 1', 'x': 153, 'y':...
2	Annotated String: Balcony 1 is on the northwes...	[{'room_type': 'balcony 1', 'x': 61, 'y': 90, ...
3	Annotated String: Balcony 1 is in the far sout...	[{'room_type': 'living room', 'x': 102, 'y': 1...
4	Annotated String: The balcony is at the South ...	[{'room_type': 'bathroom', 'x': 181, 'y': 64, ...

Fig 5.5: Dataset Overview

Training Result:

Fig 5.6: Training Result of Gemini Mode

Step 4: Visualization and Graph Plotting:

The room specifications generated by the fine-tuned model are then visualized using the Matplotlib library.

Purpose:

The data is plotted on a graph to create a visual representation of the floor plan, ensuring that all rooms, spatial relationships, and boundary constraints are accurately displayed.

Process:

Each room is represented as a box plotted according to its specifications (x, y, w, h).

Relationships and proximity constraints are visually represented.

Output:

A final, annotated graph depicting the complete floor plan, adhering to the user's input and the detected boundary constraints.

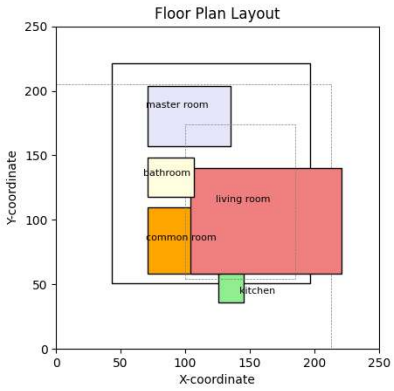

Fig 5.7: Generated Floor Plan of YOLO Model

5.5 Deep Convolution GAN(DCGAN) for Floorplan Generation:

Objective:

DCGAN (Deep Convolutional Generative Adversarial Network) is one of the widely used generative models based on convolutional layers to create realistic images from noise. Our objective was to train a DCGAN over the T2D dataset, which would be able to synthesize floorplans analogous to those in the dataset.

Approach:

- **Dataset Preparation:**
  - We acquired and pre-processed the **floorplan images** from the **T2D dataset** with equal resolution and aspect ratios.

- The images were resized and normalized to a uniform size to match the DCGAN model architecture.

DCGAN Training:

- The generator network learned to map a random noise vector zzz to a realistic floorplan.
- The discriminator network was trained to distinguish between real floorplans from the dataset and the generated ones.
- The networks were optimized utilizing the Adam optimizer with learning rate 0.0002 and batch size of 64.
- The training was for 200 epochs, and we observed the loss values and produced samples.

5.5.1 DCGAN Architecture:

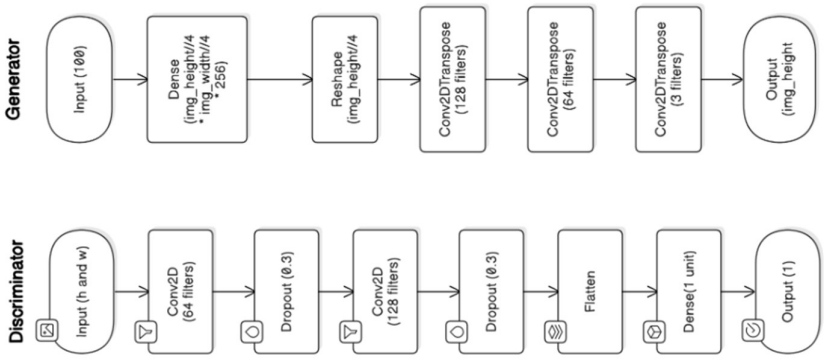

Fig 5.8 Architecture of DCGAN

Output:

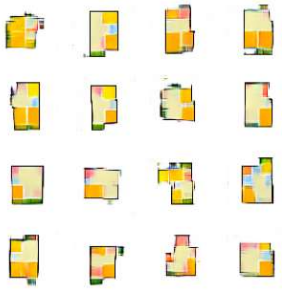

Fig 5.9: Output of DCGAN

Challenges:

- Despite 200 epochs, the resulting images were still blurry and structurally inaccurate, which suggests that DCGAN was not able to learn the subtle details of architectural schemes.
- The main restriction was that DCGAN acts on noise only, without any semantic interpretation of floorplan directives

5.6 Stack GAN

Because DCGAN could not produce good-quality floorplans, we shifted to **Stack GAN**, where text descriptions are included in image generation.

StackGAN (Stacked Generative Adversarial Network) generalizes DCGAN by **producing images conditioned on text descriptions**. Our aim was to train StackGAN on the **T2D dataset** so it could produce floorplans conditioned on user-input textual instructions.

Approach:

1. **Dataset Preparation:**
    - Each floorplan in the T2D dataset was paired with its corresponding textual description, defining room positions, sizes, and architectural features.
    - Text descriptions were tokenized and converted to embeddings with the help of pre-trained word embeddings (e.g., Word2Vec or BERT).
2. **StackGAN Training:**
    - **Stage 1:** Generates a coarse, low-resolution image from the textual description.
    - **Stage 2:** Refines the generated image, adding more details and improving spatial coherence.
    - The generator used **Recurrent Neural Networks (RNNs)** to process text, while the discriminator ensured the generated floorplans matched real ones.

Challenges:

- Though StackGAN was promising, it was not possible because of the huge nature of floorplan instructions.

- Composed of highly structured spatial relationships, floorplans are unlike traditional image generation problems (e.g., flowers, birds), for which StackGAN had difficulty capturing.
- The model did not adequately position rooms according to textual constraints and tended to generate overlapping or missing components.

5.7 Bin Packing:

Overview:

Bin Packing has also been researched as a rival solution for efficient use of space during floor plan generation. The approach treats the floor plan as a bin and attempts to pack rooms (objects) efficiently into the available space. The algorithm focuses on minimizing left-over space with care to room adjacency constraints.

Process:

- **Room Representation:** Each room is treated as a rectangular block with predefined size constraints.
- **Packing Strategy:** Rooms are sequentially placed using heuristics to reduce gaps.
- **Space Optimization:** The method attempts to minimize wasted space while keeping rooms aligned properly within boundaries.

Outcome:

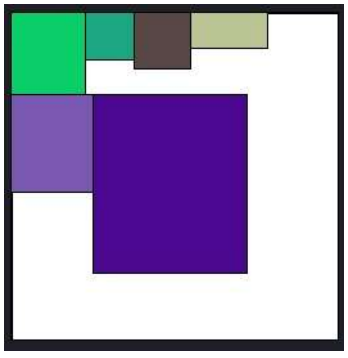

Fig:5.10: Bin Packing

Challenges:

- **Rigid Placement:** The algorithm tended to produce rigidly packed layouts, which made it challenging for user-defined adjustments.
- **Adjacency Constraints:** Although space-efficient, maintaining logical room adjacency (e.g., bathrooms adjacent to bedrooms) took extra post-processing.

- **Shape Constraints:** The method was better adopted for rectangular room plans, with little flexibility for irregular room design.

5.8 HouseGAN++:

Overview:

HouseGAN++ is an open-source model, which is trained on a Graph Neural Network. The model is given a graph as input and produces a corresponding floorplan layout. It maintains adjacency between rooms based on the nodes of the graph and the edges between them.

Outcome:

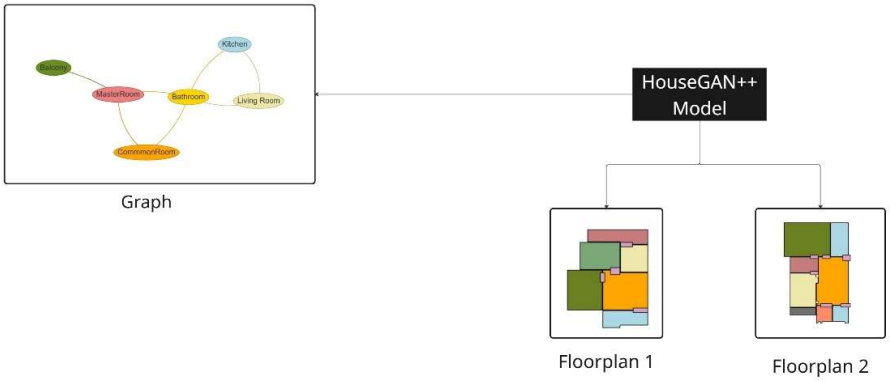

Fig:5.11: HouseGAN++ output for sample input

Challenges:

- **Boundary Violations:** The generated layouts tended to be larger than the given floorplan sizes, making them inapplicable at the moment.
- **Variable Outputs:** Since the model is not defined well for boundary constraints, the model may output varying versions of floorplans from the same graph input data.

5.9 Flood Filling Algorithm for Floorplan Segmentation:

The flood-fill algorithm is comprehensively applied in computer graphics for region-based coloration. For our floorplan generation system, we used flood-filling to see to it that every room is well labeled and identifiable visually in the resultant layout.

Approach:

Flood-filling was done through **Breadth-First Search (BFS)** to spread a color from a seed point so that neighboring areas belonging to the same room would be given a distinct color.

Working Mechanism:

1. **Seed Point Initialization:**
    - Each room is assigned a unique seed point based on its initial placement.
    - Each seed is associated with a distinct color to differentiate rooms.

2. **Color Propagation:**
    - The algorithm expands from each seed, spreading the assigned color to neighboring pixels within the room boundary.
    - This ensures that each room is fully covered while maintaining spatial separation from adjacent rooms.

3. **Stopping Condition:**
    - The flood-fill stops when it encounters predefined boundaries (walls, doors) or already colored areas.
    - This prevents color leakage into other rooms or empty spaces.

Outcome:

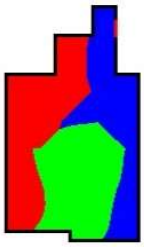

Fig:5.12: Output of Simple Flood Filling

Challenges:

- **Random Seed Placement**: The flood-fill algorithm initially placed seeds randomly, resulting in room placement inconsistencies. For instance, if the user requested "Living room in the north", the system would sometimes place it somewhere else because there was no integration between seed placement and user requests.

- **Integration with User Constraints**: The algorithm needed to be improved to align seed placement with user-defined spatial preferences, ensuring that rooms were positioned correctly before the flood-fill process began.

5.10 Flood Filling and Simulated Annealing:

Flood filling is used extensively in computer graphics for color filling and relies on Breadth-First Search (BFS). In our method, it makes sure all rooms are kept connected and open within the produced floor plan.

Working Process:

- Seed Points Initialization
- Color Spreading
- Stopping Condition

Simulated Annealing for Floor Plan Optimization:

Simulated Annealing is a probabilistic optimization technique inspired by the process of annealing metals.

Key Properties:

- The system is slowly cooled to find the lowest energy state.
- Helps escape local minima and reach a global optimum.
- Occasionally accepts worse solutions with a decreasing probability to avoid getting stuck in suboptimal states.
- The probability of accepting worse solutions decreases as the temperature lowers, ensuring convergence to an optimal floor plan.

Cost Function of Simulated Annealing:

1) Total pixels-based loss function:

i. The dataset containing 80,788 images was statistically analyzed to get room sizes relative to the boundary.
ii. The difference between the generated floor plan and the actual size is calculated using the below function.

$$Actual\ Percentage_i = room\_counts[i] \div total\_inside * 100$$

2) Optimized loss function:

To improve robustness, we designed an optimized loss function that penalizes deviations in room pixel percentages within a specified radius of each seed point from the target percentage

```
    Input: Image dimensions (width, height), seed points,
room radius

    Output: Pixel-count for each room

    1. For y in range(height):

       2. For x in range(width):

          a. Get pixel color (r, g, b).

          b. If pixel belongs to a room, compute distance
    to seed.

          c. If distance ≤ room radius, increment count
    for corresponding room.

    2. Return room pixel counts.
```

Outcome & Challenges:

Fig:5.13: Flood Fill and Simulated Annealing

Challenges:

Uncontrolled Growth: Room placements grew randomly, occasionally crossing boundary limits, necessitating extra constraints to allow controlled expansion.

5.11 Voronoi Based Partitioning:

Voronoi diagrams segmented the floor plan into adaptive space regions. This gave a first placement guideline to each room to ensure an equal distribution of space.

Outcome:

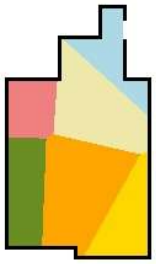

Fig 5.14: Output for Voronoi expansion

Challenge:

The Voronoi-partitioning method first allocated regions of equal size to all the rooms, disregarding functional distinctions (e.g., a living room and a bedroom were allocated the same area). To solve this, we improved the process by adding room-specific size limitations based on user-specified parameters.

5.12 Voronoi with Weighted Growth:

To resolve the problem of equal-sized rooms in typical Voronoi partitioning, we proposed weighted Voronoi growth, where the rooms grew dynamically according to their area requirements. Larger rooms (such as living rooms) had larger weights so that they grew proportionally but still kept spatial balance.

Outcome:

Fig:5.15: Output of Voronoi expansion with weights

Challenges:

This method improved layout realism but introduced new challenges in maintaining boundary constraints and avoiding excessive overlaps, requiring additional refinement through Simulated Annealing for optimal room adjustments.

CHAPTER 6

RESULTS AND DISCUSSION

Chatbot Page:

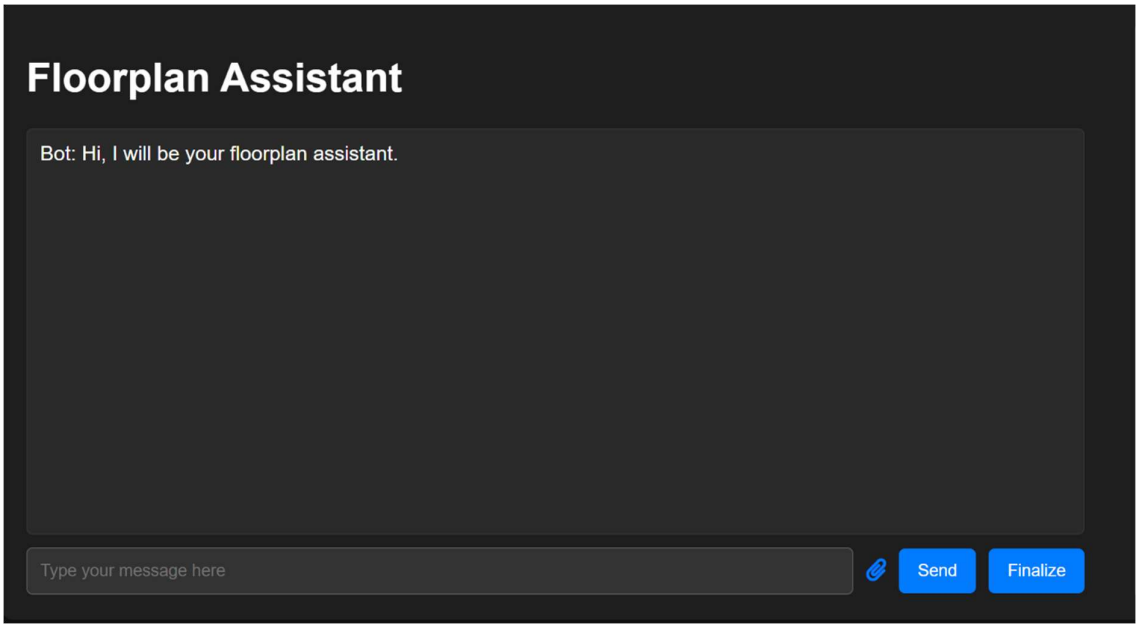

Fig 6.1: Web UI of Chatbot

Chatbot Page after uploaded Boundary:

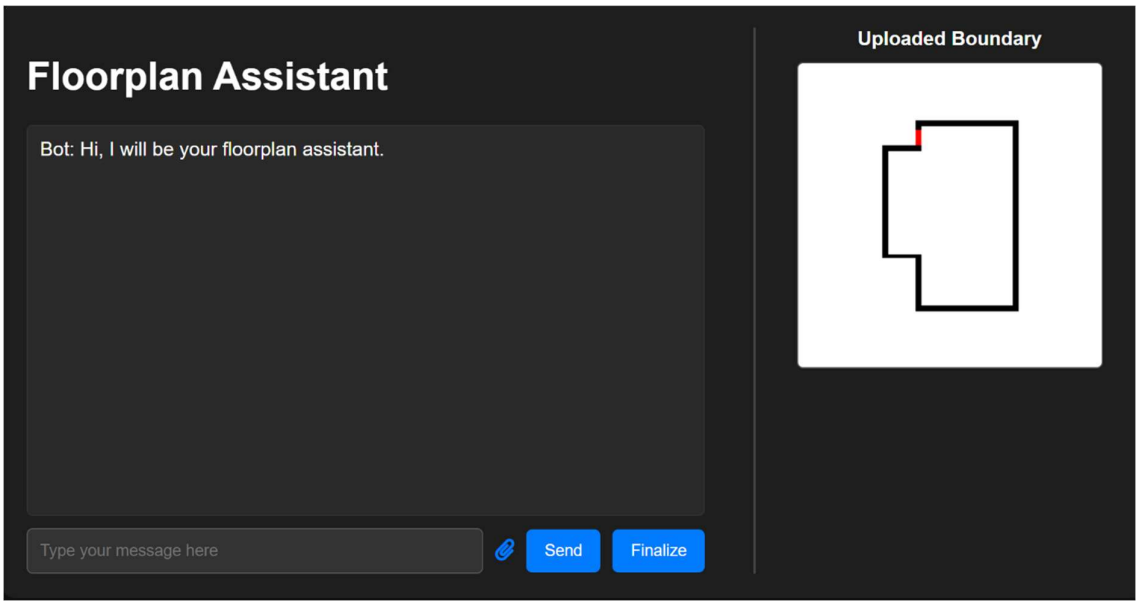

Fig 6.2: Chatbot Page with Boundary uploaded

Interaction with Chatbot:

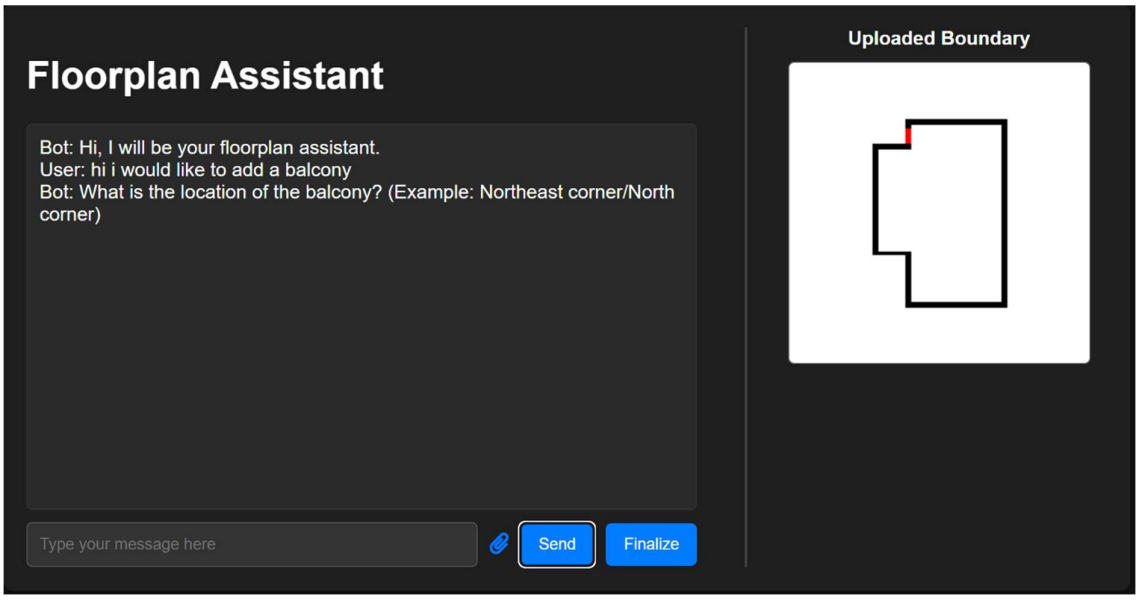

Fig:6.3: User interaction with Chatbot

Error Handling for indefinite Rooms:

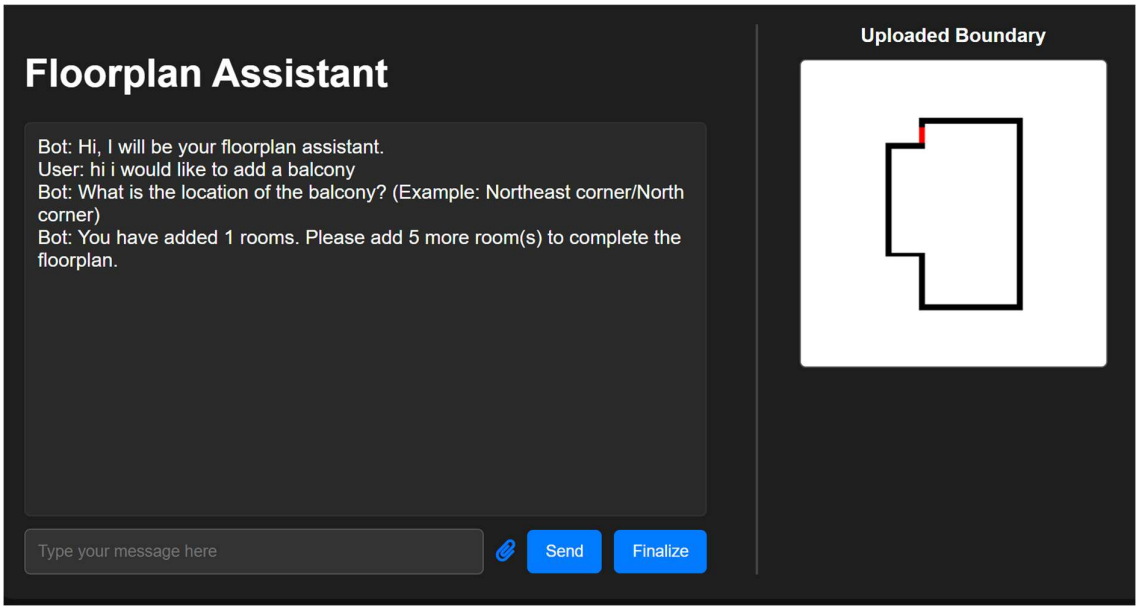

Fig:6.4: Error for indefinite rooms

After adding 2 Rooms:

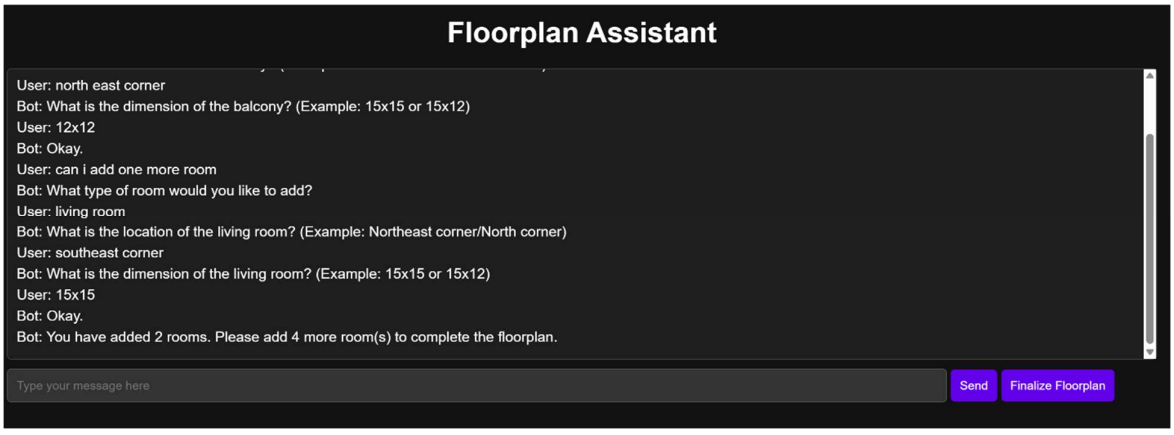

Fig:6.5: Error for not adding 6 rooms

Interactive Graph page:

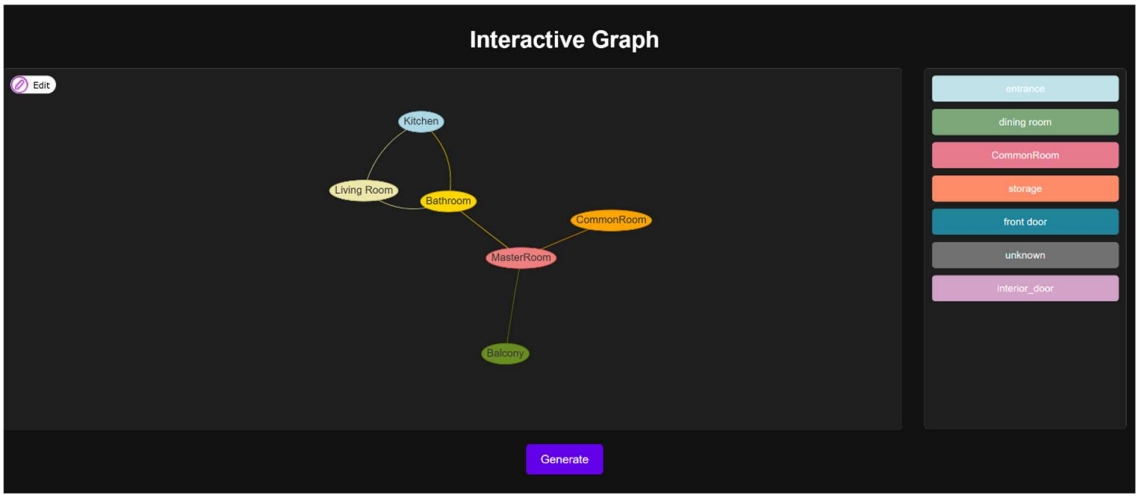

Fig:6.6: Interactive Graph page

Graph Page with added nodes:

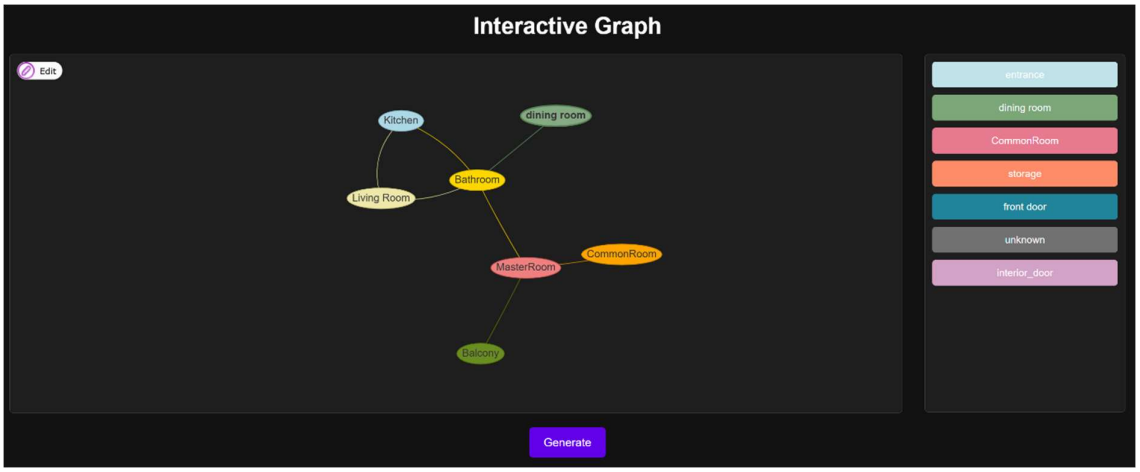

Fig:6.7: Dining room added by user

Floorplan Generated output:

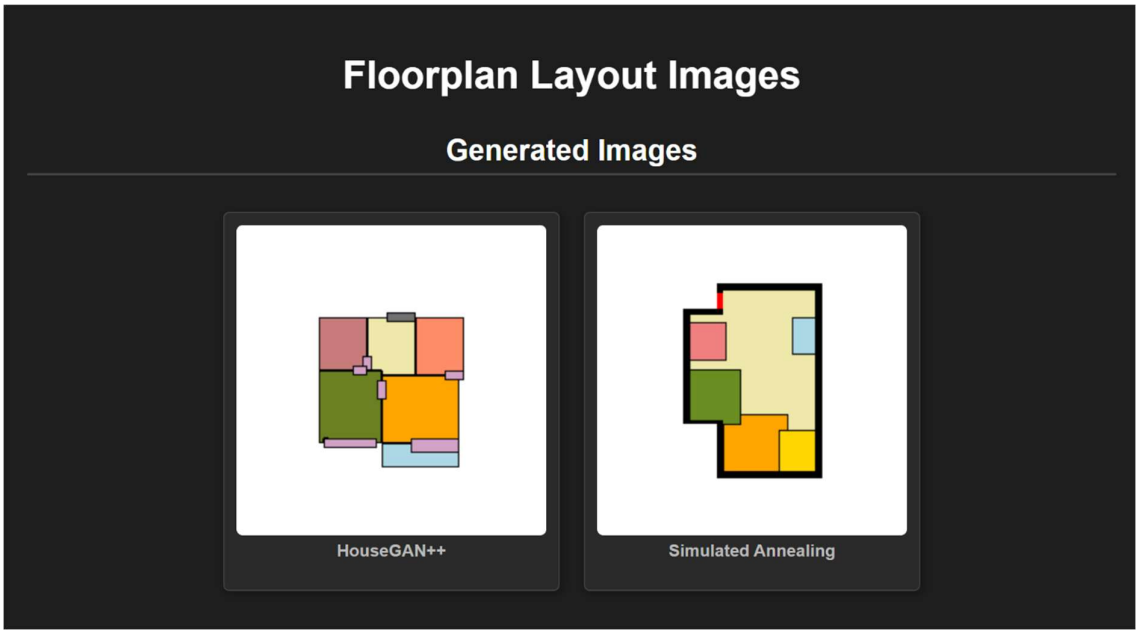

Fig:6.8: Generated Floorplan Output

Retrieved Floorplan Images:

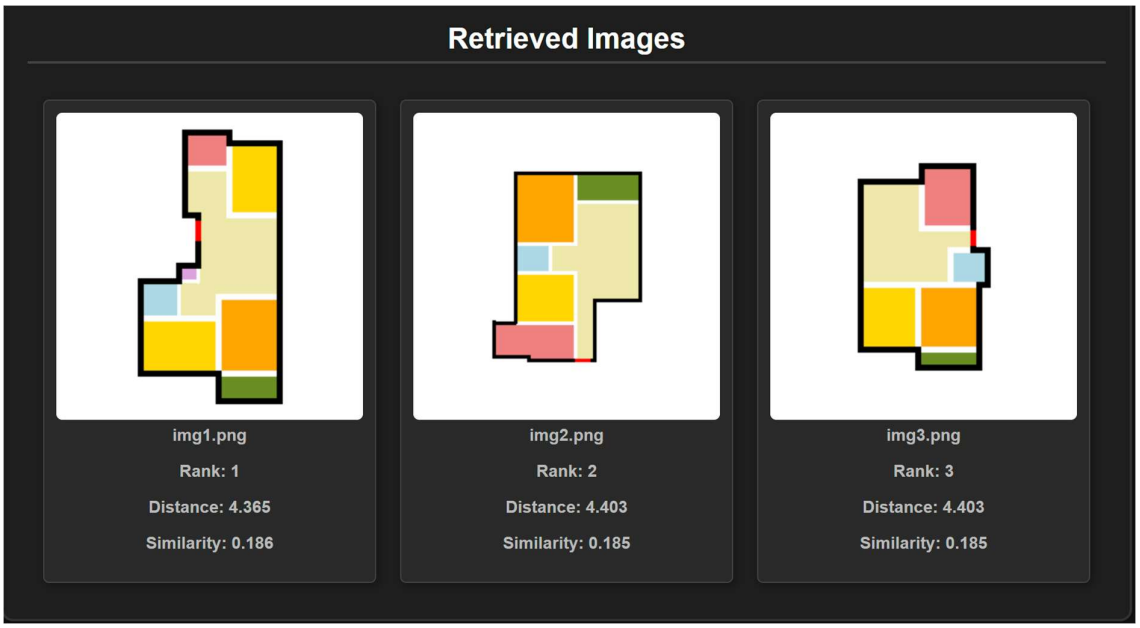

Fig:6.9: Retrieved Image based on User Floorplan Instruction

Vasthu-Based Room Placement Analysis:

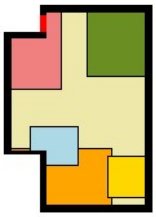

Fig:6.10: Vasthu-Compliant Room Placement in Floor Plans

6.1 PERFORMANCE METRICS:

6.1.1 Similarity Score:

Similarity score for each image is found using the formula

$$\text{Similarity\_score} = \frac{1}{1 + \text{Distance}}$$

the layout with more similarity it considered as the most suitable layout for the particular instruction

Sample Results:

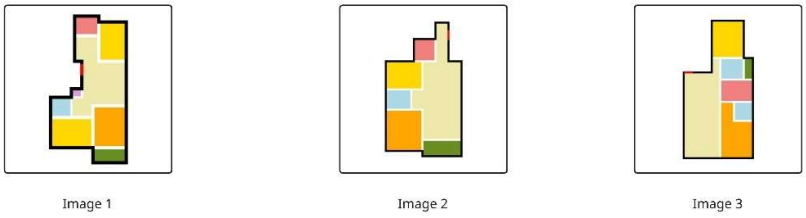

Fig:6.11: Visualization of Top Retrieved Floor Plans using FAISS

Rank	Similarity	Distance
1	0.92	0.08
2	0.89	0.11
3	0.87	0.13

Table 2: Top 3 Retrieved Images based on FAISS Similarity

6.1.2 Micro IoU and Macro IoU:

Intersection over Union (IoU) is a popular evaluation metric to measure the accuracy of segmentation tasks such as floorplan generation and object detection. It measures the extent to which the predicted regions overlap with the ground truth.

Intersection over Union (IoU) Formula:

$$IOU = A \cap B \div A \cup B$$

Where:

- A = Predicted area
- B = Ground truth area
- |A∩B| = **Intersection** (common overlapping area)
- |A∪B| = **Union** (total area covered by both predicted and ground truth)

Micro IoU:

Micro IoU computes IoU over all samples together by aggregating intersection and union over all classes before computing the ratio. Micro IoU assigns equal weight to all single pixels and is therefore affected by class imbalance.

Formula for Micro IOU:

$$Micro\ IoU = \frac{\sum_{i=1}^{N} |A_i \cap B_i|}{\sum_{i=1}^{N} |A_i \cup B_i|}$$

Where:

- N = Total number of floorplans or segmentation instances
- A_i = Predicted region for instance i
- B_i = Ground truth region for instance i

Macro IoU

Macro IoU computes IoU for each class individually and then averages across classes. It doesn't treat classes differently and is therefore less affected by class imbalance.

Formula for Macro IoU:

$$Macro\ IoU = \frac{1}{C}\sum_{c=1}^{C}\frac{|A_c \cap B_c|}{|A_c \cup B_c|}$$

Where:

- C = Number of classes (e.g., different room types in floorplan segmentation)
- A_c = Predicted region for class c
- B_c = Ground truth region for class c

Performance Analysis of Simulated Annealing:

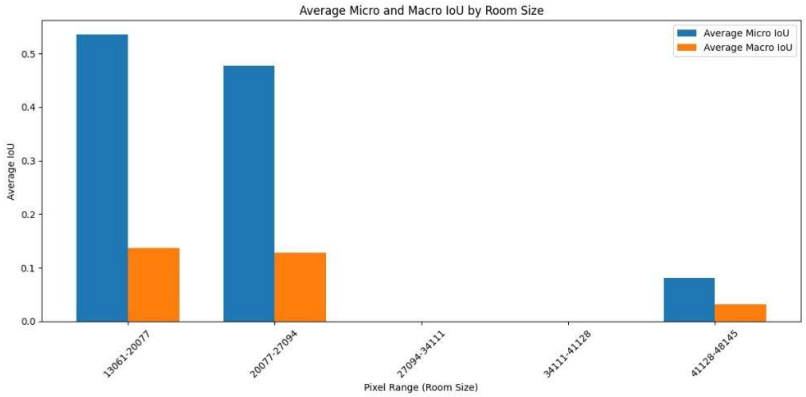

Fig:6.12: Average IOU vs Pixel Range

Comparative Study:

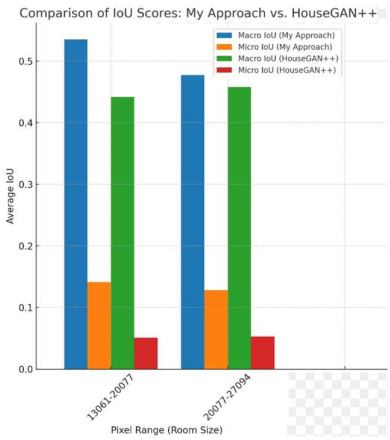

Fig:6.13: HouseGAN++ vs Simulated Annealing

6.1.3 Boundary Fit Analysis:

Boundary fit analysis estimates the quality of a generated floor plan based on how closely it adheres to pre-established spatial boundaries. The Boundary Leakage Percentage (BLP) provides the percentage of pixels leaking beyond permissible boundary and may be calculated from the formula:

$$\text{BLP} = \left(\frac{\text{Number of leaked pixels}}{\text{Total inside pixels}} \right) \times 100$$

where:
• **Leaked pixels:** Pixels that extend beyond the predefined boundary while not being part of the allowed region.
• **Inside pixels:** The total number of pixels that should be contained within the boundary

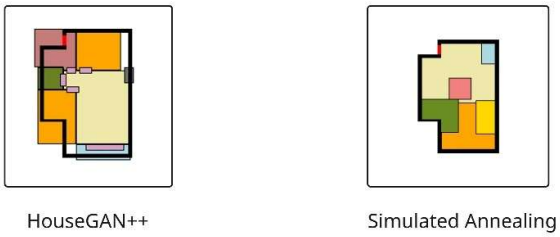

Fig:6.14: HouseGAN++ (24% leakage) vs. Simulated Annealing (0% leakage)

Boundary Spill Analysis in HouseGAN++

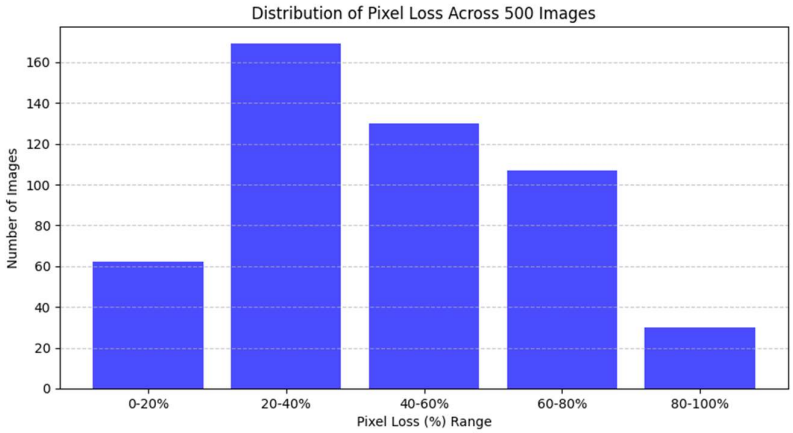

Fig:6.15: Pixels Spilled from Boundary Distribution in HouseGAN++

Room Deviation Analysis:

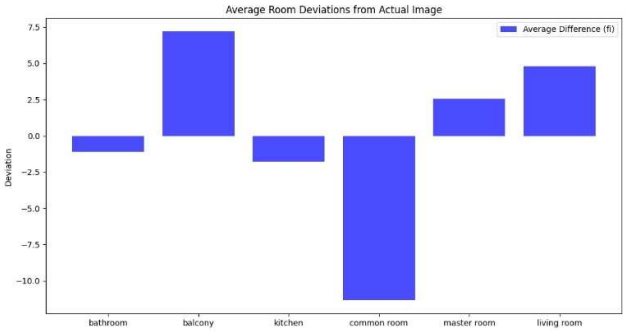

Fig:6.16: Room Deviation with Actual Image

Analysis of Simulated Annealing:

The following plots shows the working of the Simulated Annealing:

1) Temperature vs Iterations:

This graph effectively illustrates the gradual decrease in temperature over successive iterations in the simulated annealing process.

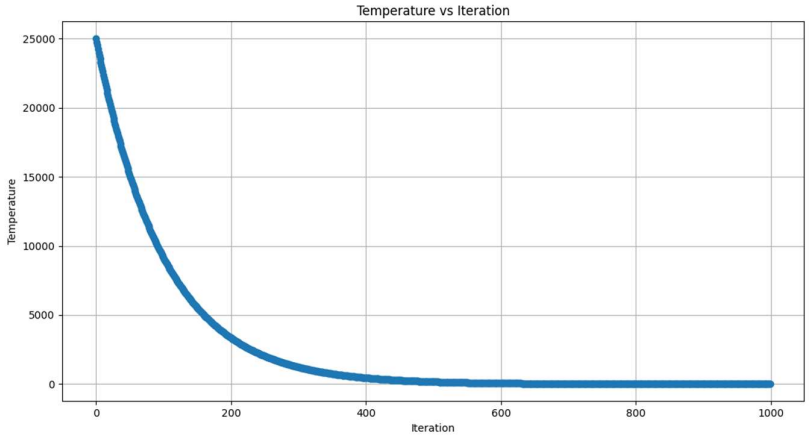

Fig:6.17: Temperature vs Iterations

2) Cost Function vs Iterations:

This plot very nicely illustrates how the cost function changes with more iterations in the simulated annealing algorithm. The cost function drops rapidly at the beginning as the algorithm tries out an extremely large number of potential solutions. As iterations increase, the cost converges towards a constant value, which represents convergence to an optimal or close-to-optimal solution.

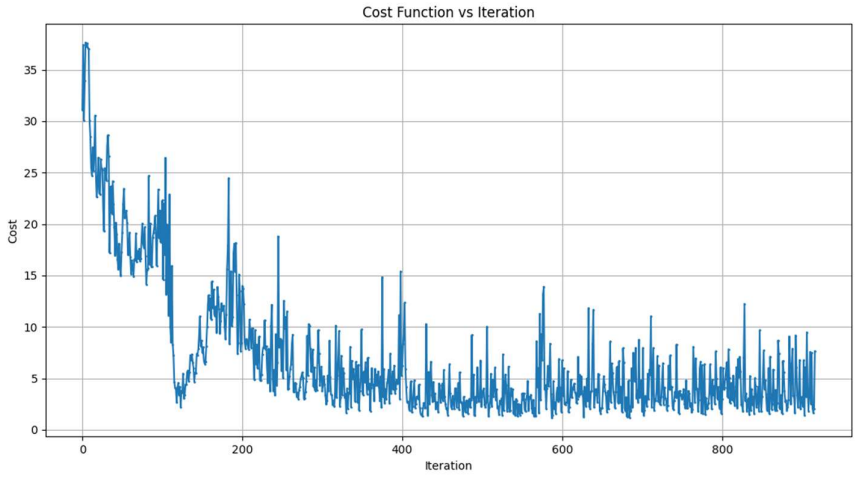

Fig:6.18: Cost Function vs Iterations

3) Percentage of Accepted Moves vs Temperature:

- This graph effectively shows that at higher temperatures, a larger percentage of moves are accepted, while at lower temperatures, fewer moves are accept

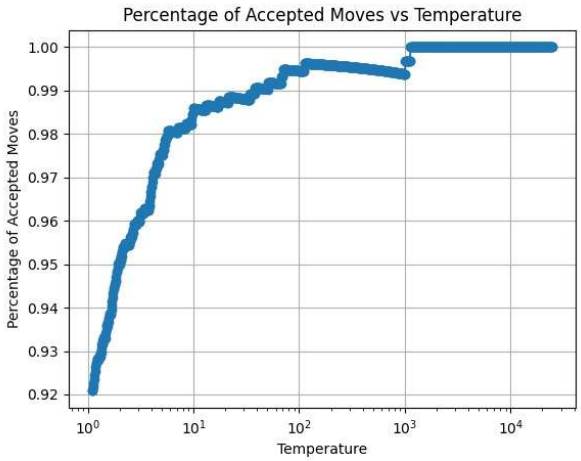

Fig:6.19: Percentage of Accepted Moves vs Temperature

4) Delta Cost vs Iterations:

The graph illustrates that in the initial iterations, both cost-increasing and cost-decreasing moves are accepted.

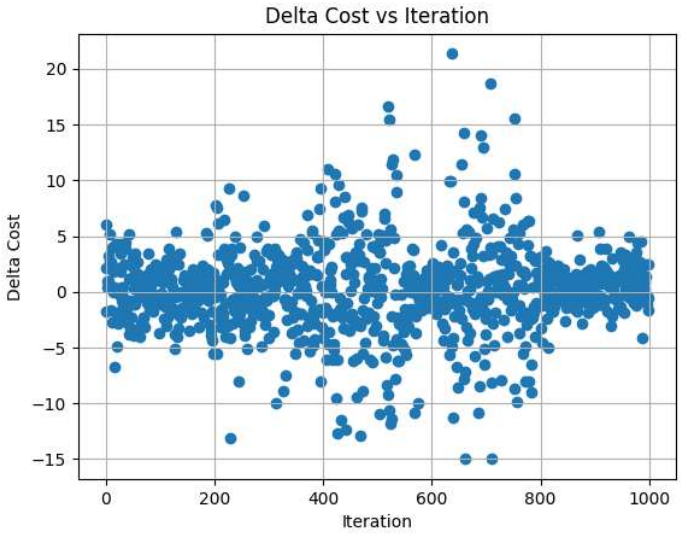

Fig:6.20: Delta Cost vs Iteration

CHAPTER 7

SOCIAL IMPACT AND SUSTAINABILITY

The proposed floor plan retrieval and generation system is in line with a series of United Nations Sustainable Development Goals (SDGs), reflecting its ability to drive positive social change and induce sustainable behavior. Democratizing the access to high-end architectural solutions, the system encourages **SDG 11: Sustainable Cities and Communities.** The system empowers individuals, architects, and city planners to design efficient, personalized floor plans, thus resulting in smart and sustainable city planning. The system reduces obstacles to users with no professional design training, is inclusive, and offers equal opportunity to access professional-level tools, thereby promoting SDG 10: Reduced Inequalities.

From a sustainable perspective, the system promotes **SDG 12: Responsible Consumption and Production** in maximizing resource utilization in building designs. By supporting efficient layout generation and retrieval, it minimizes material wastage during construction and aids in green building designs. The ability of the system to combine floor plans into specific spatial and functional needs enables energy-efficient designs, reducing the environmental impact of urban expansion. With the use of advanced machine learning methods like FAISS for planning and retrieval to interact with GANs or Stable Diffusion to create, the system is computationally efficient in its functions and reduces energy usage, operating in harmony with global sustainability goals.

Moreover, the system solves urbanization and housing deficit problems, which are accounted towards **SDG 9: Industry, Innovation, and Infrastructure.** It allows architects and designers to create innovative plans for homes and workplaces, scalable solutions for rising urban residents. This project does not only respond to current requirements for floor planning efficiency but also as a solution for sustainable development in the future. By integrating SDG aligned values, it encourages sustainable city planning, human-oriented access to housing, and environment care, which make it a socially responsible, future-oriented project.

CHAPTER 8

CONCLUSION AND FUTURE SCOPE

8.1 CONCLUSION

This project effectively proves the combination of FAISS-based retrieval and Simulated Annealing-based generation to design an intelligent floorplan system that effectively maps textual descriptions to architectural plans efficiently. Through the high-speed vector indexing of FAISS, our system retrieves the most suitable floorplans with semantic precision, much improving retrieval efficiency than conventional methods.

In addition, we discussed floorplan generation methods with Simulated Annealing, Voronoi partitioning, and GAN-based models, optimizing spatial organization based on user-provided constraints. Flood-filling and bin-packing algorithms were incorporated as well to further optimize layout feasibility, guaranteeing logical room assignment and connectivity. Gemini LLM-driven chat interface offered an interactive and ease-of-use experience, facilitating dynamic adjustments with real-time user input.

Although the project has shown promising outcomes, many avenues are left for future improvement. Increasing the dataset through various architectural styles and intricate multi-room floor plans can enhance model generalization. Also, optimizing weighted Voronoi growth and boundary-constrained optimization can improve the accuracy of the generated layouts. Incorporating user feedback-based reinforcement learning can further improve retrieval and generation to be even more accurate, enabling the system to learn over time.

This project stands out for foregrounding the potential of AI-driven floorplan retrieval and generation, bridging the gap between text inputs and geometrically abstracted spatial data. Integrating machine learning, optimisation algorithms, and NLP-based interactions, this system offers an effective, scalable, and intelligent solution for architects, designers, and real-estate professionals in transforming the way floorplans are navigated and designed.

8.2 FUTURE SCOPE

For future advancement, we plan to extend the system beyond rule-based generation and retrieval by introducing neural networks and cutting-edge generative models for improved floorplan prediction, real-world feasibility, and spatial composition.

Automatic prediction of direction in room placements under incomplete user constraints is one of these areas.At present, if there is no directional information, the system randomly places rooms. To rectify this, we intend to train a neural network (NN) to predict the best initial seed points for every room based on architectural patterns. This will improve the logical flow of room placement, providing functionally correct layouts.

In addition, the Simulated Annealing optimization algorithm and loss function need more experimentation to enhance floorplan quality. In particular, we intend to:

- Explore using **Mean Intersection over Union (MIOU) as a loss function** to guide the optimization process.
- Integrate the **expansion rate of Voronoi cells** as a dynamic factor in the optimization, ensuring room growth aligns with architectural constraints.

Yet another limitation of the existing method is that the resultant floorplans don't adequately address room adjacency relationships. Though Voronoi-based partitioning properly divides up space, it doesn't encode functional dependencies (e.g., keeping bathrooms close to bedrooms, kitchens adjacent to dining rooms). In future, graph-based learning will be included to define spatial relations between rooms prior to layout generation, resulting in more realistic and usable floorplans.

Through its shift from a retrieval-based system to an interactive generative design assistant, our method will equip architects, interior designers, and real estate developers with an effective tool for fast prototyping and customized layout generation. Such innovations will not only render the system more flexible and smarter but also a beneficial tool in real-world architectural design and planning.

CHAPTER 9

REFERENCES

[1] Michalek, J. J.; Choudhary, R.; Papalambros, P. Y. Architectural Layout Design Optimization. Engineering Optimization 2002, 34(5), 461–484.

[2] Fu, X.-M.; Tang, R.; Wang, Y.; Qi, Y.-H.; Liu, L. Data-Driven Interior Plan Generation for Residential Buildings. In Proceedings of the ACM Conference on Computer Graphics and Interactive Techniques, 2019.

[3] Nauata, N.; Zhang, Y.; Wang, Y.; Zhang, Y.; Wang, Y. House-GAN: Graph-constrained House Layout Generation. arXiv:2003.06988v1 [cs.CV] 16 Mar 2020.

[4] Hu, R.; Huang, Z.; Tang, Y.; van Kaick, O.; Zhang, H.; Huang, H. Graph2Plan: Learning Floorplan Generation from Layout Graphs. ACM Trans. Graph. 2020.

[5] Upadhyay, A.; Dubey, A.; Arora, V.; Kuriakose, S.M.; Agarawal, S. FLNet: Graph Constrained Floor Layout Generation. 2022 IEEE International Conference on Multimedia and Expo Workshops (ICMEW), 2022.

[6] Upadhyay, A.; Basha, N.K.; Ananthakrishnan, B. FloorGAN: A Generative Adversarial Network for Floor Plan Generation Guided by User Constraints. CODS-COMAD 2023, January 4–7, 2023, Mumbai, India.

[7] Chen, Q.; Wu, Q.; Tang, R.; Wang, Y.; Wang, S.; Tan, M. Tell2Design: A Dataset for Language-Guided Floor Plan Generation. arXiv 2023.

[8] Prieto, S.A.; Mengiste, E.T.; García de Soto, B. Investigating the Use of ChatGPT for the Scheduling of Construction Projects. Buildings 2023, 13, 857.

[9] Vaidya, B.; Pimpale, P.; Khartadkar, G. Generative Floor Plan Design Using Deep Learning: An AI-Powered Approach. International Journal of Novel Research and Development 2024, 9, 1-9.

10] Van Engelenburg, C.; Khademi, S.; van Gemert, J. MSD:A Dataset for Floor Plan Generation of Building Complexes.In: Int. Conf. Comput. Vis. Worksh. (2023)

[11] Chowdhery, A.; Narang, S.; Devlin, J.; Bosma, M.; Fiedel, N. PaLM: Scaling Language Modeling with Pathways.arXiv:2001.08361, 2022.

[12] Radford, A.; Kim, J.W.; Hallacy, C.; Ramesh, A.; Goh,G.; Agarwal, S.; Sastry, G.; Askell, A.; Mishkin, P.; Clark, J.;Krueger, G.; Sutskever, I. Learning Transferable Visual Models From Natural Language Supervision. arXiv 2021, 2103.00020.

[13] Anonymous authors ; GenPlan: A Novel Deep Learning Architecture for Generating Architectural Floor Plans. Under review as a conference paper at ICLR 2025.

[14] Shabani, M.A.; Hosseini, S.; Furukawa, Y. HouseDiffusion: Vector Floorplan Generation via a Diffusion Model with Discrete and Continuous Denoising. CVPR 2023.

[15] Ziyang Zong. ; Zhaohuan Zhan. ; Guang Tan. ; House-LLM: LLM-Assisted Two-Phase Text-to-Floorplan Generation. arXiv 2024.