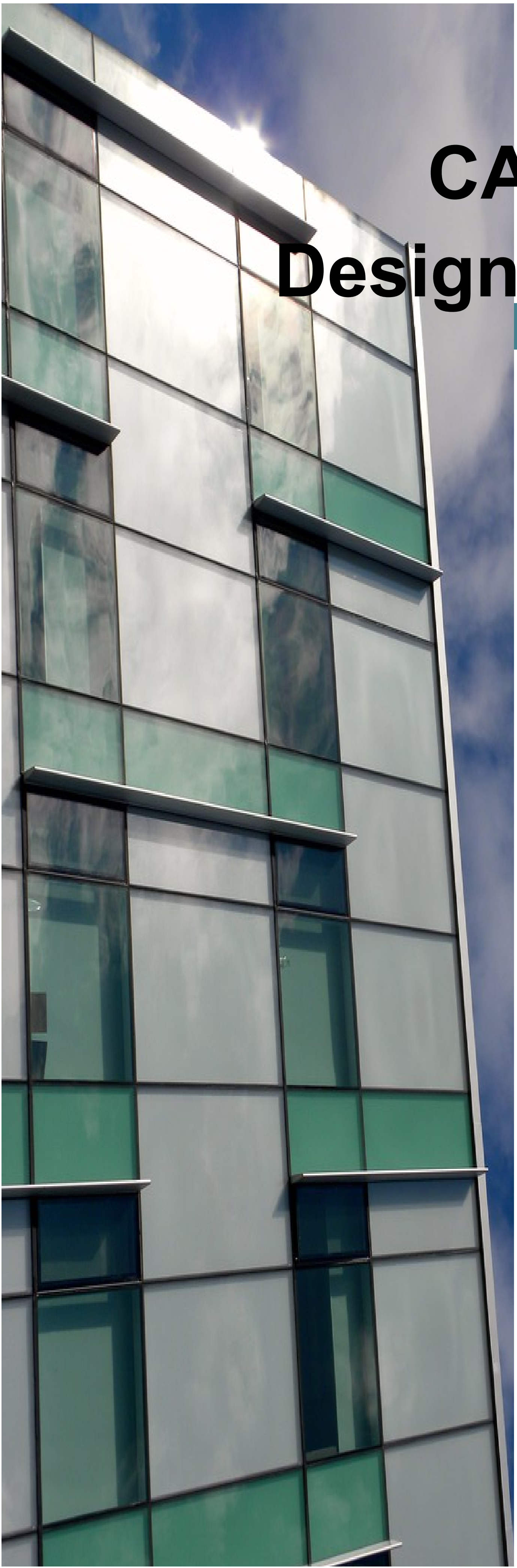


CATKINS - System Design Documentation

By: Mario Liao
Patricia Nagatani
Ilya Kostin
Yangkun Li
Howie Chen
Ran Zi

Table of Contents

Page 1 - Title Page
Page 2 - Table of Contents
Page 3 - 4 - CRC Cards
Page 5 - **7** - Architectural Diagram

CRC Cards

Class Name: Interface - DAO	
Subclasses (if any): List all the subclasses separated by a comma	
Responsibilities ● Create a post ● Create a group	**Collaborators** ● Post ● User ● Group

Class Name: User	
Subclasses (if any): Student, Professor, Department	
Responsibilities ● Has an ID ● Knows username/password ● Knows groups that user is following	**Collaborators**

Class Name: Group	
Subclasses (if any): Courses, Clubs	
Responsibilities ● Stores users as members ● Stores its own information ● Will store posts related to that group ● Determines if a club is official or not based on the owners' email	**Collaborators** ● User

Class Name: Post	
Parent Class (if any): List the parent class if applicable **Subclasses (if any):** List all the subclasses separated by a comma	
Responsibilities ● Has ID ● Knows posting user ● Knows posting time	**Collaborators** ● User ● Group

Class Name: Comment	
Parent Class (if any): List the parent class if applicable **Subclasses (if any):** List all the subclasses separated by a comma	
Responsibilities ● Has ID ● If it’s root ● Replies ● Knows commenting user ● Knows commenting time	**Collaborators** ● User ● Post

Architectural Diagram

App.js - the page with properties applied to all other pages (e.g. navbars)

Left_Navbar - part of the Navbars

Top_Navbar - another part of the navbar. Users can navigate to club creation, user search pages etc

Search_bar.js - component responsible for searching functionality. Gets as props placeholder value and the function, which will run when the form is submitted

Club-list-search-interface - wrapper component for searching for posts. Adds functionality of searching clubs by tags or club names. Gets a function as a prop to change the state of the father component

User-list-element.js - component responsible for styling user data

Users-list.js - component for rendering users, returned in accordance with Search_bar.js input

Clubs-feed - component, which renders all posts and does the filtering for them (
Comment: later will be done using another endpoint
). Also posts can be searched by title there using search bar

Clubs-post - component, which takes props of data from the Model and applies styling to it. Used in Clubs-feed component

Clubs-list-component - gets and renders all clubs in the Model and allow them to access that specific club page

Create-club-component - creates club by values passed from the user (
Comment: not yet implemented
)

Edit-club-component - allows user to edit chosen club if the user is club owner (
Comment: not yet implemented
)

Post-create-component - allows users to create posts that have can have vary properties like if the post is public or private, if it is an announcement, etc

Club-Page-Component - Allows users to see all posts for the specific club they accessed. Only the Owner of the club will be able to see the form to submit a Post for that club (The owner is hardcoded as it depends on User Sessions)

Login_component - Allows users to login (and be redirected to the home page) if the valid username and password is provided. Will return an error if invalid username or password is provided.

comment_management_component - (props: post ID and comments array) - Responsible for handling commenting posts (not replying), also globally stores which comment/post we are commenting/replying

comment - (props: post ID, handler for replying, author, comment ID, content, date, and replies) - Responsible for comment UI and main functionalities: conditionings for opening text area for replying, replies functionalities, what we do when submitting the reply, where we store text we typing to text area related to replies

textAreaForm.js (props: onChange function running when text changed in textarea, placeholder, button text, onSubmit function running when submitting) - Responsible for textarea UI and functionalities

The backend endpoints do what their name says (except for /:id's ones, they can get and delete data, depending on the request type)
Another assumption made about our document is that it is assuming you are using our default database which is:
ATLAS_URI=mongodb+srv://Manager:ManagerPassword@cluster0.mc9b6f1.mongodb.net/?retryWrites=true&w=majority
And it should be located in .env which is inside the setup/backend folder

We are using a Three - tier architecture ->
https://en.wikipedia.org/wiki/Multitier_architecture#/media/File:Overview_of_a_three-tier_application_vectorVersion.svg

A picture can be found on next page

Presentation Tier

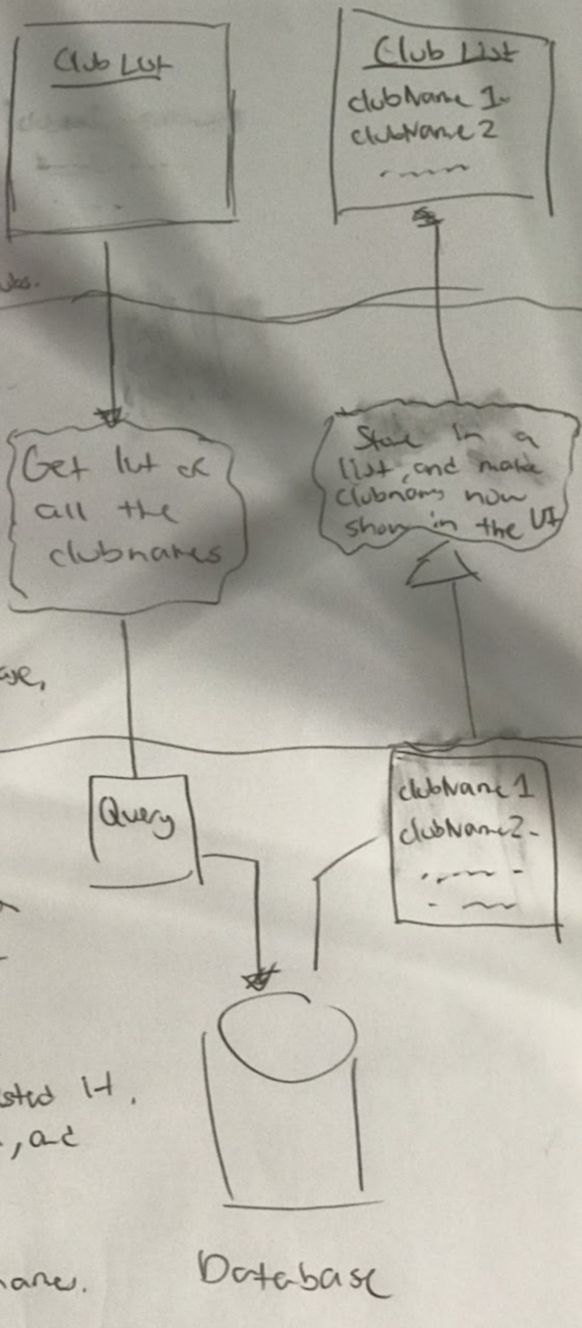
Our Website made using React.
This is where the our user interface is.
Like a page for users to follow clubs.

Logic Tier

This moves and is the inner working mechanism between our database and user interface, Process data.
Like getting a list of all club name in our database, and sends it to frontend.

Data Tier

Where all the information is stored and is where you are able to fetch information and send it back to whoever requested it.
Logic tier processes the data, and sent it back to user.
Like giving the logic tier a list of all club names.



Club List
Club List
clubName 1
clubName 2
Get list of all the clubnames
Store in a list, and make clubnames now show in the UI
Query
clubName 1
clubName 2
Database