Project Architecture

Blog Creator Web Application

– BlogVerse –

Revision Number: 1.0
Last date of revision: 04/10/2023

Document Version Control

Date Issued	Version	Description	Author
04/10/2023	1	Initial Arch - V1.0	Abhijit Paul

Contents

Abstract

The Low-Level Design (LLD) document for the Blog Creator Web Application developed using Python Flask provides a detailed insight into the internal architecture, components, and interactions within the system.

This document outlines the key technical aspects of the application, including the frontend and backend components, data flow, data tables endpoints, and security measures.

It serves as a comprehensive guide for developers and stakeholders, ensuring a clear understanding of the application's design principles, facilitating efficient implementation, and ensuring consistency in the development process. The document emphasizes the use of Python Flask as the backend framework, outlining its specific functionalities in the context of the Blog Creator Web Application.

This document helps everyone involved understand how the blog website is designed and built, making it easier for developers to create a functional and user-friendly platform for blogging.

1 Introduction

1.1 Why this Low-Level Design Document?

The purpose of this Low-Level Design (LLD) Document is to add the detailed description of the Blog Creator Web Application. It will explain the purpose and features of the system, the interfaces of the system, what the system will do, the constraints under which it must operate and how the system will react to external stimuli. This document is intended for both the stakeholders and the developers of the system and will be proposed to the higher management for its approval. The Blog Creator Web Application is designed to allow users to create, edit, and manage their blogs online. This document provides a detailed description of the application's internal design, focusing on various components and their interactions.

The LLD will:

- Present all of the design expects and define them in detail
- Describe the user interface being implemented
- Describe the hardware and software interfaces
- Describe the performance requirements
- Include designs which are the architecture of the project
- List and describe the non functional attributes like:
    - ☐ Security
    - ☐ Reliability
    - ☐ Maintainability
    - ☐ Portability
    - ☐ Reusability

1.2 Scope

This software will be a web application. This system will be designed to allow users to create, edit, and manage their blogs online. LLD documentation presents the detailed structure of the system, such as database architecture, application architecture (layers), application flow (Navigation), and technology architecture.

1.3 Constraints

We will only be developing the Web application interface.

1.4 Risks

Document specific risks that have been identified or that should be considered.

1.5 Out of Scope

Exposing API endpoints are out of scope for the project.

2 Technology Stack

2.1 Server Development

Package Name	Class/Method
flask	• Flask (class) • Blueprint (class) • render_template • redirect • url_for • request • flash • current_app
flask_user	• UserMixin (class) • UserManager (class) • login_required • current_user
flask_mongoengine	• MongoEngine (class)

	● MongoEngineSessionInterface (class)

2.2 Front End Development

Module	Usages
HTML	Static Pages
CSS	Page Design
Jinja2	Template Engine for Dynamic Pages
Bootstrap	Open-source CSS framework

2.3 Database Stack

- MongoDB

2.4 Deployment

- Render
- AWS

3 Database Schema

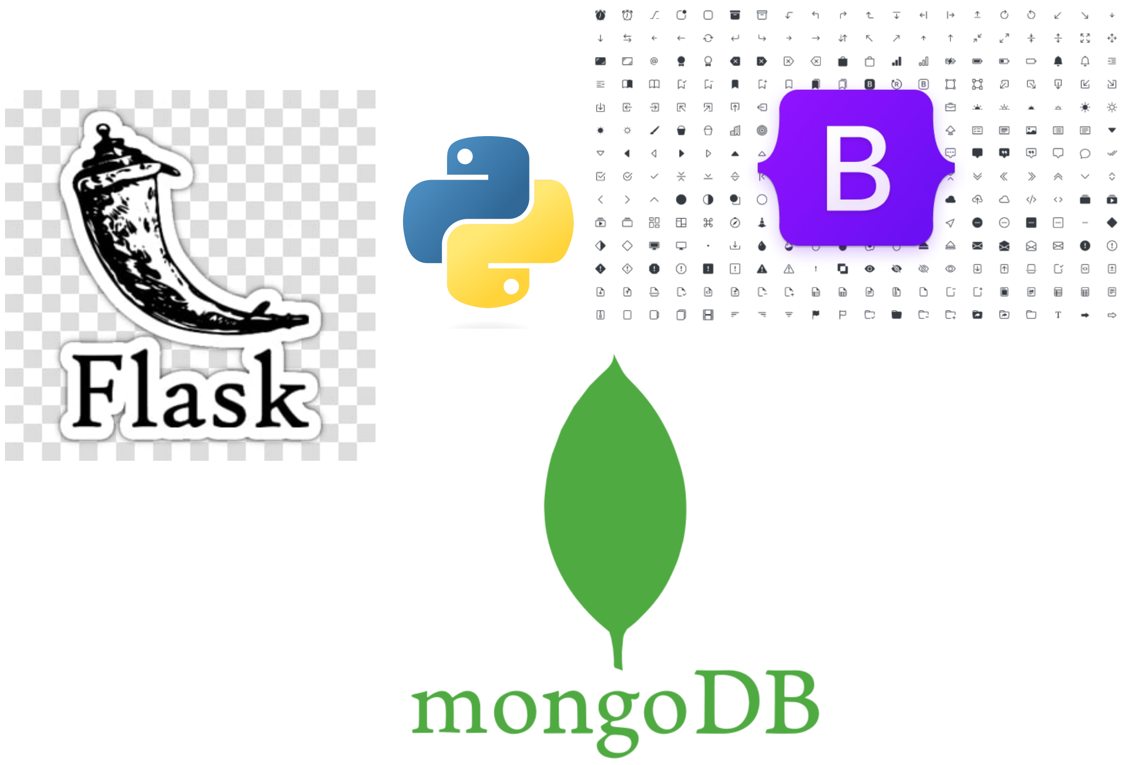
Collection Name	Schema Layout
USER	active – BooleanField username - StringField first_name - StringField last_name - StringField email - StringField roles - ListField password - StringField email_confirmed_at - DateTimeField
CATEGORY	category_name - StringField
BLOG	category_id - ObjectID blog_user_id - ObjectID blog_text - StringField blog_creation_date - DateTimeField blog_read_count - IntField blog_rating_count - IntField
COMMENT	blog_id - ObjectID blog_comment - StringField comment_user_id - ObjectID blog_rating - IntField blog_comment_date - DateTimeField

4 Proposed Solution

The solution proposed here is a web application based on Flask technology supporting the backend framework and HTML, CSS, JINJA templating and constructing the front end architechture. Users can view a gist of blogs posted on the platform. But to view the entire blog, users have to register on the platform which will ensure user engagement. Email verified users will have access to login to the platform and create blogs using HTML formatting tags, update blogs and view other’s blogs. Users will be able to share the link of their blogs to others.

4.1 Tools used

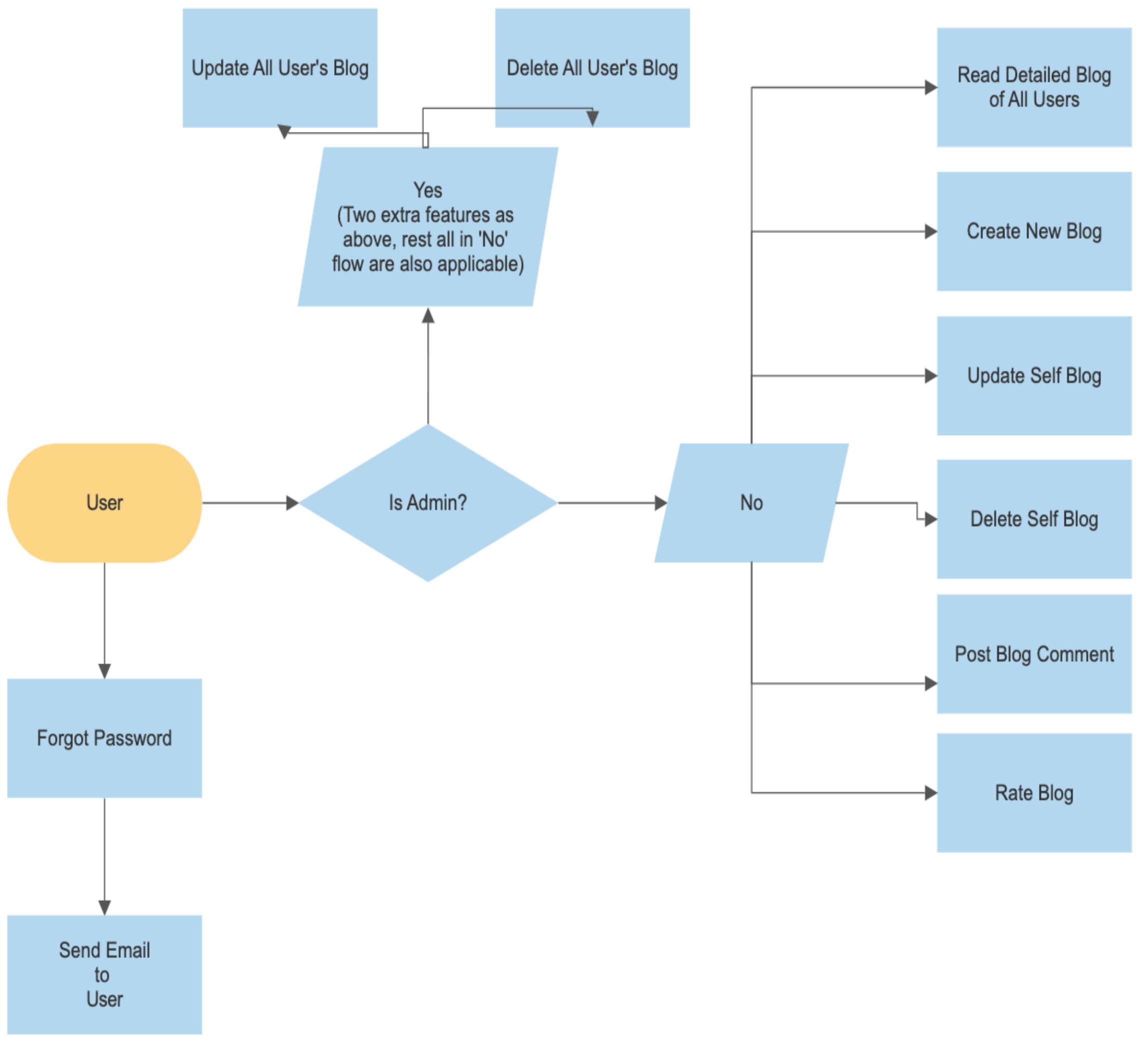
Python programming language and libraries such as Flask, FlaskLogin, BluePrints, Werkzeug, PyMongo, HTML, CSS, BootStrap are used to build the whole web application.



4.2 Assumptions

The main objective of this project is to implement the use-cases as previously mentioned (2.2 Problem Statement) to facilitate the blog creation capabilities.The front end web interface is created using HTML, CSS & BootStrap. Users are allowed to use all HTML tags to format the contents of the blog posts.

5 User I/O Workflow



6 Test Cases

Test case	Steps to perform test case	Module	Pass/Fail
1	Add new user	User Model	
2	Add new blog	Blog Model	
3	Add new comment	Comment Model	
4	Delete comment	Comment Model	
5	Delete blog	Blog Model	
6	Delete User	User Model	

7 Conclusion

The blogging website will serve CRUD(Create, Read, Update and Delete) functionality by using the Flask web framework in Python. It will prioritise user experience, offering a seamless login process that ensures the security and privacy of every user, so that users can have a pleasant environment and enjoy blogging!