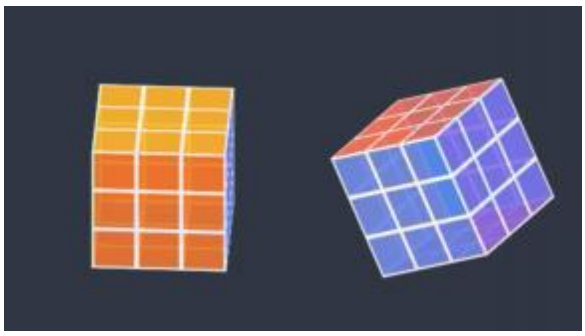
K. Kisi-Kisi

SPEEDTEST

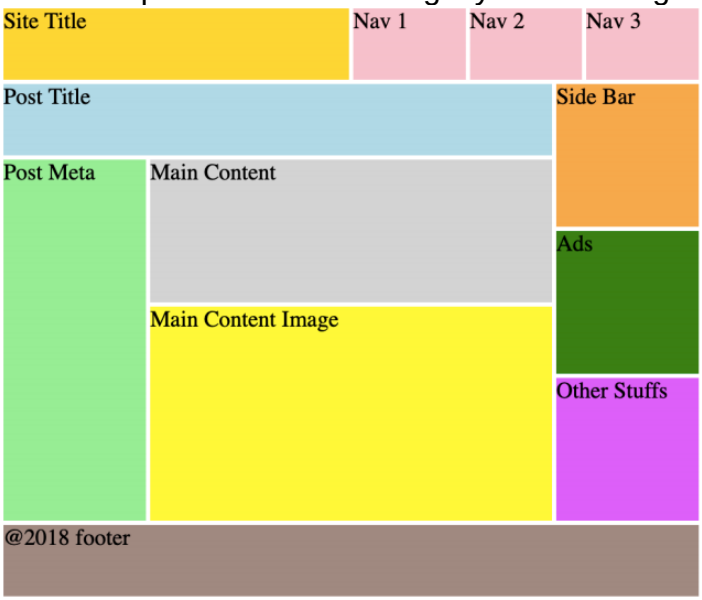
Task 1: Cube Rotation



We want to create the following cube rotation effect.
1. You should create two cubes using the index.html and the cubes should be rotated same as video.mp4.
2. Left cube should be rotated left to right.
3. Right cube should be rotated up to down.
4. color codes
    - front - #fa5252
    - back - #f76707
    - right - #12b886
    - left - #4c6ef5
    - top - #fab005
    - bottom - #7950f2
5. You can only use style.css for the Cube Animation, JavaScript is not allowed.

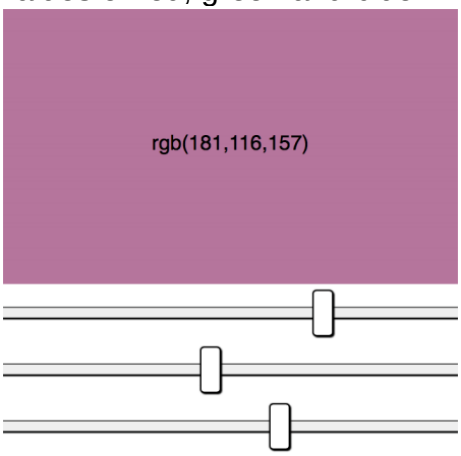
Task 2: CSS Grid

Please implement the following layout in CSS grid.



Task 3: RGB Slider

Please implement an RGB slider as following. There are 3 sliders for adjusting values of red, green and blue.



Task 4: PHP Array

Given the code with some PHP arrays in media files, please implement a PHP function that compare the given two arrays and return a new array that contains the common elements from both given arrays.

Task 5: Installing Wordpress

Now a client needs a wordpress blog to go up and live very quickly. They don't care about the layout so default theme should work. But they want to have a "Steelblue" top bar and "Light Cyan" background color on their blog. Please install a new wordpress website with the following requirement: • Title: Handy Cafe • Sub-title: A cafe that you never forget. • Top color bar on all page: 5 px steel blue • Background color on all page: light cyan • Categories: products, closing time, events • There are menus on home page that links to posts listing of the mentioned categories.

MODULE CLIENT SIDE

CONTENTS

This module has the following files:

1. MODULE_CLIENT_SIDE.doc
2. MODULE_CLIENT_SIDE_MEDIA.zip

INTRODUCTION

In recent years internet has become our basic daily source and needs, enabling the dissemination of information in an inexhaustible content and interaction. Games uses has gained a promined role in nowadays computer usage, allowing people to get access to entertainment from any places.

You are asked to develop game called **Flipping** using HTML and CSS and develop client-side programming using JavaScript and its open source libraries. Some media files are available to you in a zip file. You can create more media and modify anything in the media if you want.

Your game needs to be developed in a tablet resolution (960x600 pixels). In bigger resolution, the game must be centred in the screen both horizontally and vertically.

DESCRIPTION OF PROJECT AND TASKS

This is a module of 5 hours. Your first 2.5 hours must be used to create the design of the game in three PNG images and the initial layout using HTML/CSS. Your layout should follow the design that you created. The final 2.5 hours you will create the functionality of game using JavaScript that allows the game to work correctly in different web browsers.

Flipping game board are described below:

1. Board consist of **8x8 board** and 64 discs, 32 black and 32 white.
2. **Place 2 black and 2 white discs at the centre of the board**. Arrange the discs with matching colour diagonal to each other.
3. Total discs form each colour.

FIRST 2.5 HOURS – DESIGN AND INITIAL LAYOUT:

1. **Deliver at least 3 PNG image files that present:**
   1.1. Game Instructions: The first screen of the game presents the instructions to the player, a text field for player's name, and the "Colour" buttons for choosing color. The instructions for the game are included in the media files. The "Colour" button should be disabled if the text field is empty.
   1.2. Game board layout: It must present all elements described above in the game screen.
   1.3. Game Icon which represent the game immediately.

2. **Develop the initial markup (HTML + CSS) of your game application.** When the layout is accessed the game is presented to the user with the game instructions and the button "Start Game". The instructions must be presented in an animated way.

3. **"Colour" buttons must have active and hover effects.** The background of the buttons in hover state must be: #e2e9fc for white and #272729 for black. The active state must use #0069d9.

4. **The HTML and CSS** code must be valid in the W3C standards for HTML5 and CSS3 rules.

FINAL 2.5 HOURS – GAME FUNCTIONALITIES:

1. **Black always goes first in Flipping**. Player can choose between white or black; opposite colour will be run by computer.
2. **Pressing the "Colour" button in the initial screen will begin** the countdown from 3 before the game starts. Player cannot do anything in this state. After the countdown reaches 0, player can start playing the game with the 8x8 board.
3. **Game will shows every possible legal move suggestion** so it would be easier for players to make decisions. The suggested board should be highlighted.
4. **Player can click any suggested legal moves** to choose it. The chosen spot should be animated to show the board are chosen. If Player click on board without suggestion moves, animate and highlight all possible legal move.
5. Disc with colour based on player's colour will **appear** on spot player had chosen in advance.
6. **Place disc in a spot that surrounds an opponent's disc**. This is also known as "outflanking". A "row" consists of one or more discs that form a line horizontally, vertically, or diagonally.
    6.1. For example, if the opponent has a disc next to 1 of your discs in a vertical row, then place a disc on the open side of their disc in the same row to outflank your opponent's disc
7. **Flip the outflanked disc to its opposite side.** Once a disc is outflanked, flip it over to the opposite colour. However, the same disc may be turned over again if it is part of a row that is outflanked.
8. **Create score counters** which will count every disc on each colour in order to determine who will win at the end of the game.
9. **Pass the turn to your opponent to continue playing.** Each Player goal is to place a disc in a spot that outflanks at least 1 of the other player's discs.
10. **Continue taking turns placing discs until a legal move isn't possible.** Always place discs in a position where they can outflank a row of the opponent's discs. If this isn't possible, you must forfeit your turn until you can perform a legal move. If neither player can perform a legal move, then the game is over.
11. **Save the current score as high score in the local storage** when the game is over (whether the black or white won) with higher score than the current score.
12. **Use your talent to increase the usability of the game** as much as possible to permit a better experience for the user.
13. **There must be an option to disable/enable game sounds.** If the sound is disabled, none of sounds should be played. If sound is enabled all sounds must be played.
14. **Your game should work without JavaScript errors** or messages shown in the browser console.
15. **Maintain your HTML/CSS and JavaScript code organized and clean to facilitate future maintenance.** Use correct indentation and comments. Use meaningful variable names and document your code as much as possible so another developer would be able to modify your work in the future
16. The game needs to work correctly in **Google Chrome**

INSTRUCTIONS TO THE COMPETITOR

1. The media files are available in the ZIP file. You can modify the supplied files and create new media files to ensure the correct functionality and improve the application.
2. Save your design files in a folder call "**XX_CLIENT_SIDE_MODULE/design**" where XX is your computer number.
3. You should create additional images for each of the requested resolution to highlight hidden elements, animations, interactions, or any additional information that will assist in the presentation of the game design.
4. Additional file names
    a. Instructions: XX_instructions_2.png, XX_instructions_3.png

b. Game board: XX_game_board_2.png, XX_game_board_3.png

5. Save any image source files to a folder named "**source**" inside the "**XX_CLIENT_SIDE_MODULE/design**" folder. The source files are the files that contain the layers, development files, ie .psd, .ai, .svg, .jpg.
6. Save the working game to the directory on the server named "**XX_CLIENT_SIDE_MODULE**". Be sure that your main file is called index.html.
7. You are responsible for the time management in your development. If you finalize some tasks you can continue to other tasks. The initial 2.5 hours only define what will be evaluated first.

EXAMPLE

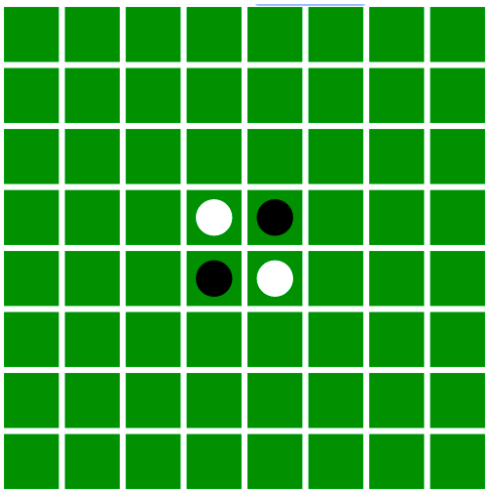


Image 1 Shows initial position Flipping Game

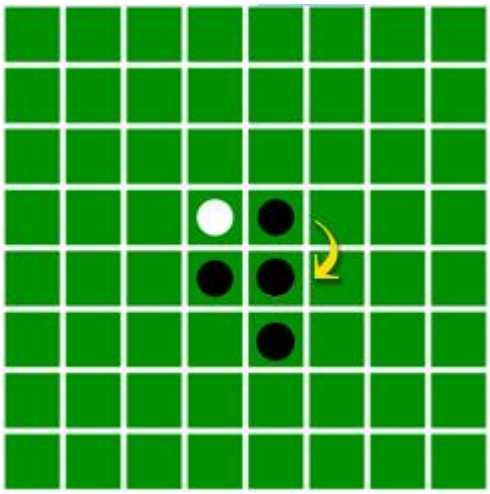


Image 3 Black Outflank White

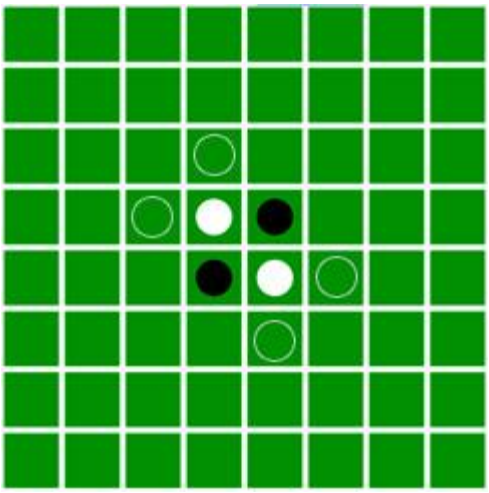


Image 2 Legal Move Suggestion

MODULE CMS

INTRODUCTION

EventX is a brand new music event that is to be held in different locations with different artists. The company running this event needs help in designing a new and vibrant website in order to market the event and sell tickets.

DESCRIPTION OF PROJECT AND TASKS

This module involves knowledge about website design, website layout techniques, client side scripting, and server side scripting, all combined in one CMS project. The CMS used for this module is the most common global CMS: WordPress.

DESIGN

Competitors will design a responsive landing page for the music event 'EventX'. The website will be developed as a WordPress child-theme for the Blankslate theme. The child-theme will be supplemented with a number of plugins. The company has prepared content and has requested you to implement the provided layout. Your tasks are to design the logo, style guide and website. The logo and website design needs to be developed for the target audience for EventX: Young people between 18 and 25 years old.

Create a design for a responsive front page for the website so it will be accessible and look beautiful in other devices:

Desktop: 1440px,

Tablet: 768px,

Mobile: 320px.

Add the designs to a mockup to present your website.

You should save the mockup file as: Mockup_EventX.*. The mockup should be saved as *.png or *.psd or *.ai.

The company wants their main page to have at least these elements:

- Image Banner
- About the company (placeholder text)
- Contact Info
- Footer with copyrights and three links to social media
- Tap to open the hidden sidebar

Style Guide

1. Color scheme for
    a. Typography
    b. Buttons
    c. Labels
    d. Background
2. Typography
    a. Main heading
    b. Subpage title
    c. Section header
    d. Buttons
    e. Labels
    f. Paragraphs
3. Grids and spacing, viewports
    a. Mobile
    b. Tablet
    c. Desktop

4. Layout
    a. Navigation
    b. Tables
    c. Lists
    d. Dialog
    e. Form elements
    f. Buttons

Because there will be another developer that will later develop the website too. You need to make a style guide for future development.

The designs should be saved as: Desktop_EventX.*, Tablet_EventX.* and Mobile_EventX.*.

The designs should be saved as *.png and .psd or .ai.

TECHNIQUE

The website layout should be developed with the ability to add menu items and content without damaging the design. In the future some widgets will be placed in the sidebar. Make sure this sidebar can easily be activated or deactivated without damaging the design.

The layout of the website should be identical to your designs, but also needs to scale without damaging the design when scaling the browser window between 320px and 1440px. You should implement the HTML and CSS W3C standards for proper SEO support.

Clicking at post items may not cause a page refresh, but post content will be loaded asynchronous from the server.

The company wants you to use the JQuery image slider they paid for a year ago. You need to rebuild the slider to a proper WordPress plugin.

CONFIGURATION

For safety reasons two user profiles needs to be created:

1. Admin - access to the complete WordPress dashboard.
    a. Username: adMinX
    b. Password: Never4get
2. Contributor - access to the WordPress dashboard except the Appearance, Plugins, Users and Tools.
    a. Username: uSure
    b. Password: Never4get2
3. Editor - Able to accept or decline news created by contributor
    a. Username: eDitor
    b. Password: Never4get3

The Wordpress login page should have the logo of EventX.

The provided plugins should be used to make the website SEO friendly, handle website security and to analyze the visitors of the website.

WEBSITE STRUCTURE

The website consists of several sections described below:

FEATURED SECTION

The Featured section contains a headline, sub headline and an interactive image slider with images of the booked artists. When hovering an image, an mp3 track of the corresponding artist is played. The track is paused when the mouse leaves the image. For the image slider you need to use the delivered JQuery slider. The header section should have an energetic appearance, appropriate to the event. The client wants to have the ability to add, remove or change the images and mp3 files of the slider using the WordPress dashboard.

NEWS SECTION

The News section contains posts about the event. The News section should have a readable and clean design. The client does not have news items; you can use placeholder text to show at least 3 news items. Each news item contains title, date, shortened content and a 'read more' button. By clicking the button the full content will be loaded from the server, without a page refresh (asynchronous).

SAVE THE DATE SECTION

EventX starts at Thursday, 19 October 2019 at 18.00. The Save The Date section contains the date and place of the event and a countdown timer with days, hours, minutes and seconds until the start of the event. The Save The Date section should have an exciting design to engage the target audience. The client wants to have the ability to add, remove or change the date and place of the event using the WordPress dashboard.

BUY YOUR TICKETS SECTION

The Buy Your Tickets section leads to the external website of www.ticketmaster.com This section should be designed attractive and with a clear call to action. The client wants to have the ability to add, remove or change the Internet address linked at the button using the WordPress dashboard.

SIDEBAR SECTION

The sidebar section will be used to show the sponsors of the event and there should be space to add some widgets.

The sidebar should be added to the left side of all pages. The sidebar is hidden but has a small tab to open the sidebar. When user clicks the sidebar, give additional slide or fade in animation to make the entrance of the sidebar appealing.

The client wants to have the ability to add, remove or change logo images and company names for sponsors and website widgets using the WordPress dashboard.

FOOTER SECTION

The footer section contains social media icons and a possibility to leave an e-mail address for subscribing at the newsletter. The client wants to have the ability to add, remove or change social media icons and their corresponding links using the WordPress dashboard.

INSTRUCTIONS TO THE COMPETITOR

1. Save your files in your working directory on the server called "**XX_Module_CMS**".
2. Save your design, style guide and mockup files in this location: **module_cms/design**
3. Save your database dump into a file named "**db-dump.sql**" placed inside the module_cms directory
4. Assessment will be done on the files and data in the database named "**module_cms_db"** on your computer.

MODULE SERVER

CONTENTS

This module has the following files:

- MODULE_SERVER.doc
- MODULE_SERVER_MEDIA.zip

INTRODUCTION

Trilu, task management website, ask you to make their minimum viable product. Your task is to implement the backend with Laravel PHP Framework and frontend with JavaScript Framework (VueJS, AngularJS, Angular, or ReactJS). The front-end design skeleton is provided. The detail description and tools that you can use will be described below.

DESCRIPTION OF PROJECT AND TASKS

API List:

Use provided ERD to make your database. Create dummy users on users table (password is hashed using bcrypt):

username	password	First name	Last name
john.doe	12345	John	Doe
richard.roe	12345	Richard	Roe
jane.poe	12345	Jane	Poe

These are the list of web service endpoint that requested by **Trilu**:

1. **Authentication**
    a. **Register**

    URL: [domain]/v1/auth/register

    Description: For client to register new user

    Method: POST

    Request Parameter:

    - Body:
        - First Name
        - Last Name
        - Username
        - Password

    Validation:

    - First Name must be alphabet only, length between 2 – 20
    - Last name must be alphabet only, length between 2 – 20
    - Username only consist of alphanumeric, underscore ‘_’, or dot ‘.’, length between 5 – 12
    - Username must be unique
    - Password length between 5 - 12

    Response:

    - If register success, registered user automatically logged in:

      Header: response status: 200

Body:

- Token (authorization token generated by the system from logged in user id with bcrypt hashing method)
- Role

- If input validation failed:

  Header: response status: 422

  Body: message: invalid field

b. Login

URL: [domain]/v1/auth/login

Description: For client to generate and get login token using username and password. Username and password must be valid.

Method: POST

Request Parameter:

- Body:
  - Username
  - Password

Response:

- If login success:

  Header: response status: 200

  Body:
  - Token (authorization token generated by the system from logged in user id with bcrypt hashing method)
  - Role
- If login failed (username or password do not match or empty):

  Header: response status: 401

  Body: message: invalid login

c. Logout

URL: [domain]/v1/auth/logout?token={*AUTHORIZATION_TOKEN*}

Description: For server to make token invalid

Method: GET

Response:

- If logout success:

  Header: response status: 200

  Body:
  - message: logout success
- If logout failed (token invalid):

  Header: response status: 401

  Body: message: unauthorized user

2. Board

a. Create new board

URL: [domain]/v1/board?token={*AUTHORIZATION_TOKEN*}

Description: For client to create new board. Assign creator as team member when board created.

Method: POST

Request Parameter:

- Body:
    - Name

Validation:

- name must be filled

Response:

- If success:
  Header: response status: 200
  Body: message: create board success
- If input validation failed:
  Header: response status: 422
  Body: message: invalid field
- If unauthorized user access it (only logged in user can access this endpoint):
  Header: response status: 401
  Body: message: unauthorized user

b. Update board

URL: [domain]/v1/board/{*board_id*}?token={*AUTHORIZATION_TOKEN*}

Description: For client to update existing board by id

Method: PUT

Request Parameter:

- Body:
    - Name

Validation:

- name must be filled

Response:

- If success:
  Header: response status: 200
  Body: message: update board success
- If input validation failed:
  Header: response status: 422
  Body: message: invalid field
- If unauthorized user access it (only team member can access this endpoint):
  Header: response status: 401
  Body: message: unauthorized user

c. Delete board

URL: [domain]/v1/board/{*board_id*}?token={*AUTHORIZATION_TOKEN*}

Description: For client to delete existing board by id

Method: DELETE

Response:

- If success:

  Header: response status: 200

  Body: message: delete board success
- If unauthorized user access it (only creator can access this endpoint):

  Header: response status: 401

  Body: message: unauthorized user

d. Get all boards

URL: [domain]/v1/board?token={*AUTHORIZATION_TOKEN*}

Description: For client to get all boards data based on logged in user, as a member or creator.

Method: GET

Response:

- If success:

  Header: response status: 200

  Body: all board data in json (consists of id, name, creator_id)
- If unauthorized user access it (only logged in user can access this endpoint):

  Header: response status: 401

e. Open board

URL: [domain]/v1/board/{*board_id*}?token={*AUTHORIZATION_TOKEN*}

Description: For client to get board detail based on board_id. Lists and cards sorted by column order.

Method: GET

Response:

- If success:

  Header: response status: 200

  Body: eager load all board data in json (consists of all team members, all lists of board and cards inside list)

Format:

```
{
  “id”: [board_id],
  “name”: [board_name],
   creator_id: [creator_id],
   “members”: [
      {
         “id”: [user_id],
         “first_name”: [first name],
         “last_name”: [last name],
         “initial”: [generated from first name and last name]
      },
      …
   ],
   “lists”: [
      {
         “id”: [list_id],
         “name”: [list_name],
         “order”: [order],
         “cards”: [
            {
               “card_id”: [card_id],
               “task”: [card_task],
               “order”, [order]
            },
            …
         ]
      },
      …
   ]
}
```

- If unauthorized user access it (only team member can access this endpoint):
  Header: response status: 401

f. Add team member

URL: [domain]/v1/board/*{board_id}*/member?token={*AUTHORIZATION_TOKEN*}
Description: For client to add team member to the board.
Method: POST
Request Parameter:

- Body:
  - Username

Validation:

- username must be existing in user table

Response:

- If success:
  Header: response status: 200
  Body: message: add member success
- If input validation failed:
  Header: response status: 422
  Body: message: user did not exist
- If unauthorized user access it (only team member can access this endpoint):
  Header: response status: 401
  Body: message: unauthorized user

g. Remove team member

URL: [domain]/v1/board/*{board_id}*/member/*{user_id}*?token={*AUTHORIZATION_TOKEN*}

Description: For client to delete team member from the board using user id.

Method: DELETE

Response:

- If success:
  Header: response status: 200
  Body: me ssage: remove member success
- If unauthorized user access it (only team member can access this endpoint):
  Header: response status: 401
  Body: message: unauthorized user

3. List

a. Create new list

URL: [domain]/v1/board/*{board_id}*/list?token={*AUTHORIZATION_TOKEN*}

Description: For client to create new list to the board

Method: POST

Request Parameter:

- Body:
  - Name

Validation:

- name must be filled

Response:

- If success:
  Header: response status: 200
  Body: message: create list success
- If input validation failed:
  Header: response status: 422

Body: message: invalid field

- If unauthorized user access it (only board team member can access this endpoint):

  Header: response status: 401

  Body: message: unauthorized user

b. Update list

URL: [domain]/v1/board/{*board_id*}/list/*{list_id}*?token={*AUTHORIZATION_TOKEN*}

Description: For client to update existing list by id

Method: PUT

Request Parameter:

- Body:
  - Name

Validation:

- name must be filled

Response:

- If success:

  Header: response status: 200

  Body: message: update list success
- If input validation failed:

  Header: response status: 422

  Body: message: invalid field
- If unauthorized user access it (only board team member can access this endpoint):

  Header: response status: 401

  Body: message: unauthorized user

c. Delete list

URL: [domain]/v1/board/{*board_id*}/list/*{list_id}*?token={*AUTHORIZATION_TOKEN*}

Description: For client to delete existing list by id

Method: DELETE

Response:

- If success:

  Header: response status: 200

  Body: message: delete list success
- If unauthorized user access it (only board team member can access this endpoint):

  Header: response status: 401

  Body: message: unauthorized user

d. Move list to right

URL:
[domain]/v1/board/{*board_id*}/list/*{list_id}*/right?token={*AUTHORIZATION_TOKEN*}

Description: For client to switch the order of the selected list with the list on the right

Method: POST

Response:

- If success:

  Header: response status: 200

  Body: message: move success
- If unauthorized user access it (only board team member can access this endpoint):

  Header: response status: 401

  Body: message: unauthorized user

e. Move list to left

URL:
[domain]/v1/board/{*board_id*}/list/{*list_id}*/left?token={*AUTHORIZATION_TOKEN*}

Description: For client to switch the order of the selected list with the list on the left

Method: POST

Response:

- If success:

  Header: response status: 200

  Body: message: move success
- If unauthorized user access it (only board team member can access this endpoint):

  Header: response status: 401

  Body: message: unauthorized user

4. Card

a. Create new card

URL:
[domain]/v1/board/*{board_id}*/list/*{list_id}*/card?token={*AUTHORIZATION_TOKEN*}

Description: For client to create new card to the list

Method: POST

Request Parameter:

- Body:
  - Task

Validation:

- Task must be filled

Response:

- If success:

  Header: response status: 200

  Body: message: create card success
- If input validation failed:

Header: response status: 422

Body: message: invalid field

- If unauthorized user access it (only board team member can access this endpoint):

  Header: response status: 401

  Body: message: unauthorized user

b. Update card

URL: [domain]/v1/board/{*board_id*}/list/*{list_id}*/card/*{card_id}*?token={*AUTHORIZATION_TOKEN*}

Description: For client to update existing card by id

Method: PUT

Request Parameter:

- Body:
  - Task

Validation:

- Task must be filled

Response:

- If success:

  Header: response status: 200

  Body: message: update card success

- If input validation failed:

  Header: response status: 422

  Body: message: invalid field

- If unauthorized user access it (only board team member can access this endpoint):

  Header: response status: 401

  Body: message: unauthorized user

c. Delete card

URL: [domain]/v1/board/{*board_id*}/list/*{list_id}*/card/*{card_id}*?token={*AUTHORIZATION_TOKEN*}

Description: For client to delete existing card by id

Method: DELETE

Response:

- If success:

  Header: response status: 200

  Body: message: delete card success

- If unauthorized user access it (only board team member can access this endpoint):

  Header: response status: 401

  Body: message: unauthorized user

d. Move up card

URL: [domain]/v1/card/*{card_id}*/up?token={*AUTHORIZATION_TOKEN*}

Description: For client to switch the order of the selected card with the card above

Method: POST

Response:

- If success:

  Header: response status: 200

  Body: message: move success
- If unauthorized user access it (only board team member can access this endpoint):

  Header: response status: 401

  Body: message: unauthorized user

e. Move down card

URL: [domain]/v1/card/*{card_id}*/down?token={*AUTHORIZATION_TOKEN*}

Description: For client to switch the order of the selected card with the card below

Method: POST

Response:

- If success:

  Header: response status: 200

  Body: message: move success
- If unauthorized user access it (only board team member can access this endpoint):

  Header: response status: 401

  Body: message: unauthorized user

f. Move card to another list

URL: [domain]/v1/card/*{card_id}*/move/*{list_id}*?token={*AUTHORIZATION_TOKEN*}

Description: For client to move card to another list. List must be in the same board.

Method: POST

Response:

- If success:

  Header: response status: 200

  Body: message: move success
- If list id is not in the same board:

  Header: response status: 422

  Body: message: move list invalid
- If unauthorized user access it (only board team member can access this endpoint):

  Header: response status: 401

  Body: message: unauthorized user

Make sure that all validations are done in **both backend and frontend**. The complete minimum viable product for **Trilu** system should cover the following requirement:

Menu	Detail
Login & Register	• User can register into the system on the register page • User can login (and logout) into the system on the start page of the application • After login, user directed to “home” menu • Links to the Home and Logout are always visible on the top part of the page while the user is logged in
Home	• On “home” menu, user can add new board or access board if the user is the team member • User can update board’s name by clicking the board name, type new name, then press Enter to submit • Creator of the board can delete a board by deleting its entire name, then press Enter to submit • Make sure your system is preventing users to view and access unauthorized board
Board	• On “board” menu, user can manage board: members, lists, and cards **Member** • User can add new member using username • User can remove member by clicking its initial then click confirmation button **List** • User can add new list to the board • User can update list’s name by clicking the list name, type new name, then press Enter to submit • User can delete a list by deleting its entire name, then press Enter to submit • User can move list left and right **Card** • User can add new card to a list • User can update card’s task by clicking the card, type new task, then press Enter to submit • User can delete a card by deleting its entire task, then press Enter to submit • User can move card up and down in the list • User can move card to another list by click the card, then click another list.

ERD

You can use and improve ERD below: