Software Architecture Design

for

SkyBar Photo Management System

Group ID: R01

Abhyuday Choumal-CS20B1001,
Beerelly Srinitha-CS20B1004,
Kaustubh Kesharwani-CS19B1015,
Rakesh Kumbhkar-CS20B1017,
Sikander Kathat-CS20B1020,
Yugal Garg-CS20B1027.

Instructor: Manish Singh
Course: Software Engineering (CS210)
TA's: Suryamukhi K, Ashish Kishor
Date: 11th March 2022

Contents

1. INTRODUCTION

In this report, the purpose of our system will be briefly mentioned again. And then we aim to briefly explain Software Architecture.

1.1. Purpose

SkyBar photo management system aims to provide people with an efficient medium for photo management and image processing/organization. This system also intends to satisfy people with its mutual online sharing option.

The targeted audience of this system is people of various ages with a minimum requirement of technical knowledge of smart devices.

1.2. Design Goals

High performance, database maintenance, load balancing on servers, scalability, ease of access, low response time, user-friendliness are the main design goals. We make sure that it is easy to maintain as the system is expected to be used for a long time. We try our best to make it as simple as possible in order to reduce the complexity.

Furthermore, we make sure it is portable as we have different developers from different environments. Even adaptability is a considerable issue, as new changes made at later stages should be adaptable.

We do consider the Design trade-offs during the implementation of the system. Here we will be considering the usability-utility trade-off.

Usability - Utility

We plan efficiently in order to balance usefulness and ease of use. We make sure that the system has several useful functionalities for users. Likewise, we will prefer the ease of use over many other functionalities, though. Since our main goal is to provide user-friendliness, we can leave out some secondary functionalities if they worsen usability.

1.3 Definitions, Acronyms, and Abbreviations

React.js - UI library developed by Facebook.
MongoDB - NoSQL database
Express - A Node.js framework
REST - REpresentational State Transfer
API - Application Program Interface
SPA - Single Page Application
OTHERS ARE GIVEN IN SRS

1.4 Stakeholders of SkyBar

The main stakeholders for this system are individual users and system designers/developers. The main concerns of each are as follows:

- For Users: The usability of the system and providing the net worth information and a reasonable response time.
- For designer/developer: The system should be easy to modify and should be adaptable to the new changes which may occur in the future.

1.5. Non-Functional requirements

- High performance: The system must be able to receive large data and should be able to process the head count and store it on the database and maintain it each second.
- User-friendly: The final users will be common people, and it should not be assumed that they are technology experts at all.
- Failure tolerance: The system should be fail-proof and should be able to recover or back up and keep working in a matter of seconds.

- Maturity: The system has to run tests every time so that a change is made, and new tests have to be built for new features. Both unit and inter-module tests should be done.
- Changeability: Everything is very likely to change, so the system must be able to handle any kind of changes in its features in future.

2. Proposed Software Architecture

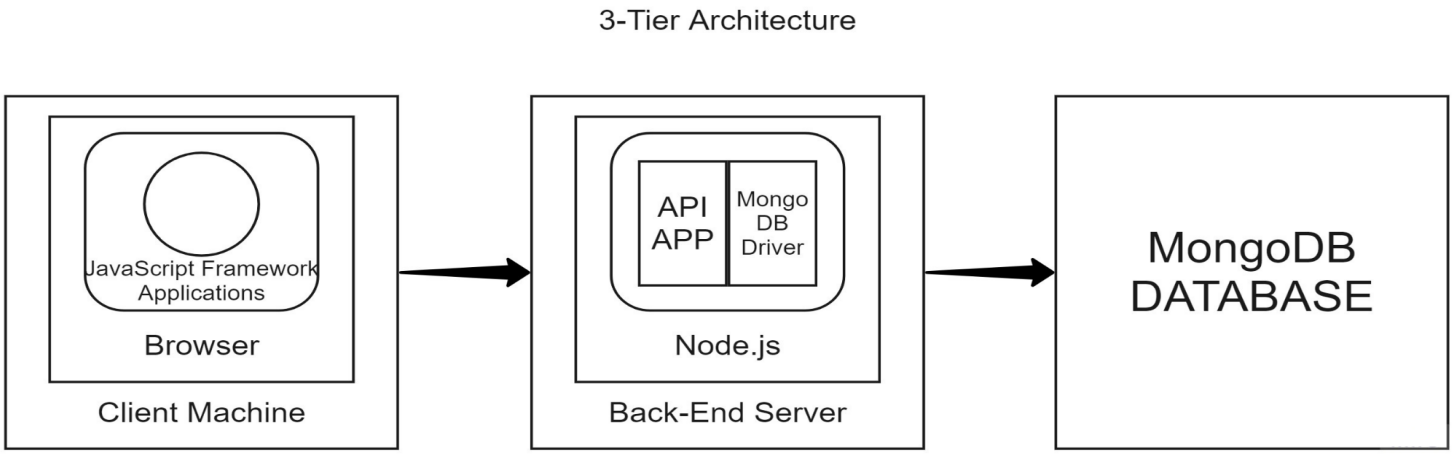
2.1 Architecture Overview

This is an approach to present the information of this system in a structured fashion and discuss its architecture. It also provides the guidelines for the upcoming future developments. Whenever possible, we make use of the existing technology and the usability of the system is taken into account as the first priority.
The structure that the remaining of the document will follow is:
1. A summarized description of the software architecture, including major components and their interactions.
2. Architectural constraints and decisions.
3. A brief description of each component.

2.2 Architecture 1 (3 tier architecture)

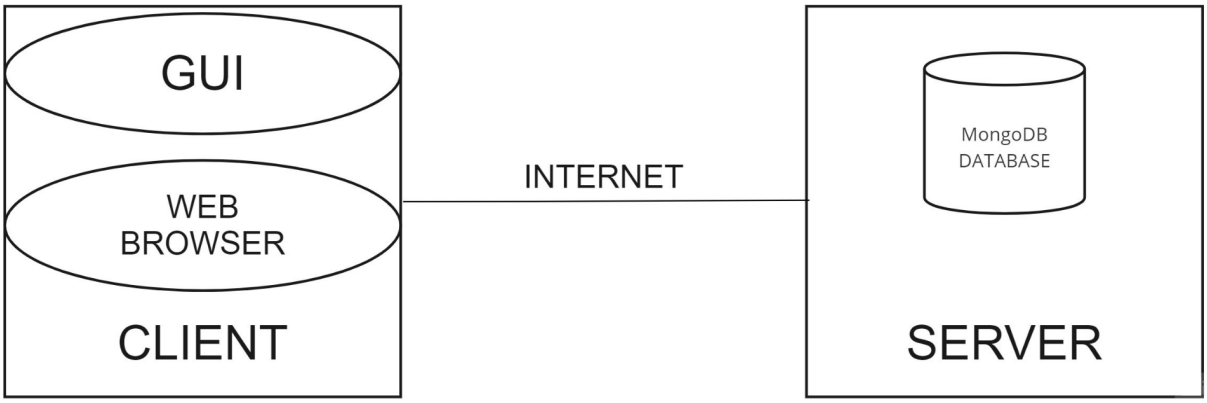
2.2.1 Design

We will be using 3 - tier architecture. The first-tier is the frontend developed by React.js, the second tier is the application (middleware) tier that manages UI and database, and the third tier is the database driver that is responsible for data management.



2.2.2 Hardware/Software Mapping

The architecture of the system is client-server architecture , and thus we have client machines and server machines. As shown in the figure below, the client makes a request and the server sends back the requested data.



2.2.3 Boundary Conditions

- Initialization - A normal flow of events occurs during the initial web page loading. The process remains the same in any web application.
- Failure - Sometimes an internet connection may be timed out. Thus, we should prepare the system for this kind of situation. The system will keep displaying the page instead of the error page during internet connection timeout. The connection timeout page is shown only if the user clicks for any functionality. This may be done by storing all the data locally on the browser. If this is done by the browser itself, then the system should ensure that the data of timeout timestamp is saved. So that the user can continue from where it was left off.
- Termination - The system undergoes termination when the browser window or web application’s tab is closed.

2.3 Architecture 2 (The Repository Model)

2.3.1 Design

The repository model is based on the data repository which contains all the posts, profiles, images, user comments and other data that needs non-volatile storage. This data repository is the central point of this architecture, and all the other modules are developed based on it.

2.3.2 Components in the architecture

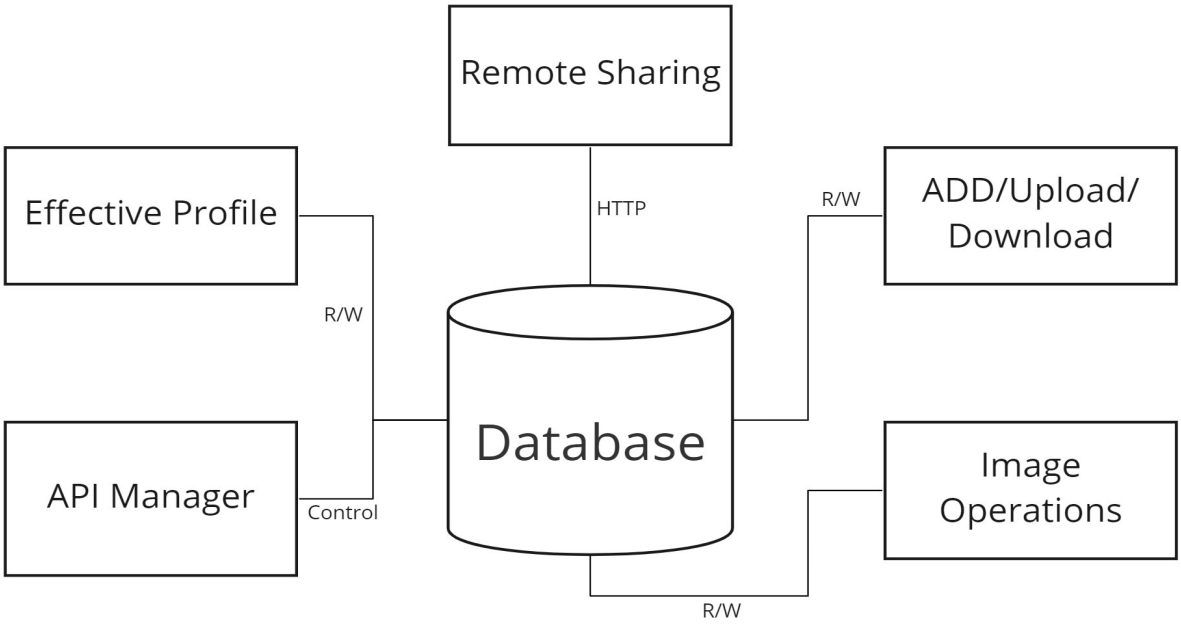
S.N.	Component	Component Type	Description(Detailed overview)
1.	*Data Repository*	Database	Database would have details about all the users, their profiles, photos and all the others.
2.	Master controller	Processing(interface design)	This is the graphical interface design that leads decisions and makes the interaction between the user and other domains and makes proper calls to serve the user requests.
3.	Effective Profile	Processing	Display all the information about the user in an attentive manner.
4.	Add(/upload) and download	Processing(updating database)	For efficient use of the storage, users can upload their photos. This will store all the photos on the database, from where they can take back these at any time.
5.	Create, View, Edit, Delete and Sort etc.	Processing(Accessing, Modification in database)	With this domain users can create new folders(/albums), view their photos from the database, edit photos(As mentioned in SRS), delete the photos from the database or folder, and sort the photos in a folder as mentioned in SRS.
6.	Remote sharing	Processing service	Users have the option to share the

			photos with any available services and devices after acceptance of the request.
7.	API manager	Processing(API)	With this domain, Skybar gets access to perform tasks as mentioned in the SRS.

2.3.3 Connectors in the architecture

S.R.	Connector	Connector Type	Description(Detailed overview)
1.	Write only Connector	Access Database	These connectors work between the data repository and domains which want to modify the data from the database.
2.	Read only connector	Modify the database	These connectors work between the data repository and domains which want to access the data from the database.
3.	Read/Write connector	Access and Modify the database	These connectors work between the data repository and domains which want to access the data as well as modify the data from the database.
4.	URL connector	HTTP	These connectors work between the data repositories, servers and APIs.
5.	Control connector	Function calls(invocation)	These connectors work between the master controller and domains which are granted by this controller.

The diagram for the architecture is given below:



2.4 Comparison of architectures

Criteria	3-tier Model	Repository Model
Changes to data repository layer	Easy	Not Easy
Data security	More secure, as only server accessing the database	Less secure, as all the components are accessing the database
Creating new features	Relatively easier	Easy
Understanding code for developers	Easy	Hard
Debugging	Easy	Hard
Switching database	Easy	Hard(as all the components are connected)
Addition in functionality	Easy	Easy

2.5 Selection and Summary

Based on all the points mentioned in the comparison of architectures, the 3-tier model seems better. Because the future functionalities and features can be developed more easily and can be adapted easily.

Hence, we choose architecture 1 which is 3-tier architecture for SkyBar Photo Management System.

3. Subsystem Services

● UI management - This component is responsible for frontend : showing data and getting input from the user.
● System management - This is the main component that includes and connects all other components.
● User management - This component manages authentication and authorization in the system.
● Data management - This component is a MongoDB driver responsible for managing all the data stored in the database.

4. Glossary

MERN stack - Modern JavaScript full stack for web development (MongoDB, Express.js, React.js, Node.js)