# **טכנולוגיות אינטרנט מתקדמות - 61776 (WEB)**

**הגשת פרויקט**

**<crypto exchange simulator> < B6>< 23>**

|  |  |
| --- | --- |
| **שם חבר.ת הצוות** | **תז** |
| רועי שמאייב | 207776568 |
| דניאל פריימוביץ | 315880963 |
| שמרון יפרח | 312423247 |
| אלכס בבושין | 310926415 |

A Cryptocurrency Exchange Simulator web application built with React, JavaScript, CSS, and HTML. It allows users to simulate trading cryptocurrencies in real-time, view market data, and track their portfolio balance. The application features real-time price updates, user authentication, leaderboard, and learning resources.

**Database (DB) Details:** The project uses a MongoDB database to store user data, including usernames, passwords, balances, and coin holdings. The database schema includes collections for users. interacts with the database using queries and transactions to handle user authentication, data retrieval, and updates.

**Special Code Features:**

* **React Router**: Used for client-side routing and navigation between different pages of the application.
* **Tailwind CSS**: A utility-first CSS framework used for styling the application.
* **Fetch API**: Used to make HTTP requests from the client-side to the server and external APIs.

**Local Storage**: Used to store user-specific data like authentication tokens

**APIs Used:**

**CoinGecko API**: Used to fetch real-time cryptocurrency prices and market data. The API endpoint is https://api.coingecko.com/api/v3/coins/markets.

and balance on the client-side.

<[Github link](https://github.com/rsCode1/B6) >

[Site link](https://b6-client.vercel.app/)

[MTW Link](https://www.morethanwallet.com/app/710)

# **טכנולוגיות אינטרנט מתקדמות - 61776 (WEB)**

**הגשת פרויקט**

1. עליכם להמשיך את בניית האתר לפי האלמנטים המתקדמים שלמדתם
2. יש למנות מהנדס מערכת בכל צוות, אשר יהיה אחראי על הגדרת והקצאת המשימות בתרגיל זה.  
   נא לרשום את שם הסטודנט בתרגיל זה. על מהנדס המערכת לכתוב כיצד נעשתה חלוקת העבודה מול הצוות, מה היו המשימות של כל חבר צוות, האם היה ממשק בין חברי הצוות, והאם המשימות מולאו:

|  |  |  |
| --- | --- | --- |
| **שם חבר הצוות** | **משימות שהוקצו** | **משימות שהושלמו** |
| דניאל | חיבור משתמשים | חיבור משתמשים |
| אלכס | ארנק משתמש פעיל | ארנק משתמש פעיל |
| שימרון | פריסה לריאקט | פריסה לריאקט |
| רועי | פריסה לVERCEL | פריסה לVERCEL |

1. הציגו רשימת דרישות פונקציונליות ולא פונקציונליות (בנפרד, יש לסווג דרישות לא פונקציונליות לפי wikipedia NFR).

**Functional requirements**

1.User Registration​

1.1 Users will be able to create a new account​.

1.2 Users will able to log into their accounts​

​

2. Login Validation​

2.1 Implement validation checks to ensure that user is in the database​

​

3. Trading Features​

3.1 the platform will use real-time market data​

3.2 Users should be able to buy orders for different coins.​

3.3 Users should be able to sell orders for different coins.​

3.4 Users should be able to view their holdings.​

3.5 Users should be able to set stop-loss and take-profit orders to manage risk.​

​

4. Trading Competitions​

4.1 Users can join trading competitions​.

4.2 the platform will show leaderboard​.

​

5. Learning Resources​

5.1 Educational materials will be available​.

**Nonfunctional requirements**

**1. Usability​**

1.1 Intuitive and user-friendly interface​

1.2 Responsive - max of 0.5 sec of delay on max users' workload​

1.3 Fast and efficient order execution to provide a smooth trading experience.​

1.4 The system will work seamlessly across different devices and screen sizes.​

**2. Security​**

2.1 The user will register with username password​.

2.2 the user will log in with username and password​.

2.3 logged user gets a security token for 10 mins​.

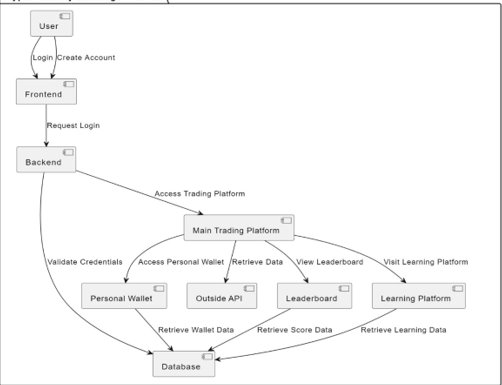
​

**3. Scalability​**

3.1 Ability to handle a large number of users.​

3.2 Ability to handle many Cryptocurrency's coins type.​

1. הציגו ארכיטקטורה מעודכנת של האתר (תרשים הכולל את האלמנטים המרכזיים).



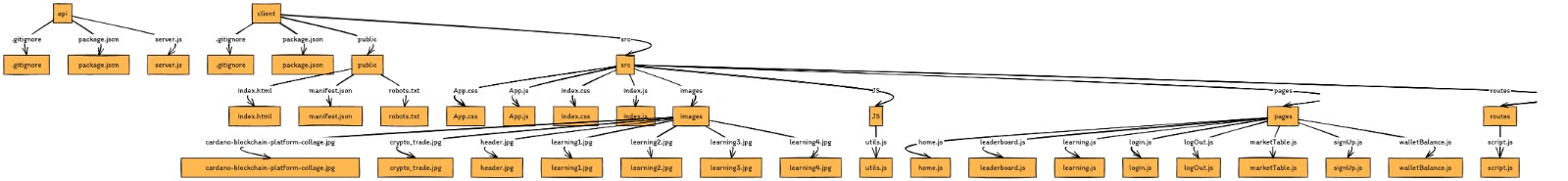
A diagram of a diagram

Description automatically generated

1. הציגו דיאגרמת use case המתארת את השימוש באתר.

A diagram of a cryptocurrency exchange

Description automatically generated

5.יש להציג מבנה סופי של האתר שלכם:   
א. האתר ימומש ב -react/preact, וכן שימוש ב Tailwind - נא להציג דיאגרמה המתארת את התיקיות והקבצים השונים.

ב. יש לפרט את פריטי המידע - יש להשתמש במידע אמיתי ורלוונטי לפרויקט שלכם (בשליפה ממסד נתונים חיצוני או מ - API). יש להראות דיאגרמת מבנה DB.

A screenshot of a computer

Description automatically generated

5.יש להגיש תיק למתכנת:

א. יש לתעד ולהסביר את כל הפונקציות המרכזיות בקוד שלכם.

ב. יש לפרט את כל ה -API שהשתמשתם בהם, קישור ל- DB,סביבות מיוחדות שהתקנתם, קטעי קוד מיוחדים שלקחתם ממקורות שונים (כולל רפרנסים) , פרומפטים שהשתמשתם בהם בכלי AI.

**תיק למתכנת - CryptocurrencyExchange Simulator**

This project is a Cryptocurrency Exchange Simulator web application built with JavaScript, CSS, and HTML. It allows users to simulate trading cryptocurrencies in real-time, view market data, and track their portfolio balance. The application features real-time price updates, user authentication, leaderboard, and learning resources.

File Structure: The project is divided into two main directories: **client and server**

**client directory:**

* public: Contains the main HTML file (index.html) and other static assets like icons and manifest.
* src: Contains the React components, pages, routes, and other JavaScript files.  
  + components: Directory for reusable React components.
  + pages: Directory containing the main pages of the application (e.g., Home, Login, SignUp, Leaderboard, etc.).
  + routes: Directory with JavaScript files handling API requests and user authentication.
  + utils: Directory containing utility functions (sorting users by balance).
  + App.js: The main React component that renders the navbar, footer, and routes.
  + index.js: The entry point for the React application.

The main entry point for the client-side (front-end) is **index.js**, which renders the App component.

**Key Functions and Code Snippets:**

1. **User Authentication**:
   * client/src/routes/script.js:
     + authenticateUser: Handles user authentication by sending a POST request to the server with the provided username and password.
     + userNameAvailable: Checks if a username is available by making a POST request to the server.
     + register: Registers a new user by sending a POST request to the server with the provided username and password.
2. **Cryptocurrency Data Fetching**:
   * client/src/pages/home.js:
     + CreateCryptoDataTable: Fetches cryptocurrency data from the CoinGecko API and displays it in a table.
3. **Trading Functionality**:
   * client/src/pages/marketTable.js:
     + CreateCryptoDataTable: Allows users to buy and sell cryptocurrencies by rendering input fields and buttons for each coin.
     + handleBuy: Handles the buy action by sending a POST request to the server with the purchase details.
     + handleSell: Handles the sell action by sending a POST request to the server with the sell details.
4. **Wallet Balance**:
   * client/src/pages/walletBalance.js:
     + CreateUserWalletTable: Fetches the user's wallet data from the server and displays their coin holdings, prices, and values.
     + DepositFundsButton: Allows users to deposit funds by sending a POST request to the server with the deposit amount.
5. **Leaderboard**:
   * client/src/pages/leaderboard.js:
     + Fetches all user data from the server and sorts users by their balance to display a leaderboard.

**APIs Used:**

* **CoinGecko API**: Used to fetch real-time cryptocurrency prices and market data. The API endpoint is https://api.coingecko.com/api/v3/coins/markets.

**Database (DB) Details:** The project uses a MongoDB database to store user data, including usernames, passwords, balances, and coin holdings. The database schema includes collections for users. interacts with the database using queries and transactions to handle user authentication, data retrieval, and updates.

**Special Code Features:**

* **React Router**: Used for client-side routing and navigation between different pages of the application.
* **Tailwind CSS**: A utility-first CSS framework used for styling the application.
* **Fetch API**: Used to make HTTP requests from the client-side to the server and external APIs.
* **Local Storage**: Used to store user-specific data like authentication tokens and balance on the client-side.

**Getting Started Guide:**

1. Clone the repository to your local machine.
2. Install the required dependencies by running npm install in both the client and server directories.
3. Configure the server environment variables (e.g., database connection string, API keys) if needed.
4. Start the server by running the appropriate command (e.g., npm start) in the server directory.
5. Start the client-side development server by running npm start in the client directory.
6. The application should now be accessible at http://localhost:3000 (or the specified client-side port).

**Back-end (API)**

* api/server.js: The main entry point for the back-end API server.

1. **Back-end**
   * app.post('/deposit', ...): This route handler in server.js processes user deposits by updating the user's balance in the database.
   * app.post('/walletBalance', ...): This route handler retrieves the user's wallet balance and cryptocurrency holdings from the database and sends them back to the client.
   * app.post('/login', ...): This route handler authenticates the user by verifying their username and password, and generates a JSON Web Token (JWT) upon successful login.
   * app.post('/register', ...): This route handler registers a new user by inserting their details into the database.
   * app.post('/purchase', ...): This route handler processes a user's cryptocurrency purchase by updating their balance and coin holdings in the database.
   * app.post('/updateSell', ...): This route handler processes a user's cryptocurrency sale by updating their balance and coin holdings in the database.
   * generateToken(user): This function generates a JSON Web Token (JWT) for the authenticated user, containing their user ID, username, and password.

**APIs Used** The project integrates with external APIs to fetch real-time cryptocurrency prices. These APIs may require authentication keys or follow specific rate-limiting policies. The API integration code is located in the api.js file on the front-end.

**Database (DB) Details** The project uses MongoDB as the database to store user information, including usernames, passwords, balances, and cryptocurrency holdings. The database schema consists of a single collection called users with the following fields:

* userName (string): The user's unique username.
* password (string): The user's password.
* balance (number): The user's account balance.
* coins (array): An array of objects representing the user's cryptocurrency holdings, with each object containing the coinName and amount.

Example query to fetch a user's data:

**const user = await collection.findOne({ userName: username });**

**Special Code Features** The project utilizes the following libraries and frameworks:

* **Express.js**: A Node.js web application framework used for building the API server.
* **MongoDB Node.js Driver**: The official MongoDB driver for Node.js, used for interacting with the MongoDB database.
* **JSON Web Tokens (JWT)**: A standard for securely transmitting information between parties as a JSON object. Used for user authentication and authorization.
* **Express-Session**: A middleware for handling user sessions in Express.js applications.
* **Connect-Mongo**: A MongoDB session store for Express.js applications.

**Getting Started Guide**

1. **Prerequisites**
   * Node.js and npm (Node Package Manager) installed on your machine.
   * MongoDB installed and running locally, or a MongoDB Atlas cluster set up.
2. **Installation**
   * Clone the project repository from the provided source.
   * Navigate to the project directory: cd project-directory
   * Install the required dependencies: npm install
3. **Configuration**
   * Set up the MongoDB connection string in api/server.js by replacing the uri variable with your MongoDB connection details.
   * Replace the SECRET\_KEY variable in api/server.js with a secure secret key for JWT token generation.
4. **Running the Project**
   * Start the API server: node api/server.js
   * Open the front-end application in a web browser by navigating to http://localhost:5500 (or the specified port in server.js).

6. יש להגיש תיק משתמש הכולל צילומי מסך והסברים כיצד להשתמש באתר שלכם.

7. בתאריכים 25.3 ו -31.3 תציגו את תוצרי הפרויקט שלכם. כל צוות יציג 10 דקות את המערכת, וכן אלמנטים מרכזיים של העבודה (סעיפים 1-4 של מסמך זה).בסיום ההצגה נשאל את חברי הצוות שאלות בנוגע להצגה ולנושאי הקורס.

**תיק למשתמש - CryptocurrencyExchange Simulator**



**Introduction:**Welcome to our Demo Crypto Trade Platform! This platform is designed to provide a risk-free environment for those who want to learn and experience cryptocurrency trading.

**Main Pages description:**

**Landing page**

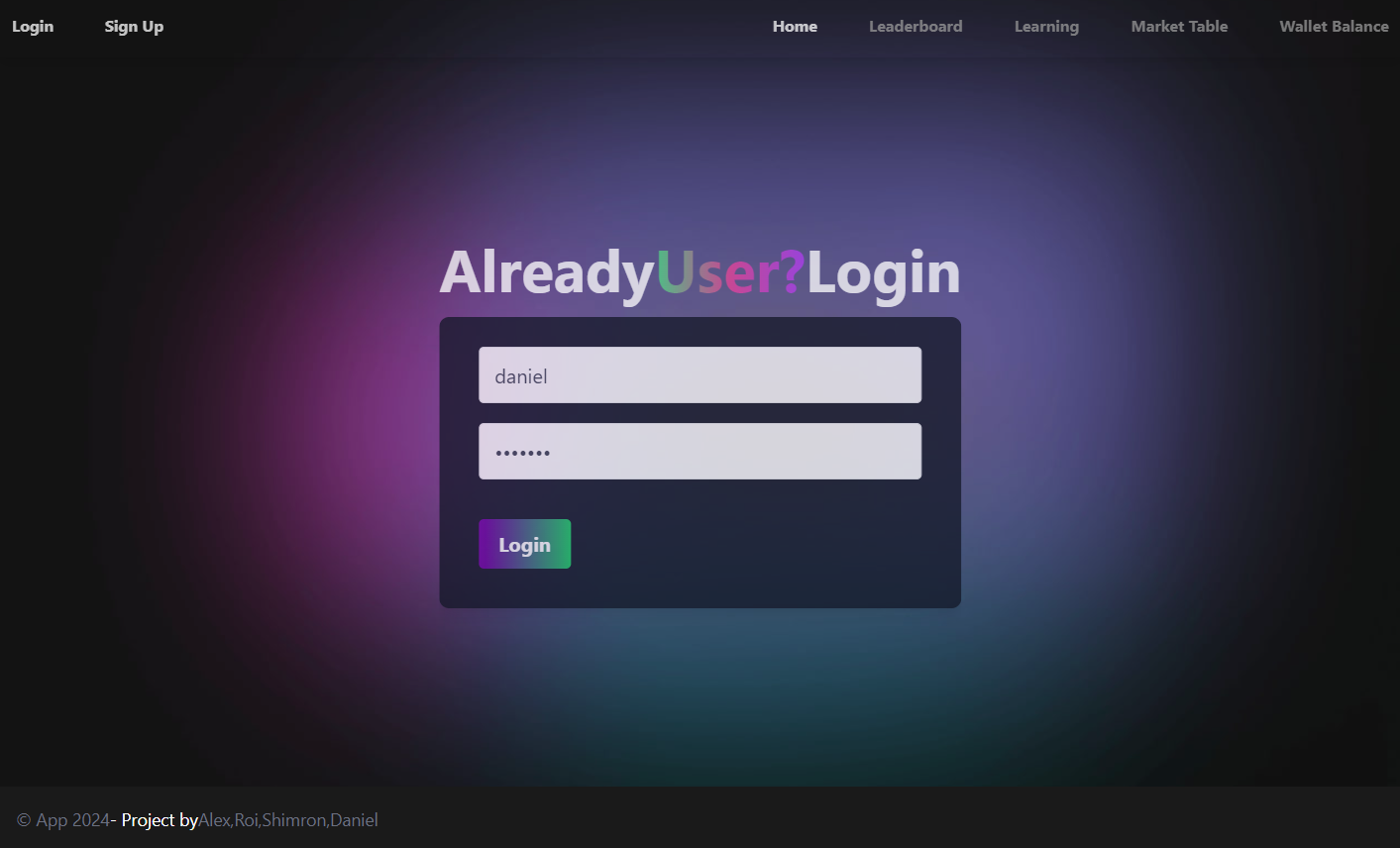
Our primary objective on the landing page is to present crucial data about specific cryptocurrencies, including their current value and fluctuations over the past 24 hours. Additionally, we provide users with the ability to log in or register for our simulator. This feature enables them to execute transactions, offering a hands-on experience of operating on a real trading platform.



**Page description:**

* displays a page dedicated to show an engaging information about the coins value.
  + **Navbar** actions and other options are locked and showed with grey color, unlocks by signing in.
  + **Table** shows information about specific Coins and their value and changes in the last 24 hours
* **Login Navbar** allows the user to login into the platform and make actions.
* **Sign up Navbar** allows the user to create a new account.

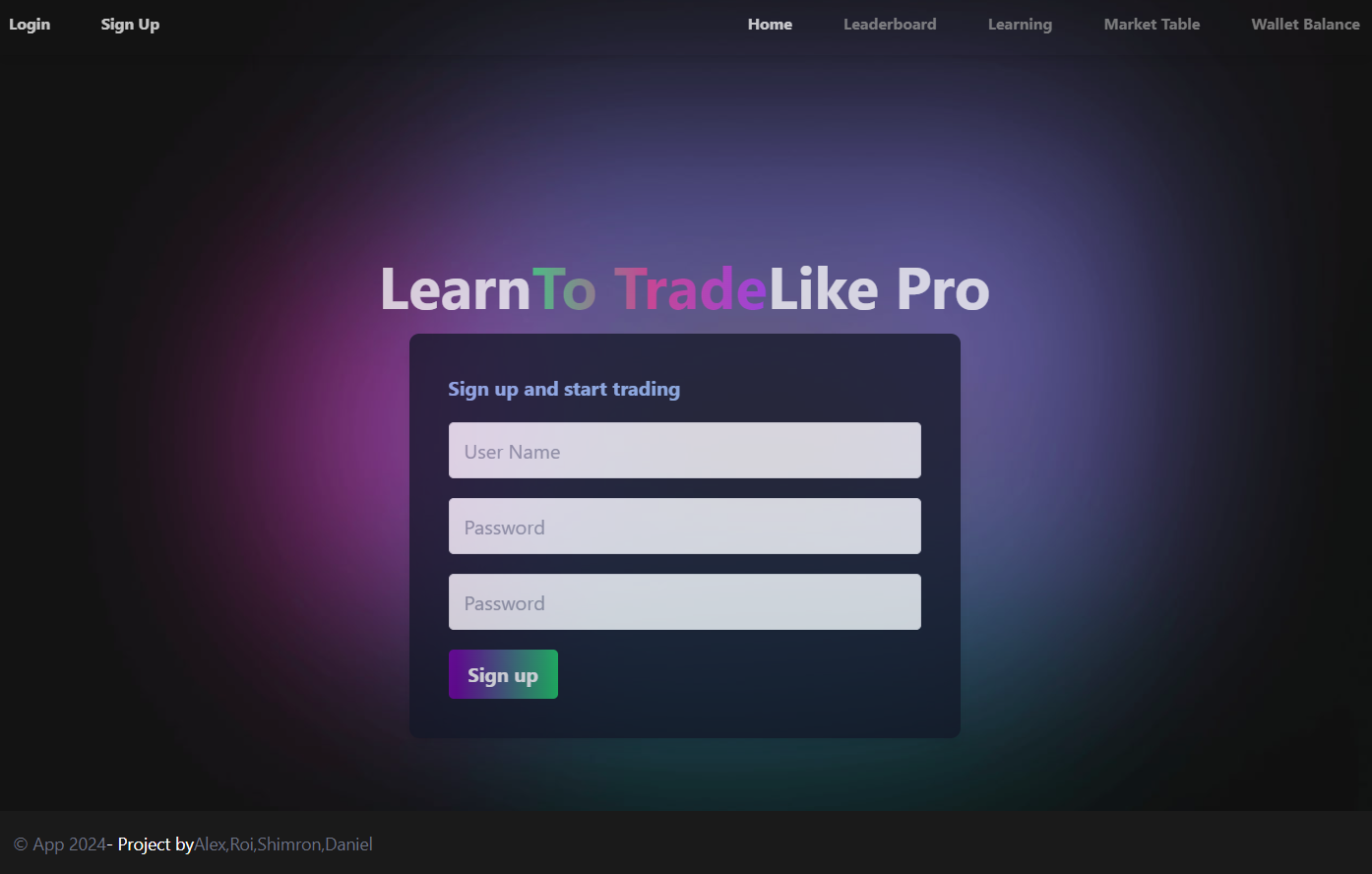
**Login screen**



**Page description:**

* Displays a login screen with option to write existing user name and password
* **Login Navbar** allows the user to login into the platform and make actions.
* **Sign up Navbar** allows the user to create a new account.
* **Login bottom** after entering user data allows to send the data and go through authentication.

**Sign up screen**



**Page description:**

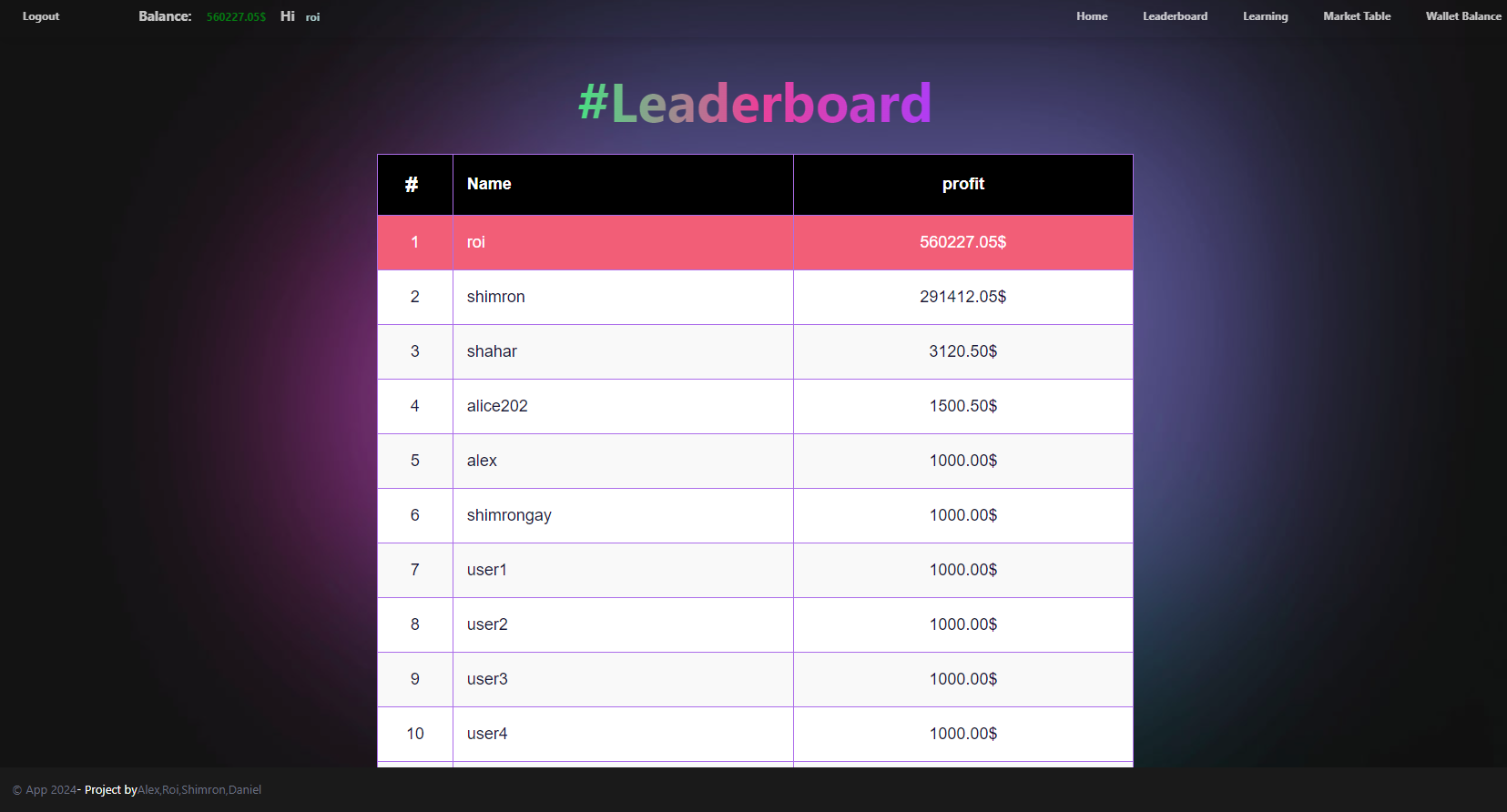
* Displays a sign-up screen allowing the user to enter an non existing user name, a password and confirming he password to register to the platform.
* **Login Navbar** allows the user to login into the platform and make actions.
* **Sign up Navbar** allows the user to create a new account.
* **Sign up bottom** after entering user data allowing the send the data to the server.

**Navbar update after login**



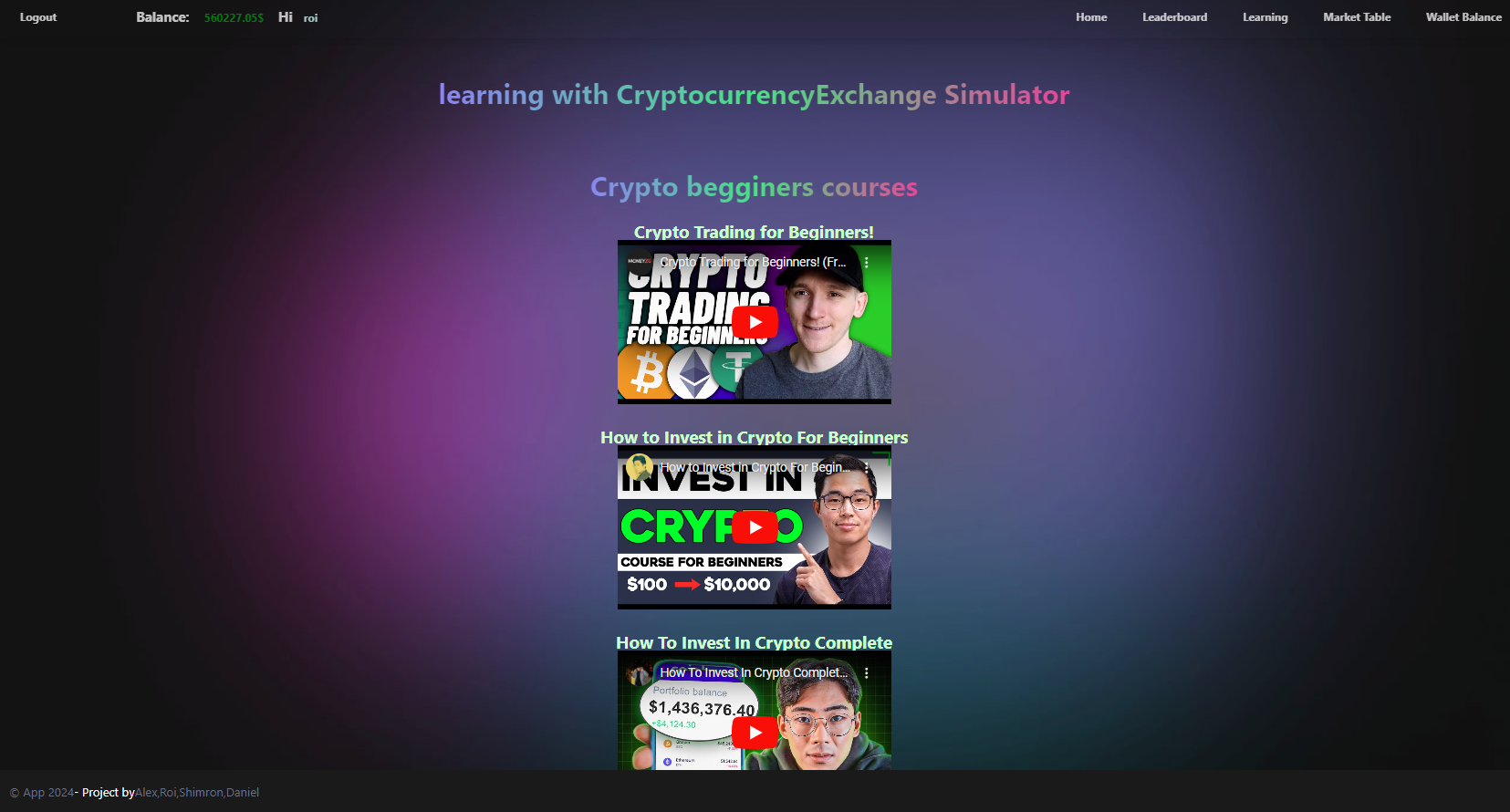
* Displays user name and current Balance
* Allows the user to watch **leaderboard**
* Allow the user to watch the **learning page**
* Allows the user to check **balance** information
* Allows the user to trade and exchange coins at the **Market Table**

**Leaderboard screen**



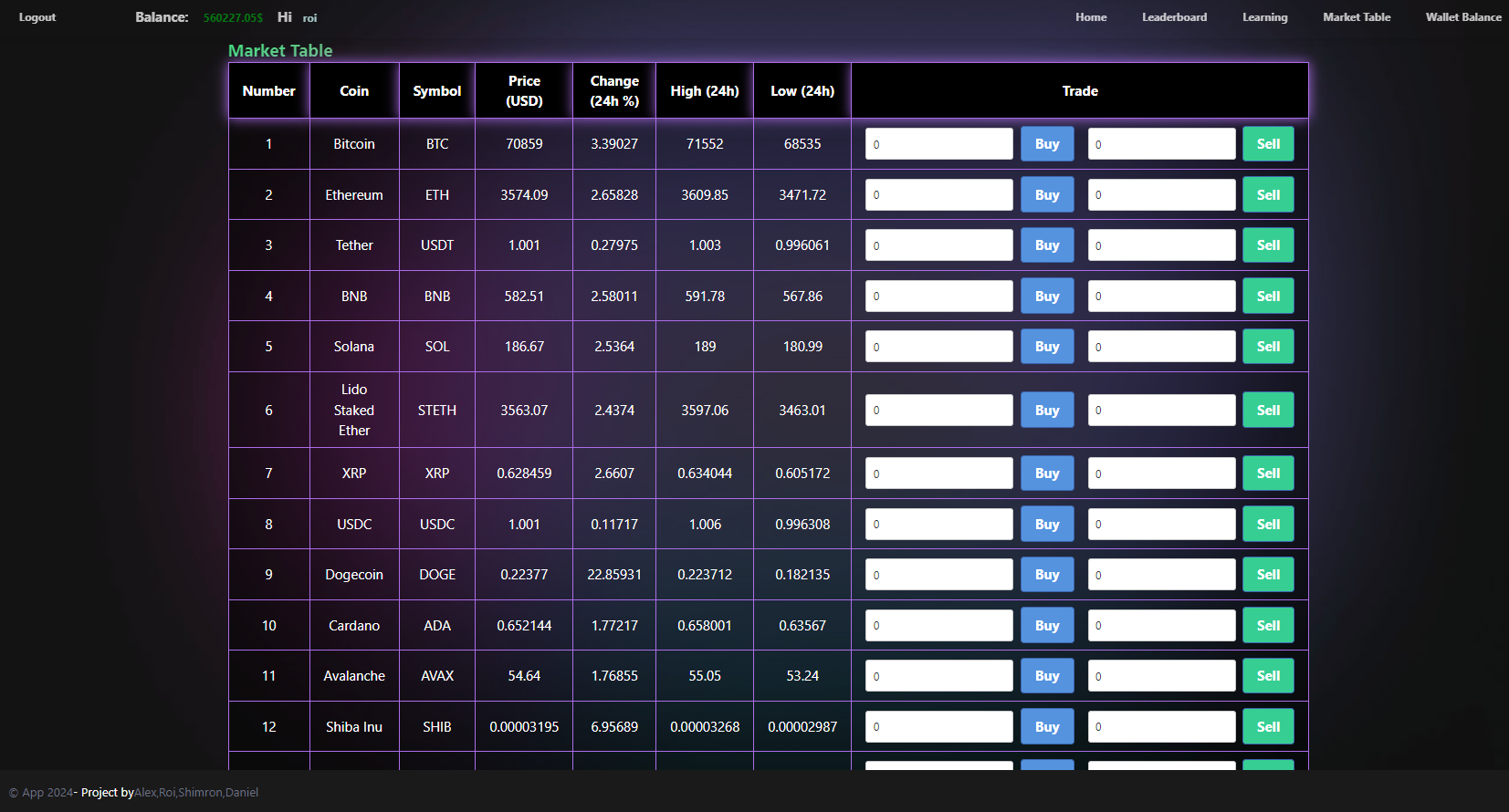
* Displays a leaderboard of highest current balance holders in the platform

**Learning page screen**



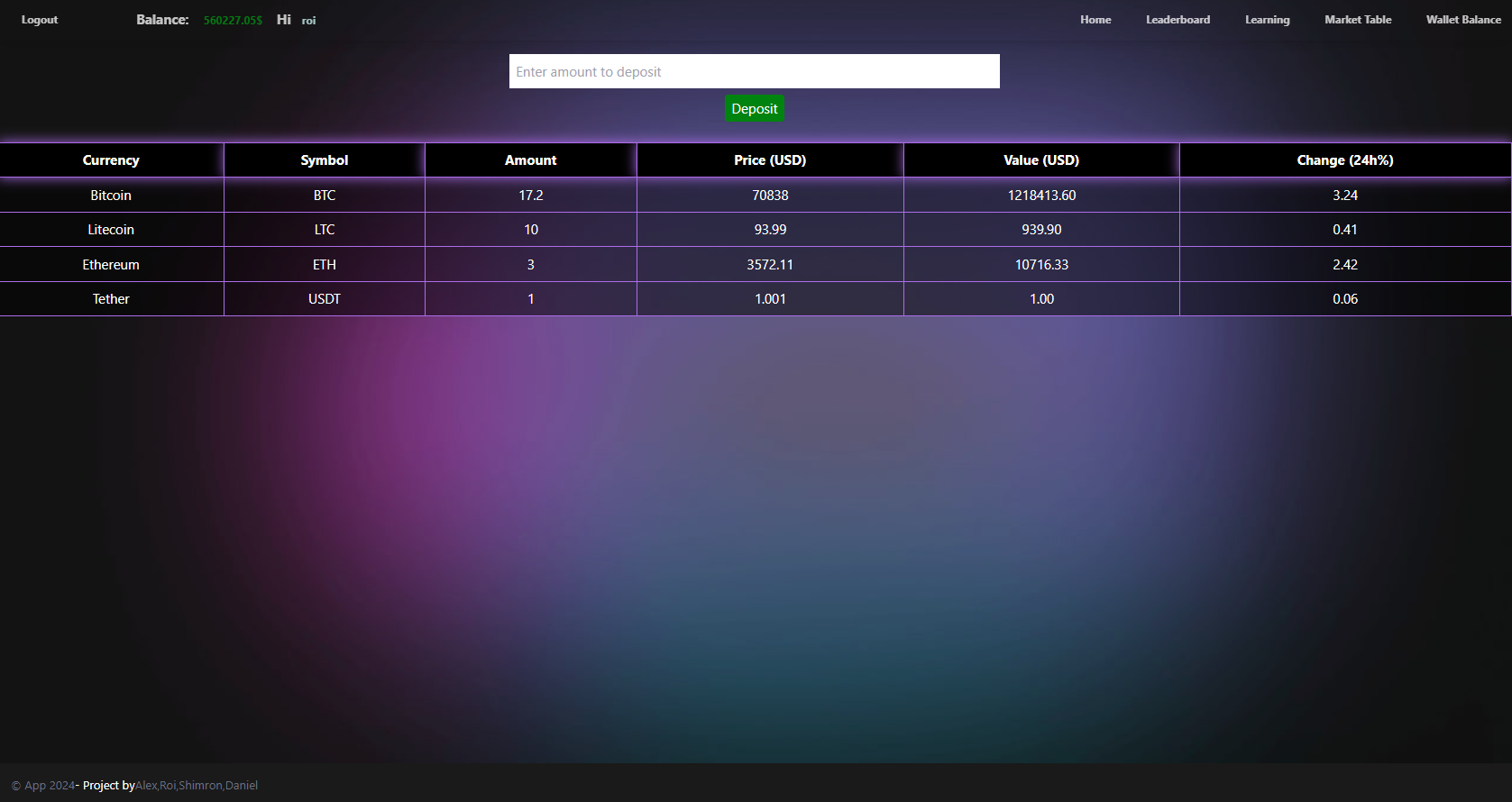
* Displays a learning page of YouTube videos the user can watch on the platform.

**Market Table and exchange screen**



* Displays the exchange table of all the coins you can buy and sell.
* **Coin, symbol** information about the coin
* **Price (USD)** current price of the coin
* **Change (24 %)** the change of the specific coin in percentage
* **High (24h)** the highest price the coin ever been in the last 24 hours
* **Low (24h)** the lowest price the coin ever been in the last 24 hours
*  **Buy text-box & button** allows the user the option to buy the amount of a specific coin that desires.
* **Sell text-box & button** allows the user to sell a specific amount of coin that exists in the user’s wallet.

**Wallet balance screen**



* Displays the wallet balance of the user
*  **Deposit Text-box & button** allows the user to deposit (USD) to their wallet.
* **Balance Table** shows the user information about the different coin that are in the wallet

יש להגיש את כל התוצרים בסביבת MTW :

* GIt repository - קישור (הריפו שפתחתם לצוות)
* Git Pages - קישור
* MoreThanWallet.com Gallery

להזכירכם , יש לבצע פריסה (deployment) ב - morethanwallet.com App Gallery

תוך שימוש באחת מהפלטפורמות GitHub pages או vercel.

הסבר על הגשה זו ניתן למצוא ב:

https://www.morethanwallet.com/appStore/gettingStarted

בנוסף, יש לוודא כי תיקיית ה- GIT ציבורית וכוללת את כל הקבצים של הפרויקט, כולל המסמכים, המצגת וכן הוראות הרצה.

**הנחיות הגשה:**

1.באחריותכם לוודא שהגשתם את כל התוצרים כנדרש ושהם הגיעו ליעדם.

לא תתאפשר בדיקה מחודשת של העבודה עקב טעויות בעת ההגשה!

2 .יש להגיש את התרגיל בקבוצות שהוגדרו בקורס.

3 .יש להגיש את התוצר הסופי של מסמך הפרויקט בפורמט WORD שייכלל בריפו של הפרויקט.

4.ניתן להפנות שאלות במייל לצוות הקורס, נא לשלוח שאלות