CliffBuddy

High Level Design

Team Name: Navite.io

Hunter Lewis (*Team Leader*)

Jay Park

Jerry Belmonte

Ji Park

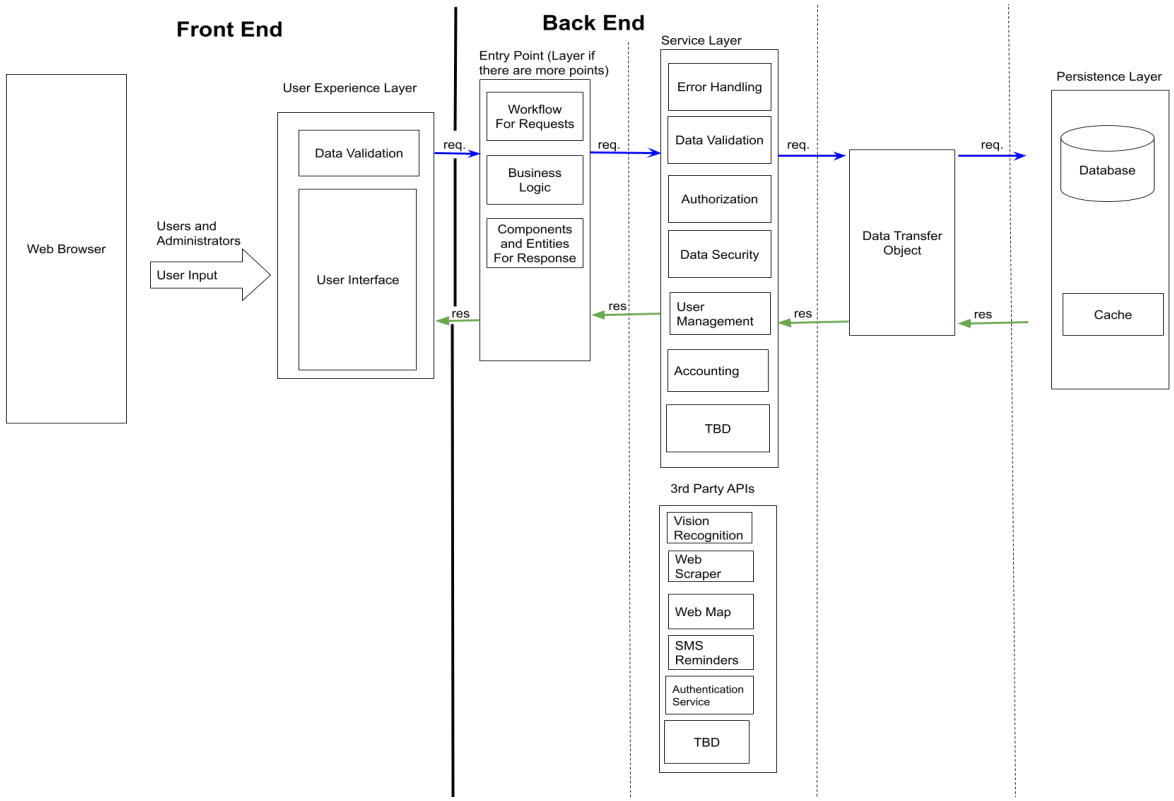
California State University: Long Beach

CECS 491A SEC 05

10/5/21

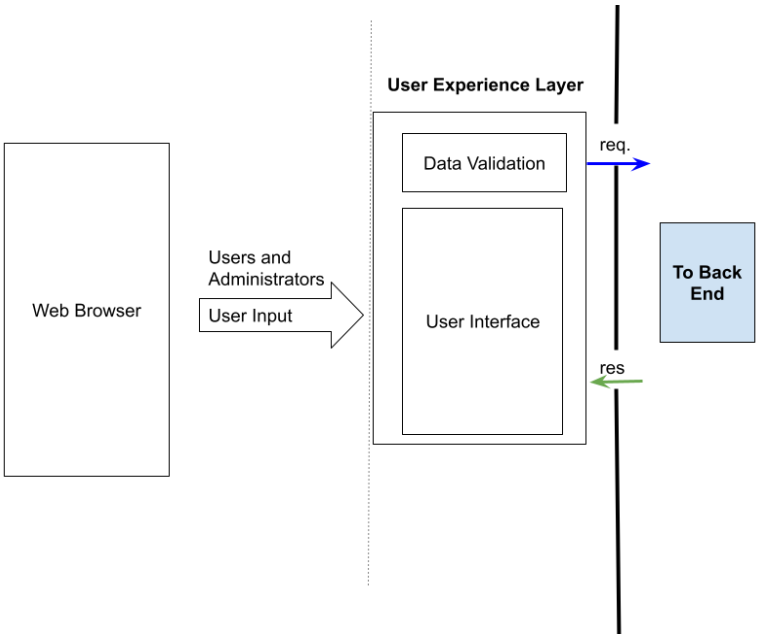
Table of Contents

Abstract Overview



The following diagram displays CliffBuddy's design from a high-level as a whole. Various aspects of both the front end and the back end are subject to change. Throughout all of the layers, there exists a potential for errors to occur, and it will be handled through an error handling service. All information logged throughout all layers will be managed through an accounting service. Security for the system will also have potential to be compromised on all layers, but it will be handled mainly through a security service.

Front End



User Experience Layer

This layer is circulated around the User Interface, aiming to provide a clear and straightforward experience for both users and administrators. This will display the content and features that CliffBuddy delivers, making inputting and displaying information in an easy way.

Through the web browser, the user's general flow would be to:

- Register
- Login/Logout
- Upload puzzles
- Map Solutions
- Visualize Performance
- Discover Other Maps and Solutions
- Utilize the Glossary
- Schedule and Set Reminders for Climbs
- Share Events

Through the web browser, the administrator's general flow would be to:

- Manage Users at a Glance
- Access User Analysis Dashboard
- Set Notifications for Certain Events
- See Event Information

The Data Validation Service is also utilized in the Front End. Any user input will be validated to ensure incorrect information will not be placed. For example, if a user were to input a date that has not occurred yet in “Date of Birth”, the Data Validation Service will notify the user that they need to input valid data. All data except for visual input, password validation, and solution mapping will be handled in the front end’s Data Validation Service. This is because the data being validated will not require a lot of time for the front end to confirm, whereas the data specified will require another layer of Data Validation within the Back End.

Because CliffBuddy is a Web Application, HTTP requests are also specified within the front end as well.

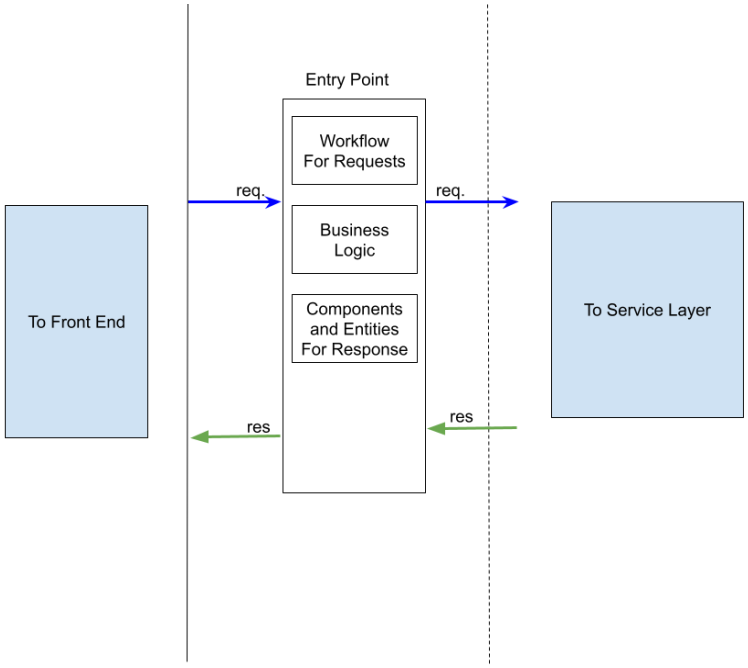
All HTTP Requests include but are not limited to:

- GET
- POST
- HEAD
- PUT
- DELETE

Because CliffBuddy has various functions, other HTTP requests will also be utilized to fit the needs of the requirements.

Back End

Entry Point

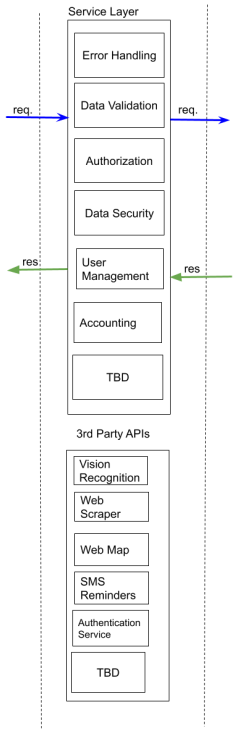


The main function of the entry point is to secure a clean bridge between the flow of data from the front end to the back end of the system. From the User Experience layer, packets are sent over TCP, where all the data within the packet needs to be processed into the server in an organized manner. The workflow will handle what to do with a specific HTTP request, it provides all logic for the application and will send a piece of data to a specific service.

All business logic will also be handled within the entry point, and the requests will not continue on its way to the persistence layer if the data is invalid under business rules. This means that all application constraints will be applied within the data sent.

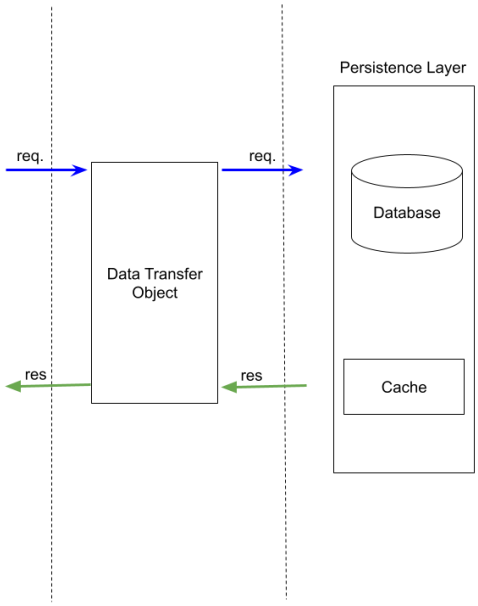
All dependencies and components for the Back End being displayed at the front end will be present. They will be necessary to handle all responses that the server brings back to the entry point.

Service Layer



There will be a service layer to handle requests coming from the front-end and routing them to the back-end. The load balancer will be able to handle multiple requests from having multiple users trying to access our back-end. We want to implement security at this service layer to avoid malicious actors having access to private information that we store about our users. We will manage user information when the backend is sending a response back to the front-end. We plan to avoid bottlenecking for our users by having authenticated users establish a secure connection with our backend through mutual TLS that will make sure our front-end client and back-end server are communicating securely over HTTPS. We will use 3rd party APIs to aid our target system in providing unique features to our application related to vision recognition, SMS reminders, Web Scraping, Web Map, and Authentication Services.

Data Transfer Object and Persistence Layer



After data is processed from the service layer, a DTO will be responsible to carry that processed data to the persistence layer. Because each call is an expensive operation that requires resources, the DTO will aggregate data into one single call to the next layer. This means it will transform all processed data into a readable format that the Database will use. The DTO will also allow data to be updated and deleted before making the call to transfer.

The final manipulated data will be determined to stay within the database or cache. An HTTP response will be made, and will travel back to the front end through the system.