

Low Level Design (LLD)

Blog Creator Web Application

– BlogVerse –

Revision Number: 2.0

Last date of revision: 22/09/2023

Contents

Document Version Control	2
Abstract	4
1 Introduction	5
1.1 Why this High-Level Design Document?.....	5
2 General Description	6
2.1 Product Perspective.....	6
2.2 Problem Statement.....	6
2.3 Proposed Solution.....	7
2.4 Further Improvements.....	7
2.5 Tools used.....	7
2.6 Constraints.....	8
2.7 Assumptions.....	8
3 Design Details	9
3.1 Process Flow.....	9
3.2 Event log.....	9
3.3 Error Handling.....	9
4 Performance	10
4.1 Reusability.....	10
4.2 Application Compatibility.....	10
4.3 Resource Utilisation.....	10
4.4 Deployment.....	10
5 Conclusion	10

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

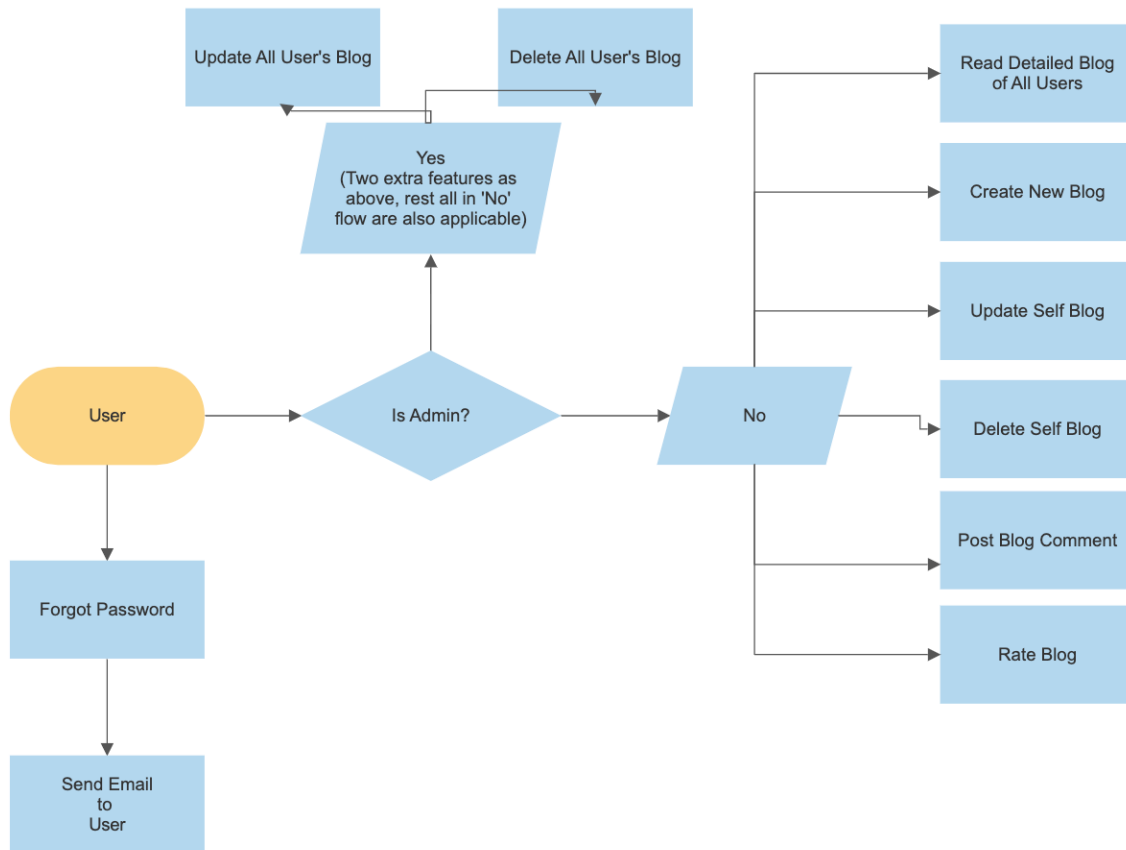
1.1 Why this Low-Level Design Document?

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.

1.2 Scope

Low-level design (LLD) is a component-level design process that follows a step-by-step refinement process. This process can be used for designing data structures, required software architecture and source code. Overall, the data organisation may be defined during requirement analysis and then refined during data design work.

2 Architecture



3 Architecture Description

3.1 Product Perspective

The Blogging Website is a web application developed using Python Flask, designed to provide a platform for users to create, publish, and interact with blog posts.

3.2 Front End Web Interface

In order to create the web interface we will need to use

- HTML
- Native CSS along with Bootstrap
- Jinja2

3.3 Server Development

To create a robust and simple backend server process we will use **flask & flask_user** which will cater creating routes and models and handling the user requests and serve responses.

3.4 Database

- a. Database Creation and connection - Create a database on MongoDB. If the database is already created, open the connection to the database using the URI.
- b. Collection creation in the database using MongoEngine ORM
- c. Insertion of documents in the collections.

3.5 Insertion of data into Database using ORM

To create a swift mechanism of creating schema-less collections in MongoDB, we will use **flask_mongoengine** which is an ORM.

This schema will never be passed on to MongoDB — this will only be enforced at the application level, making future changes easy to manage. Also, the User documents will be stored in a MongoDB collection rather than a table.

3.6 User Registration

As soon as the root (`/`) route of the Web Application will be called, by default an 'Admin' user will be created in the database.

To get other users registered on the web application, we will make use of the 'Registration Form'.

3.7 User Login

Once the registered user activates their account using the email link, users will be able to login to the web-application and access the features of it thoroughly.

3.8 Blog Creation

To create a blog, logged in users will be able to use this feature. Once the blog gets created, it will be immediately visible on the dashboard to all the other users.

3.9 Blog Reading

Logged in users will be able to read the blogs posted by other users. Once the blog is opened in full view mode, read count gets incremented by '1'. If the user opens up any of the blogs written by him/herself, it will be launched in update/delete mode.

3.10 Blog Updation

Users will be able to modify the blog written by him/herself only. Modification will be immediately visible on the dashboard to all the other users.

3.11 Blog Deletion

Users will be able to delete the blog written by him/herself only. As soon as a Blog post gets deleted, it will be immediately removed from the application.

3.10 Rating & Commenting Blogs

Logged in users will be able to comment on the blogs posted by other users. Also they will be able to choose a rating on the scale of 1 to 5. This feature will not be available on blogs written by the logged in user.



4 Deployment

We will be deploying the web application to AWS as well as Render. For AWS, we will make use of AWS CodePipeline for CI/CD and AWS ElasticBeanstalk for orchestration services such as EC2.



5 Unit Test Cases

Test Case Description	Pre-Requirement	Expected Result
Verify whether the Web Application URL is accessible to the user	<ol style="list-style-type: none"> 1.Application URL should be defined 2.Application should be hosted and live 	Application URL should be accessible to the user
Verify whether user is able to sign up in the application	<ol style="list-style-type: none"> 1.Application is accessible 2.Application should be hosted and live 	Users should be able to sign up in the application
Verify whether user is able to successfully login to the application	<ol style="list-style-type: none"> 1. Application is accessible 2. User is signed up to the application 	Users should be able to successfully login to the application
Verify whether user is able to 'add a new user' on application	<ol style="list-style-type: none"> 1. Application is accessible 2. User is signed up to the application 3. User is logged in to the application 	Users should be able to create a new user
Verify whether user is able to 'add a new blog post' on application	<ol style="list-style-type: none"> 1. Application is accessible 2. User is signed up to the application 3. User is logged in to the application 	Users should be able to post a new blog
Verify whether user is able to 'add a new comment' on a blog on the application	<ol style="list-style-type: none"> 1. Application is accessible 2. User is signed up to the application 3. User is logged in to the application 	Users should be able to post a new comment on a blog post

 High Level Design (HLD) Verify whether user is able to 'delete a comment' on a blog on the application	<ol style="list-style-type: none"> 1. Application is accessible 2. User is signed up to the application 3. User is logged in to the application 	 Users should be able to delete the comment on a blog post
Verify whether user is able to 'delete a blog' on the application	<ol style="list-style-type: none"> 1. Application is accessible 2. User is signed up to the application 3. User is logged in to the application 	Users should be able to delete the blog post.
Verify whether user is able to 'delete user' on the application	<ol style="list-style-type: none"> 1. Application is accessible 2. User is signed up to the application 3. User is logged in to the application 	User should be able to delete the user from the web application.

6 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!