# טכנולוגיות אינטרנט מתקדמות - 61776 (WEB)

**הגשת פרויקט**

**<קריפתו..> <A7>**

|  |  |
| --- | --- |
| **שם חבר\ת הצוות** | **תז** |
| נור עמורי | 212263289 |
| מאהר סלמאן |  |
| אסד ספורי | 314656919 |
| עומר סאלח |  |
| עבד אללטיף זועבי |  |

תקציר הפרויקט - עד חצי עמוד- פונקציונליות מרכזיות לפי משתמשים:

### תקציר פרויקט

הפרויקט מתמקד בפיתוח מערכת לניהול ושקיפות של נכסי קרפטו עבור משתמשים פרטיים. המערכת מציעה מגוון פונקציות עיקריות המיועדות לשפר את חוויית המשתמש ולהבטיח ניהול אפקטיבי של הנכסים הדיגיטליים.

**פונקציות עיקריות לפי משתמשים:**

1. **משתמשי פרטיים:**
   * **הוספת נכסים**: המשתמשים יכולים להוסיף את אחזקותיהם במטבעות קריפטוגרפים למערכת.
   * **צפייה במחירי מטבעות בזמן אמת**: ניתן לעקוב אחר מחירי המטבעות הנוכחיים דרך ממשק המשתמש.
   * **ניהול תיקים וניתוחים**: המערכת מספקת תצוגה מפורטת של התיק הנוכחי וניתוחים של ביצועי הנכסים.
   * **צפייה בהיסטוריית עסקאות**: המשתמשים יכולים לעיין בהיסטוריה של כל העסקאות שביצעו.
   * **הפקדה ומשיכה**: המערכת מאפשרת לבצע פעולות של הפקדת כספים ומשיכת נכסים.
   * **חיבור ארנק חיצוני**: תמיכה בחיבור ובניהול ארנקים חיצוניים לצורך גמישות וניהול נרחב יותר של נכסים.

**מימוש- הטכנולוגיות המרכזיות בפרויקט:**

**סביבת עבודה:** fronted) – VS Code using React)

**סביבת עבודה:** backend) – VS Code using API)

**ספריות תקשורת צד שרת לקוח:** fronted)– Axios)

**ספריות ניווט**: (fronted)– React Router

**עיצוב:** fronted)- CSS and tailwind)

**כלי לניהול:** - git

להוסיף קישור git ,moretahwallet..

**מהנדס המערכת**, עומר סאלח, בעל רקע עשיר בבניית אתרים, הוסמך לתפקיד זה בזכות ניסיונו. במהלך העבודה על הפרויקט, נוהל ממשק פעיל בין חברי הצוות באמצעות פלטפורמת zoom שבה חולקנו את העבודה למטלות בלתי תלויות אחת בשנייה, דבר שתרם לייעול התהליך. כל העדכונים שבוצעו הועלו ל-GitHub, עם עדכון שוטף של חברי הצוות בכל שינוי.

בתהליך הקבצת הקוד, ננקטו צעדים קפדניים להבטחת איכות, כולל ביצוע בדיקות שוטפות ותיקון בעיות או באגים שהתגלו. כאשר הגיע הזמן לאיחוד הקוד, התכנסנו או ב- zoom או במכללה כדי לבצע את האיחוד בצורה חלקה ויעילה.

התקשורת המתמשכת ושיתוף הפעולה האפקטיבי בצוות תרמו להשלמת המשימות באופן מקצועי ומוצלח, תוך שמירה על איכות גבוהה בכל שלב.

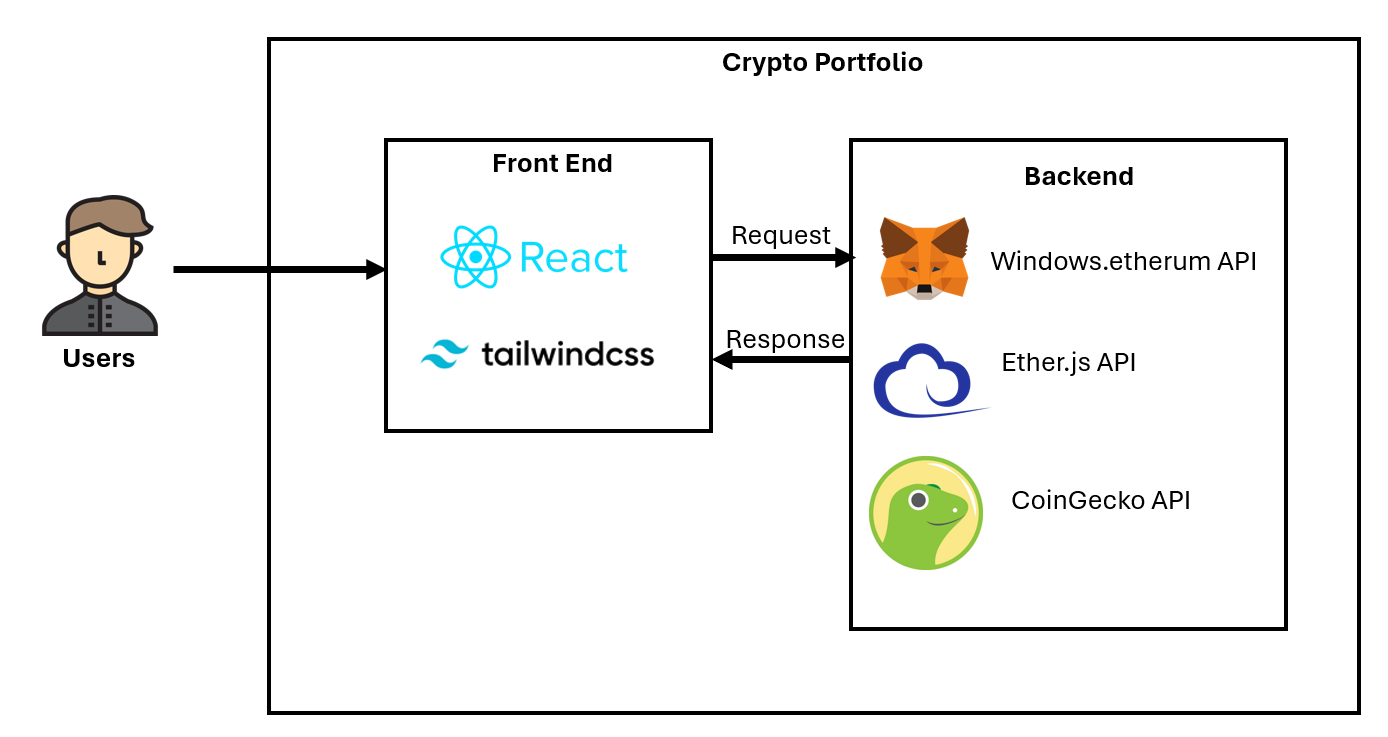
|  |  |  |
| --- | --- | --- |
| **שם חבר הצוות** | **משימות שהוקצו** | **משימות שהושלמו** |
| מאהר סלמאן | * Wallet connect , fetch main data | All |
| עומר סאלח  System Engineer | * Transactions History | All |
| נור עמורי | * Fetch wallet Assets , assets data . | All |
| אסד ספורי | * Best performing assets | All |
| עבד אלטיף זועבי | * Use case diagram + architecture design | All |

**Requirements**

|  | **דרישות פונקציונליות** |
| --- | --- |
| 1 | The system allows adding cryptocurrency holdings . |
| 2 | The system allows viewing token prices in real time . |
| 3 | The system allows viewing portfolio details and analytics. |
| 4 | The system allows viewing the best-performing assets. |
| 5 | The system allows viewing the wallet Transaction Histories. |
| 6 | The system allows Depositing and Withdraws assets. |
| 7 | The system allows to connect external wallet |

|  |  |  |
| --- | --- | --- |
| דרישות לא פונקציונליות | |  |
| 1 | The system shall fetch current prices of cryptocurrencies from a reliable external API (CoinGecko API). | Interoperability |
| 2 | The system shall support viewing historical price data over various time ranges (1 day, 1 week, 1 month, 1 year). | Usability |
| 3 | The System support multiple cryptocurrency networks (e.g., Bitcoin, Ethereum, BNB) to enable users to manage assets across different blockchain platforms. | Interoperability |
| 4 | Provide users with the ability to view detailed information for each transaction, including source address, destination address, network, amount, and fee, within the application interface | Usability |
| 5 | The user interface should be intuitive and easy to navigate for first-time users | Usability |
| 6 | The application must obtain user permission for each operation involving the wallet, including data retrieval and transactions. | Security |

Architecture System :



**1.Frontend**:

* **ReactJS:** Create the user interface using ReactJS for structure, Tailwind for styling, and JavaScript for interactivity.
* **Tailwind CSS:** Utilize Tailwind CSS for quickly styling your UI components.
* **JavaScript:** Implement client-side logic for handling user interactions, making API requests, and updating the UI dynamically.

**2.APIs**:

* + **External APIs**: Integrate with external APIs for fetching data or performing specific actions. Use JavaScript's fetch API or libraries like Axios to make HTTP requests.

**3.Application Architecture**:

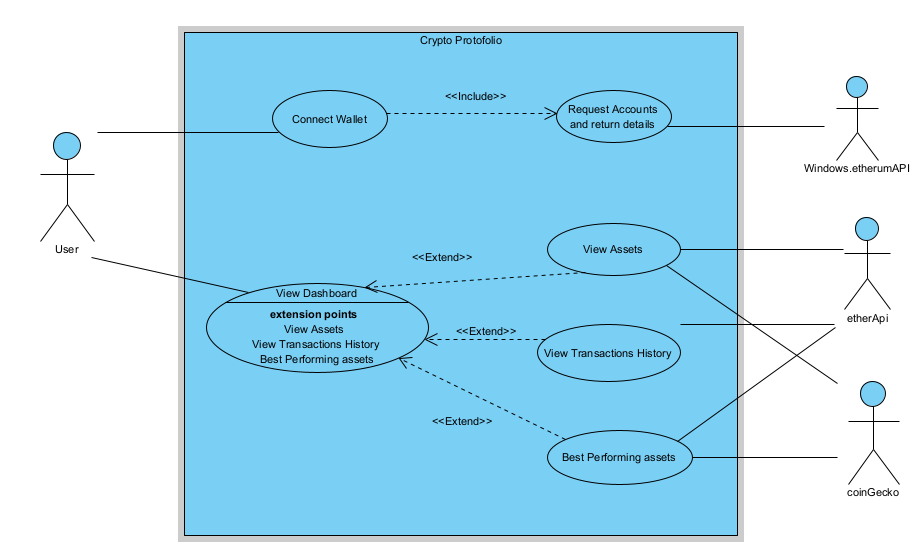
* + **Client-Server Architecture**: Follow the traditional client-server architecture where the frontend interacts with the backend through HTTP requests.

**4.Deployment:**

* **Server (Dashboard render):** Handles backend logic, requests, and database interactions.
* **Client (Vercel):** Hosts frontend code for user interface and interaction.

**דיאגרמה המתארת את התיקיות והקבצים השונים.**

**Use case**



**תיק מתכנת:**

**ספריות שהשתמשנו:**

API שהשתמשנו:

**תיק משתמש:**