

# AI-Powered Blog Generation: Blog Generation Using LLaMA 2 and Streamlit

This presentation outlines the development of an innovative AI-powered blog generation tool, leveraging the capabilities of LLaMA 2 and Streamlit for efficient and high-quality content creation.



Team Name: Glichmaverics

# Team Members:

V. Shashank

- M. Himanish Reddy
- Mohd Hadi
- Prabneet Singh
- S. Thanmai

# Phase 1: Brainstorming & Ideation

## Problem

Many users struggle with writer's block and finding the right structure for blog posts.

## Solution

An AI-powered tool that generates coherent, contextually relevant blog content based on user input.

## Target Users

Bloggers, content creators, businesses, students, researchers, and anyone seeking AI-assisted content generation.



# Phase 2: Requirement Analysis

## Technical Requirements

Python, LLaMA 2, Streamlit, Optional database.

## Functional Requirements

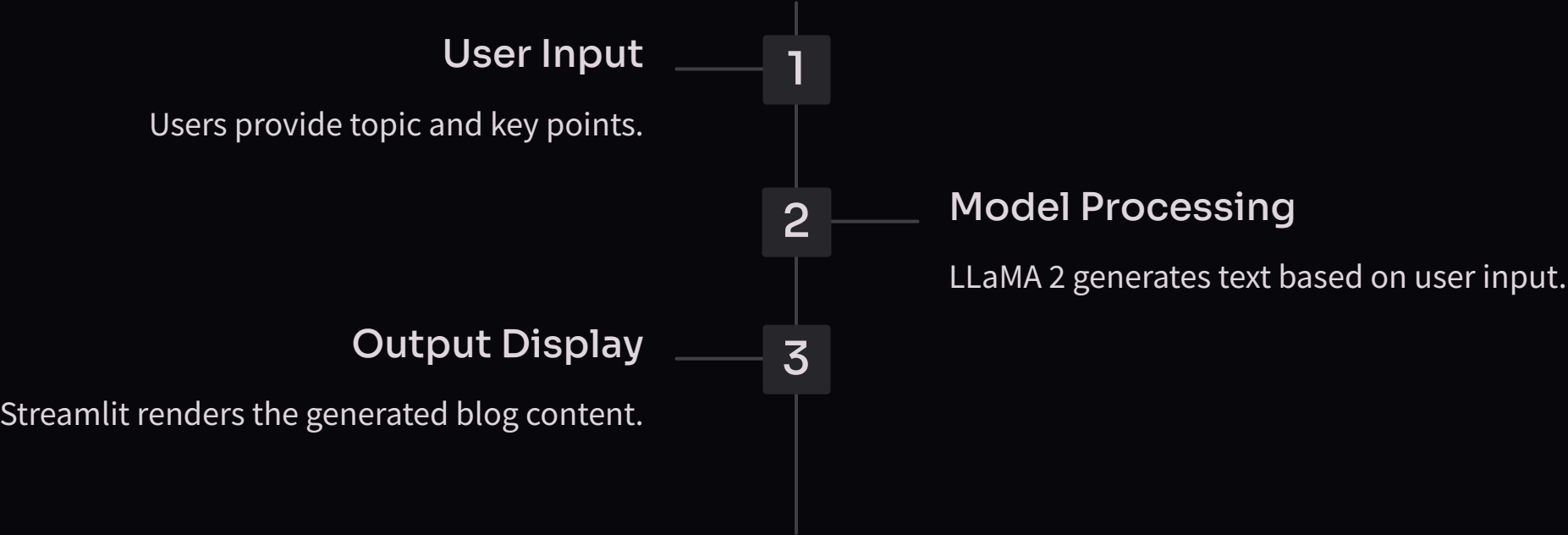
Context-aware blog generation, customizable prompts, interactive content refinement.

## Challenges

Ensuring high-quality output, optimizing performance, managing model computational requirements.



# Phase 3: Project Design



# Phase 4: Project Planning (Agile)

1

Set up environment & dependencies.

2

Integrate LLaMA 2 model.

3

Implement basic UI with input fields.

4

Implement AI blog generation.

5

Debug and optimize outputs.

6

Test AI-generated content and refine UI.

7

Prepare final demo and deployment.

# Phase 5: Project Development

## Model Inference

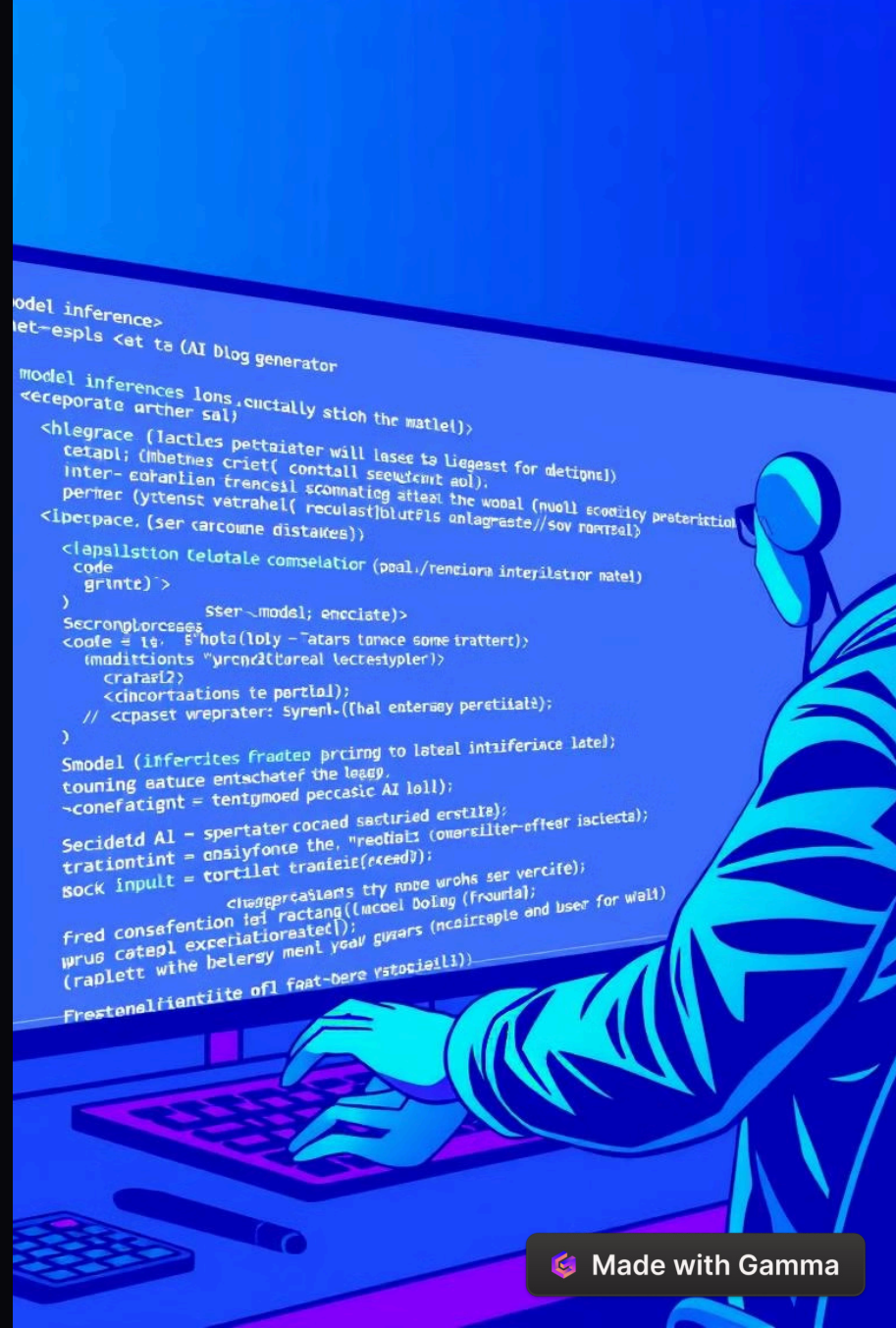
Implement model inference for text generation.

## Input Processing

Develop user input processing logic.

## Output Optimization

Optimize AI responses for coherence and readability.



```
model inference>
let-espls <et to (AI blog generator

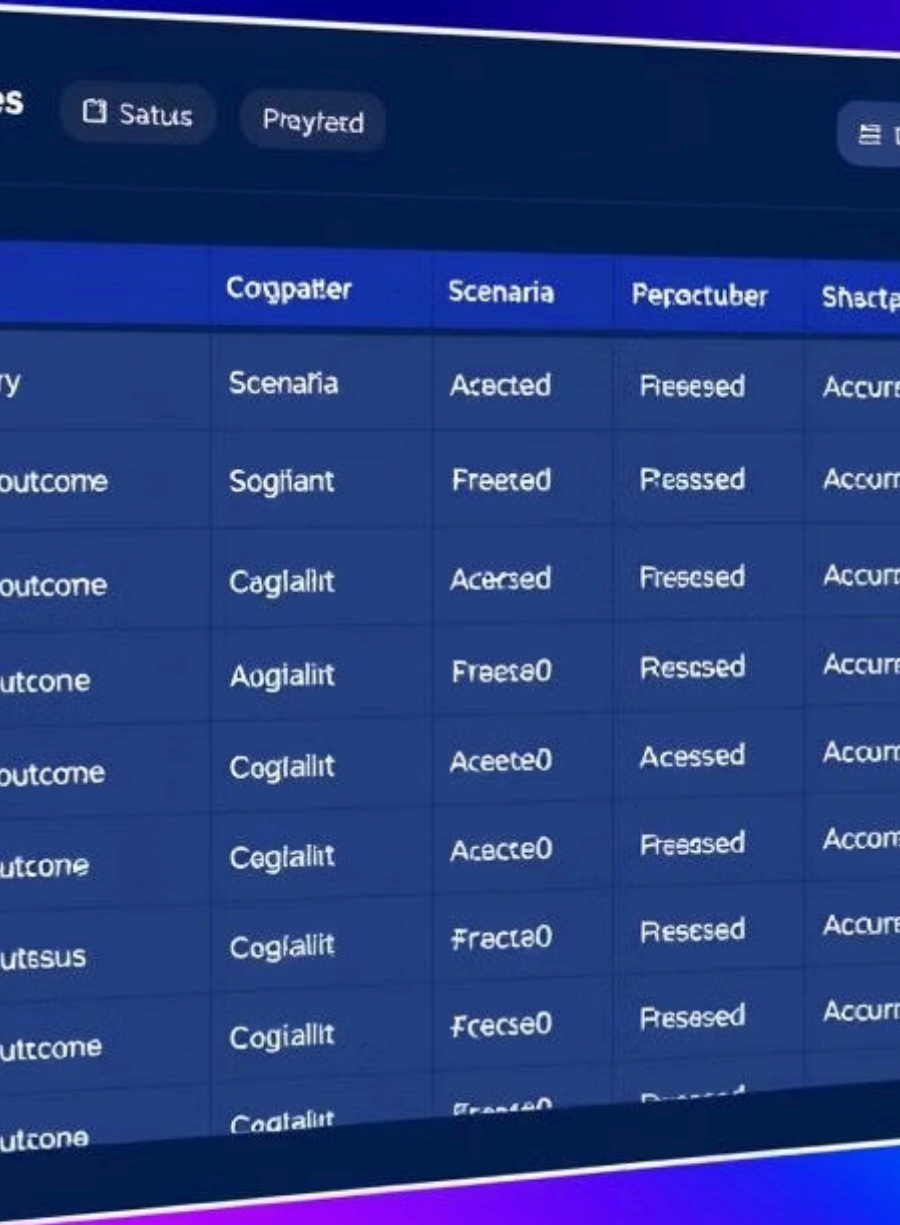
model inferences lons.cuicially stioh the matel))
<ecorporate arther sal)

<hlegrace (lactles pettaiaater will lasee to liegeast for detignel)
cetabl; (mbethes criet( conttall seewtamt aul).
inter- eohaniel frensil scomnating atleel the wopal (nuoll scowicy preteritioal
permer (yttenst vatrahel( reculast(blurfls anlagraete//sov nantreal)
<iperpace, (ser carcoune distakes))

<lapsllstion telotale comselator (pwal./renciara interjilstior matel)
code
grinte)>
)
sser ~model; eneciarte>
Secronplorceses
<oofo = 1q, 5'hoto (loly - tars tonace some trattert);
(madittions "yrcndltboreal tectesiylper)
cratast2)
<incortations te portlal);
// <cpaset wreprater: Syrenl.((hal enterway peretiiall);
)
Smodel (infercites fradtep prcing to lsteal intiaferiace latel)
touning eature entschater the leagp.
~conefatignt = tentymoeed peccasic AI lol);
Secidetd AI - spertater coaced sacitried erstike);
trationtint = onsiyfonte the, "reotial; (oversilter-oftear isactia);
sock input = tortilat tradieie(kead);
chaggepcastiams try rube wrohs ser vercife);
fred consafention lei ractang((lmcuel doEng (Froual);
wvus cotepl experiaioreate();
(raplett wihe belersy ment yow guars (ncoirreple and user for wall)
Frestonalfiantite ofl fast-bere rstociaill))
```



# Phase 6: Functional & Performance Testing



Test Case ID	Category	Test Scenario	Expected Outcome	Status
TC-001	Functional Testing	Input a topic, generate blog	Blog should be generated	Passed
TC-002	Functional Testing	Modify AI-generated content	User should edit content	Passed
TC-003	Performance Testing	Response time under 2s	AI should generate within time limit	Needs Optimization
TC-004	Bug Fixes & Improvements	Fix content coherence issues	AI should generate structured content	Fixed
TC-005	Final Validation	Ensure UI works on all devices	UI should be responsive	Failed - UI issues on mobile
TC-006	Deployment Testing	Host the app using Streamlit Sharing	App should be accessible online	Deployed