SYSTEM DESIGN DOCUMENT

Blog Platform

25-05-2023

1. Introduction

1.1 Purpose of the SDD:

The purpose of a System Design Document (SDD) is to provide a comprehensive overview of a system's design and architecture. This includes detailing the system's components and their interactions, describing functional specifications, explaining key design choices, establishing design patterns and best practices, facilitating better understanding and communication among stakeholders, serving as future reference documentation, and aiding in risk management. In the specific example given, the SDD is used to define and explain the design and functionality of a Blog Platform.

2. General Overview and Design Guidelines

2.1 General Overview:

The Blog Platform is a web-based application that supports publishing and reading blogs, with roles for Admins, Authors, and Readers. Admins manage the platform, Authors create content, and Readers consume it. It operates on a client-server system and employs a three-tier architecture:

1. The Presentation Layer is the user-facing front-end.
2. The Application Layer handles the business logic and mediates between the front-end and back-end.
3. The Data Layer, which is responsible for storing and retrieving data.

This design allows for centralized control, security, and updates, and ensures flexibility, maintainability, and scalability.

2.2 Assumptions/Constraints/Risks

2.2.1 Assumptions:

- Users have access to modern web browsers: This assumption is fundamental for the design and development of the Blog Platform. It suggests that users will be utilizing up-to-date web browsers, which will support the latest web technologies, design aesthetics, and security standards that the platform employs.
- The system will be developed using a specific programming language and framework (e.g., Golang with Mux): This implies that the system's developers have decided on the technology stack, in this case, Golang as the programming language, and Mux as the

web framework. These technologies are typically chosen for their efficiency, scalability, and robustness, which can greatly benefit the Blog Platform.
- Adequate server resources will be available to handle the expected user load: This assumption assumes that the necessary server infrastructure will be in place to handle the expected traffic and requests from users, ensuring that the platform performs optimally without overloading servers.

2.2.2 Constraints:

- The system must be developed within a specific budget and timeline: This constraint signifies that the development process has set limits in terms of budget and time. Developers will need to effectively manage resources and schedules to deliver the system within these boundaries.
- Compatibility with multiple web browsers and devices must be ensured: This constraint necessitates that the Blog Platform be designed and developed to function seamlessly across various web browsers (like Chrome, Firefox, Safari, etc.) and devices (like desktops, tablets, smartphones, etc.). This ensures a broad reach and optimal user experience.
- The system should adhere to industry best practices for security and performance: This implies that the Blog Platform should incorporate standard security measures to protect against cyber threats and maintain optimal performance levels. Practices might include secure coding, regular testing, performance optimizations, etc.

2.2.3 Risks:

- Potential risks include security vulnerabilities: The Blog Platform, like any other web application, is susceptible to various security threats and vulnerabilities. These might include injection attacks, data breaches, or unauthorized access, among others.
- Scalability issues: As the Blog Platform grows and attracts more users, it may face challenges in efficiently scaling up to meet the increased demand. This could lead to slow response times, service outages, or other performance problems.
- Data loss: This risk is associated with the potential loss of data due to a system crash, server issues, or security breaches. Data loss can have significant consequences, impacting user trust and platform integrity.

2.3 Alignment with Federal Enterprise Architecture:

The Blog Platform will be designed to align with relevant architectural frameworks and standards, ensuring compatibility and interoperability with other systems.

3. Design Considerations

3.1 Goals and Guidelines:

- Provide a user-friendly interface for blog creation, reading, and commenting: The Blog Platform aims to offer an intuitive and easy-to-navigate interface. This involves a well-structured layout, clear navigational elements, and user-friendly functionalities, making it simple for users to create, read, and comment on blogs.
- Ensure system security and protect user data: The platform places high importance on security. Measures to protect the system and user data might include data encryption, secure user authentication, regular system updates and patches, and compliance with data protection regulations.
- Optimize system performance and scalability: The Blog Platform aims to deliver high-speed performance, even under high user loads. Scalability considerations are also important, so the system can efficiently handle a growing number of users and increased data over time.

3.2 Development Methods & Contingencies:

- Agile development methodology will be followed, with iterative development and continuous feedback: This method involves breaking the development process into smaller, manageable stages or "sprints." Each sprint results in a usable part of the system and allows for regular feedback and adjustments to be made.
- Contingency plans will be established to address potential risks and challenges during development: These plans are fallback measures to ensure that work can continue even when unexpected problems occur. This could involve alternative development paths, additional resource allocation, or back-up systems.

3.3 Architectural Strategies:

Client-server architecture with a three-tier structure (presentation, application, and data layers).

- The client-server architecture will be employed to enable the separation of concerns between the client-side and server-side components. This allows for efficient resource management and improved scalability.
- The three-tier structure will consist of the presentation layer, responsible for user interface rendering and interaction, the application layer, which handles business logic and processing, and the data layer, responsible for data storage and retrieval.

Use of web frameworks, such as Golang(Mux), to facilitate development and maintainability.

- The Golang(Mux) web framework will be utilized to streamline the development process and enhance maintainability. It provides a robust set of tools and libraries for building web applications in Go, offering features like routing, middleware, and request handling.

4. System Architecture and Architecture Design

4.1 Logical View:

User management: This component will be responsible for managing users, including creating, deleting, and updating user accounts. It will also be responsible for managing user permissions, such as which users can create blogs, comment on blogs, and view other users' profiles.
Blog management: This component will be responsible for managing blogs, including creating, deleting, and updating blogs. It will also be responsible for managing blog permissions, such as which users can view the blogs.
Blog display: This component will be responsible for displaying blogs to users. It will retrieve blogs from the database and render them in a user-friendly format.
Profile management: This component will be responsible for managing user profiles. It will allow users to create and update their profiles, and it will also allow users to view other users' profiles.
Admin controls: This component will be responsible for providing administrative controls for the system. It will allow administrators to create and delete blogs.

The logical view of the system is a high-level overview of the system's components and their interactions. It is used to understand the overall structure of the system and to identify the key components that need to be developed.

4.2 Hardware Architecture:

4.2.1 Security Hardware Architecture:

- Firewalls will block unauthorized access to the system.
- Intrusion detection systems will detect and block unauthorized attempts to access the system.
- Web application firewalls will protect the system from common web application attacks.
- Data encryption will protect sensitive data from unauthorized access.

4.2.2 Performance Hardware Architecture:

High-performance servers and network infrastructure will be utilized to handle user traffic and ensure system responsiveness.

- High-performance servers will handle user traffic and ensure system responsiveness.

- The high-speed network infrastructure will ensure that users can access the system quickly and easily.

4.3 Software Architecture:

4.3.1 Security Software Architecture:

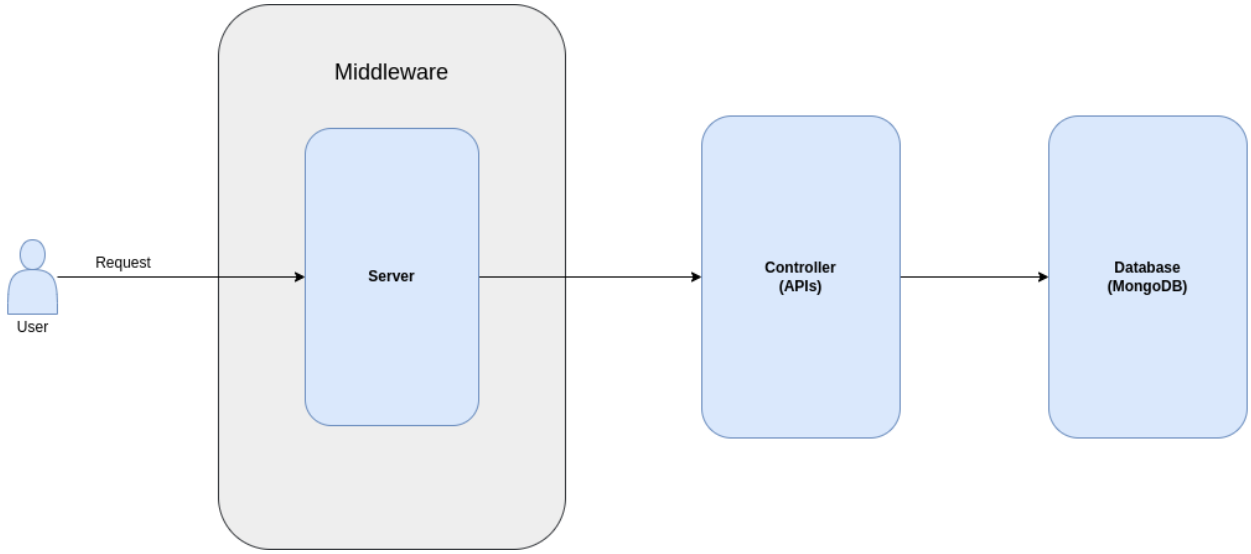
- Encryption will be used to protect sensitive data from unauthorized access.
- Secure authentication will be used to verify the identity of users before they are granted access to the system.
- Access control will be used to restrict users' access to system resources based on their permissions.

4.3.2 Performance Software Architecture:

- Efficient database queries will be used to reduce the amount of time it takes to retrieve data from the database.
- Resource utilization will be optimized to ensure that the system is using its resources efficiently.
- The software architecture of the system is designed to protect the system from unauthorized access and to ensure that the system can handle a large number of users.

In addition to the security and performance measures mentioned above, the software architecture will also include the following features:

- **Scalability:** The software architecture will be designed to be scalable so that it can be easily adapted to handle a growing number of users.
- **Reliability:** The software architecture will be designed to be reliable so that it can minimize the likelihood of system outages.



4.4 Information Architecture:

4.4.1 Records Management:

- **Storage:** Blog records will be stored in a database. The database will be designed to be scalable and reliable.
- **Retrieval:** Blog records will be retrieved using a search engine. The search engine will be designed to be efficient and user-friendly.
- **Indexing:** Blog records will be indexed to improve the efficiency of retrieval. The indexing will be performed using a variety of techniques, including full-text indexing and keyword indexing.

The information architecture of the system is designed to meet the storage, retrieval, and indexing requirements of blog records.

In addition to the storage, retrieval, and indexing requirements, the information architecture will also include the following features:

- **Security:** The information architecture will be designed to protect blog records from unauthorized access.
- **Compliance:** The information architecture will be designed to comply with applicable laws and regulations.

4.5 Internal Communications Architecture:

Communication protocols and mechanisms will be established to facilitate interactions between system components. This will include the use of standard protocols such as HTTP and TCP/IP, as well as custom protocols that are designed to meet the specific needs of the system.

4.6 Security Architecture:

A comprehensive security architecture will be designed, including authentication, authorization, data encryption, and protection against common vulnerabilities. This will involve the use of a variety of security measures, such as firewalls, intrusion detection systems, and data encryption.

4.7 Performance:

Performance testing and optimization will be conducted to ensure the system can handle the expected user load. This will involve testing the system under a variety of load conditions, and making adjustments to the system architecture as needed to improve performance.

4.8 System Architecture Diagram:

A system architecture diagram will be created to illustrate the relationships and interactions between system components. This diagram will be used to communicate the system architecture to stakeholders and to help with the development and maintenance of the system.

5. System Design

5.1 Business Requirements:

- **User Registration:** Users will be able to register for an account on the system. This will require them to provide their username, role, email address, and password.
- **Blog Creation:** Users will be able to create blogs on the system. This will require them to provide a title and description for their blog.
- **Blog Display:** Blogs will be displayed on the system in a list view. Users will be able to filter the list view by title, author, and date created.
- **Profile Management:** Users will be able to manage their profiles on the system. This will allow them to change their username, email address, role, and bio.
- **Admin Controls:** Admins will be able to control the system. This will allow them to add and remove users, create and delete blogs.

The system design will address the business requirements by providing a user-friendly interface that allows users to easily register for an account, create blogs, comment on blogs, and manage their profiles. The system design will also provide admins with the tools they need to control the system.

In addition to the business requirements, the system design will also consider the following factors:

- **Security:** The system design will be secure and protect user data from unauthorized access.

- **Performance:** The system design will be performant and handle a large number of users.
- **Scalability:** The system design will be scalable and can be easily adapted to handle a growing number of users.
- **Reliability:** The system design will be reliable and minimize the likelihood of system outages.

5.2 Database Design:

5.2.1 Data Objects and Resultant Data Structures:

The database design will include entities such as users, blogs, and profiles, with appropriate relationships and attributes.

5.2.2 File and Database Structures:

The file and database structures will be designed to efficiently store and retrieve user and blog-related data.

5.3 User Machine-Readable Interface:

5.3.1 Inputs:

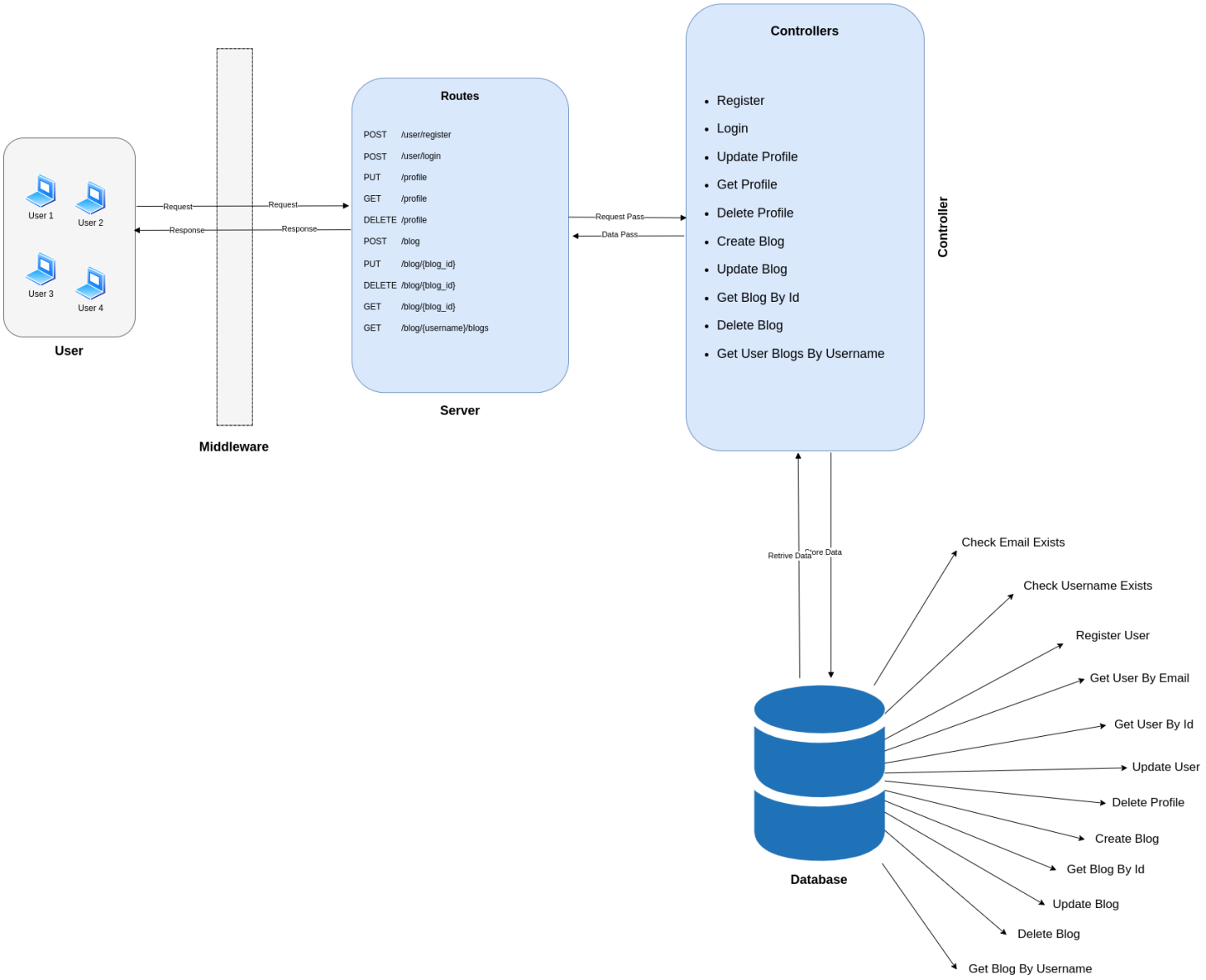
- **User registration details:** This includes the user's name, email address, role, and password.
- **Blog content:** This includes the blog's title and description.
- **Profile information:** This includes the user's name, email address, role, and bio.

5.3.2 Outputs:

- **Blog content:** This includes the blog's title and, description.
- **Profile information:** This includes the user's name, email address, role, and bio.

The user interface will be designed to be user-friendly and easy to use. It will use a variety of techniques, such as graphical user interfaces (GUIs), to make it easy for users to interact with the system.
The user interface will be designed to meet the following requirements:

- **Usability:** The user interface will be easy to use and understand.
- **Security:** The user interface will be secure and protect user data from unauthorized access.
- **Performance:** The user interface will be performant and handle a large number of users.


User 1
User 2
User 3
User 4
User
Request
Response
Request
Response
Middleware
Routes
POST /user/register
POST /user/login
PUT /profile
GET /profile
DELETE /profile
POST /blog
PUT /blog/{blog_id}
DELETE /blog/{blog_id}
GET /blog/{blog_id}
GET /blog/{username}/blogs
Server
Request Pass
Data Pass
Controllers
Register
Login
Update Profile
Get Profile
Delete Profile
Create Blog
Update Blog
Get Blog By Id
Delete Blog
Get User Blogs By Username
Controller
Retrive Data
Store Data
Database
Check Email Exists
Check Username Exists
Register User
Get User By Email
Get User By Id
Update User
Delete Profile
Create Blog
Get Blog By Id
Update Blog
Delete Blog
Get Blog By Username