Architecture

The Chessko web application that employs frontend and backend architecture. The frontend is mainly tasked with handling user interactions and displaying information, while the backend handles data storage, authentication, and business logic. The website was made using HTML, CSS, JavaScript, and PHP.

Algorithms

Signup:

* Validate “email”, “password”, “password confirmation” and “uploaded file” fields by frontend and backend.
* If all validations (checks) are successful, a new user object is created, and it is saved to the "sign\_up.json" file. (Passwords are saved in hashed form)
* Store the user’s email, ID, and image file name in the session.
* Reload the forum page to enable the button for creating a new post and activate the link to the profile page.

Login:

* Validate “email” and “password” fields by frontend and backend.
* If all validations (checks) are successful, save the user’s email, ID, and image file name in the session.
* Reload the forum page to enable the button for creating a new post and activate the link to the profile page.

Create post:

* Validate “title” and “content” fields by frontend.
* If all validations (checks) are successful, save the new post to the posts.json file.
* Fetch and display all posts, including the new post, by using the 'fetch' method to access the 'load\_posts.php' file.

Index page:

* Displays images as a background periodically using intro.js script.

Forum page:

* Fetch and display all posts in a paginated manner by using the 'fetch' method to access the 'load\_posts.php' file.

Profile page:

* Display the user’s image and email.
* Fetch and display all user’s posts by using the 'fetch' method to access the 'load\_user\_posts.php' file.

Edit Account:

* Validate the entered email by frontend.
* If all validations (checks) are successful, save the new email instead of current to the sign\_up.json file.
* Reload profile page to display the updated user’s email.

Edit post:

* Validate the entered post title and content by frontend.
* If all validations (checks) are successful, save the new post title and content instead of current to the posts.json file.
* Fetch and display all user’s posts with edited post by using the 'fetch' method to access the 'load\_user\_posts.php' file.

Main files

* index.html: serves as a welcome page featuring a changing background of chess-related images and includes a button redirecting to the 'forum.php' page.
* forum.php: displays all posts with pagination, includes a button for creating new posts (available for registered users), features a section with site rules, and provides options for navigating to other pages based on user actions.
* data: the directory containing data files for user accounts, posts, and user images.
* scripts: the directory containing JavaScript scripts for user interactions, validation, and post creation management.
* css: the directory containing CSS files for styling the website.
* source: the directory containing website-related resources, such as images.
* php: the directory containing PHP for various functionalities, including signup, login, validation, post management, authentication, and user data management.
* intro.js: creates an automatic image slider that switches between pictures every 5 seconds on an HTML page.
* signup\_login.js: includes scripts that handle the processes of signing up, logging in, and logging out.
* create\_post.js: controls the process of creating and loading forum posts, handling user interactions such as creating new posts and updating the forum content.
* profile.js: manages the display and updating of user posts, along with the functionality to change the user's email.
* change\_email.php: updates the user's email in the data file and the session, redirecting to the profile page afterward.
* CheckUser.php: class that is used to return an object value for the login PHP script.
* create\_post.php: handles the process of generating a new post.
* find\_email.php: returns a boolean value based on whether the passed email was found in the sign\_up.json file.
* find\_email\_res.php: contains the function that checks whether the passed email already exists in the sign\_up.json file.
* load\_posts.php: retrieves and paginates posts from two JSON files, matching user IDs using a function to find a user by ID in the sign\_up.json user array. Finally, it displays all posts with pagination.
* load\_user\_posts.php: retrieves and displays posts for a specific user, filtering them based on their email. It also provides an option to edit each post.
* login.php: handles the login process with validation.
* sign\_up.php:handles the signup process with validation.
* logout.php: handles the logout process.
* Post.php: class for the post object that is displayed on the forum and profile pages.
* update\_post.php: updates an existing user’s post based on received data.
* User.php: class for the user object.
* verify\_pw\_res.php: verifies user login credentials by comparing the provided email and password with data from a JSON file, returning a boolean result.