2023

System Design Document

CALYPSO

CSCC01 SUMMER 2023

Table of Contents

Introduction 3

Design Overview 3

Architecture 4

System Decomposition 5

Class Responsibility Collaborator (CRC) Cards 6

Strategy for dealing with errors and edge cases 8

Introduction

Calypso is a media marketplace for users to buy and sell their creative work, whether it is art, music, movies, and any other art form. Calypso aims to create a marketplace that aims to satisfy the needs of a niche of upcoming artists wanting to gain recognition.

Design Overview

Dependencies:

Frontend

- React
- TypeScript
- Tailwind CSS

Backend

- Node.js
- Express

Database

- MongoDB and Neo4j

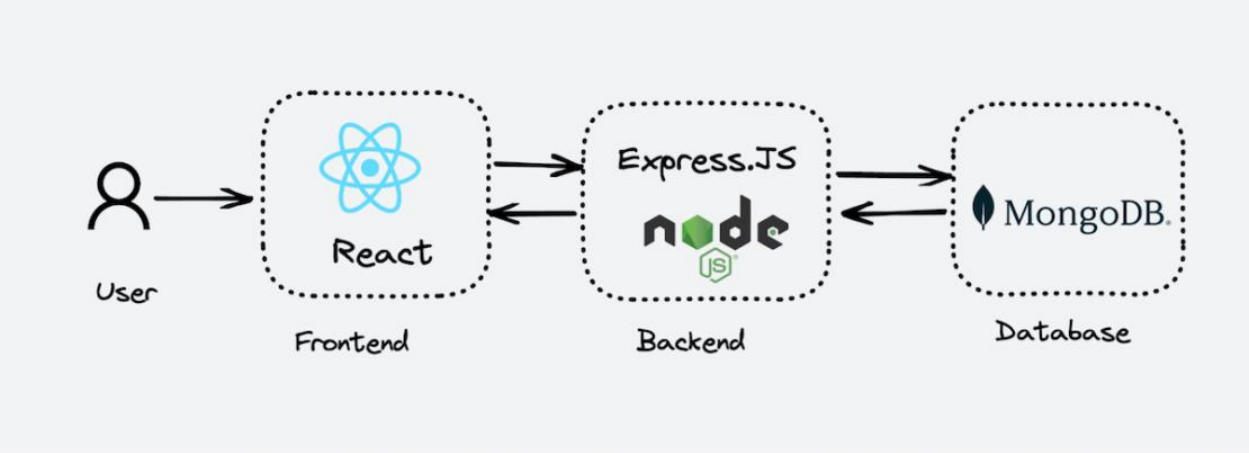


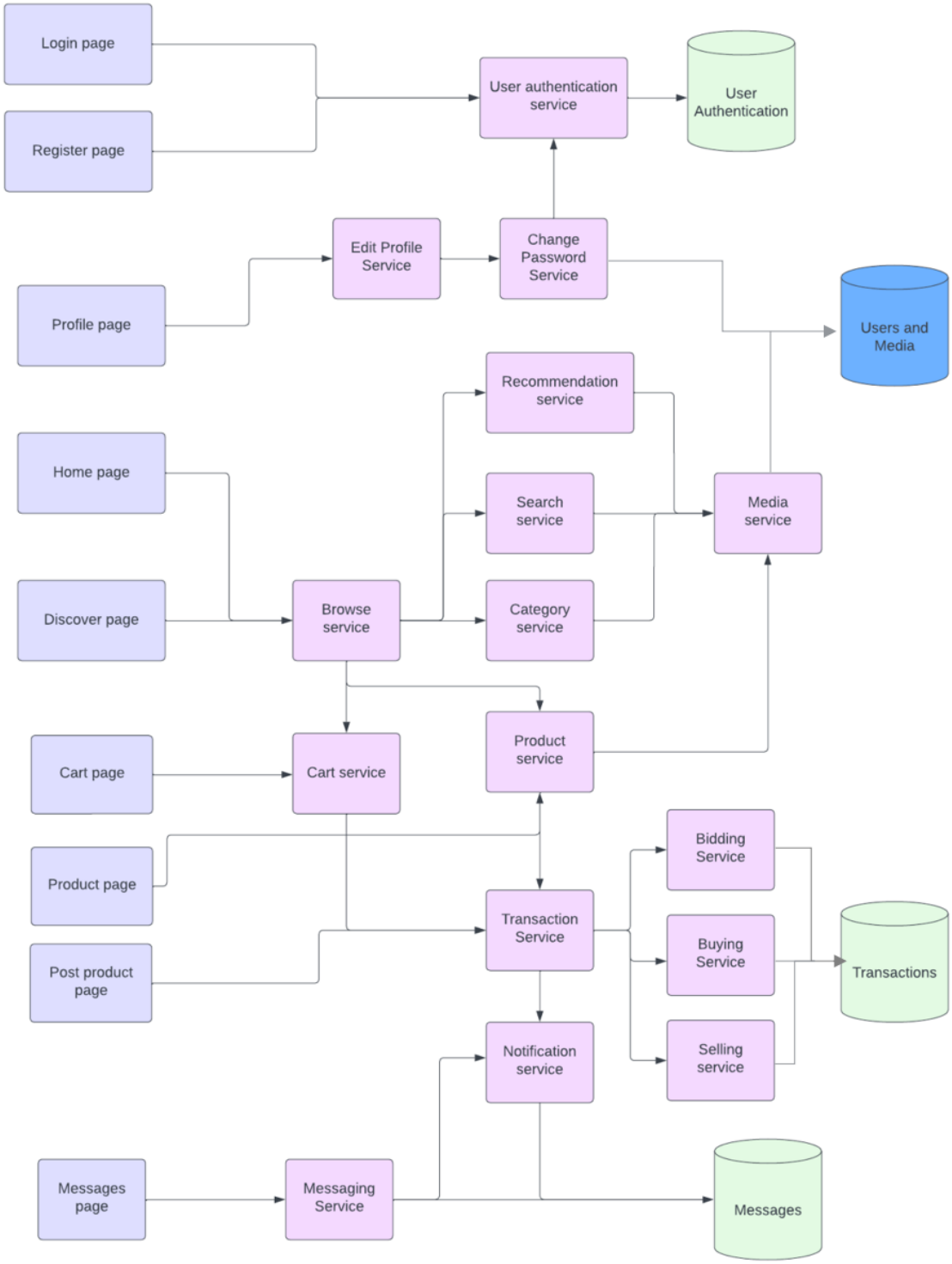
Image source:
https://www.docker.com/wp-content/uploads/2022/09/User-MERN-Flow-Chart-1110x406.png.we bp

Assumptions:

It is assumed that all users have access to a web-browser

Architecture

Our system is divided into specific pages that interact with the services available by our product.



System Decomposition

Components

1. User authentication service
    - To create, edit, or remove a user and store them in the user database
2. Change password service
    - To change the password of a user's account
3. Message service
    - To send and retrieve messages from users from the messages database
4. Edit profile service
    - To edit user data
5. Media service
    - To retrieve art and music with metadata from the media database
6. Browse Service
    - To request media and categories
7. Cart service
    - To add, or remove items to the cart of a user and complete a transaction
8. Recommendation Service
    - To show recommended media to a user based on user data
9. Search Service
    - To search music, art, or artists with or without filters
10. Category service
    - To organize media by category, request a specific category
11. Transaction service
    - To complete a transaction as a bid, buy, or sell, or retrieve transaction history from the transaction database
12. Bidding service
    - To hold an auction for an item, and to place a bid as a user
13. Buying service
    - To complete a purchase of an item as a user
14. Selling service
    - To sell items to a user
15. Notification service
    - To create, and request notifications for a user

Class Responsibility Collaborator (CRC) Cards

User Auth	
• First Name • Last Name • Email • Password	• Arielle

User Profile	
• ID • Username • Description • Profile Pic • Banner	• Tasif

Products	
• ID • Name • Artist • Style • Price • Link • ImageSrc • ImageAlt	• Maaneth • William

Message	
• Sender • Content • Chat • Timestamp	• Cassandra

Chat	
• Name • isGroupChat • Users • Latest Message • Group Admin • Timestamp	• Cassandra

Strategy for dealing with errors and edge cases

Since we are running unit tests to verify the functionality of the program, dealing with errors will not be too time-consuming. Unit tests test a single piece of functionality and so if that returns an error then we do not need to take more time to troubleshoot and find the error. We can just directly go to the problem and solve it.

A strategy to deal with edge cases is a strategy that we learned in CSCA08 called the zero, one, many approaches. where the functionality is tested in the case of 0 items, 1 item, and many items. By doing so, the edge cases are covered.

For anticipated errors and exceptions, a summary of how the software will respond in these situations.

1. *TimeoutException*: occurs when a page takes too long to load.
    a. Abort the client and report the timeout.
2. *CommunicationException*: occurs when a client is unable to interact with a service.
    a. Abort the client and report the communication failure.