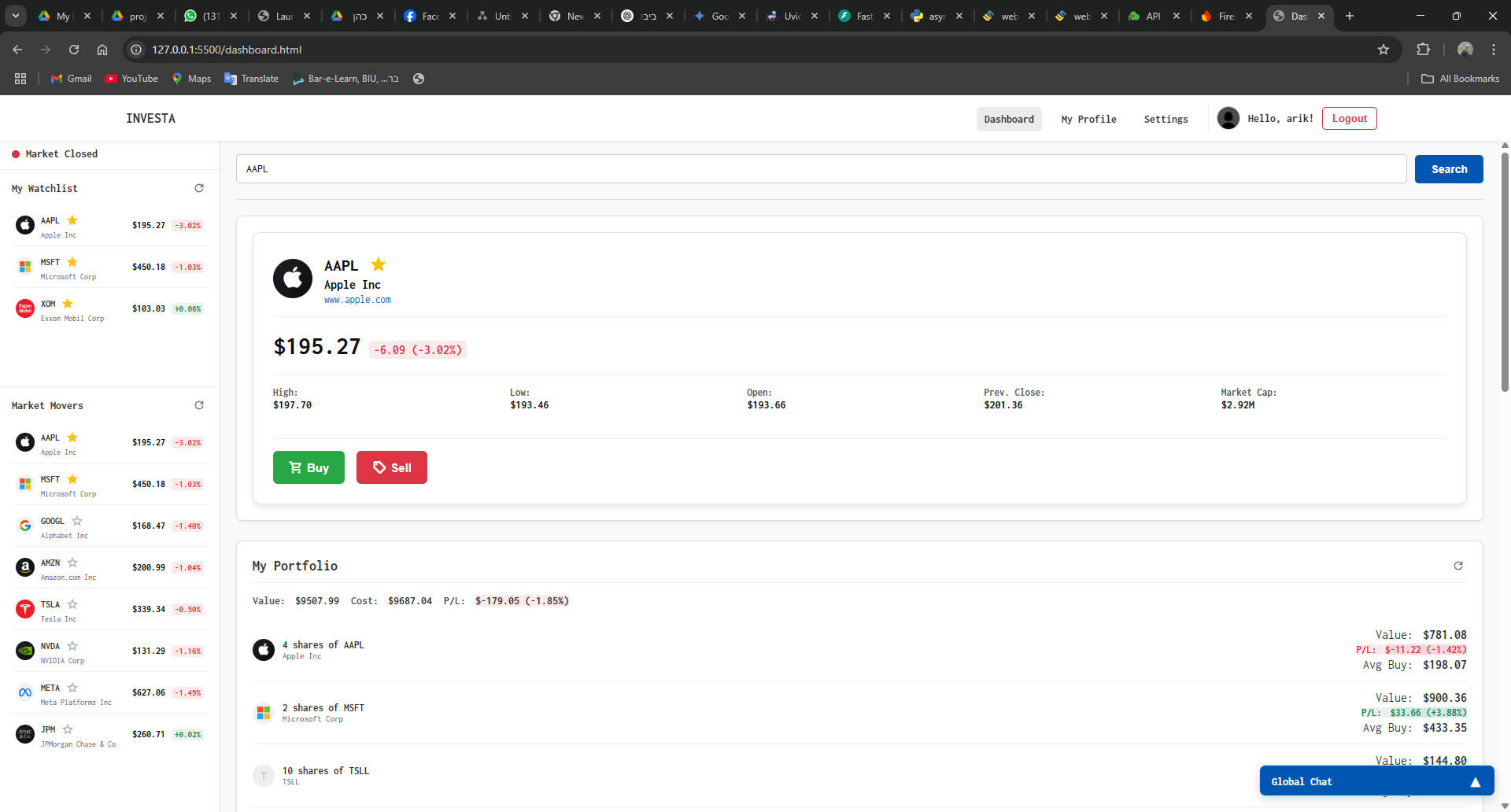


**עבודת גמר בתכנון ותכנות מערכות  
התמחות הגנת סייבר**

5 יח"ל - שאלון 883589



**שם:** ליאור כהן

**ת.ז:** 329579841

**שם המנחים:** אירית סעדון, שרית שוורץ

**שם המערכת:** INVESTA

**תאריך ההגשה:** כ"ד באייר תשפ"ה 22/05/2025

**תוכן עניינים**

[מבוא 6](#_Toc199105128)

[הרקע לפרויקט 6](#_Toc199105129)

[תיאור ראשוני של המערכת 6](#_Toc199105130)

[הגדרת הלקוח 6](#_Toc199105131)

[הגדרת מטרות 6](#_Toc199105132)

[בעיות תועלות וחסכונות 7](#_Toc199105133)

[סקירת פתרונות קיימים 7](#_Toc199105134)

[אפיון המערכת 8](#_Toc199105135)

[סקירת טכנולוגיות הפרויקט 8](#_Toc199105136)

[תיחום הפרויקט 8](#_Toc199105137)

[תיאור מפורט של המערכת 8](#_Toc199105138)

[פירוט היכולות שהיא תעניק לכל סוג משתמש במערכת 8](#_Toc199105139)

[פירוט הבדיקות 9](#_Toc199105140)

[תכנון וניהול לו"ז זמנים לפיתוח המערכת 10](#_Toc199105141)

[תיאור תחום הידע 11](#_Toc199105142)

[פירוט היכולות בצד שרת 11](#_Toc199105143)

[פירוט היכולות בצד לקוח 12](#_Toc199105144)

[מבנה 15](#_Toc199105145)

[תיאור הארכיטקטורה של המערכת המוצעת 15](#_Toc199105146)

[תיאור הטכנולוגיה הרלוונטית 15](#_Toc199105147)

[תיאור זרימת המידע במערכת 16](#_Toc199105148)

[תיאור האלגוריתמים מרכזיים בפרויקט 31](#_Toc199105149)

[אימות וניהול משתמשים 31](#_Toc199105150)

[אחזור ועיבוד נתוני מניות 32](#_Toc199105151)

[ניהול תקשורת בזמן אמת (צ'אט) 33](#_Toc199105152)

[תיאור סביבת הפיתוח 34](#_Toc199105153)

[תיאור פרוטוקול התקשורת 34](#_Toc199105154)

[תיאור מסכי המערכת 37](#_Toc199105155)

[תרשים המסכים 37](#_Toc199105156)

[מסך נחיתה 38](#_Toc199105157)

[מסך התחברות 39](#_Toc199105158)

[מסך הרשמה 40](#_Toc199105159)

[מסך בית 41](#_Toc199105160)

[מסך פרופיל אישי 42](#_Toc199105161)

[מסך הגדרות 43](#_Toc199105162)

[מסך פרופיל (של משתמש אחר) 44](#_Toc199105163)

[תיאור מבני הנתונים 45](#_Toc199105164)

[סקירת החולשות 47](#_Toc199105165)

[שכבת האפליקציה 47](#_Toc199105166)

[שכבת התעבורה 48](#_Toc199105167)

[מימוש הפרויקט 50](#_Toc199105168)

[סקירת המודלים 50](#_Toc199105169)

[מודלים בpython 50](#_Toc199105170)

[מודלים בJS 50](#_Toc199105171)

[מודלים\מחלקות שפותחו 51](#_Toc199105172)

[קבצי JavaScript 52](#_Toc199105173)

[קטעי קוד ממרכיבים מרכזיים 53](#_Toc199105174)

[פירוט הבדיקות 60](#_Toc199105175)

[תקינות מערכת ההרשמה וההתחברות 60](#_Toc199105176)

[תקינות חיפוש מניות 60](#_Toc199105177)

[תקינות מערכת הצ'אט 60](#_Toc199105178)

[תקינות מערכת הפורטפוליו 60](#_Toc199105179)

[מדריך למשתמש 62](#_Toc199105180)

[פירוט כלל קבצי המערכת 62](#_Toc199105181)

[התקנת המערכת 63](#_Toc199105182)

[אופן הפעלת המערכת 64](#_Toc199105183)

[רפלקציה 68](#_Toc199105184)

[ביבליוגרפיה 70](#_Toc199105185)

[נספחים 72](#_Toc199105186)

[chat\_server.py 72](#_Toc199105187)

[getStocksData.py 75](#_Toc199105188)

[dashboard.js 77](#_Toc199105189)

[login.js 106](#_Toc199105190)

[profile.js 115](#_Toc199105191)

[settings.js 129](#_Toc199105192)

[user\_profile.js 138](#_Toc199105193)

[dashboard.html 154](#_Toc199105194)

[dashboard.css 158](#_Toc199105195)

[index.html 177](#_Toc199105196)

[login.html 182](#_Toc199105197)

[profile.css 188](#_Toc199105198)

[profile.html 198](#_Toc199105199)

[settings.css 200](#_Toc199105200)

[settings.html 207](#_Toc199105201)

[styles.css 210](#_Toc199105202)

[user\_profile.css 218](#_Toc199105203)

[user\_profile.html 221](#_Toc199105204)

**מבוא**

# מבוא

## הרקע לפרויקט

### תיאור ראשוני של המערכת

הפרויקט הינו פלטפורמה חברתית למשקיעים. המערכת מאפשרת צ'אט עם משקיעים נוספים ומעקב אחרי תיק ההשקעות שלהם. בנוסף המערכת מציגה חדשות על השוק בפני המשתמש ומאפשרת לו לעקוב אחרי מנות שבוחר.

בחרתי בפרויקט זה מתוך התעניינות אישית בשוק ההון, וכן מתוך הבנה שקיים צורך בכלי שמשלב אפשרויות לתקשורת ישירה בין משקיעים וכמו כן צפייה ומעקב אחרי מניות בעזרת תיק השקעות.

האתגרים העיקריים שצפיתי בפרויקט כללו מציאת מקור אמין לנתוני מניות בזמן אמת ויצירת ממשק משתמש ידידותי שמשלב בצורה חלקה את כל הפונקציונליות המורכבת של המערכת.

### הגדרת הלקוח

המערכת מיועדת למגוון רחב של משקיעים, החל ממשקיעים מתחילים המחפשים הדרכה והכוונה בצעדיהם הראשונים בעולם ההשקעות ע"י משקיעים מנוסים יותר, ועד למשקיעים מנוסים המעוניינים ברשת קשרים עם עמיתים למקצוע. המשתמשים יכולים להיות משקיעים פרטיים, סטודנטים לכלכלה המעוניינים בהדרכה וחובבי שוק ההון בכל הרמות.

### הגדרת מטרות

1. לאפשר תקשורת ישירה בין משקיעים באמצעות צ'אט פרטי לשיתוף ידע ותובנות.
2. לספק מערכת לניהול פורטפוליו דמה שמאפשרת למשתמשים לעקוב אחר ההשקעות שלהם ולצפות ברווחים/הפסדים.

### בעיות תועלות וחסכונות

הבעיות העיקריות שהמערכת מנסה לפתור:

1. קושי של משקיעים לנתח כמויות גדולות של מידע פיננסי וחדשות רלוונטיות.
2. חוסר בפלטפורמות שמשלבות יכולות ניהול תיק השקעות עם יכולות תקשורת בין משקיעים.

התועלות הצפויות מהמערכת:

1. קבלת החלטות השקעה מושכלות יותר בעזרת רעיונות של משקיעים אחרים.
2. יצירת קהילת משקיעים תומכת המאפשרת שיתוף ידע ותובנות.
3. מעקב פשוט וויזואלי אחר ביצועי ההשקעות בפורטפוליו.
4. נגישות למידע פיננסי מורכב בצורה ברורה וידידותית למשתמש.

חסכונות:

1. חיסכון בזמן שמושקע בחיפוש רעיונות של אחרים ע"י שיתוף בפלטפורמה.
2. צמצום הצורך במספר פלטפורמות שונות (אחת לתקשורת, אחת למעקב).
3. הקטנת הסיכון בהשקעות דרך שיפור איכות המידע וההחלטות בעקבות כך שניתן לראות האם כל משקיע הוא מרווח\מופסד.

### סקירת פתרונות קיימים

1. TradingView – מאפשרת ניתוח טכני של מניות, אין קשר בין משקיעים.
2. Reddit – מאפשרת קשר בין משקיעים, אין תיק דמה.
3. eTorro – מאפשרת ניתוח בעזרת בינה מלאכותית לא מאפשרת קשר בין משקיעים.

## אפיון המערכת

### סקירת טכנולוגיות הפרויקט

לא מדובר בטכנולוגיות חדשות, כל הטכנולוגיות קיימות שמשלבים ביניהם על מנת ליצור את הפרויקט המוגמר.

### תיחום הפרויקט

הפרויקט שלי עוסק ברשתות בשביל ליצור את הצ'אט בין המשקיעים, בנוסף הפרויקט שלי יוצר קריאות בAPI על מנת לקבל מידע מעודכן על מצב השוק, חדשות השוק ומצב התיק של משתמשים אחרים.

### תיאור מפורט של המערכת

המערכת מאפשרת למשקיעים ליצור פרופיל ותיק דמה ולעקוב אחרי מניות. בנוסף המערכת מציגה את חדשות השוק, משתמשים אחרים, מאפשרת לחפש משתמשים וכן המערכת מציגה את הפורטפוליו שלהם, מצב הרווח\הפסד ואפשרות לעקוב אחריהם.

### פירוט היכולות שהיא תעניק לכל סוג משתמש במערכת

כל היכולות המפורטות לעיל כגון: יצירת משתמש בחינם, מעקב אחרי מניות, יצירת קשר עם משקיעים ומעקב אחריהם.

### פירוט הבדיקות

#### בדיקה 1: תקינות מערכת ההרשמה וההתחברות

* מה הבדיקה בודקת: בדיקת תהליך ההרשמה וההתחברות למערכת.
* אופן הבדיקה: ניסיון להירשם עם פרטים תקינים, פרטים שגויים, ניסיון להתחבר עם משתמש קיים.

#### בדיקה 2: תקינות חיפוש מניות

* מה הבדיקה בודקת: דיוק המידע המתקבל מהAPI של המניות.
* אופן הבדיקה: חיפוש מניות וקבלת נתונים, השוואת הנתונים לאלו המוצגים במקורות מידע מוכרים.

#### בדיקה 3: תקינות מערכת הצ'אט

* מה הבדיקה בודקת: יכולת לשלוח ולקבל הודעות.
* אופן הבדיקה: שליחת הודעות בין שתי חשבונות שונים ובדיקת ההגעה בזמן אמת.

#### בדיקה 4: תקינות מערכת הפורטפוליו

* מה הבדיקה בודקת: יכולת להוסיף ולהסיר מניות מהפורטפוליו וחישוב נכון של רווח/הפסד
* אופן הבדיקה: הוספת מניות לפורטפוליו, מעקב אחר שינויי מחיר ובדיקת חישוב הרווח/הפסד.

### תכנון וניהול לו"ז זמנים לפיתוח המערכת

|  |  |
| --- | --- |
| שבוע 1 | - חשיבה על רעיון הפרויקט ואיפיון ראשוני שלו. - הגדרת מטרות ויעדים כלליים. |
| שבוע 2 | - כתיבת החלק הראשוני של פרק המבוא בתיק הפרויקט. - המשך אפיון מפורט יותר של המערכת. |
| שבוע 3 | - התחלת כתיבת קוד ראשוני (שלד המערכת, מבנה בסיסי). - למידה מעמיקה של טכנולוגיות ספציפיות הנדרשות לפרויקט. |
| שבוע 4 | - המשך פיתוח המערכת. - עיצוב ראשוני של מסכי המערכת. |
| שבוע 5 | - המשך פיתוח המערכת. - התחלת כתיבת פרק "תיאור תחום הידע" בתיק הפרויקט. |
| שבוע 6 | - המשך פיתוח המערכת. |
| שבוע 7 | - התמקדות בבדיקות ראשוניות של המערכת (בדיקות יחידה). |
| שבוע 8 | - המשך פיתוח ותיקון באגים.  - התחלת כתיבת פרק "מבנה / ארכיטקטורה של הפרויקט" בתיק הפרויקט. |
| שבוע 9 | - בדיקות מעמיקות יותר (בדיקות אינטגרציה, בדיקות מערכת).  - המשך פיתוח ותיקון באגים. |
| שבוע 10 | - תיקון באגים ושיפורים נוספים.  - התחלת כתיבת פרק "מימוש הפרויקט" בתיק הפרויקט. |
| שבוע 11 | - סיום פיתוח המערכת. - התחלת כתיבת "מדריך למשתמש" בתיק הפרויקט. |
| שבוע 12 | - כתיבת פרק "סיכום אישי / רפלקציה" בתיק הפרויקט.  - תיקון באגים. |
| שבוע 13 | - הכנת תיק הפרויקט להגשה סופית (עריכה, עיצוב). |
| שבוע 14 | הגשת הפרויקט.🥳 |

## תיאור תחום הידע

### פירוט היכולות בצד שרת

#### יכולת 1: אימות משתמשים

שם היכולת: מערכת אימות משתמשים

מהות היכולת: זיהוי ואימות משתמשים ברישום והתחברות למערכת

אוסף היכולות/פעולות: קליטת נתוני משתמש, בדיקת תקינות, הצפנת סיסמאות, שמירה בfirebase

אובייקטים נחוצים: מחלקת משתמש, מחלקת אימות, מחלקת הצפנה, firebase

#### יכולת 2: קבלת נתוני מניות

שם היכולת: מערכת איסוף נתוני מניות

מהות היכולת: קבלת נתוני מניות עדכניים מFinnhub API

אוסף היכולות/פעולות: יצירת בקשות , עיבוד תגובות, שליחת תגובות

אובייקטים נחוצים: תקשורת, מחלקת ניתוח תגובות

#### יכולת 4: ניהול תקשורת בין משתמשים

שם היכולת: מערכת צ'אט

מהות היכולת: ניהול תקשורת בין משתמשים

אוסף היכולות/פעולות: שליחת הודעות, קבלת הודעות

אובייקטים נחוצים: מחלקת צ'אט

### פירוט היכולות בצד לקוח

#### יכולת 1: ממשק משתמש

שם היכולת: מערכת ממשק משתמש.

מהות היכולת: הצגת הנתונים והפונקציונליות למשתמש בצורה ידידותית.

אוסף היכולות/פעולות: הצגת נתונים, קבלת קלט משתמש, ניווט בין מסכים.

אובייקטים נחוצים: ממשק משתמש, תקשורת, firebase.

#### יכולת 2: תצוגת מניות

שם היכולת: תצוגת מניות.

מהות היכולת: הצגת מידע מפורט על מניות .

אוסף היכולות/פעולות: הצגת נתונים מספריים, אפשרות להוספה לפורטפוליו, אפשרות למעקב.

אובייקטים נחוצים: תקשורת, בסיסי נתונים , ממשק משתמש.

#### יכולת 3: ממשק צ'אט

שם היכולת: ממשק צ'אט.

מהות היכולת: אפשרות לתקשורת בין משתמשים.

אוסף היכולות/פעולות: שליחת וקבלת הודעות בזמן אמת.

אובייקטים נחוצים: מחלקת ניהול הודעות, תקשורת, ממשק משתמש.

#### יכולת 4: ממשק ניהול פורטפוליו

שם היכולת: ממשק ניהול פורטפוליו.

מהות היכולת: הצגה וניהול של הפורטפוליו האישי.

אוסף היכולות/פעולות: הצגת רשימת מניות, הצגת רווח/הפסד, הוספה/הסרה של מניות.

אובייקטים נחוצים: תקשורת, בסיס נתונים, ממשק משתמש.

#### יכולת 5: הרשמה\התחברות למערכת

שם היכולת: הרשמה\התחברות למערכת

מהות היכולת: להתחבר למשתמש קיים וליצור משתמש חדש

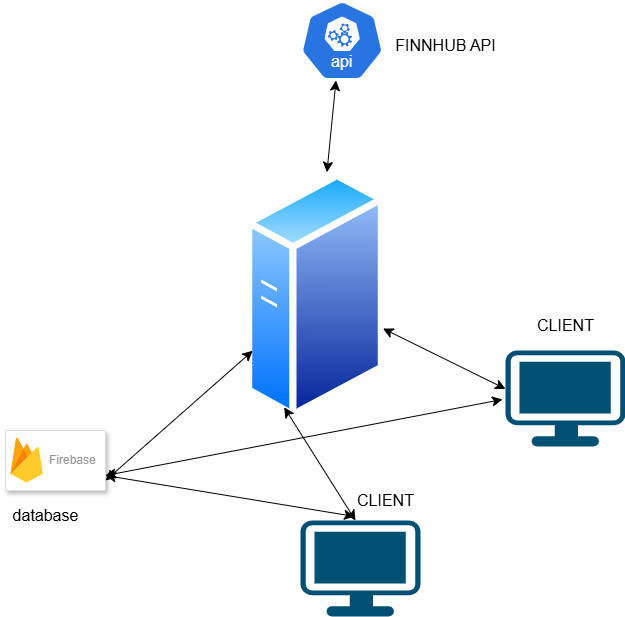
אוסף היכולות/פעולות: התחברות למערכת, יצירת משמשת חדש

אובייקטים נחוצים: תקשורת, בסיס נתונים, ממשק משתמש.

**מבנה**

# מבנה

## תיאור הארכיטקטורה של המערכת המוצעת



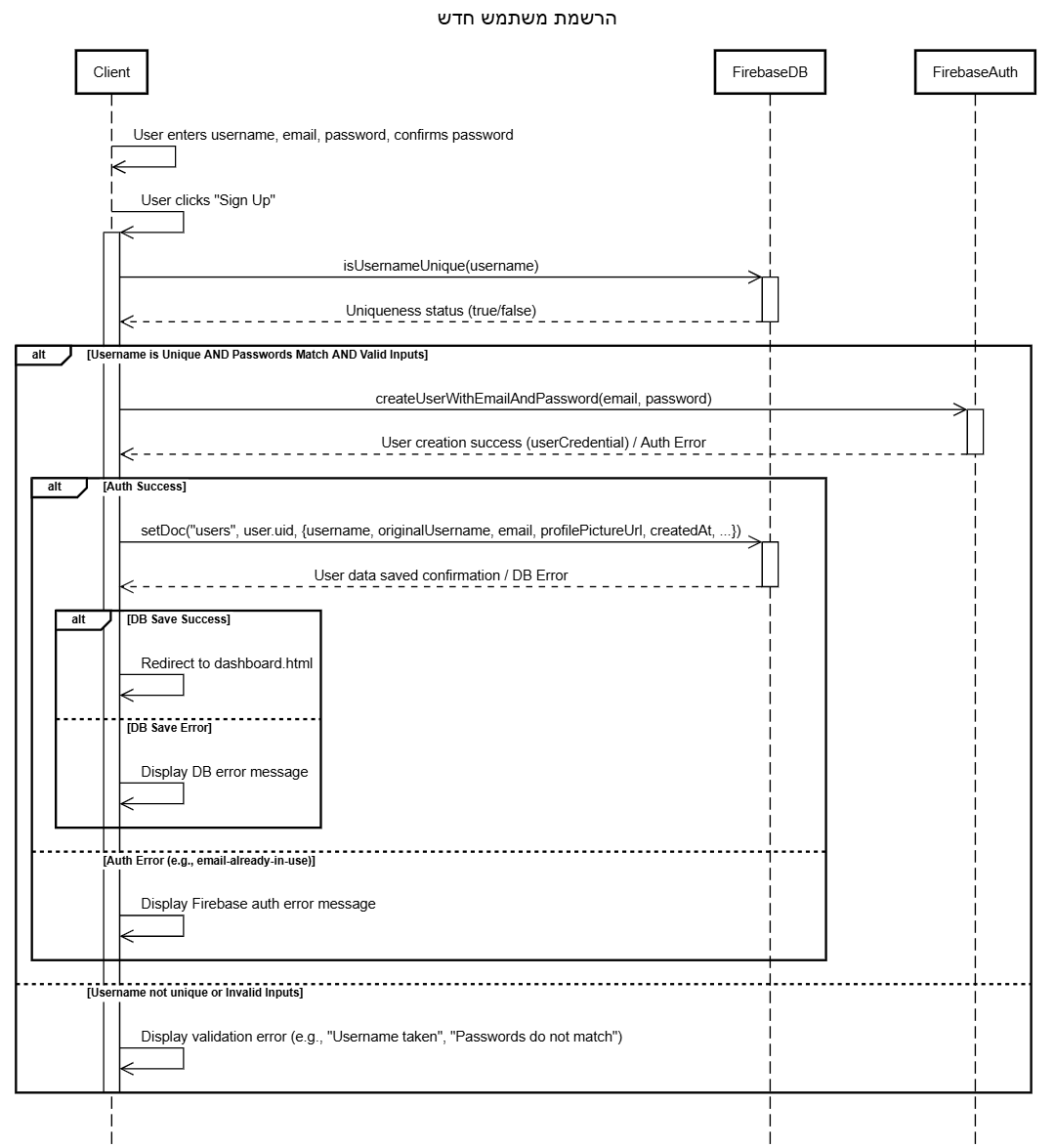
## תיאור הטכנולוגיה הרלוונטית

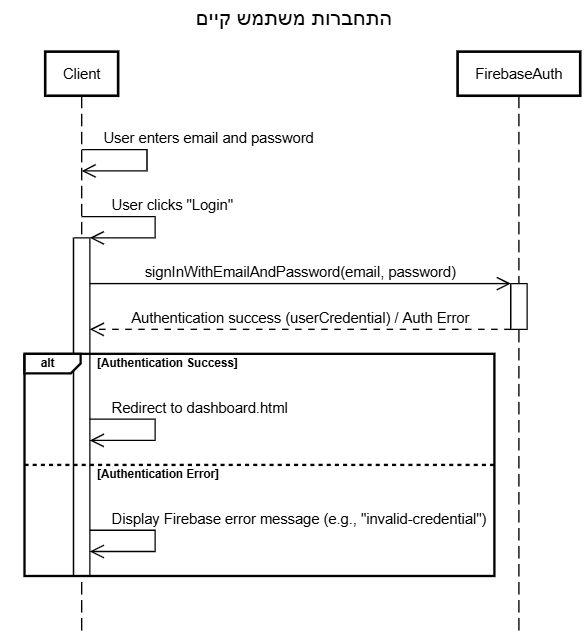
ממשק המשתמש כתוב בhtml, CSS. התקשרות עם בסיס הנתונים כתובה בJS והתקשורת בין המשתמשים וFINNHUB מתבצעת ע"י הPython אשר מחזיר תשובה לJS. מכיוון שזה אתר אז ניתן להריץ אותו על כל מערכת הפעלה אשר מריצה python, JS, html.

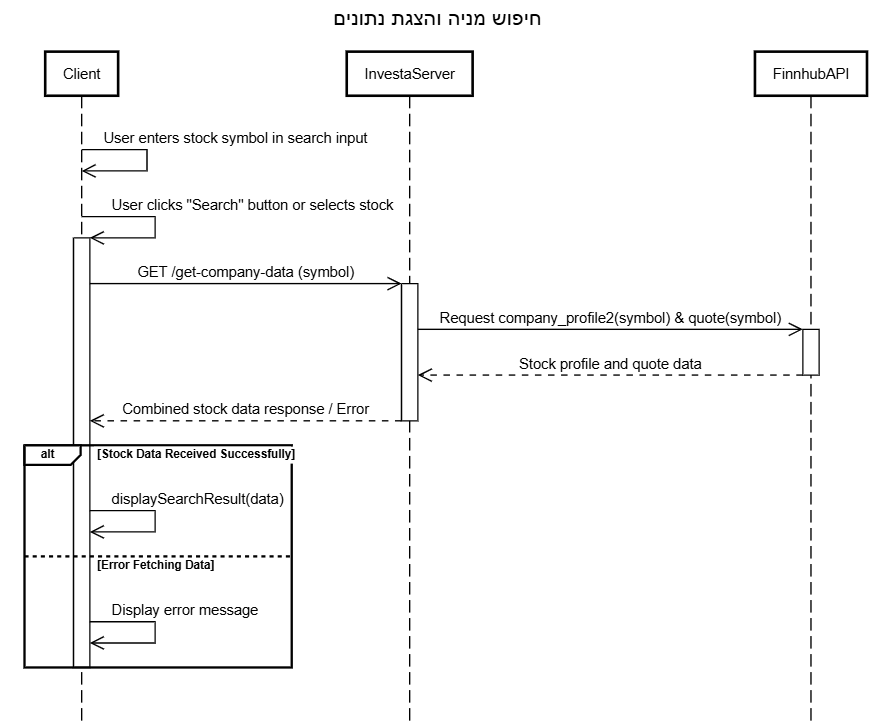
FINNHUB הוא API חינמי המאפשר מידע לגבי מניות והחברות המחזיקות אותן.

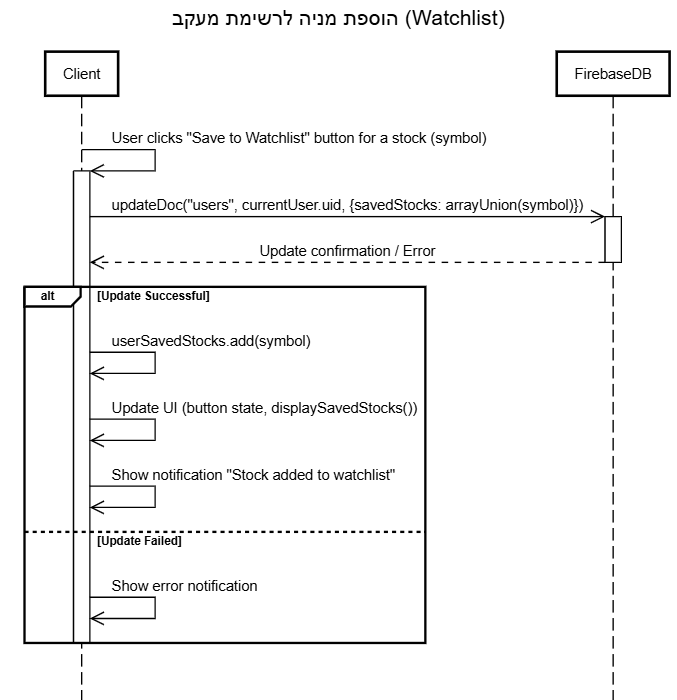
תחום העניין של הפרויקט הוא שוק ההון, רשת חברתית המקשרת בין המשקיעים

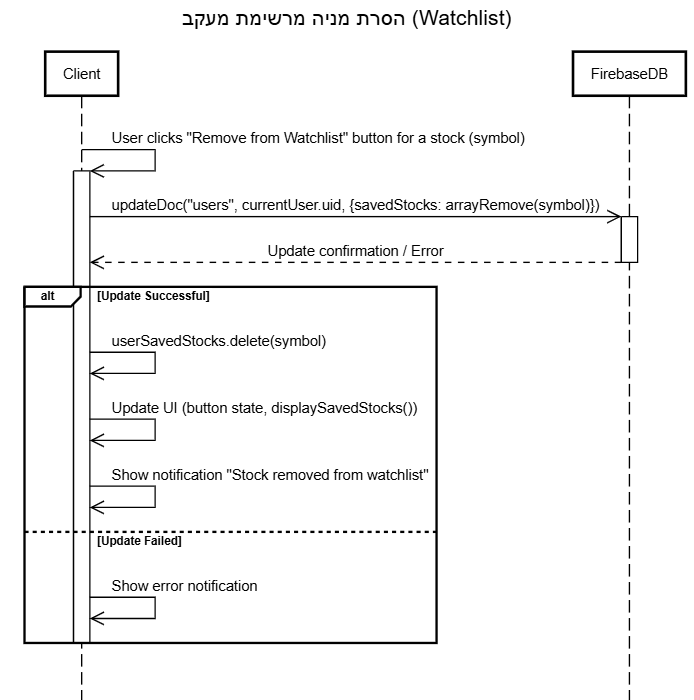
## תיאור זרימת המידע במערכת

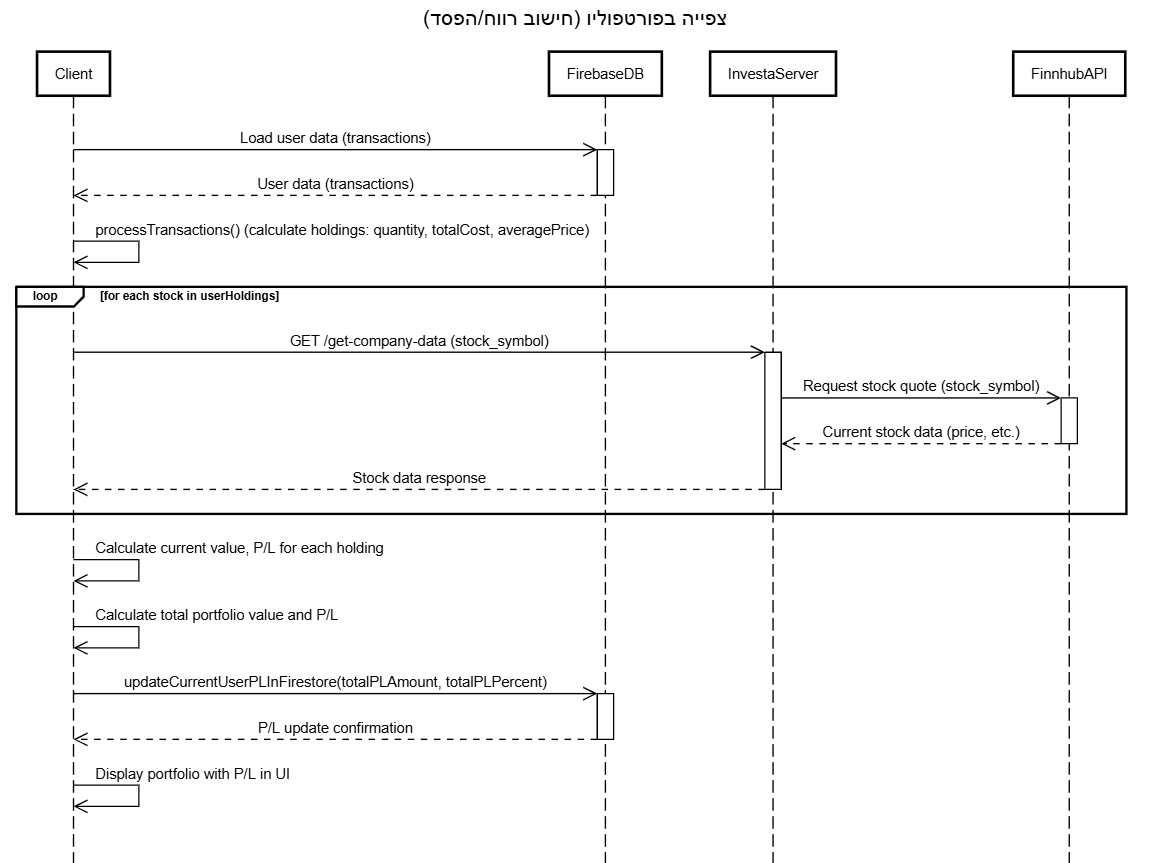
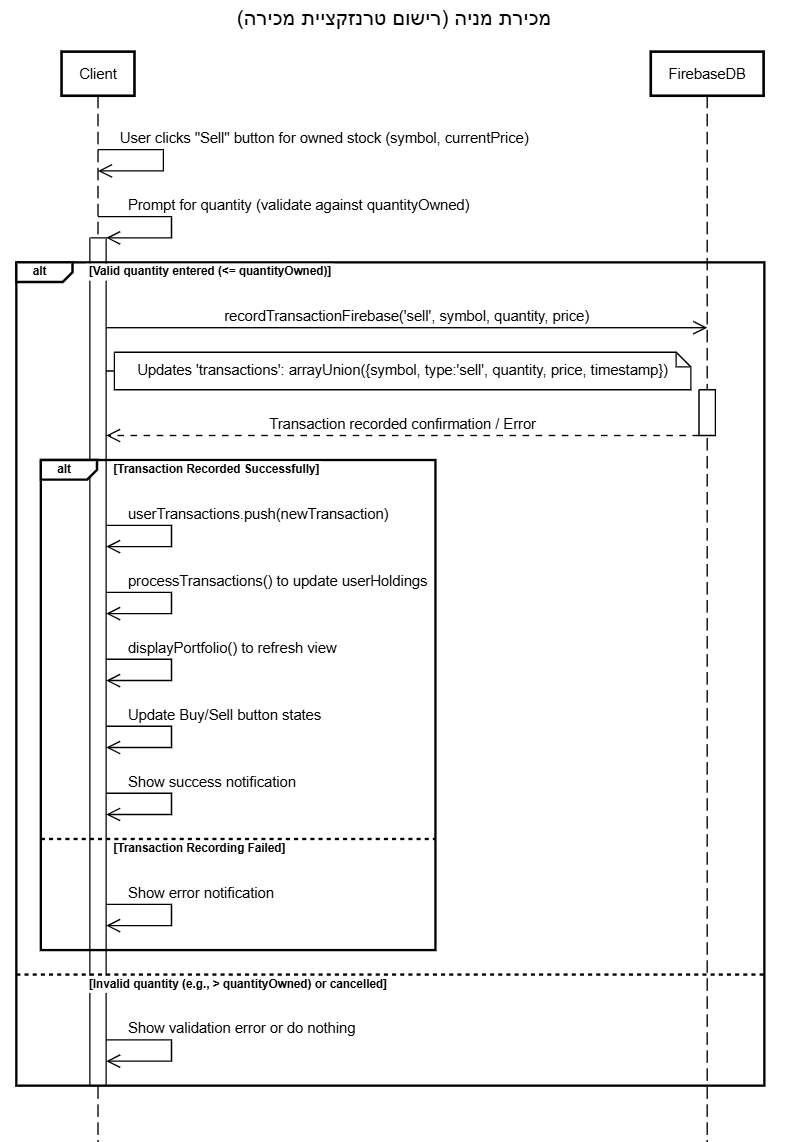
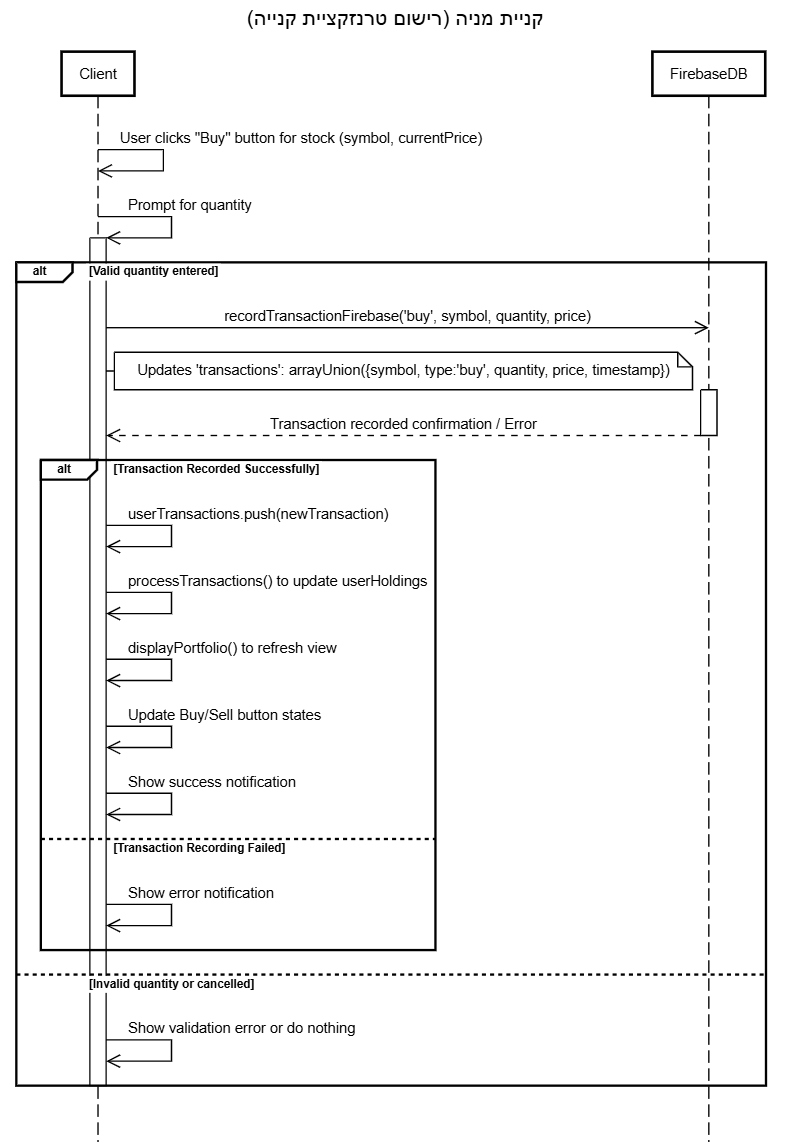


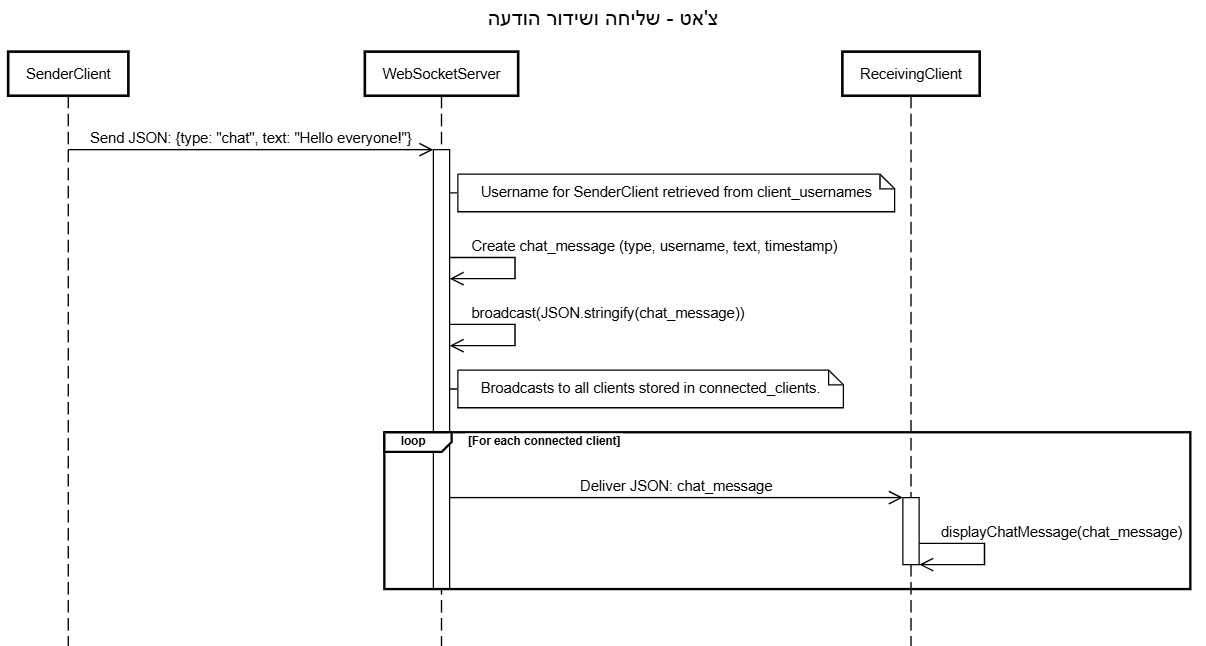


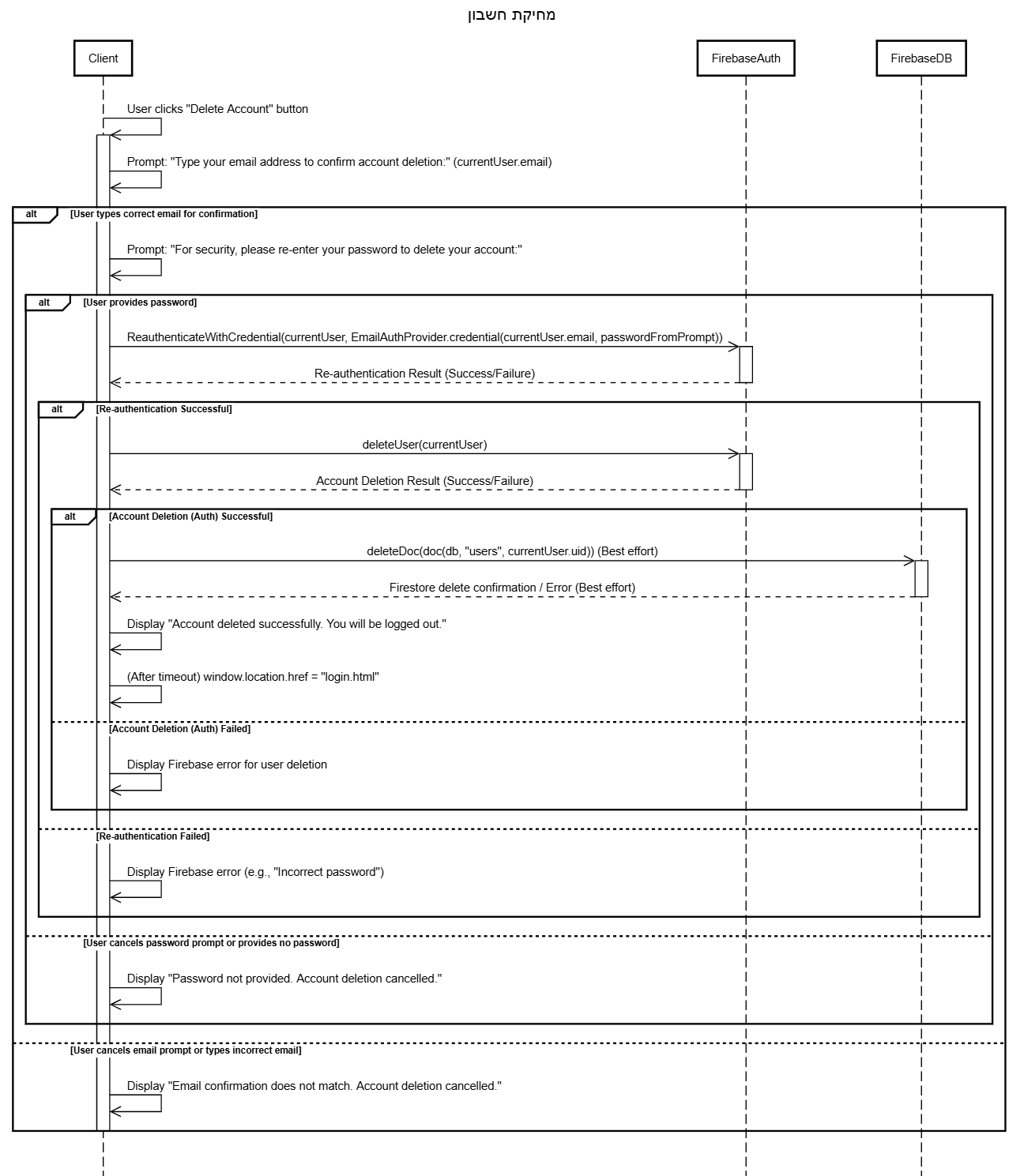
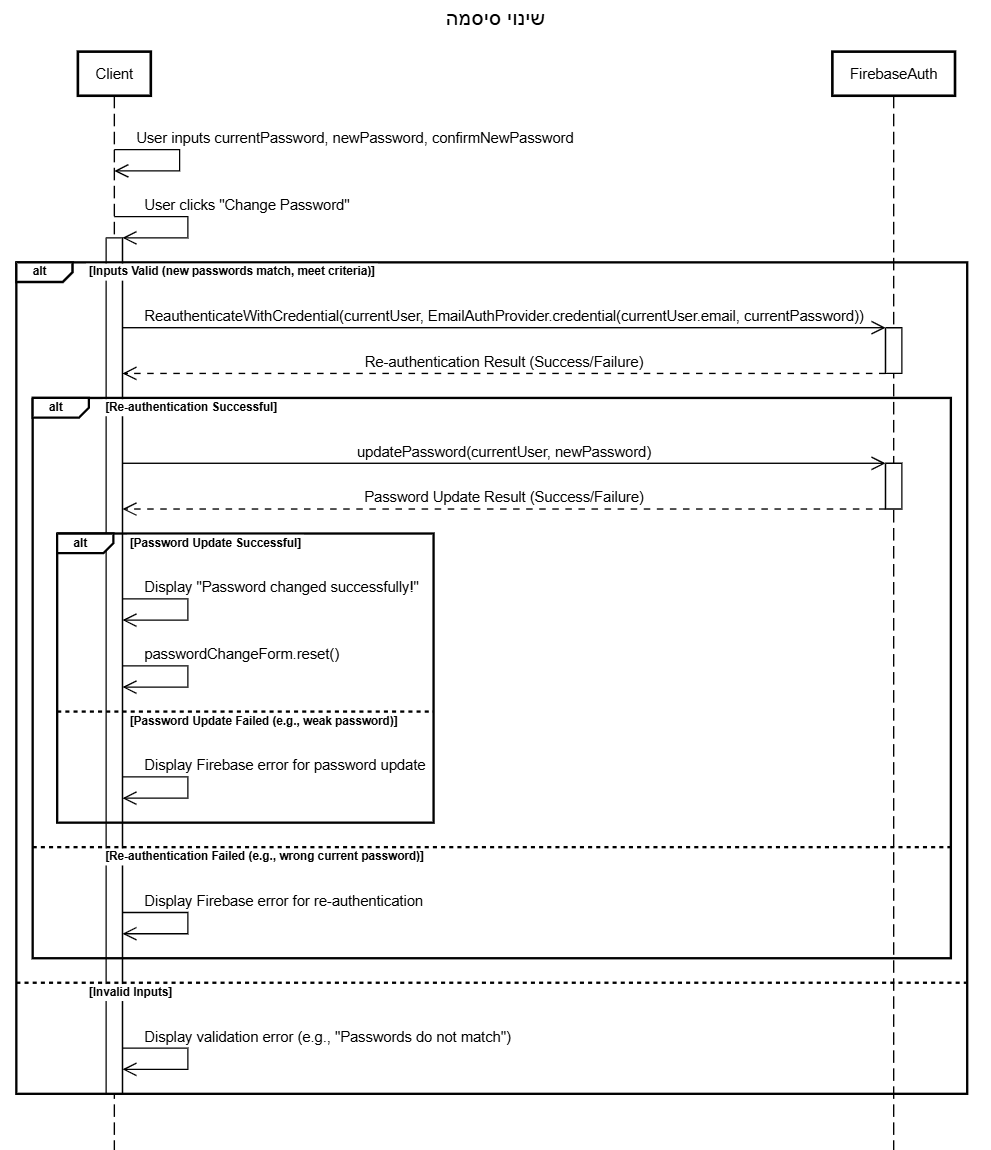
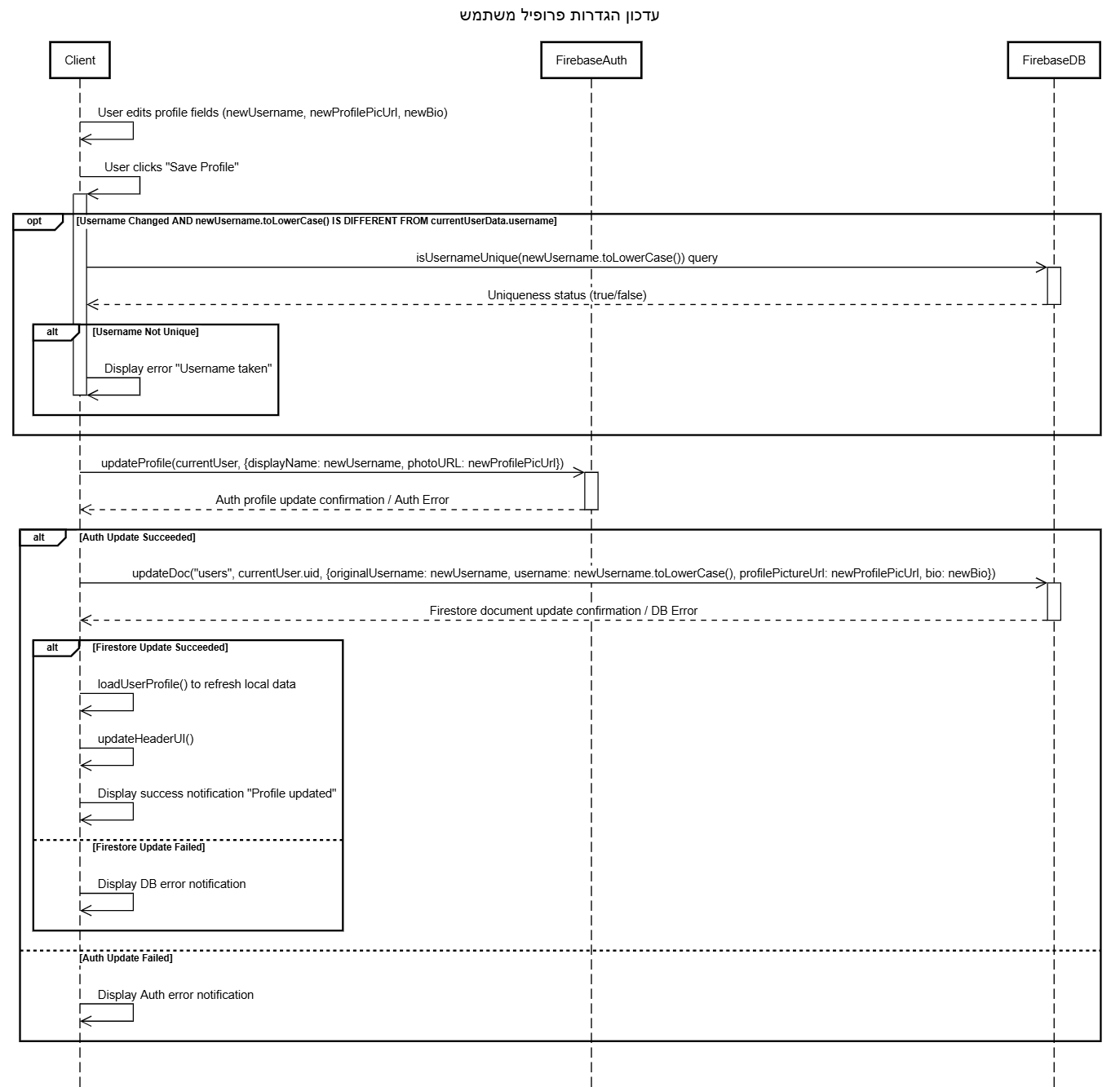
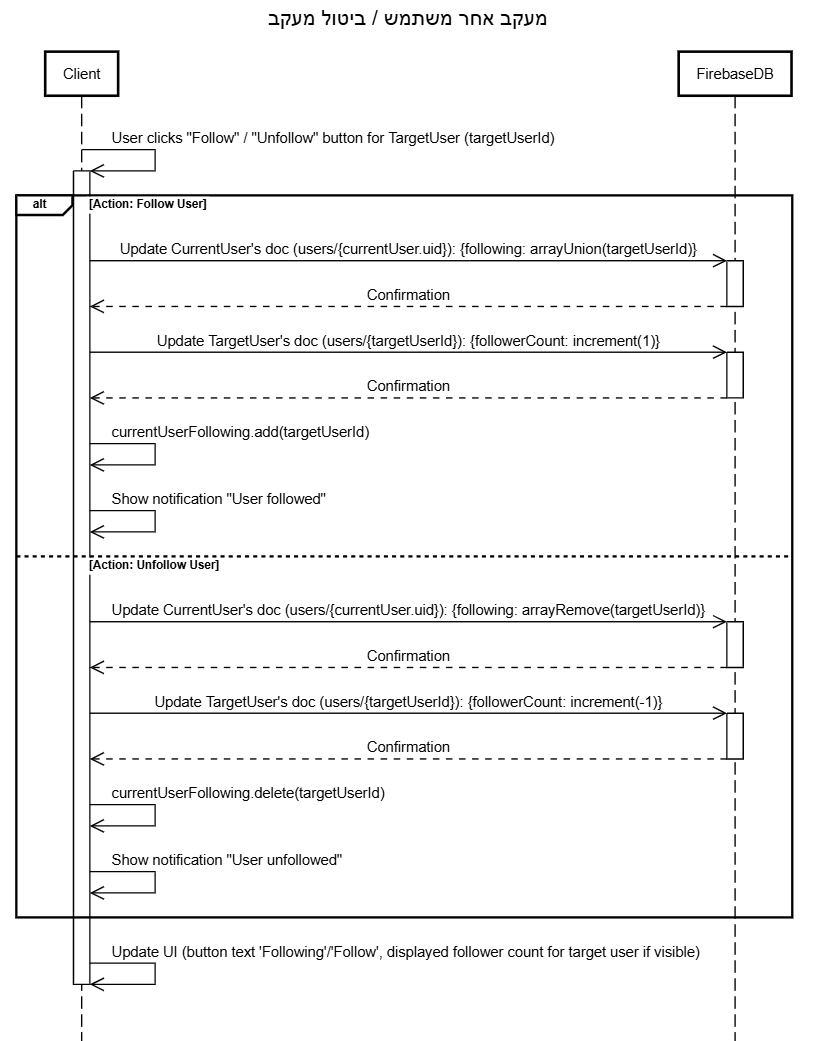
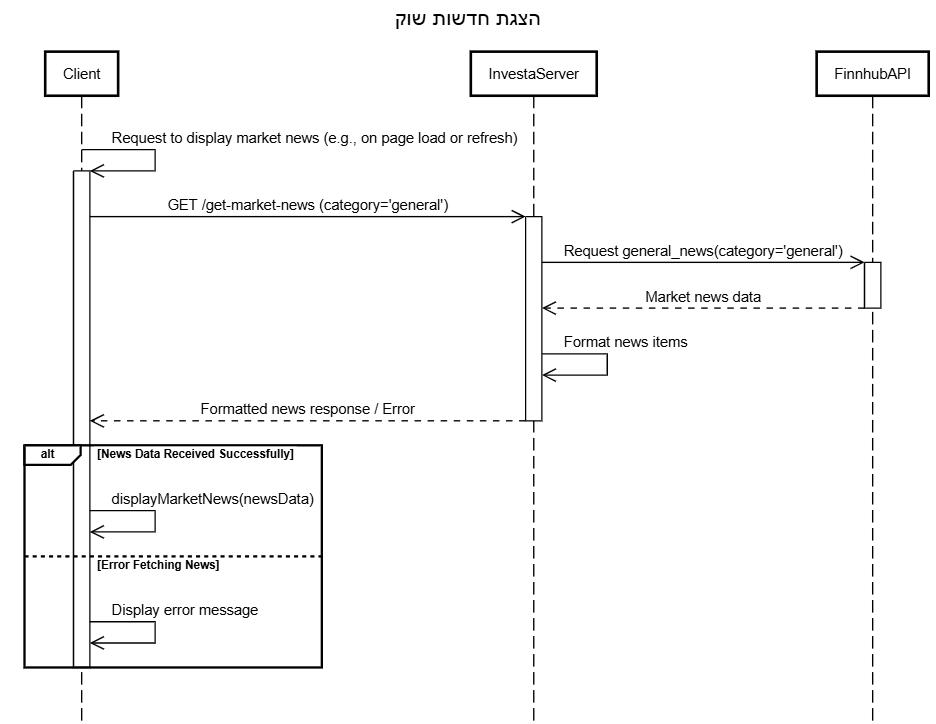
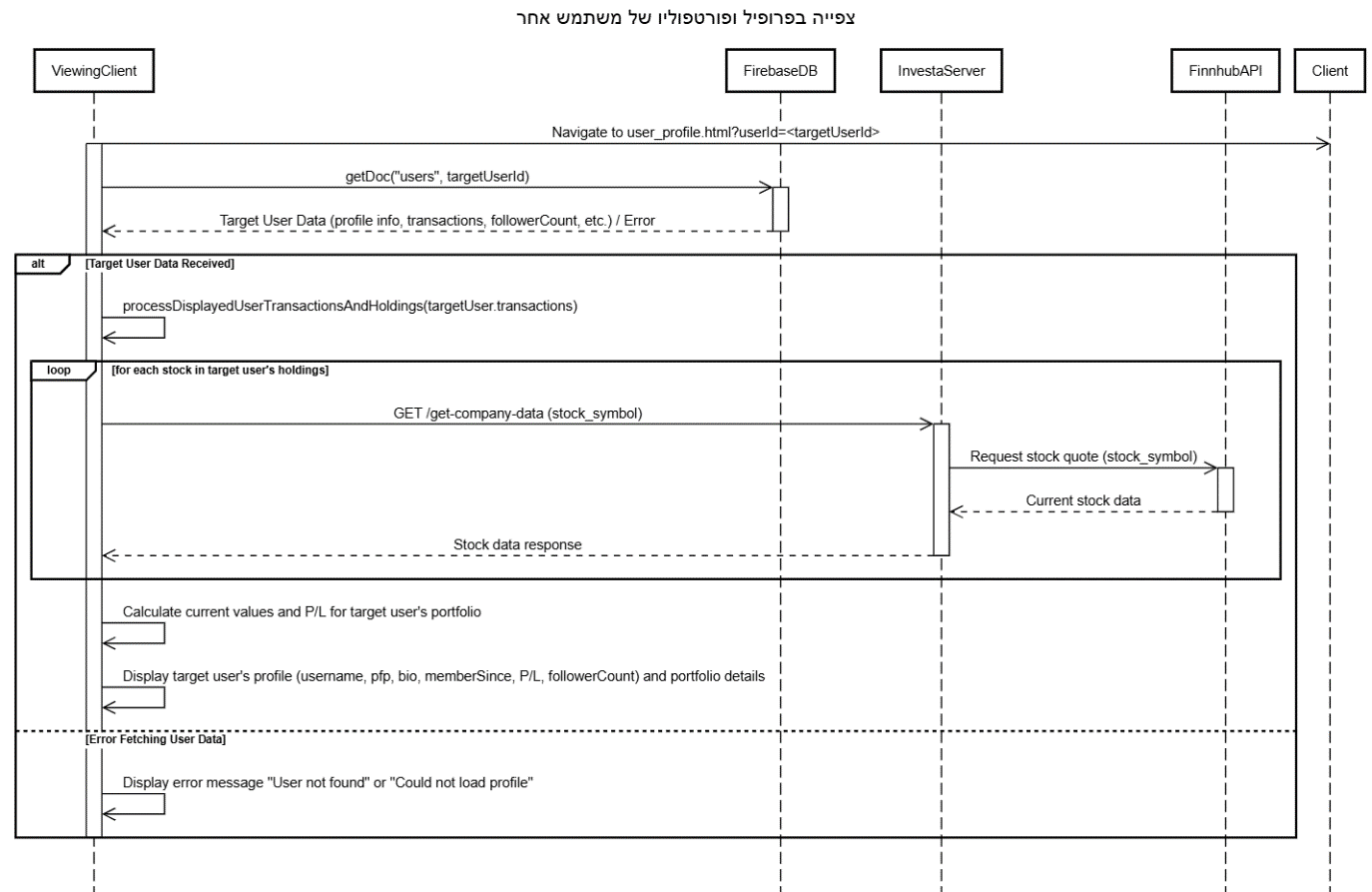












## תיאור האלגוריתמים מרכזיים בפרויקט

### אימות וניהול משתמשים

* **ניסוח הבעיה:** יש צורך לאמת משתמשים חדשים וקיימים בצורה מאובטחת, ולנהל את פרטי הפרופיל והפעילות שלהם במערכת. זה כולל בדיקת ייחודיות שם המשתמש, בדיקה אם המשתמש קיים, הצפנת סיסמאות, ושמירה מאובטחת של נתוני משתמש
* **פתרונות קיימים:**
  + פיתוח מערכת אימות עצמאית: פתרון זה דורש ידע מעמיק באבטחת מידע ותחזוקה שוטפת.
  + שימוש בשירותים צד שלישי: פתרון זה קל ליישום וגם השרת של מקורות אחרים הוא מאובטח יותר.
* **הפתרון שנבחר ונימוק הבחירה:**
  + אני בחרתי להשתמש בfirebase מכיוון שהוא בטוח וקל למימוש. כמו כן הבחירה בfirebase חוסכת לי זמן של פיתוח מערכת משל עצמי וצורך להצפין מכיוון שהסיסמאות נשמרות בfirebase בצורה מאובטחת.

### אחזור ועיבוד נתוני מניות

* **ניסוח הבעיה:** המערכת צריכה לאחזר נתוני מניות עדכניים ממקורות חיצוניים, לעבד אותם ולהציג אותם באופן קריא ורלוונטי למשתמש.
* **פתרונות קיימים:**
  + שימוש בAPI: פתרון קל ונוח אך עם מגבלה שאי אפשר לעשות אין סוף קריאות כי אתה מוגבל בשימוש(לפחות בחינמי).
  + שימוש בWeb scrapping: איסוף נתונים ישירות מאתרי אינטרנט. פתרון מורכב, לא יציב (תלוי במבנה האתר), ולוקח יותר זמן.
* **הפתרון שנבחר ונימוק הבחירה:**
  + אני בחרתי להשתמש בfinnhub API מכיוון שהוא הכיל את כל הנתונים שאני צריך והמגבלה שלו היא יחסית טובה לגרסה החינמית.

### ניהול תקשורת בזמן אמת (צ'אט)

* **ניסוח הבעיה:** יש צורך לאפשר שליחה וקבלת הודעות בין משתמשים בזמן אמת, תוך שמירה על חיבורים פתוחים ויכולת שידור הודעות לכלל המשתתפים. הבעיה כוללת ניהול חיבורים מרובים, טיפול בהודעות נכנסות ויוצאות, וטיפול בניתוקים.
* **פתרונות קיימים:**
  + להשתמש בשירות צ'אט קיים: אפשר למצוא שירות קיים ולהתממשק אליו בעזרת API, אבל דרך זו יותר מסובכת ואתה תלוי בשירות.
  + פיתוח מערכת צ'אט: ניתן לפתח מערכת צ'אט ביחסית פשטות בעזרת websockets בpython.
* **הפתרון שנבחר ונימוק הבחירה:**
  + בחרתי להשתמש בwebsockets מכיוון שזו אופציה מאוד נוחה ואמינה לפיתוח המערכת.

## תיאור סביבת הפיתוח

אני השתמשתי בvs code על מנת לפתח את התוכנה.

## תיאור פרוטוקול התקשורת

מערכת הצ'אט שלך עושה שימוש בפרוטוקול WebSockets, המאפשר תקשורת דו-כיוונית רציפה בזמן אמת בין הלקוח לשרת.

#### מבנה ההודעות הזורמות במערכת הצ'אט

כל הודעה הנשלחת בין הלקוח לשרת בצ'אט היא אובייקט JSON המכיל שדות ספציפיים המגדירים את סוג ההודעה, את תוכנה ואת השולח/הנמען.

#### הודעת רישום

שם ההודעה: register

נשלחת מ: **לקוח,** נשלחת אל: **שרת**

מבנה השדות בהודעה:

{

“type": "register",

“username”: “username”

}

#### הודעת צ'אט

שם ההודעה: chat

נשלחת מ: **לקוח,** נשלחת אל: **שרת**

מבנה השדות בהודעה:

{

“type”: “chat”,

“text”: “What do you think about $TSLL”

}

#### הודעת שידור של צ'אט

שם ההודעה: chat

נשלחת מ: **שרת,** נשלחת אל: **לקוח**

מבנה השדות בהודעה:

{

“type”: “chat”,

“username”: “username”,

“text”: “What do you think about $TSLL”,

“timestamp”: timestamp

}

#### הודעת סטטוס

שם ההודעה: status

נשלחת מ: **שרת,** נשלחת אל: **לקוח**

מבנה השדות בהודעה:

{

“type”: “status”,

“message”: “username has joined the chat!”

}

#### הודעת שגיאה

שם ההודעה: error

נשלחת מ: **שרת,** נשלחת אל: **לקוח**

מבנה השדות בהודעה:

{

“type”: “error”,

“message”: “Invalid JSON format.”

}

#### אופן הפעולה

1. כאשר משתמש מתחבר למערכת הוא שולח הודעת רישום אל השרת עם שם המשתמש שלו.

2. השרת מקבל את הודעת הרישום, רושם את הלקוח ושם המשתמש שלו, ומשדר הודעת סטטוס לכל המחוברים על הצטרפות המשתמש החדש.

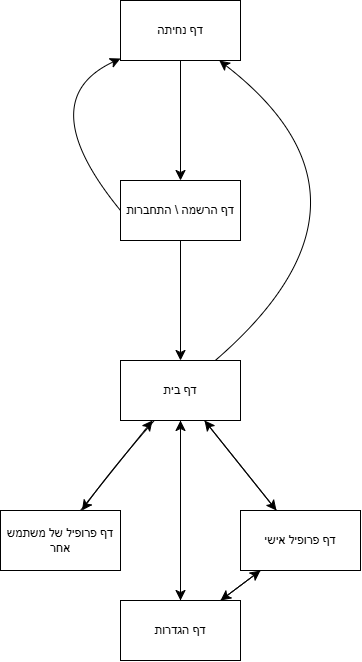
3. כאשר משתמש מקליד ושולח הודעה בצ'אט, הוא שולח הודעת צ'אט אל השרת דרך חיבור הwebsocket.

4. השרת מקבל את הודעת הצ'אט, מוסיף לה את שם המשתמש המקורי ואת חותמת הזמן, ומשדר את ההודעה לכל הלקוחות המחוברים.

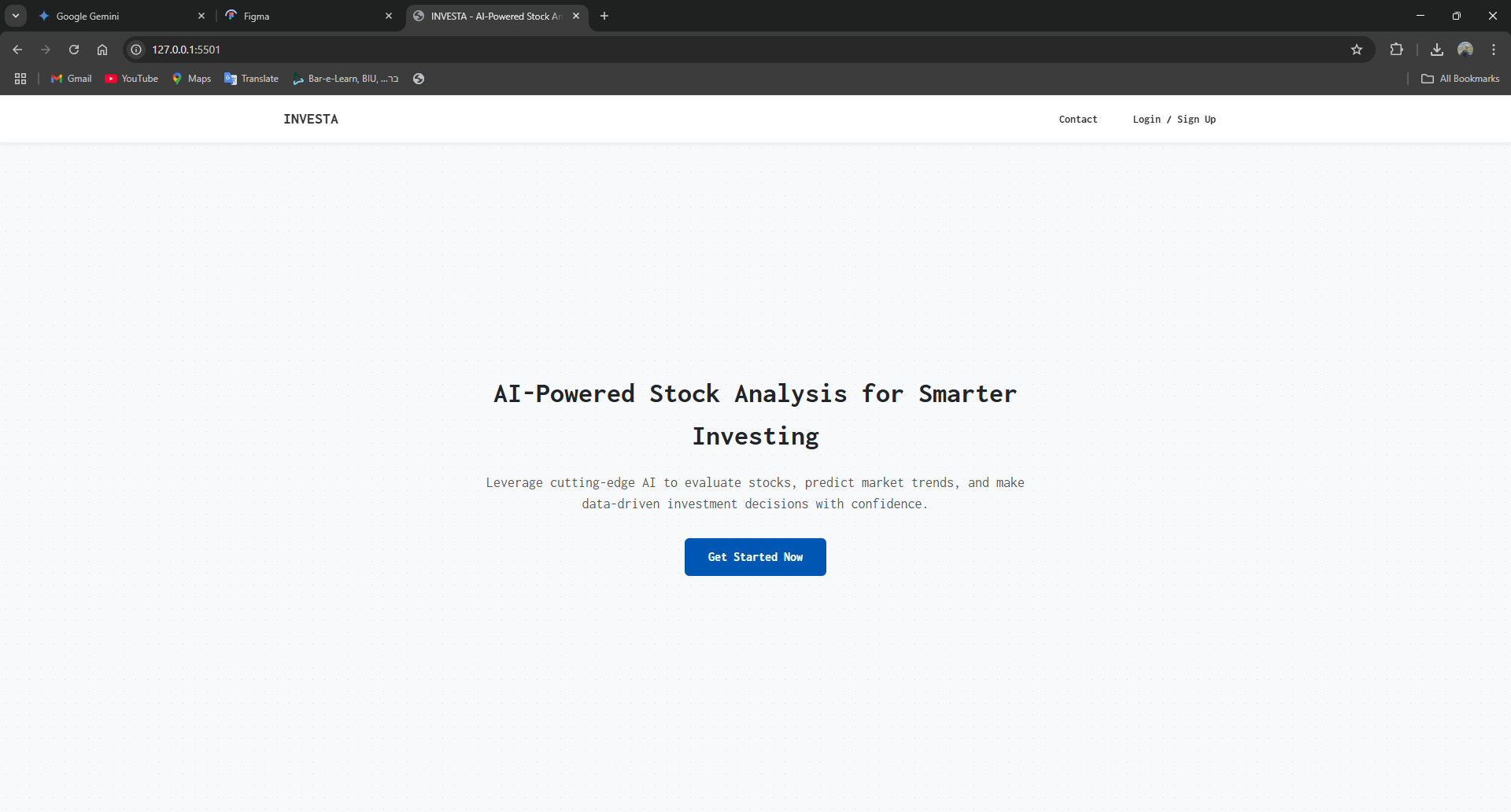
5. כל לקוח שמקבל הודעת צ'אט או סטטוס מהשרת מציג אותה בממשק המשתמש של הצ'אט.

## תיאור מסכי המערכת

### תרשים המסכים

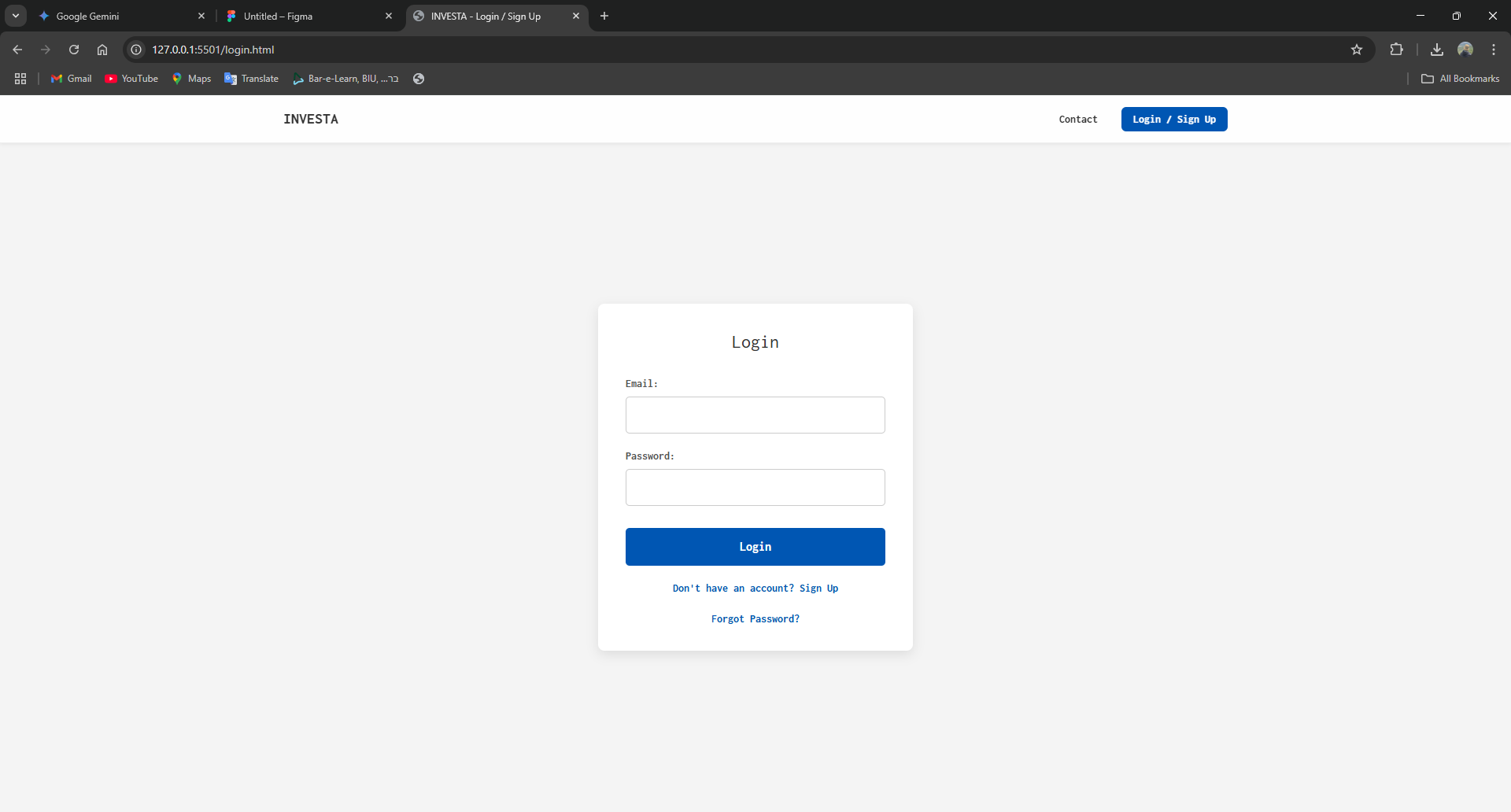


### מסך נחיתה



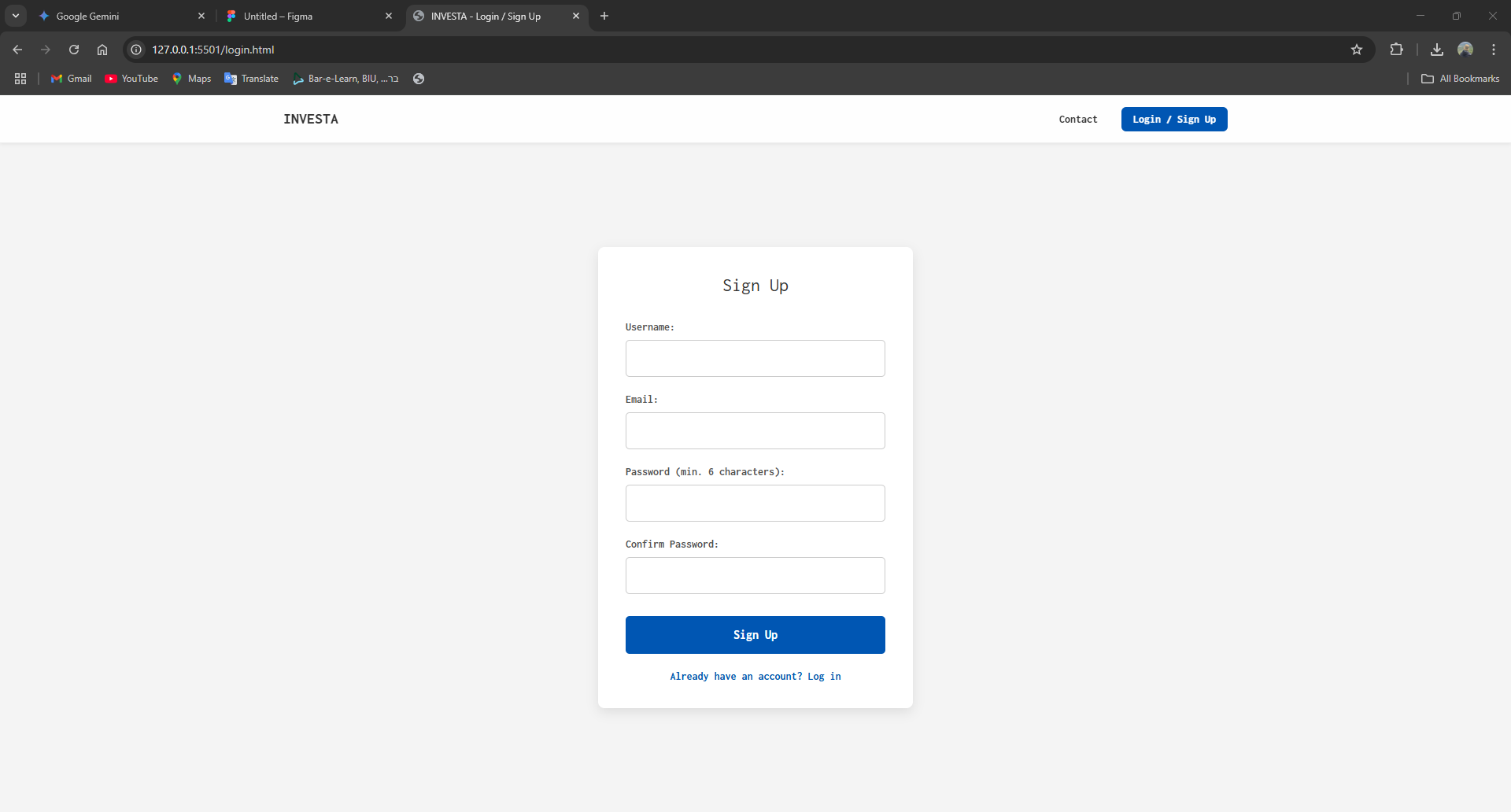
המסך מציג מידע בסיסי ממש על התוכנה וכן כפתור התחברות למערכת.

### מסך התחברות



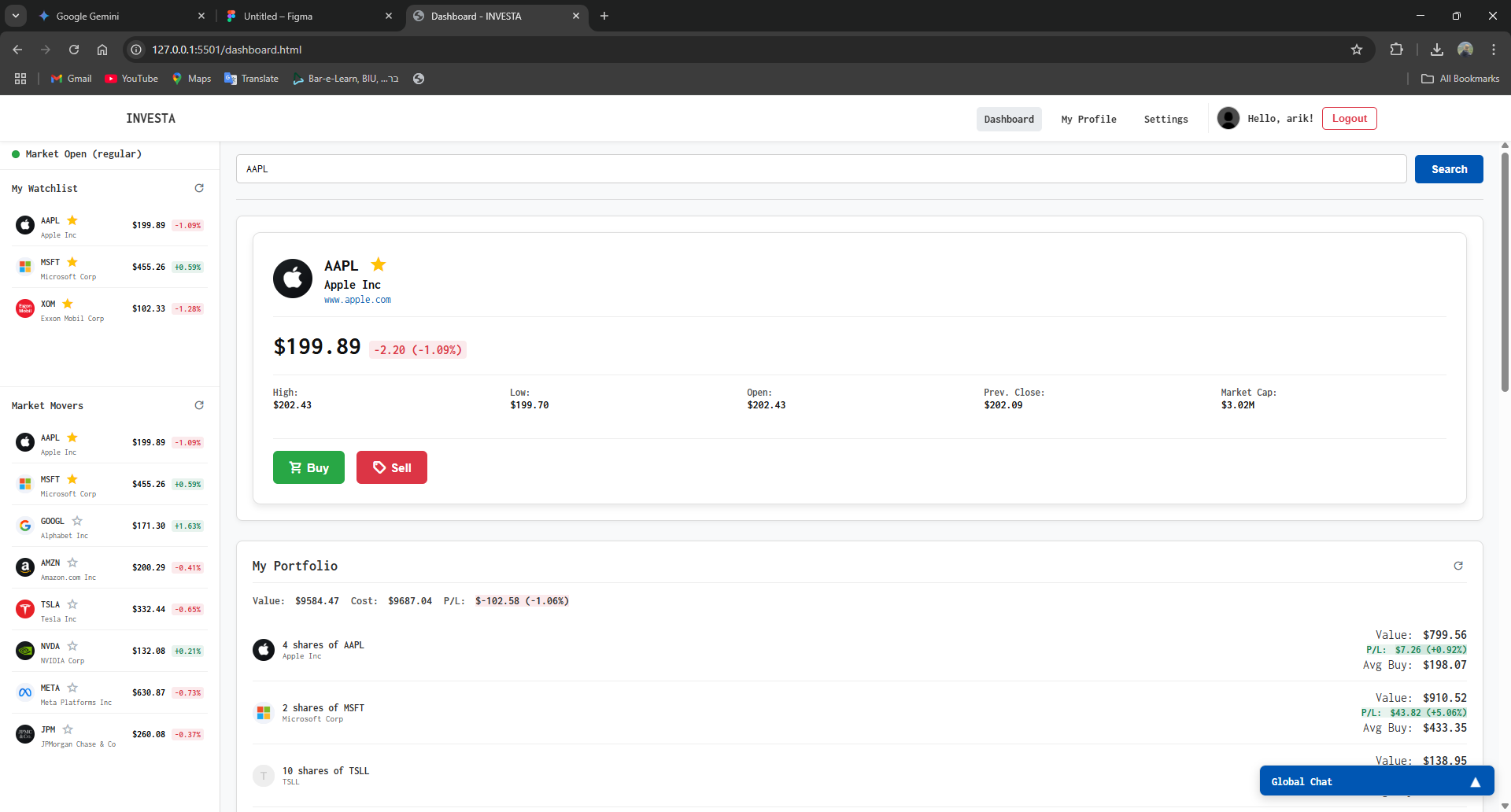
אפשרות למעבר בין התחברות להרשמה, שדות התחברות הכוללים: מייל וסיסמא.

### מסך הרשמה



אפשרות למעבר בין התחברות להרשמה, שדות הרשמה הכוללים: שם, מייל וסיסמא.

### מסך בית



המסך הוא המסך העיקרי של הפרויקט ממנו ניתן לבצע הכל. במסך מופיעים:

מצב השוק: פתוח/סגור.

רשימת המניות אחריהן המשתמש עוקב ואפשרות להסיר אותן.

רשימת מניות פופולריות וכפתור הוספה לרשימת המעקב.

חיפוש מניה אפשרות לקניה\מכירה ומידע עליה.

תיק ההשקעות שלי + מצב רווח\הפסד ופירוט על כל מניה בה מחזיק.

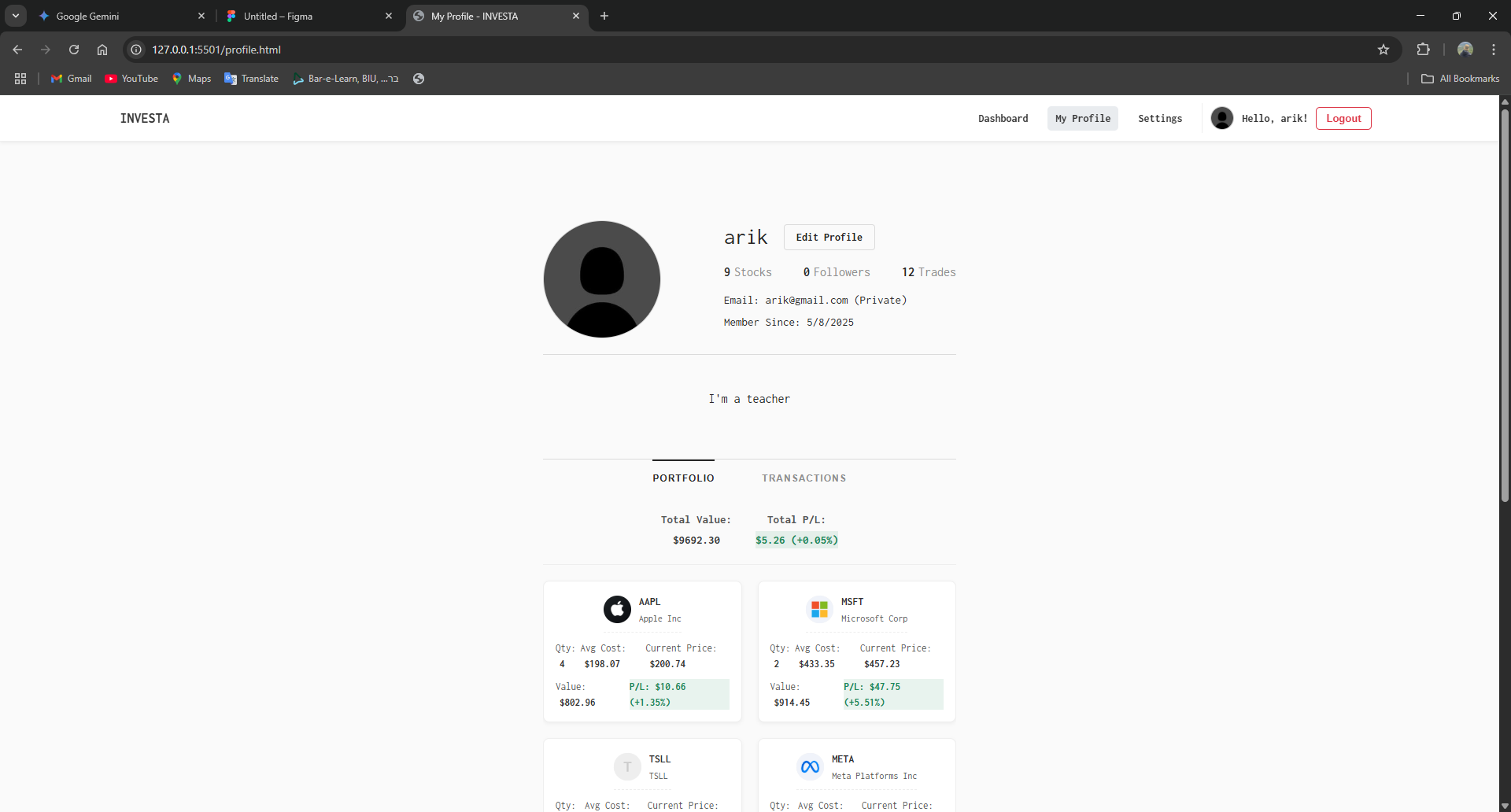
צ'אט גלובלי בין כל חברי הפלטפורמה.

חדשות שוק עדכניות.

משתמשים אחרים.

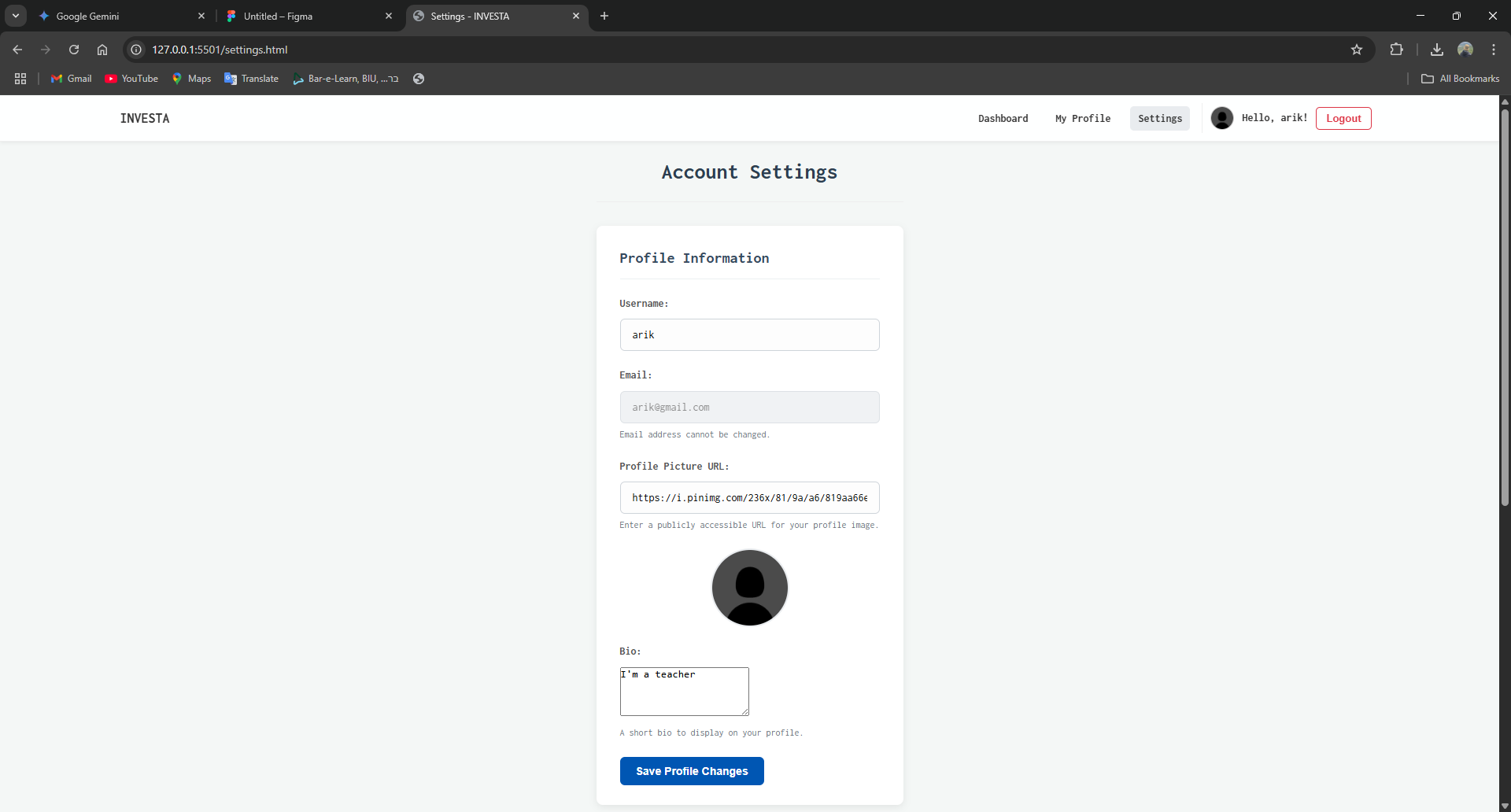
כפתורים לניווט בין – הפרופיל שלי, הגדרות וכפתור יציאה.

### מסך פרופיל אישי



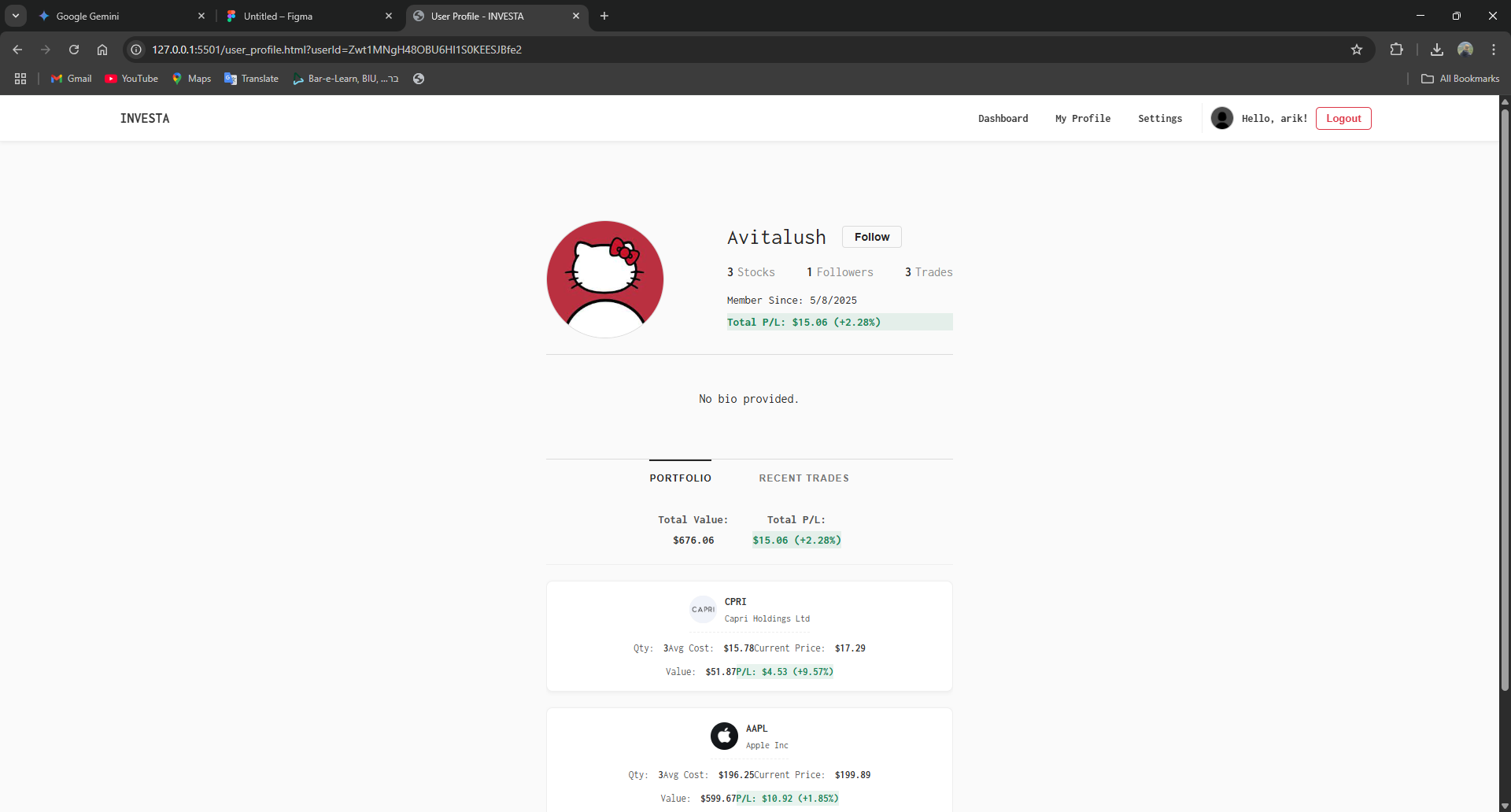
במסך זה ניתן לצפות בפרופיל שלך כפי שאחרים רואים אותו וכמו כן ניתן לערוך את הפרופיל. מה שרואים יפורט בהמשך במסך הפרופיל של משתמש אחר.

### מסך הגדרות



במסך זה ניתן לערוך את השם משתמש, תמונת פרופיל, ביוגרפיה, לשנות סיסמא ולמחוק את החשבון.

### מסך פרופיל (של משתמש אחר)



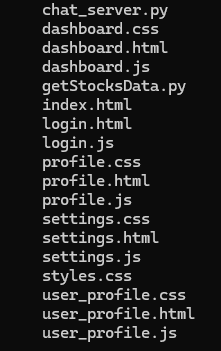
במסך זה ניתן לראות את תמונת לראות את הפרופיל של משתמש אחר וכן את מצב הרווח\הפסד שלו, מניות שהמשתמש מחזיק בהן, היסטוריית קניות/מכירות, שווי תיק וכו'.

## תיאור מבני הנתונים

בfirebase המידע נשמר כך:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| timestamp | | createdAt | user\_uid (String) | **users** |
| string | | email |
| number | | followerCount |
| user\_uid(string) | array | following |
| timestamp | | lastPlUpdate |
| string | | originalUsername |
| string | | bio |
| string | | profilePictureUrl |
| number | | profitLossAmount |
| number | | profitLossPercentage |
| symbol(string) | array | savedStocks |
| price(number) | array | transactions |
| quantity(number) |
| symbol(string) |
| timestamp(timestamp) |
| type(string) |
| string | | username |

מבנה הקבצים: הכל תחת תייקיה אחת

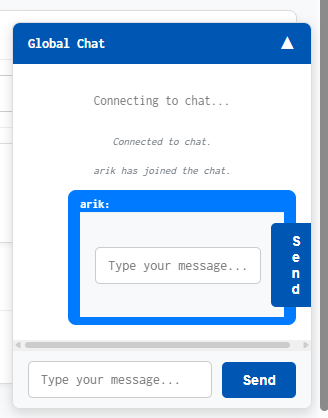


## סקירת החולשות

### שכבת האפליקציה

* + - 1. Cross-site scripting

המשמעות של cross-site scripting היא שמשתמשים אחרים יכולים להכניס תכונות זדוניות דרך האתר עצמו. איך זה מתבטא בתוכנה שלי? לדגומה משתמש יכול להכניס קוד בhtml דרך הצאט. כמו בדוגמה שעשיתי כאן:



ניתן לראות בתמונה שהכנסתי קוד דרך ההודעה בצ'אט. בדרך זו יכולים משתמשים להכניס תוכנות זדוניות.

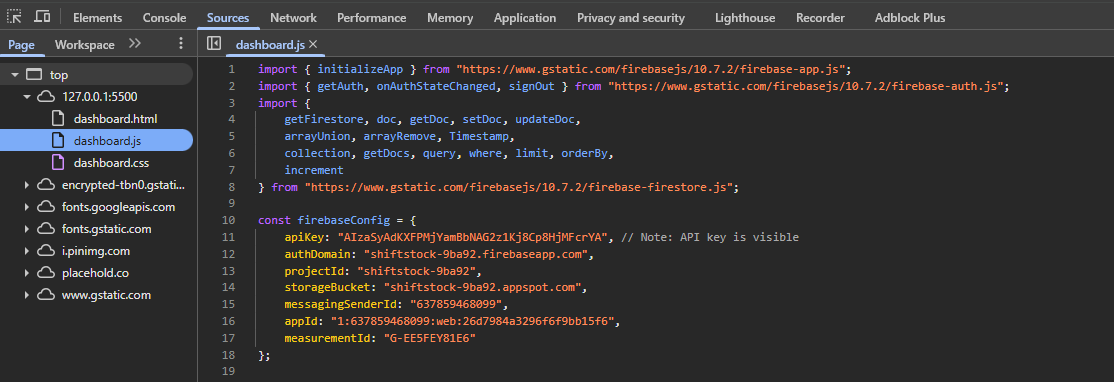
* + - 1. Denial of Service (DoS/DDoS)  
         ההתקפה זאת שולחת משתמשים רבים אל האתר, מציפה את השרתים ובכך מונעת מתן שירות לכלל המשתמשים. השרתים שלי מוגבלים מאוד. מכיוון שאני משתמש בAPI חינמי למידע על המניות, כמות הבקשות שאני יכול לשלוח לFinnhub מוגבלת מאוד, ולכן אם מספר רב של משתמשים יכנסו אל האתר, האתר יקרוס ולא יהיה שירות.
      2. Men In The Middle (MITM)

בהתקפה זו מחשב עומד באמצע מסניף את התקשורת אליו ומתנהג כמו השרת אל הלקוח, האדם הזה שולח את מה שמקבל אל השרת כמו לקוח ואז מחזיר את התגובה שמקבל אל הלקוח. בצורה זו אדם יכול לקרוא את כל התקשורת שבין האדם לשרת ובגלל שהסיסמא לא מוצפנת כאשר היא נשלחת אל firbaseAuth אז מישהו יכול לגנוב סיסמאות.

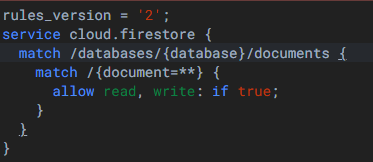
### שכבת התעבורה

API של Firebase נגיש לכולם

בגלל שהAPI KEY של הFIREBASE שמור כמשתנה בתוך הscript עצמו כל אחד ניתן לגשת אליו על ידי בדיקה בסיסית של הקוד בדפדפן כך:



ובגלל החוקים בשרת הfirebase שלי שמאפשרים תקשורת מכל מקום, אפשר לגשת אליהם מכל תוכנה ובכך לגנוב מידע וכו'.



**מימוש הפרויקט**

# מימוש הפרויקט

## סקירת המודלים

### מודלים בpython

|  |  |
| --- | --- |
| שם מודל | תפקיד |
| **asyncio** | לתכנות אסינכרוני בפיתון, נצרך בשביל לעשות כמה פעולות במקביל. לדגומה, חיבור של כמה משתמשים בצ'אט. |
| **websockets** | משמש ליצירת שרת ולקוח לתקשורת בזמן אמת בצ'אט. |
| **json** | ספרייה שמאפשרת לקרוא json וליצור מחרוזות. משמש בצ'אט לכתיבת הודעות. |
| **logging** | לכתיבת אזהרות ושגיאות, משמש בשביל להבין מה קרה אם השרת נפל או משהו. |
| **finnhub** | כך אני מקבל מידע על מניות בpython. |
| **uvicorn** | משמש להרצת שרתים. בקוד של היא מריץ את הapi |
| **fastAPI** | משמש ליצירת תבנית של api. כך אני יוצר את התבנית לבקשות. |

### מודלים בJS

|  |  |
| --- | --- |
| שם מודל | תפקיד |
| **firebase-app** | משמש בשביל לאתחל את האפליקציה של firebase בתוכנה שלנו. |
| **firebase-auth** | משמש בשביל לתקשר עם הauthentication של firebase |
| **firebase-firestore** | משמש בשביל לתקשר עם הDB של firebase |

### מודלים\מחלקות שפותחו

**מחלקת ChatServer בchat\_sever.py**

**תפקיד המחלקה:** ניהול של כל המשתמשים המחוברים לצ'אט ושידור הודעות.

**תכונות המחלקה:**

* connected\_clients – מכיל את כל האוביקטיים של הwebsockets המחוברים לצ'אט.
* client\_usernames – מכיל רשימה של שמות כל המשתמשים המחוברים כעת.

**פעולות במחלקה:**

* \_\_init\_\_
* **טענת כניסה:** אין
* **טענת יציאה:** מאתחלת את כל המשתנים במחלקה.
* register
* **טענת כניסה:** אובייקט הwebsocket של המשתמש היוצא.
* **טענת יציאה:** מקשר בין אובייקט הwebsocket לשם המשתמש, מוסיף אותו אל רשימת המחוברים, משדר הודעה לכל המשתמשים שהמשתמש הצטרף לצ'אט.
* unregister
* **טענת כניסה:** מקבלת את שם המשתמש ואת אובייקט הwebsocket שלו.
* **טענת יציאה:** מוריד אותו מרשימת המחוברים, משדר הודעה לכל המשתמשים שהמשתמש התנתק מהצ'אט.
* broadcast
* **טענת כניסה:** מקבלת את ההודעה של המשתמש.
* **טענת יציאה:** משדרת את ההודעה לכולם.
* handler
* **טענת כניסה:** את אובייקט הwebsocket שלו.
* **טענת יציאה:** מנהלת את החיבור הראשוני לשרת, וכמו כן את ההודעות שנשלחות ובעיות בצ'אט.

### קבצי JavaScript

* **login.js** – מנהל את תהליכי ההרשמה וההתחברות לתוכנה אל מול firebase.
* **dashboard.js** – מנהל את כל הדף הראשי: חישוב רווח\הפסד, כל מה שקשור לגבי חיפוש והצגת מניות, הצגת חדשות השוק, הצ'אט משתמשים אחרים וכו'.
* **profile.js** – מנהל את דף הפרופיל האישי מציג את התיק ואת היסטוריית הרכישות.
* **settings.js** – אחראי על שינויי הפרטים של המשתמש.
* **user\_profile.js** – אחראי על הצגת פרופיל משתמש אחר.

## קטעי קוד ממרכיבים מרכזיים

#### אימות וניהול משתמשים

ביכולת זו מתבצעת בדיקה של ייחודיות שם המשתמש, של תקינות הפרטים ולאחר מכן רישום של המשתמש בfirebase ויצרת המשתמש בfirestore.

// From login.js

async function isUsernameUnique(username) {

if (!db) return false;

const lowerCaseUsername = username.toLowerCase();

try {

const usersRef = collection(db, "users");

const q = query(usersRef, where("username", "==", lowerCaseUsername));

const querySnapshot = await getDocs(q);

return querySnapshot.empty;

} catch (error) {

displayMessage("Error checking username availability. Please try again.", 'error');

return false;

}

}

function generateProfilePictureUrl(username) {

const initial = username ? username.charAt(0).toUpperCase() : '?';

const size = 30;

const bgColor = 'eee';

const textColor = '333';

return `https://placehold.co/${size}x${size}/${bgColor}/${textColor}?text=${initial}`;

}

if (auth && db) {

signupForm.addEventListener("submit", async (event) => {

event.preventDefault();

clearMessages();

showLoader();

isProcessingAuth = true;

const usernameInput = document.getElementById("signup-username");

const username = usernameInput.value.trim();

const email = document.getElementById("signup-email").value.trim();

const password = document.getElementById("signup-password").value;

const passwordConfirmation = document.getElementById("signup-confirm-password").value;

if (!username || !email || !password || !passwordConfirmation || password !== passwordConfirmation || password.length < 6 || looksLikeEmail(username)) {

if (!username || !email || !password || !passwordConfirmation) displayMessage("Please fill in all required fields.", 'error');

else if (password !== passwordConfirmation) displayMessage("Passwords do not match.", 'error');

else if (password.length < 6) displayMessage("Password must be at least 6 characters long.", 'error');

else if (looksLikeEmail(username)) displayMessage("Please enter a valid username, not an email address.", 'error');

hideLoader();

isProcessingAuth = false;

return;

}

try {

const isUnique = await isUsernameUnique(username);

if (!isUnique) {

displayMessage(`Username "${username}" is already taken. Please choose another.`, 'error');

hideLoader();

isProcessingAuth = false;

return;

}

const userCredential = await createUserWithEmailAndPassword(auth, email, password);

const user = userCredential.user;

const profilePictureUrl = generateProfilePictureUrl(username);

const userData = {

username: username.toLowerCase(),

originalUsername: username,

email: user.email,

profilePictureUrl: profilePictureUrl,

createdAt: new Date(),

savedStocks: [],

transactions: [],

following: [],

profitLossAmount: 0,

profitLossPercentage: 0,

followerCount: 0

};

const userDocRef = doc(db, "users", user.uid);

await setDoc(userDocRef, userData);

isProcessingAuth = false;

window.location.href = "dashboard.html";

} catch (error) {

const friendlyMessage = getFirebaseErrorMessage(error.code);

displayMessage(friendlyMessage || `Sign-up failed: ${error.message}`, 'error');

hideLoader();

isProcessingAuth = false;

}

});

}

#### אחזור ועיבוד נתוני מניות

קטע קוד זה דואג לקחת את כל הנתונים הנדרשים מfinnhub ליצור עבורם API ייחודי ולהריץ אותו על שרת פרטי אליו יכול לשלוח קריאות הלקוח.

# From getStocksData.py

import finnhub

import uvicorn

from fastapi import FastAPI, HTTPException

from fastapi.middleware.cors import CORSMiddleware

app = FastAPI()

app.add\_middleware(

CORSMiddleware,

allow\_origins=["\*"],

allow\_credentials=True,

allow\_methods=["\*"],

allow\_headers=["\*"],

)

finnhub\_client = finnhub.Client(api\_key=" cvijij9r01qks9qat59gcvijij9r01qks9qat5a0")

@app.get("/get-company-data")

async def get\_company\_data(symbol: str):

try:

profile = finnhub\_client.company\_profile2(symbol=symbol)

quote = finnhub\_client.quote(symbol)

if not profile and not quote:

raise HTTPException(status\_code=404, detail=f"Data not found for symbol: {symbol}")

combined\_data = {}

if profile:

combined\_data.update(profile)

if quote:

combined\_data.update(quote)

if 'ticker' not in combined\_data and symbol:

combined\_data['ticker'] = symbol.upper()

if 'ticker' not in combined\_data:

raise HTTPException(status\_code=404, detail=f"Could not retrieve data for symbol: {symbol}")

return combined\_data

except Exception as e:

raise HTTPException(status\_code=500, detail=f"Failed to fetch data for {symbol}: {str(e)}")

if \_\_name\_\_ == "\_\_main\_\_":

uvicorn.run("getStocksData:app", host="127.0.0.1", port=8000, reload=True)

#### ניהול תקשורת בזמן אמת (צ'אט)

**א. צד שרת**

קטע קוד זה יוצר שרת ומריץ אותו על שרת פרטי ומאפשר תקשורת במקביל.

# From chat\_server.py

import asyncio

import websockets

import json

import logging

logging.basicConfig(level=logging.INFO)

logger = logging.getLogger(\_\_name\_\_)

class ChatServer:

def \_\_init\_\_(self):

self.connected\_clients = set()

self.client\_usernames = {}

async def register(self, websocket, username):

self.connected\_clients.add(websocket)

self.client\_usernames[websocket] = username

logger.info(f"Client {websocket.remote\_address} registered as {username}. Total connected: {len(self.connected\_clients)}")

await self.broadcast(json.dumps({"type": "status", "message": f"{username} has joined the chat."}))

async def unregister(self, websocket):

if websocket in self.connected\_clients:

username = self.client\_usernames.get(websocket, "Unknown User")

self.connected\_clients.remove(websocket)

if websocket in self.client\_usernames:

del self.client\_usernames[websocket]

logger.info(f"Client {websocket.remote\_address} ({username}) disconnected. Total connected: {len(self.connected\_clients)}")

await self.broadcast(json.dumps({"type": "status", "message": f"{username} has left the chat."}))

async def broadcast(self, message):

tasks = [asyncio.create\_task(client.send(message)) for client in self.connected\_clients]

if tasks:

await asyncio.wait(tasks)

async def handler(self, websocket):

username = None

try:

username\_message = await websocket.recv()

try:

data = json.loads(username\_message)

if data.get("type") == "register" and data.get("username"):

username = data["username"]

await self.register(websocket, username)

else:

await websocket.send(json.dumps({"type": "error", "message": "Please send a registration message with your username first."}))

return

except json.JSONDecodeError:

await websocket.send(json.dumps({"type": "error", "message": "Invalid JSON format for registration."}))

return

async for message in websocket:

try:

data = json.loads(message);

if data.get("type") == "chat" and data.get("text"):

chat\_message = {

"type": "chat",

"username": username,

"text": data["text"],

"timestamp": asyncio.get\_event\_loop().time()

}

await self.broadcast(json.dumps(chat\_message))

else:

await websocket.send(json.dumps({"type": "error", "message": "Invalid message format."}))

except json.JSONDecodeError:

await websocket.send(json.dumps({"type": "error", "message": "Invalid JSON format."}))

except websockets.exceptions.ConnectionClosedOK:

logger.info(f"Connection closed normally by {websocket.remote\_address}")

except websockets.exceptions.ConnectionClosedError as e:

logger.error(f"Connection closed with error by {websocket.remote\_address}: {e}")

except Exception as e:

logger.error(f"Unexpected error in handler for {websocket.remote\_address}: {e}")

finally:

await self.unregister(websocket)

async def main():

server = ChatServer()

port = 8822

logger.info(f"WebSocket server starting on ws://127.0.0.1:{port}")

async with websockets.serve(server.handler, "127.0.0.1", port):

await asyncio.Future()

if \_\_name\_\_ == "\_\_main\_\_":

try:

asyncio.run(main())

except KeyboardInterrupt:

logger.info("Server stopped by user.")

except Exception as e:

logger.critical(f"Server crashed: {e}")

**ב. צד לקוח**

קטע קוד זה דואג להקמת התקשורת מצד הלקוח אל השרת, מטפל בבעיות של השרת, מטפל בהצגת ההודעות ושליחתן.

// From dashboard.js

let websocket = null;

let chatUsername = "Guest";

function connectWebSocket() {

if (websocket && websocket.readyState === WebSocket.OPEN) {

return;

}

websocket = new WebSocket(WEBSOCKET\_URL);

websocket.onopen = () => {

const username = currentUser?.displayName || currentUser?.email || 'Guest';

chatUsername = username;

websocket.send(JSON.stringify({ type: "register", username: username }));

displayChatMessage({ type: "status", message: "Connected to chat." });

if (chatMessagesContainer.querySelector('.info-message') && chatMessagesContainer.children.length === 1) {

chatMessagesContainer.innerHTML = '';

}

};

websocket.onmessage = (event) => {

try {

const message = JSON.parse(event.data);

displayChatMessage(message);

} catch (error) {

}

};

websocket.onerror = (error) => {

displayChatMessage({ type: "status", message: "WebSocket error. Attempting to reconnect..." });

};

websocket.onclose = (event) => {

displayChatMessage({ type: "status", message: "Disconnected from chat. Reconnecting..." });

setTimeout(connectWebSocket, 5000);

};

}

function displayChatMessage(message) {

if (!chatMessagesContainer) return;

const messageElement = document.createElement('div');

messageElement.classList.add('chat-message');

if (message.type === "chat") {

const isSent = message.username === chatUsername;

messageElement.classList.add(isSent ? 'sent' : 'received');

messageElement.innerHTML = `

<strong>${message.username}:</strong>

<span>${message.text}</span>

`;

} else if (message.type === "status") {

messageElement.classList.add('status');

messageElement.textContent = message.message;

} else if (message.type === "error") {

messageElement.classList.add('status', 'error-message');

messageElement.textContent = `Error: ${message.message}`;

} else {

return;

}

chatMessagesContainer.appendChild(messageElement);

chatMessagesContainer.scrollTop = chatMessagesContainer.scrollHeight;

}

function sendChatMessage() {

if (!websocket || websocket.readyState !== WebSocket.OPEN) {

showNotification("Chat is not connected.", "info");

return;

}

if (!chatInput || chatInput.value.trim() === "") {

return;

}

const messageText = chatInput.value.trim();

const chatMessage = {

type: "chat",

text: messageText

};

websocket.send(JSON.stringify(chatMessage));

chatInput.value = "";

}

if (sendChatButton) {

sendChatButton.addEventListener('click', sendChatMessage);

}

if (chatInput) {

chatInput.addEventListener('keypress', (event) => {

if (event.key === 'Enter') {

event.preventDefault();

sendChatMessage();

}

});

}

## פירוט הבדיקות

### תקינות מערכת ההרשמה וההתחברות

* מה הבדיקה בודקת: בדיקת תהליך ההרשמה וההתחברות למערכת.
* מה בוצע בפועל:

ניסיון הרשמה והתחברות עם פרטים תקינים

ניסיון התחברות עם סיסמא לא תקינה  
ניסיון הרשמה עם מייל קיים  
ניסיון הרשמה עם סיסמאות שונות ולא תיקנות

ניסיון התחברות קיים

* תוצאות הבדיקה:

הכל תקין

### תקינות חיפוש מניות

* מה הבדיקה בודקת: דיוק המידע המתקבל מהAPI של המניות.
* מה בוצע בפועל:

בדיקה של המידע בזמן אמת אל מול tradingView

* תוצאות הבדיקה:

הכל תקין

### תקינות מערכת הצ'אט

* מה הבדיקה בודקת: יכולת לשלוח ולקבל הודעות.
* מה בוצע בפועל:

שליחת הודעות בין שתי חשבונות

* תוצאות הבדיקה:

הכל תקין

### תקינות מערכת הפורטפוליו

* מה הבדיקה בודקת: יכולת להוסיף ולהסיר מניות מהפורטפוליו וחישוב נכון של רווח/הפסד
* מה בוצע בפועל:

קניית\מכירת מניות ומערקב אחריהן על מנת לבדוק אם חישוב הרווח\הפסד נכון

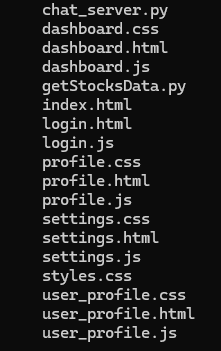
* תוצאות הבדיקה:

הכל תקין

**מדריך למשתמש**

# מדריך למשתמש

## פירוט כלל קבצי המערכת



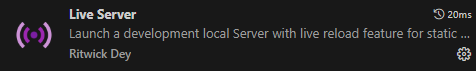
chat\_server.py – השרת שעליו רץ הצ'אט.

getStocksData.py – משמש בשביל להשיג את כל המידע על המניות.

שאר הקבצים אלו קבצי html, js, css פשוטים שעליהם רץ האתר.

## התקנת המערכת

הסביבה הנדרשת היא visual studio code וכן צריך להוריד את ההרחבה הנ"ל אליה:

  
בשביל הpython צריך להתקין את הספריות הבאות(ע"י pip install):

uvicorn, fastapi, finnhub-python, websockets, asyncio, logging

על כל הקבצים צריכים להיות בתיקייה אחת.

נתונים התחלתיים:

* אימייל: [arik@gmail.com](mailto:arik@gmail.com)
* סיסמא: abcd1234

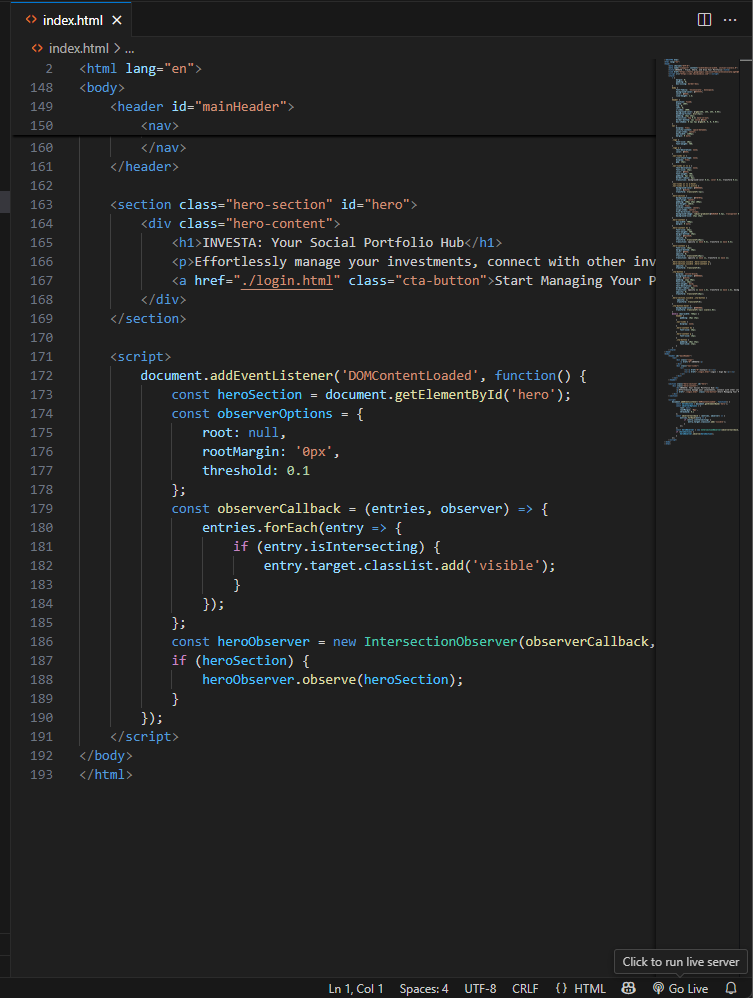
## אופן הפעלת המערכת

נתחיל מלפתוח את הcmd על התיקייה של התיק ונריץ את שני קבצי הפייתון ע"י הפקודות:

python chat\_server.py

python getStocksData.py

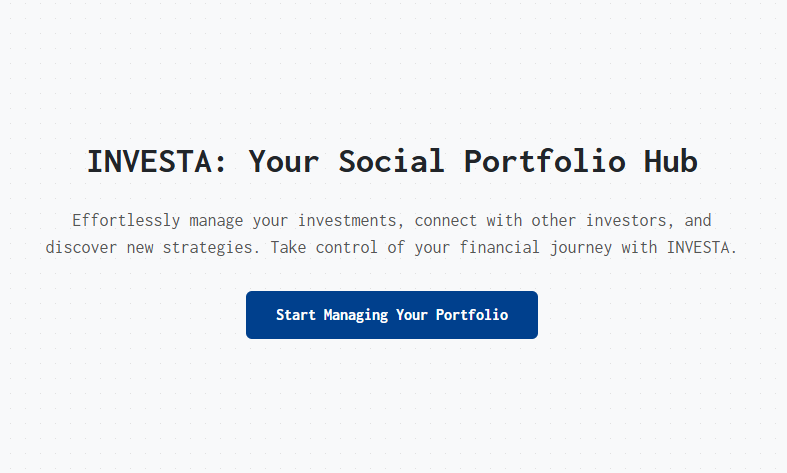
(כל אחד רץ בחלון שונה)



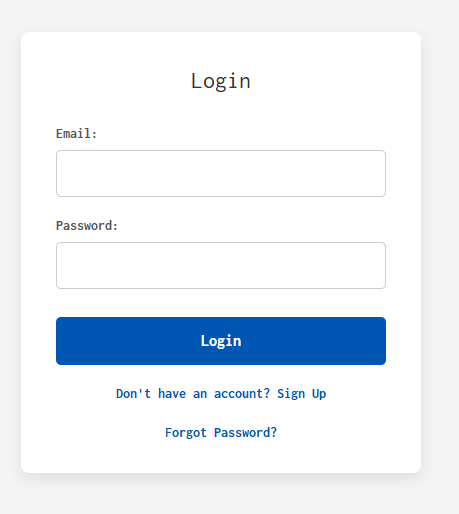
לאחר מכן, כאשר אנחנו על דף הindex.html בvisual studio code נלחץ על go live בפינה הימנית למטה:



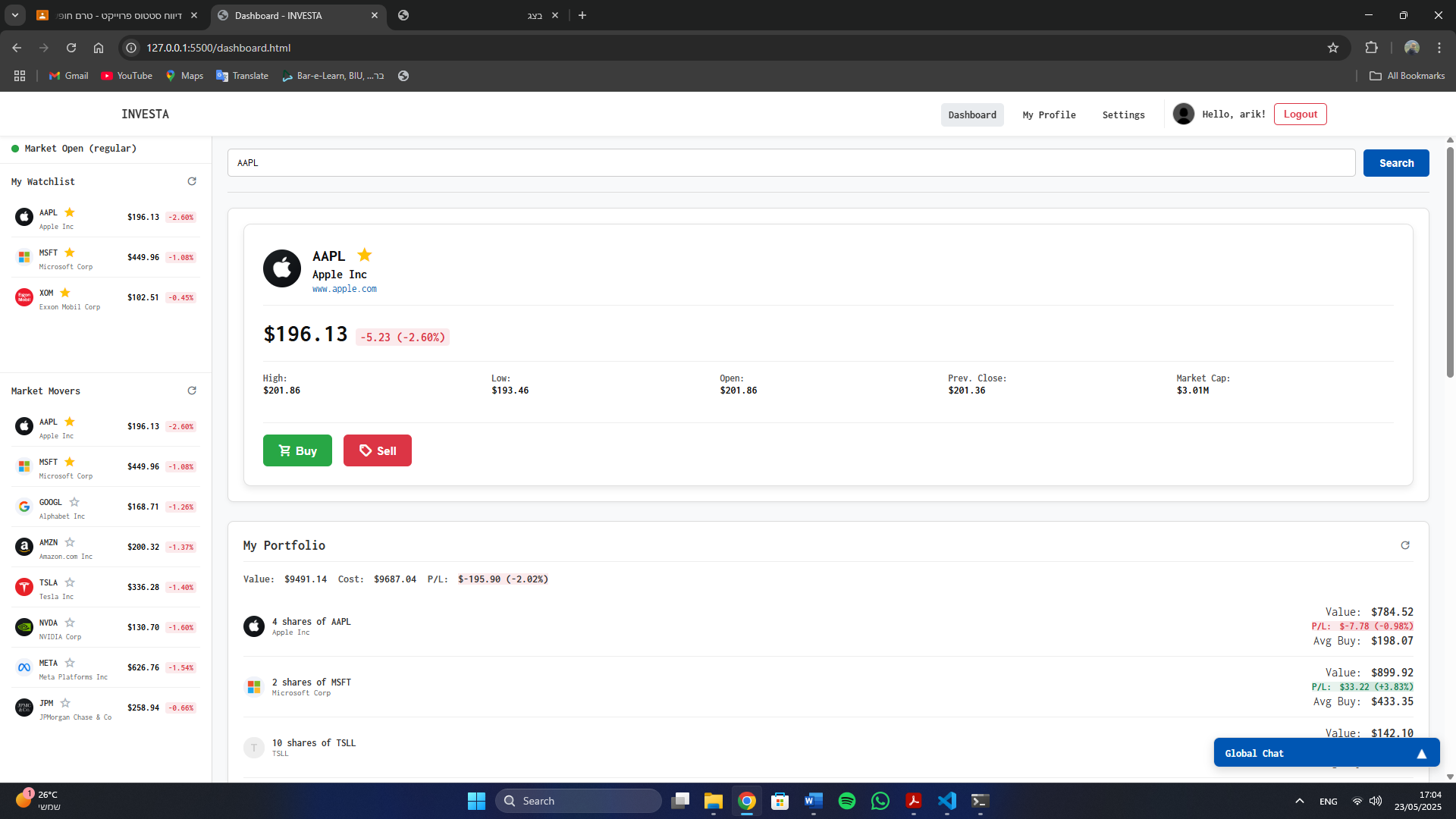
לאחר לחיצה יפתח לנו חלון בדפדפן על עמוד הנחיתה.



כאשר אנחנו במסך שכאן נוכל ללחוץ על start manging your portfolio או על login/signup.



במסך פה נכניס את שם המשתמש והסיסמא הנתונים.



כעת כשאנחנו בדף הבית נוכל לעשות מספר דברים:

* לחפש מנייה בעזרת שורת החיפוש
* לרכוש מניות
* להוסיף מניות לתיק
* להסתכל בחדשות
* לצפות במשתמשים אחרים
* לערוך פרופיל
* לצפות בפרופיל שלנו

כל הפועלות שניתן לעשות ניתן לראות במסכי הפרויקט שבעמודים 34-39.

**רפלקציה**

# רפלקציה

#### רפלקציה

העבודה על הפרויקט היה מסע מאתגר, מלמד ומספק עבורי. בהתחלה, לא רציתי לעשות בכלל את הפרויקט ולמורים לקח זמן לשכנע אותי שכן. הרעיון לשלב בין פלטפורמה חברתית לבין משקיעים עלה לי תוך כדי שאני חיפשתי מענה לצרכים אישיים.

האתגרים הראשונים שצפיתי, כמו מציאת מקור אמין לנתוני מניות בזמן אמת ויצירת ממשק משתמש ידידותי, אכן התבררו כמשמעותיים. ההחלטה להשתמש בfirebase לאימות משתמשים ובסיס נתונים, בfinnhub כמקור לנתוני מניות בזמן אמת ובwebsockets ליצירת הצ'אט, התבררה כנכונה והיא חסכה לי זמן רב בפיתוח.

אחת ההצלחות המרכזיות בפרויקט לדעתי היא הממשק משתמש שיצרתי מכיוון שהוא נראה מאוד ידידותי ומקצועי למשתמש. לראות את הצ'אט מתעורר לחיים, את נתוני המניות מתעדכנים, ואת תיקי ההשקעות האישיים היה רגע של סיפוק גדול.

לצד ההצלחות, היו גם לא מעט אתגרים. ניהול התקשורת בצ'אט דרש למידה מעמיקה של asyncio ו-websockets בpython. כמו כן היו גם אתגרים בחיים האישיים שכללו שילוב בין עבודה, כושר קרבי, מיונים לצה"ל, חיי חברה וכו'.

בראייה לאחור, ייתכן והייתי מקדיש יותר זמן בשלבים המוקדמים לתכנון של המערכת ולא משאיר את רוב הטיפול לשלבים מאוחרים יותר.

במידה והיו לי משאבים נוספים, הייתי שואף להרחיב את הפרויקט: לפתח אלגוריתמים לניתוח והמלצות מבוססות בינה מלאכותית, ואף להרחיב את היכולות החברתיות, כמו מתן התראות על שינוי וכו'.

#### תודות

אני רוצה להודות למנחות שלי, אירית סעדון ושרית שוורץ, על ההכוונה, התמיכה והסבלנות לאורך כל הדרך. רק בזכותן עשיתי את הפרויקט ואם לא הן הייתי מוותר עוד בשנה שעברה, רק הן דחפו אותי לבצע ועל כך אני מודה להן מאוד.

**ביבליוגרפיה**

# ביבליוגרפיה

Google. (n.d.). Firebase Authentication documentation. Firebase. Retrieved May 25, 2025, from <https://firebase.google.com/docs/auth>

Google. (n.d.). Cloud Firestore documentation. Firebase. Retrieved May 25, 2025, from <https://firebase.google.com/docs/firestore>

Finnhub. (n.d.). API documentation. Finnhub. Retrieved May 25, 2025, from <https://finnhub.io/docs/api>

Augustin, A. (n.d.). websockets: Library for building WebSocket servers and clients in Python. Read the Docs. Retrieved May 25, 2025, from <https://websockets.readthedocs.io/en/stable/>

Python Software Foundation. (n.d.). asyncio — Asynchronous I/O. Python documentation. Retrieved May 25, 2025, from <https://docs.python.org/3/library/asyncio.html>

Ramírez, S. (n.d.). FastAPI documentation. FastAPI. Retrieved May 25, 2025, from <https://fastapi.tiangolo.com/>

Encode. (n.d.). Uvicorn documentation. Uvicorn. Retrieved May 25, 2025, from <https://www.uvicorn.org/>

**נספחים**

# נספחים

## chat\_server.py

import asyncio

import websockets

import json

import logging

logging.basicConfig(level=logging.INFO)

logger = logging.getLogger(\_\_name\_\_)

class ChatServer:

    def \_\_init\_\_(self):

        self.connected\_clients = set()

        self.client\_usernames = {}

    async def register(self, websocket, username):

        self.connected\_clients.add(websocket)

        self.client\_usernames[websocket] = username

        logger.info(f"Client {websocket.remote\_address} registered as {username}. Total connected: {len(self.connected\_clients)}")

        await self.broadcast(json.dumps({"type": "status", "message": f"{username} has joined the chat."}))

    async def unregister(self, websocket):

        if websocket in self.connected\_clients:

            username = self.client\_usernames.get(websocket, "Unknown User")

            self.connected\_clients.remove(websocket)

            if websocket in self.client\_usernames:

                del self.client\_usernames[websocket]

            logger.info(f"Client {websocket.remote\_address} ({username}) disconnected. Total connected: {len(self.connected\_clients)}")

            await self.broadcast(json.dumps({"type": "status", "message": f"{username} has left the chat."}))

    async def broadcast(self, message):

        tasks = [asyncio.create\_task(client.send(message)) for client in self.connected\_clients]

        if tasks:

            await asyncio.wait(tasks)

    async def handler(self, websocket):

        username = None

        try:

            username\_message = await websocket.recv()

            try:

                data = json.loads(username\_message)

                if data.get("type") == "register" and data.get("username"):

                    username = data["username"]

                    await self.register(websocket, username)

                else:

                    await websocket.send(json.dumps({"type": "error", "message": "Please send a registration message with your username first."}))

                    return

            except json.JSONDecodeError:

                await websocket.send(json.dumps({"type": "error", "message": "Invalid JSON format for registration."}))

                return

            async for message in websocket:

                try:

                    data = json.loads(message)

                    if data.get("type") == "chat" and data.get("text"):

                        chat\_message = {

                            "type": "chat",

                            "username": username,

                            "text": data["text"],

                            "timestamp": asyncio.get\_event\_loop().time()

                        }

                        await self.broadcast(json.dumps(chat\_message))

                    else:

                         await websocket.send(json.dumps({"type": "error", "message": "Invalid message format."}))

                except json.JSONDecodeError:

                    await websocket.send(json.dumps({"type": "error", "message": "Invalid JSON format."}))

        except websockets.exceptions.ConnectionClosedOK:

            logger.info(f"Connection closed normally by {websocket.remote\_address}")

        except websockets.exceptions.ConnectionClosedError as e:

            logger.error(f"Connection closed with error by {websocket.remote\_address}: {e}")

        except Exception as e:

            logger.error(f"Unexpected error in handler for {websocket.remote\_address}: {e}")

        finally:

            await self.unregister(websocket)

async def main():

    server = ChatServer()

    port = 8822

    logger.info(f"WebSocket server starting on ws://127.0.0.1:{port}")

    async with websockets.serve(server.handler, "127.0.0.1", port):

        await asyncio.Future()

if \_\_name\_\_ == "\_\_main\_\_":

    try:

        asyncio.run(main())

    except KeyboardInterrupt:

        logger.info("Server stopped by user.")

    except Exception as e:

        logger.critical(f"Server crashed: {e}")

## getStocksData.py

import finnhub

import uvicorn

from fastapi import FastAPI, HTTPException

from fastapi.middleware.cors import CORSMiddleware

app = FastAPI()

app.add\_middleware(

    CORSMiddleware,

    allow\_origins=["\*"],

    allow\_credentials=True,

    allow\_methods=["\*"],

    allow\_headers=["\*"],

)

finnhub\_client = finnhub.Client(api\_key="cvijij9r01qks9qat59gcvijij9r01qks9qat5a0")

@app.get("/get-company-data")

async def get\_company\_data(symbol: str):

    try:

        profile = finnhub\_client.company\_profile2(symbol=symbol)

        quote = finnhub\_client.quote(symbol)

        if not profile and not quote:

             raise HTTPException(status\_code=404, detail=f"Data not found for symbol: {symbol}")

        combined\_data = {}

        if profile:

            combined\_data.update(profile)

        if quote:

            combined\_data.update(quote)

        if 'ticker' not in combined\_data and symbol:

             combined\_data['ticker'] = symbol.upper()

        if 'ticker' not in combined\_data:

             raise HTTPException(status\_code=404, detail=f"Could not retrieve data for symbol: {symbol}")

        return combined\_data

    except Exception as e:

        raise HTTPException(status\_code=500, detail=f"Failed to fetch data for {symbol}: {str(e)}")

@app.get("/get-market-status")

async def get\_market\_status():

    try:

        status = finnhub\_client.market\_status(exchange='US')

        if status and isinstance(status.get('isOpen'), bool):

             return status

        else:

             return {"isOpen": False, "session": "unknown", "endTime": None, "openTime": None}

    except Exception as e:

        return {"isOpen": False, "session": "error", "endTime": None, "openTime": None, "error": str(e)}

@app.get("/get-market-news")

async def get\_market\_news(category: str = 'general'):

    try:

        news = finnhub\_client.general\_news(category=category)

        formatted\_news = []

        if isinstance(news, list):

            for item in news:

                formatted\_news.append({

                    "title": item.get("headline"),

                    "url": item.get("url"),

                    "image": item.get("image"),

                    "summary": item.get("summary"),

                    "source": item.get("source"),

                    "datetime": item.get("datetime")

                })

        return formatted\_news

    except Exception as e:

        raise HTTPException(status\_code=500, detail=f"Failed to fetch market news: {str(e)}")

if \_\_name\_\_ == "\_\_main\_\_":

    uvicorn.run("getStocksData:app", host="127.0.0.1", port=8000, reload=True)

## dashboard.js

import { initializeApp } from "https://www.gstatic.com/firebasejs/10.7.2/firebase-app.js";

import { getAuth, onAuthStateChanged, signOut } from "https://www.gstatic.com/firebasejs/10.7.2/firebase-auth.js";

import {

    getFirestore, doc, getDoc, setDoc, updateDoc,

    arrayUnion, arrayRemove, Timestamp,

    collection, getDocs, query, where, limit, orderBy,

    increment

} from "https://www.gstatic.com/firebasejs/10.7.2/firebase-firestore.js";

const firebaseConfig = {

    apiKey: "AIzaSyAdKXFPMjYamBbNAG2z1Kj8Cp8HjMFcrYA", // Note: API key is visible

    authDomain: "shiftstock-9ba92.firebaseapp.com",

    projectId: "shiftstock-9ba92",

    storageBucket: "shiftstock-9ba92.appspot.com",

    messagingSenderId: "637859468099",

    appId: "1:637859468099:web:26d7984a3296f6f9bb15f6",

    measurementId: "G-EE5FEY81E6"

};

const MAIN\_API\_BASE\_URL = "http://127.0.0.1:8000";

const WEBSOCKET\_URL = "ws://127.0.0.1:8822";

let app, auth, db;

try {

    app = initializeApp(firebaseConfig);

    auth = getAuth(app);

    db = getFirestore(app);

} catch (error) {

    showNotification("Error connecting to services.", "error");

    const greetingEl = document.getElementById("user-greeting");

    if (greetingEl) greetingEl.textContent = "Error";

    const profilePicEl = document.getElementById("user-profile-picture");

    if (profilePicEl) profilePicEl.src = 'https://placehold.co/30x30/eee/ccc?text=!Error';

}

const stockListContainer = document.getElementById("stock-list");

const savedStockListContainer = document.getElementById("saved-stock-list");

const userGreetingSpan = document.getElementById("user-greeting");

const userProfilePictureImg = document.getElementById("user-profile-picture");

const logoutButton = document.getElementById("logout-button");

const stockSearchInput = document.getElementById("stock-search-input");

const searchButton = document.getElementById("search-button");

const searchResultDisplay = document.getElementById("search-result-display");

const portfolioListContainer = document.getElementById("portfolio-list");

const totalPortfolioValueElement = document.getElementById('total-portfolio-value')?.querySelector('span');

const totalCostBasisElement = document.getElementById('total-cost-basis')?.querySelector('span');

const totalProfitLossElement = document.getElementById('total-profit-loss')?.querySelector('.value');

const marketStatusIndicator = document.getElementById("market-status-indicator");

const marketStatusDot = document.getElementById("market-status-dot");

const marketStatusText = document.getElementById("market-status-text");

const notificationArea = document.getElementById("notification-area");

const newsListContainer = document.getElementById("news-list");

const refreshWatchlistBtn = document.getElementById("refresh-watchlist-btn");

const refreshMoversBtn = document.getElementById("refresh-movers-btn");

const refreshPortfolioBtn = document.getElementById("refresh-portfolio-btn");

const refreshNewsBtn = document.getElementById("refresh-news-btn");

const usersDirectoryContainer = document.getElementById("users-directory-container");

const usersListContainer = document.getElementById("users-list");

const userSearchInput = document.getElementById("user-search-input");

const userSearchButton = document.getElementById("user-search-button");

const userSearchResultsDisplay = document.getElementById("user-search-results-display");

const conversationsTab = document.getElementById("conversations-tab");

const toggleConversationsButton = document.getElementById("toggle-conversations");

const conversationsTabContent = document.querySelector("#conversations-tab .tab-content");

const chatMessagesContainer = document.getElementById("chat-messages");

const chatInput = document.getElementById("chat-input");

const sendChatButton = document.getElementById("send-chat-button");

let currentUser = null;

let userSavedStocks = new Set();

let userTransactions = [];

let userHoldings = {};

let stockDataCache = {};

const CACHE\_DURATION = 5 \* 60 \* 1000;

let currentDisplayedSymbol = null;

const baseStockSymbols = ["AAPL", "MSFT", "GOOGL", "AMZN", "TSLA", "NVDA", "META", "JPM"];

let currentUserFollowing = new Set();

let websocket = null;

let chatUsername = "Guest";

function connectWebSocket() {

    if (websocket && websocket.readyState === WebSocket.OPEN) {

        return;

    }

    websocket = new WebSocket(WEBSOCKET\_URL);

    websocket.onopen = () => {

        const username = currentUser?.displayName || currentUser?.email || 'Guest';

        chatUsername = username;

        websocket.send(JSON.stringify({ type: "register", username: username }));

        displayChatMessage({ type: "status", message: "Connected to chat." });

        if (chatMessagesContainer.querySelector('.info-message') && chatMessagesContainer.children.length === 1) {

             chatMessagesContainer.innerHTML = '';

        }

    };

    websocket.onmessage = (event) => {

        try {

            const message = JSON.parse(event.data);

            displayChatMessage(message);

        } catch (error) {

            // console.error("Error parsing WebSocket message:", error);

        }

    };

    websocket.onerror = (error) => {

        displayChatMessage({ type: "status", message: "WebSocket error. Attempting to reconnect..." });

    };

    websocket.onclose = (event) => {

        displayChatMessage({ type: "status", message: "Disconnected from chat. Reconnecting..." });

        setTimeout(connectWebSocket, 5000);

    };

}

function displayChatMessage(message) {

    if (!chatMessagesContainer) return;

    const messageElement = document.createElement('div');

    messageElement.classList.add('chat-message');

    if (message.type === "chat") {

        const isSent = message.username === chatUsername;

        messageElement.classList.add(isSent ? 'sent' : 'received');

        messageElement.innerHTML = `

            <strong>${message.username}:</strong>

            <span>${message.text}</span>

        `;

    } else if (message.type === "status") {

        messageElement.classList.add('status');

        messageElement.textContent = message.message;

    } else if (message.type === "error") {

        messageElement.classList.add('status', 'error-message');

         messageElement.textContent = `Error: ${message.message}`;

    } else {

         return;

    }

    chatMessagesContainer.appendChild(messageElement);

    chatMessagesContainer.scrollTop = chatMessagesContainer.scrollHeight;

}

function sendChatMessage() {

    if (!websocket || websocket.readyState !== WebSocket.OPEN) {

        showNotification("Chat is not connected.", "info");

        return;

    }

    if (!chatInput || chatInput.value.trim() === "") {

        return;

    }

    const messageText = chatInput.value.trim();

    const chatMessage = {

        type: "chat",

        text: messageText

    };

    websocket.send(JSON.stringify(chatMessage));

    chatInput.value = "";

}

function formatCurrency(value) {

    const num = typeof value === 'string' ? parseFloat(value) : value;

    if (typeof num !== 'number' || isNaN(num)) return 'N/A';

    return `$${num.toFixed(2)}`;

}

function formatPercentage(value) {

     const num = typeof value === 'string' ? parseFloat(value) : value;

     if (typeof num !== 'number' || isNaN(num)) return 'N/A';

     const sign = num > 0 ? '+' : '';

     return `${sign}${num.toFixed(2)}%`;

}

function formatPriceChange(changeValue, changePercent) {

    const val = typeof changeValue === 'string' ? parseFloat(changeValue) : changeValue;

    const pct = typeof changePercent === 'string' ? parseFloat(changePercent) : changePercent;

    if (typeof val !== 'number' || isNaN(val) || typeof pct !== 'number' || isNaN(pct)) return 'N/A';

    const valueSign = val >= 0 ? '+' : '';

    const percentSign = pct >= 0 ? '+' : '';

    return `${valueSign}${val.toFixed(2)} (${percentSign}${pct.toFixed(2)}%)`;

}

function getChangeClass(value) {

    const num = typeof value === 'string' ? parseFloat(value) : value;

    if (typeof num !== 'number' || isNaN(num)) return '';

    return num >= 0 ? 'positive' : 'negative';

}

function formatMarketCap(value) {

     const num = typeof value === 'string' ? parseFloat(value) : value;

     if (typeof num !== 'number' || isNaN(num)) return 'N/A';

     if (num >= 1e12) return `$${(num / 1e12).toFixed(2)}T`;

     if (num >= 1e9) return `$${(num / 1e9).toFixed(2)}B`;

     if (num >= 1e6) return `$${(num / 1e6).toFixed(2)}M`;

     return `$${num.toFixed(0)}`;

}

function timeAgo(timestamp) {

    let past;

    if (typeof timestamp === 'number') {

        past = new Date(timestamp \* 1000);

    } else if (typeof timestamp === 'string') {

        try {

            const year = parseInt(timestamp.substring(0, 4), 10);

            const month = parseInt(timestamp.substring(4, 6), 10) - 1;

            const day = parseInt(timestamp.substring(6, 8), 10);

            const hour = parseInt(timestamp.substring(9, 11), 10);

            const minute = parseInt(timestamp.substring(11, 13), 10);

            const second = parseInt(timestamp.substring(13, 15), 10);

            past = new Date(year, month, day, hour, minute, second);

        } catch (e) {

             return 'Invalid date format';

        }

    } else {

         return 'Invalid timestamp type';

    }

    if (isNaN(past.getTime())) return 'Invalid date';

    const now = new Date();

    const diffInSeconds = Math.floor((now.getTime() - past.getTime()) / 1000);

    if (diffInSeconds < 0) return 'in the future';

    if (diffInSeconds < 60) return `${diffInSeconds}s ago`;

    const diffInMinutes = Math.floor(diffInSeconds / 60);

    if (diffInMinutes < 60) return `${diffInMinutes}m ago`;

    const diffInHours = Math.floor(diffInMinutes / 60);

    if (diffInHours < 24) return `${diffInHours}h ago`;

    const diffInDays = Math.floor(diffInHours / 24);

    if (diffInDays < 7) return `${diffInDays}d ago`;

    return past.toLocaleDateString('en-US', { month: 'short', day: 'numeric' });

}

function showNotification(message, type = 'info', duration = 3000) {

     if (!notificationArea) return;

     const notification = document.createElement('div');

     notification.classList.add('notification', type);

     notification.textContent = message;

     notificationArea.appendChild(notification);

     setTimeout(() => {

         notification.classList.add('show');

     }, 10);

     setTimeout(() => {

         notification.classList.remove('show');

         notification.addEventListener('transitionend', () => {

             if (notification.parentNode) {

                 notification.parentNode.removeChild(notification);

             }

         }, { once: true });

     }, duration);

 }

function setLoadingState(container, message = "Loading...") {

     if(container) container.innerHTML = `<p class="info-message">${message}</p>`;

}

function setErrorState(container, message = "An error occurred.") {

     if(container) container.innerHTML = `<p class="error-message">${message}</p>`;

}

 function setEmptyState(container, message = "Nothing to display.") {

     if(container) container.innerHTML = `<p class="empty-list-message">${message}</p>`;

 }

if (auth) {

    onAuthStateChanged(auth, async (user) => {

        if (user) {

            currentUser = user;

            await displayUserProfile(user.uid);

            initializeDashboard();

            connectWebSocket();

        } else {

            currentUser = null;

            userSavedStocks.clear();

            userTransactions = [];

            userHoldings = {};

            stockDataCache = {};

            currentUserFollowing.clear();

            if(userGreetingSpan) userGreetingSpan.textContent = "Guest";

            if(userProfilePictureImg) userProfilePictureImg.src = "https://placehold.co/30x30/eee/ccc?text=?";

            if (websocket) {

                websocket.close();

                websocket = null;

            }

            window.location.href = 'login.html';

        }

    });

} else {

     if(userGreetingSpan) userGreetingSpan.textContent = "Error";

     if(userProfilePictureImg) userProfilePictureImg.src = 'https://placehold.co/30x30/eee/ccc?text=!Error';

     showNotification("Authentication service unavailable.", "error");

}

async function initializeDashboard() {

    if (!currentUser) return;

    updateMarketStatus();

    setLoadingState(savedStockListContainer, "Loading watchlist...");

    setLoadingState(stockListContainer, "Loading market data...");

    setLoadingState(portfolioListContainer, "Loading portfolio...");

    setLoadingState(newsListContainer, "Loading news...");

    setLoadingState(usersListContainer, "Loading users...");

    if (userSearchResultsDisplay) setEmptyState(userSearchResultsDisplay, "Search for a user by username.");

    setLoadingState(searchResultDisplay, "Search for a stock or select one from the lists.");

    setLoadingState(chatMessagesContainer, "Connecting to chat...");

    try {

        const dataPromises = [

             loadUserData(),

             displayStocks(baseStockSymbols),

             displayMarketNews(),

             displayUsersDirectory(),

        ];

        const results = await Promise.allSettled(dataPromises);

        results.forEach((result, index) => {

            if (result.status === 'rejected') {

                // console.error(`Initial data load failed for promise index ${index}:`, result.reason);

            }

        });

         try { await displayPortfolio(); } catch(e) { setErrorState(portfolioListContainer, "Could not display portfolio."); }

         try { await displaySavedStocks(); } catch(e) { setErrorState(savedStockListContainer, "Could not display watchlist."); }

        const initialSymbol = userSavedStocks.size > 0 ? Array.from(userSavedStocks)[0] : baseStockSymbols[0];

        if (initialSymbol) await handleSearch(initialSymbol);

    } catch (error) {

         setErrorState(savedStockListContainer, "Could not load watchlist.");

         setErrorState(stockListContainer, "Could not load market data.");

         setErrorState(portfolioListContainer, "Could not load portfolio.");

         setErrorState(newsListContainer, "Could not load news.");

         setErrorState(usersListContainer, "Could not load users.");

         showNotification("Failed to load some dashboard data.", "error");

    }

}

async function displayUserProfile(userId) {

    if (!db || !userGreetingSpan || !userProfilePictureImg) return;

    try {

        const userDocRef = doc(db, "users", userId);

        const docSnap = await getDoc(userDocRef);

        if (docSnap.exists()) {

            const userData = docSnap.data();

            const usernameToDisplay = userData.originalUsername || userData.username || "User";

            const profilePictureUrl = userData.profilePictureUrl;

            userGreetingSpan.textContent = `Hello, ${usernameToDisplay}!`;

            if(userProfilePictureImg && profilePictureUrl) {

                 userProfilePictureImg.src = profilePictureUrl;

                 userProfilePictureImg.alt = `${usernameToDisplay}'s Profile Picture`;

            } else {

                 userProfilePictureImg.src = 'https://placehold.co/30x30/eee/ccc?text=?';

                 userProfilePictureImg.alt = "Profile Picture";

            }

            currentUserFollowing = new Set(userData.following || []);

            chatUsername = usernameToDisplay;

        } else {

            userGreetingSpan.textContent = "Hello, User!";

            userProfilePictureImg.src = 'https://placehold.co/30x30/eee/ccc?text=?';

            userProfilePictureImg.alt = "Profile Picture";

            currentUserFollowing = new Set();

            chatUsername = currentUser.email || 'Guest';

        }

    } catch (error) {

        userGreetingSpan.textContent = "Hello, User!";

        userProfilePictureImg.src = 'https://placehold.co/30x30/eee/ccc?text=!Error';

        userProfilePictureImg.alt = "Error loading picture";

        currentUserFollowing = new Set();

        chatUsername = currentUser?.email || 'Guest';

        showNotification("Failed to load user profile data.", "error");

    }

}

async function fetchStockData(symbol, forceRefresh = false) {

    symbol = symbol.toUpperCase();

    const cachedItem = stockDataCache[symbol];

    if (!forceRefresh && cachedItem && (Date.now() - cachedItem.timestamp < CACHE\_DURATION)) {

        return cachedItem.data;

    }

    try {

        if (!MAIN\_API\_BASE\_URL) throw new Error("Main API base URL is not configured.");

        const response = await fetch(`${MAIN\_API\_BASE\_URL}/get-company-data?symbol=${symbol}`);

        if (!response.ok) {

            if (response.status === 404) { return null; }

            let errorDetail = `HTTP error! status: ${response.status}`;

            try { const errorData = await response.json(); if (errorData?.detail) errorDetail = errorData.detail; } catch (e) { }

            throw new Error(errorDetail);

        }

        const data = await response.json();

        if (!data || typeof data !== 'object' || !data.ticker) { return null; }

        stockDataCache[symbol] = { data: data, timestamp: Date.now() };

        return data;

    } catch (error) {

        return null;

    }

}

async function fetchMarketNews() {

     try {

         if (!MAIN\_API\_BASE\_URL) throw new Error("Main API base URL is not configured.");

         const response = await fetch(`${MAIN\_API\_BASE\_URL}/get-market-news?category=general`);

         if (!response.ok) throw new Error(`HTTP error! status: ${response.status}`);

         const newsData = await response.json();

         return Array.isArray(newsData) ? newsData : [];

     } catch (error) {

         return null;

     }

}

async function loadUserData() {

    if (!currentUser || !db) return;

    const userDocRef = doc(db, "users", currentUser.uid);

    try {

        const docSnap = await getDoc(userDocRef);

        if (docSnap.exists()) {

            const data = docSnap.data();

            userSavedStocks = new Set(data.savedStocks || []);

            userTransactions = (data.transactions || []).map(tx => ({ ...tx, timestamp: tx.timestamp?.toDate ? tx.timestamp.toDate() : (tx.timestamp || null) }));

            currentUserFollowing = new Set(data.following || []);

        } else {

            const fallbackUsername = currentUser.email || "User";

            const fallbackProfilePictureUrl = `https://placehold.co/30x30/eee/ccc?text=${fallbackUsername.charAt(0).toUpperCase()}`;

            await setDoc(userDocRef, {

                 username: fallbackUsername.toLowerCase(), originalUsername: fallbackUsername, email: currentUser.email,

                 profilePictureUrl: fallbackProfilePictureUrl, createdAt: new Date(),

                 savedStocks: [], transactions: [], following: [],

                 profitLossAmount: 0, profitLossPercentage: 0, followerCount: 0

            });

            userSavedStocks = new Set(); userTransactions = []; currentUserFollowing = new Set();

        }

    } catch (error) {

        showNotification("Failed to load your saved data.", "error");

        userSavedStocks = new Set(); userTransactions = []; currentUserFollowing = new Set();

    }

    processTransactions();

}

function processTransactions() {

    const tempHoldings = {};

    userTransactions.sort((a, b) => (a.timestamp?.getTime() || 0) - (b.timestamp?.getTime() || 0));

    userTransactions.forEach(transaction => {

        const { symbol, type, quantity, price } = transaction;

        if (!symbol || (type !== 'buy' && type !== 'sell') || !(quantity > 0) || !(price > 0)) return;

        const s = symbol.toUpperCase();

        if (!tempHoldings[s]) tempHoldings[s] = { quantity: 0, totalCost: 0 };

        if (type === 'buy') {

            tempHoldings[s].quantity += quantity;

            tempHoldings[s].totalCost += quantity \* price;

        } else if (type === 'sell') {

             const quantityToSell = quantity;

             const currentQuantity = tempHoldings[s].quantity;

             const currentTotalCost = tempHoldings[s].totalCost;

             if (currentQuantity >= quantityToSell) {

                 const proportionSold = quantityToSell / currentQuantity;

                 tempHoldings[s].totalCost -= currentTotalCost \* proportionSold;

                 tempHoldings[s].quantity -= quantityToSell;

                 if (tempHoldings[s].quantity < 1e-9) { tempHoldings[s].quantity = 0; tempHoldings[s].totalCost = 0; }

             } else {

                 tempHoldings[s].quantity = 0;

                 tempHoldings[s].totalCost = 0;

             }

        }

    });

    userHoldings = {};

    for (const symbol in tempHoldings) {

        if (tempHoldings[symbol].quantity > 0) {

            userHoldings[symbol] = {

                quantity: tempHoldings[symbol].quantity,

                totalCost: tempHoldings[symbol].totalCost,

                averagePrice: tempHoldings[symbol].totalCost / tempHoldings[symbol].quantity

            };

        }

    }

}

async function saveStockToFirebase(symbol) {

    if (!currentUser || !db || userSavedStocks.has(symbol)) return;

    const userDocRef = doc(db, "users", currentUser.uid);

    try {

        await updateDoc(userDocRef, { savedStocks: arrayUnion(symbol) });

        userSavedStocks.add(symbol);

        showNotification(`${symbol} added to watchlist.`, "success");

        updateSaveButtonState(symbol, true);

        displaySavedStocks();

    } catch (error) { showNotification(`Failed to save ${symbol}.`, "error"); }

}

async function removeStockFromFirebase(symbol) {

    if (!currentUser || !db || !userSavedStocks.has(symbol)) return;

    const userDocRef = doc(db, "users", currentUser.uid);

    try {

        await updateDoc(userDocRef, { savedStocks: arrayRemove(symbol) });

        userSavedStocks.delete(symbol);

        showNotification(`${symbol} removed from watchlist.`, "info");

        updateSaveButtonState(symbol, false);

        displaySavedStocks();

        if (currentDisplayedSymbol === symbol) {

             setLoadingState(searchResultDisplay, "Select a stock to view details.");

             currentDisplayedSymbol = null;

        }

    } catch (error) { showNotification(`Failed to remove ${symbol}.`, "error"); }

}

 async function recordTransactionFirebase(symbol, type, quantity, price) {

     if (!currentUser || !db) { showNotification("You must be logged in to trade.", "error"); return false; }

     if (!symbol || (type !== 'buy' && type !== 'sell') || !(quantity > 0) || !(price > 0)) { showNotification("Invalid transaction data.", "error"); return false; }

     const userDocRef = doc(db, "users", currentUser.uid);

     const transaction = { symbol: symbol.toUpperCase(), type, quantity, price, timestamp: Timestamp.fromDate(new Date()) };

     try {

         await updateDoc(userDocRef, { transactions: arrayUnion(transaction) });

         showNotification(`${type === 'buy' ? 'Purchase' : 'Sale'} of ${quantity} ${symbol} recorded.`, "success");

         userTransactions.push({ ...transaction, timestamp: transaction.timestamp.toDate() });

         processTransactions();

         await displayPortfolio();

         if (symbol === currentDisplayedSymbol) updateBuySellButtonStates(symbol);

         return true;

     } catch (error) { showNotification(`Failed to record ${type} of ${symbol}.`, "error"); return false; }

 }

async function followUser(targetUserId) {

    if (!currentUser || !db || !targetUserId || currentUser.uid === targetUserId || currentUserFollowing.has(targetUserId)) return;

    const currentUserDocRef = doc(db, "users", currentUser.uid);

    const targetUserDocRef = doc(db, "users", targetUserId);

    try {

        await updateDoc(currentUserDocRef, { following: arrayUnion(targetUserId) });

        await updateDoc(targetUserDocRef, { followerCount: increment(1) });

        currentUserFollowing.add(targetUserId);

        showNotification("User followed.", "success", 1500);

        updateFollowButtonState(targetUserId, true);

        displayUsersDirectory();

    } catch (error) { showNotification("Failed to follow user.", "error"); }

}

async function unfollowUser(targetUserId) {

    if (!currentUser || !db || !targetUserId || !currentUserFollowing.has(targetUserId)) return;

    const currentUserDocRef = doc(db, "users", currentUser.uid);

    const targetUserDocRef = doc(db, "users", targetUserId);

    try {

        await updateDoc(currentUserDocRef, { following: arrayRemove(targetUserId) });

        await updateDoc(targetUserDocRef, { followerCount: increment(-1) });

        currentUserFollowing.delete(targetUserId);

        showNotification("User unfollowed.", "info", 1500);

        updateFollowButtonState(targetUserId, false);

        displayUsersDirectory();

    } catch (error) { showNotification("Failed to unfollow user.", "error"); }

}

async function updateCurrentUserPLInFirestore(amount, percent) {

    if (!currentUser || !db) return;

    const plAmount = (typeof amount === 'number' && !isNaN(amount)) ? amount : 0;

    const plPercent = (typeof percent === 'number' && !isNaN(percent)) ? percent : 0;

    const userDocRef = doc(db, "users", currentUser.uid);

    try {

        await updateDoc(userDocRef, { profitLossAmount: plAmount, profitLossPercentage: plPercent });

    } catch (error) { /\* console.error(`Error updating P/L for ${currentUser.uid} in Firestore:`, error); \*/ }

}

function createStockRowElement(data) {

    if (!data || !data.ticker) return null;

    const stockRow = document.createElement("div");

    stockRow.classList.add("stock-row");

    stockRow.dataset.symbol = data.ticker;

    const isSaved = userSavedStocks.has(data.ticker);

    const logoUrl = data.logo || `https://placehold.co/48x48/eee/ccc?text=${data.ticker[0]}`;

    const companyName = data.name || data.ticker;

    stockRow.innerHTML = `

        <div class="stock-info">

            <img class="stock-logo" src="${logoUrl}" alt="${companyName} Logo" onerror="this.src='https://placehold.co/48x48/eee/ccc?text=${data.ticker[0]}'; this.onerror=null;">

            <div class="stock-details">

                <div class="symbol-container">

                     <span class="symbol">${data.ticker}</span>

                     <button class="save-remove-button ${isSaved ? 'saved' : ''}" data-symbol="${data.ticker}" title="${isSaved ? 'Remove from watchlist' : 'Add to watchlist'}"></button>

                </div>

                <span class="company-name">${companyName}</span>

            </div>

        </div>

        <div class="stock-price-info">

            <span class="stock-price">${formatCurrency(data.c)}</span>

            <span class="stock-change ${getChangeClass(data.dp)}">${formatPercentage(data.dp)}</span>

        </div>`;

    stockRow.addEventListener("click", (event) => { if (!event.target.closest('.save-remove-button')) handleSearch(data.ticker); });

    stockRow.querySelector('.save-remove-button').addEventListener("click", (event) => {

        event.stopPropagation();

        const symbol = event.target.dataset.symbol;

        if (userSavedStocks.has(symbol)) removeStockFromFirebase(symbol); else saveStockToFirebase(symbol);

    });

    return stockRow;

}

function createPortfolioRowElement(data, holding) {

     if (!data || !data.ticker || !holding || holding.quantity <= 0) return null;

     const portfolioRow = document.createElement("div");

     portfolioRow.classList.add("portfolio-item");

     portfolioRow.dataset.symbol = data.ticker;

     const logoUrl = data.logo || `https://placehold.co/48x48/eee/ccc?text=${data.ticker[0]}`;

     const companyName = data.name || data.ticker;

     const symbol = data.ticker;

     const currentPrice = data.c;

     const quantityOwned = holding.quantity;

     const averageBuyPrice = holding.averagePrice;

     const currentValue = (typeof currentPrice === 'number') ? currentPrice \* quantityOwned : NaN;

     const costBasis = holding.totalCost;

     const profitLoss = (!isNaN(currentValue) && !isNaN(costBasis)) ? currentValue - costBasis : NaN;

     const profitLossPercent = (!isNaN(profitLoss) && costBasis > 0) ? (profitLoss / costBasis) \* 100 : NaN;

     const plClass = getChangeClass(profitLoss);

     portfolioRow.innerHTML = `

         <div class="portfolio-stock-info">

            <img class="portfolio-stock-logo" src="${logoUrl}" alt="${companyName} logo" onerror="this.src='https://placehold.co/48x48/eee/ccc?text=${data.ticker[0]}'; this.onerror=null;">

            <div class="portfolio-stock-details">

                 <span class="portfolio-symbol">${quantityOwned.toFixed(quantityOwned % 1 === 0 ? 0 : 2)} shares of ${symbol}</span>

                 <span class="portfolio-company-name">${companyName}</span>

            </div>

         </div>

         <div class="portfolio-value-info">

             <div class="portfolio-current-value">Value: <strong>${formatCurrency(currentValue)}</strong></div>

             <div class="portfolio-profit-loss ${plClass}">

                 P/L: <span class="${plClass}">${formatCurrency(profitLoss)} (${formatPercentage(profitLossPercent)})</span>

             </div>

             <div class="portfolio-average-price">Avg Buy: <strong>${formatCurrency(averageBuyPrice)}</strong></div>

         </div>`;

     portfolioRow.addEventListener("click", () => handleSearch(data.ticker));

     return portfolioRow;

}

function createNewsItemElement(newsItem) {

    if (!newsItem || !newsItem.title || !newsItem.url) return null;

    const item = document.createElement('div');

    item.classList.add('news-item');

    const imageUrl = newsItem.image || `https://placehold.co/160x120/eee/ccc?text=News`;

    const summary = newsItem.summary ? (newsItem.summary.length > 150 ? newsItem.summary.substring(0, 150) + '...' : newsItem.summary) : 'Read more...';

    item.innerHTML = `

        <img src="${imageUrl}" alt="News Image" class="news-image" loading="lazy" onerror="this.style.display='none';">

        <div class="news-content">

            <a href="${newsItem.url}" target="\_blank" rel="noopener noreferrer" class="news-headline">${newsItem.title}</a>

            <p class="news-summary">${summary}</p>

            <p class="news-source-time">${newsItem.source} - ${timeAgo(newsItem.datetime)}</p>

        </div>`;

    return item;

}

function createUserRowElement(userData, userId) {

    if (!userData || !userId) return null;

    const userRow = document.createElement("div");

    userRow.classList.add("user-directory-item");

    userRow.dataset.userId = userId;

    const username = userData.originalUsername || userData.username || 'Unknown User';

    const pfpUrl = userData.profilePictureUrl || `https://placehold.co/64x64/eee/ccc?text=${username.charAt(0).toUpperCase()}`;

    const isFollowing = currentUserFollowing.has(userId);

    const userPLAmount = userData.profitLossAmount !== undefined ? userData.profitLossAmount : null;

    const userPLPercent = userData.profitLossPercentage !== undefined ? userData.profitLossPercentage : null;

    const plClass = getChangeClass(userPLAmount);

    const followerCount = userData.followerCount !== undefined ? userData.followerCount : 'N/A';

    let plDisplay = 'N/A';

    if (userPLAmount !== null) {

        plDisplay = `${formatCurrency(userPLAmount)}`;

        if (userPLPercent !== null) plDisplay += ` (${formatPercentage(userPLPercent)})`;

    }

    userRow.innerHTML = `

        <div class="user-directory-info">

            <img class="user-directory-pfp" src="${pfpUrl}" alt="${username}'s Profile Picture" onerror="this.src='https://placehold.co/64x64/eee/ccc?text=?'; this.onerror=null;">

            <div class="user-directory-details">

                <span class="user-directory-username">${username}</span>

                <div class="user-directory-stats">

                     <span class="user-pl ${plClass}">P/L: ${plDisplay}</span>

                     <span>Followers: ${followerCount}</span>

                </div>

            </div>

        </div>

        <div class="user-directory-actions">

            <button class="follow-button ${isFollowing ? 'following' : ''}" data-user-id="${userId}">${isFollowing ? 'Following' : 'Follow'}</button>

        </div>`;

    userRow.addEventListener('click', () => {

        window.location.href = `user\_profile.html?userId=${userId}`;

    });

    const followButton = userRow.querySelector('.follow-button');

    if (followButton) {

        followButton.addEventListener('click', (event) => {

            event.stopPropagation(); event.target.disabled = true;

            if (currentUserFollowing.has(userId)) unfollowUser(userId).finally(() => updateFollowButtonState(userId, false));

            else followUser(userId).finally(() => updateFollowButtonState(userId, true));

        });

    }

    return userRow;

}

function updateSaveButtonState(symbol, isSaved) {

    document.querySelectorAll(`.save-remove-button[data-symbol="${symbol}"]`).forEach(button => {

        button.classList.toggle('saved', isSaved);

        button.title = isSaved ? 'Remove from watchlist' : 'Add to watchlist';

    });

}

function updateBuySellButtonStates(symbol) {

    const sellButton = searchResultDisplay.querySelector(`.sell-button-large[data-symbol="${symbol}"]`);

    const buyButton = searchResultDisplay.querySelector(`.buy-button-large[data-symbol="${symbol}"]`);

    if (sellButton) {

        const quantityOwned = userHoldings[symbol]?.quantity || 0;

        sellButton.disabled = quantityOwned <= 0;

        sellButton.title = quantityOwned <= 0 ? "You don't own shares to sell" : `Sell ${symbol}`;

    }

    if (buyButton) {

        const price = buyButton.dataset.price;

        const isPriceValid = price && !isNaN(parseFloat(price));

        buyButton.disabled = !isPriceValid;

        buyButton.title = buyButton.disabled ? "Cannot buy (price unavailable)" : `Buy ${symbol}`;

    }

}

async function displayStockList(container, symbols, emptyMessage = "No stocks to display.", forceRefresh = false) {

     if (!container) return; setLoadingState(container);

     if (!symbols || symbols.length === 0) { setEmptyState(container, emptyMessage); return; }

     const fetchPromises = symbols.map(symbol => fetchStockData(symbol, forceRefresh));

     const results = await Promise.allSettled(fetchPromises);

     container.innerHTML = ''; let hasAddedStock = false;

     results.forEach(result => {

          if (result.status === 'fulfilled' && result.value?.ticker) {

             const stockRowElement = createStockRowElement(result.value);

             if (stockRowElement) { container.appendChild(stockRowElement); hasAddedStock = true; }

         } else {

              const symbolWithError = symbols.find(sym => result.reason?.message?.includes(sym));

              if(symbolWithError) {

                  const errorRow = document.createElement('div');

                  errorRow.classList.add('stock-row', 'error-message');

                  errorRow.textContent = `Could not load data for ${symbolWithError}.`;

                  container.appendChild(errorRow); hasAddedStock = true;

              }

         }

     });

     if (!hasAddedStock) setEmptyState(container, "Could not load data for stocks.");

}

async function displayStocks(symbolsToDisplay = baseStockSymbols, forceRefresh = false) {

     await displayStockList(stockListContainer, symbolsToDisplay, "No market data available.", forceRefresh);

}

async function displaySavedStocks(forceRefresh = false) {

     await displayStockList(savedStockListContainer, Array.from(userSavedStocks), "Your watchlist is empty.", forceRefresh);

}

async function displayPortfolio(forceRefresh = false) {

     if (!portfolioListContainer || !totalPortfolioValueElement || !totalCostBasisElement || !totalProfitLossElement) return;

     const portfolioSymbols = Object.keys(userHoldings);

     if (portfolioSymbols.length === 0) {

         setEmptyState(portfolioListContainer, "Your portfolio is empty.");

         totalPortfolioValueElement.textContent = '$0.00'; totalCostBasisElement.textContent = '$0.00';

         totalProfitLossElement.textContent = '$0.00 (0.00%)'; totalProfitLossElement.className = 'value';

         await updateCurrentUserPLInFirestore(0, 0); return;

     }

     setLoadingState(portfolioListContainer, "Updating portfolio...");

     const fetchPromises = portfolioSymbols.map(symbol => fetchStockData(symbol, forceRefresh));

     const results = await Promise.allSettled(fetchPromises);

     portfolioListContainer.innerHTML = ''; let totalValue = 0, totalCost = 0, hasAddedItem = false;

     results.forEach(result => {

         if (result.status === 'fulfilled' && result.value?.ticker) {

             const data = result.value; const symbol = data.ticker; const holding = userHoldings[symbol];

             if (holding?.quantity > 0) {

                 const portfolioRowElement = createPortfolioRowElement(data, holding);

                 if (portfolioRowElement) { portfolioListContainer.appendChild(portfolioRowElement); hasAddedItem = true; }

                 if (typeof data.c === 'number' && !isNaN(data.c)) totalValue += data.c \* holding.quantity;

                 if (typeof holding.totalCost === 'number' && !isNaN(holding.totalCost)) totalCost += holding.totalCost;

             }

         } else {

              const symbolWithError = portfolioSymbols.find(sym => result.reason?.message?.includes(sym));

              if(symbolWithError && userHoldings[symbolWithError]) {

                  const errorRow = document.createElement('div');

                  errorRow.classList.add('portfolio-item', 'error-message');

                  errorRow.textContent = `Could not load data for ${symbolWithError} (${userHoldings[symbolWithError].quantity.toFixed(userHoldings[symbolWithError].quantity % 1 === 0 ? 0 : 2)} shares).`;

                  portfolioListContainer.appendChild(errorRow); hasAddedItem = true;

                  if (typeof userHoldings[symbolWithError].totalCost === 'number' && !isNaN(userHoldings[symbolWithError].totalCost)) totalCost += userHoldings[symbolWithError].totalCost;

              }

         }

     });

     const totalPLAmount = totalValue - totalCost;

     const totalPLPercent = totalCost > 0 ? (totalPLAmount / totalCost) \* 100 : 0;

     if (!hasAddedItem && portfolioSymbols.length > 0) {

         setErrorState(portfolioListContainer, "Could not load portfolio data.");

         totalPortfolioValueElement.textContent = 'Error'; totalCostBasisElement.textContent = 'Error'; totalProfitLossElement.textContent = 'Error';

         await updateCurrentUserPLInFirestore(0,0);

     } else if (!hasAddedItem) { /\* Handled by initial empty check \*/ }

     else {

          totalPortfolioValueElement.textContent = formatCurrency(totalValue);

          totalCostBasisElement.textContent = formatCurrency(totalCost);

          totalProfitLossElement.textContent = `${formatCurrency(totalPLAmount)} (${formatPercentage(totalPLPercent)})`;

          totalProfitLossElement.className = `value ${getChangeClass(totalPLAmount)}`;

          await updateCurrentUserPLInFirestore(totalPLAmount, totalPLPercent);

     }

}

async function displayMarketNews() {

     if (!newsListContainer) return; setLoadingState(newsListContainer, "Loading news...");

     const newsData = await fetchMarketNews();

     if (newsData && newsData.length > 0) {

         newsListContainer.innerHTML = '';

         newsData.forEach(newsItem => { const newsElement = createNewsItemElement(newsItem); if (newsElement) newsListContainer.appendChild(newsElement); });

         if (newsListContainer.children.length === 0) setEmptyState(newsListContainer, "No recent market news found that could be displayed.");

     } else if (newsData) { setEmptyState(newsListContainer, "No recent market news found."); }

     else { setErrorState(newsListContainer, "Could not load market news."); }

}

async function updateMarketStatus() {

     if (!marketStatusIndicator || !marketStatusDot || !marketStatusText) return;

     marketStatusText.textContent = "Checking Status..."; marketStatusIndicator.className = 'market-status-indicator market-unknown';

     try {

         if (!MAIN\_API\_BASE\_URL) throw new Error("Main API base URL is not configured.");

         const response = await fetch(`${MAIN\_API\_BASE\_URL}/get-market-status`);

         if (!response.ok) throw new Error(`HTTP error! status: ${response.status}`);

         const data = await response.json();

         if (data && typeof data.isOpen === 'boolean') {

              let sessionText = data.session && typeof data.session === 'string' ? ` (${data.session})` : "";

             marketStatusText.textContent = `Market ${data.isOpen ? "Open" : "Closed"}${sessionText}`;

             marketStatusIndicator.className = `market-status-indicator market-${data.isOpen ? "open" : "closed"}`;

         } else { marketStatusText.textContent = "Status Unknown"; marketStatusIndicator.className = 'market-status-indicator market-unknown'; }

     } catch (error) { marketStatusText.textContent = "Status Unknown"; marketStatusIndicator.className = 'market-status-indicator market-unknown'; }

}

function updateFollowButtonState(targetUserId, isFollowing) {

    document.querySelectorAll(`#users-list .follow-button[data-user-id="${targetUserId}"], #user-search-results-display .follow-button[data-user-id="${targetUserId}"]`)

    .forEach(button => {

        button.textContent = isFollowing ? 'Following' : 'Follow';

        button.classList.toggle('following', isFollowing);

        button.disabled = false;

    });

}

async function displayUsersDirectory() {

    if (!usersListContainer || !db || !currentUser) return; setLoadingState(usersListContainer, "Loading users...");

    try {

        const q = query(collection(db, "users"), where("\_\_name\_\_", "!=", currentUser.uid), orderBy("username"), limit(20));

        const querySnapshot = await getDocs(q);

        usersListContainer.innerHTML = ''; let userCount = 0;

        querySnapshot.forEach((docSnap) => {

            const userData = docSnap.data();

            if (userData.originalUsername || userData.username) {

                 const userRowElement = createUserRowElement(userData, docSnap.id);

                 if (userRowElement) { usersListContainer.appendChild(userRowElement); userCount++; }

            }

        });

        if (userCount === 0) setEmptyState(usersListContainer, "No other users found.");

    } catch (error) { setErrorState(usersListContainer, "Could not load users."); }

}

async function handleUserSearch() {

    const searchTerm = (userSearchInput?.value || "").trim();

    if (!searchTerm) { showNotification("Please enter a username to search.", "info"); setEmptyState(userSearchResultsDisplay, "Search for a user by username."); return; }

    setLoadingState(userSearchResultsDisplay, `Searching for "${searchTerm}"...`);

    try {

        const q = query(collection(db, "users"), where("username", "==", searchTerm.toLowerCase()), limit(10));

        const querySnapshot = await getDocs(q);

        userSearchResultsDisplay.innerHTML = '';

        if (querySnapshot.empty) { setEmptyState(userSearchResultsDisplay, `No users found matching "${searchTerm}".`); return; }

        let userCount = 0;

        querySnapshot.forEach((docSnap) => {

            const userData = docSnap.data();

            if (docSnap.id !== currentUser?.uid && (userData.originalUsername || userData.username)) {

                 const userRowElement = createUserRowElement(userData, docSnap.id);

                 if (userRowElement) { userSearchResultsDisplay.appendChild(userRowElement); userCount++; }

            }

        });

        if (userCount === 0) setEmptyState(userSearchResultsDisplay, `No other users found matching "${searchTerm}".`);

    } catch (error) { setErrorState(userSearchResultsDisplay, `An error occurred while searching for users.`); }

}

async function handleSearch(symbolToSearch = null) {

    const searchTerm = (symbolToSearch || stockSearchInput?.value || "").trim().toUpperCase();

    if (!searchTerm) { showNotification("Please enter a stock symbol.", "info"); return; }

    currentDisplayedSymbol = searchTerm;

    searchResultDisplay.innerHTML = `<p class="info-message">Searching for ${searchTerm}...</p>`;

    if(stockSearchInput) stockSearchInput.value = searchTerm;

    try {

        const stockData = await fetchStockData(searchTerm, true);

        if (stockData) {

            displaySearchResult(stockData);

        } else {

             searchResultDisplay.innerHTML = `<p class="error-message">Could not find data for symbol: ${searchTerm}. Please check the symbol.</p>`;

             currentDisplayedSymbol = null;

        }

    } catch (error) {

        searchResultDisplay.innerHTML = `<p class="error-message">An error occurred while searching for ${searchTerm}.</p>`;

        currentDisplayedSymbol = null;

    }

}

function displaySearchResult(data) {

     if (!searchResultDisplay || !data?.ticker || typeof data !== 'object') {

         setErrorState(searchResultDisplay, "Invalid data received for display.");

         currentDisplayedSymbol = null; return;

     }

     const symbol = data.ticker;

     const isSaved = userSavedStocks.has(symbol);

     const logoUrl = data.logo || `https://placehold.co/100x100/eee/ccc?text=${symbol[0]}`;

     const companyName = data.name && typeof data.name === 'string' ? data.name : symbol;

     let websiteDisplay = '';

     if (data.weburl && typeof data.weburl === 'string' && (data.weburl.startsWith('http://') || data.weburl.startsWith('https://'))) {

        try {

            const urlObject = new URL(data.weburl);

            websiteDisplay = `<a href="${data.weburl}" target="\_blank" rel="noopener noreferrer" class="company-website">${urlObject.hostname}</a>`;

        } catch(e){ /\* Invalid URL, do nothing \*/ }

     }

     const currentPrice = typeof data.c === 'number' ? data.c : 'N/A';

     const priceChange = typeof data.d === 'number' ? data.d : 'N/A';

     const percentChange = typeof data.dp === 'number' ? data.dp : 'N/A';

     const highPrice = typeof data.h === 'number' ? data.h : 'N/A';

     const lowPrice = typeof data.l === 'number' ? data.l : 'N/A';

     const openPrice = typeof data.o === 'number' ? data.o : 'N/A';

     const previousClose = typeof data.pc === 'number' ? data.pc : 'N/A';

     const marketCapUSD = typeof data.marketCapitalization === 'number' ? data.marketCapitalization : 'N/A';

     const priceChangeClass = getChangeClass(percentChange);

     searchResultDisplay.innerHTML = `

         <div class="large-stock-display">

             <div class="company-header">

                 <img class="large-stock-logo" src="${logoUrl}" alt="${companyName} Logo" onerror="this.src='https://placehold.co/100x100/eee/ccc?text=${symbol[0]}'; this.onerror=null;">

                 <div class="company-title">

                      <div class="title-line-1">

                         <span class="large-symbol">${symbol}</span>

                         <button class="save-remove-button ${isSaved ? 'saved' : ''}" data-symbol="${symbol}" title="${isSaved ? 'Remove from watchlist' : 'Add to watchlist'}"></button>

                      </div>

                     <h2 class="large-company-name">${companyName}</h2>

                     ${websiteDisplay}

                 </div>

             </div>

             <div class="price-info">

                 <span class="large-price">${formatCurrency(currentPrice)}</span>

                 <span class="large-price-change ${priceChangeClass}">

                     ${formatPriceChange(priceChange, percentChange)}

                 </span>

             </div>

             <div class="key-metrics">

                 <div class="metric-item"><span>High:</span> <strong>${formatCurrency(highPrice)}</strong></div>

                 <div class="metric-item"><span>Low:</span> <strong>${formatCurrency(lowPrice)}</strong></div>

                 <div class="metric-item"><span>Open:</span> <strong>${formatCurrency(openPrice)}</strong></div>

                 <div class="metric-item"><span>Prev. Close:</span> <strong>${formatCurrency(previousClose)}</strong></div>

                 <div class="metric-item"><span>Market Cap:</span> <strong>${formatMarketCap(marketCapUSD)}</strong></div>

             </div>

             <div class="stock-actions-large">

                 <button class="action-button buy-button-large" data-symbol="${symbol}" data-price="${currentPrice}">Buy</button>

                 <button class="action-button sell-button-large" data-symbol="${symbol}" data-price="${currentPrice}">Sell</button>

             </div>

            </div>`;

      const buyButton = searchResultDisplay.querySelector('.buy-button-large');

      const sellButton = searchResultDisplay.querySelector('.sell-button-large');

      const saveButton = searchResultDisplay.querySelector('.company-title .save-remove-button');

      if (buyButton) buyButton.addEventListener('click', handleBuyClick);

      if (sellButton) sellButton.addEventListener('click', handleSellClick);

      if (saveButton) {

           saveButton.addEventListener('click', (event) => {

              const clickedSymbol = event.target.dataset.symbol;

              if (userSavedStocks.has(clickedSymbol)) removeStockFromFirebase(clickedSymbol);

              else saveStockToFirebase(clickedSymbol);

          });

      }

      updateBuySellButtonStates(symbol);

}

function handleBuyClick(event) {

    const button = event.target.closest('button');

    if (!button) return;

    const symbol = button.dataset.symbol;

    const priceString = button.dataset.price;

    const price = parseFloat(priceString);

    if (isNaN(price) || price <= 0) {

         showNotification(`Cannot buy ${symbol}: Price unavailable or invalid.`, "error"); return;

    }

    const quantityString = prompt(`Enter quantity of ${symbol} to buy at ~${formatCurrency(price)}:`);

    if (quantityString === null) return;

    const quantity = parseInt(quantityString.trim(), 10);

    if (isNaN(quantity) || quantity <= 0) {

        showNotification("Invalid quantity. Please enter a positive whole number.", "error"); return;

    }

    button.disabled = true;

    recordTransactionFirebase(symbol, 'buy', quantity, price)

         .finally(() => {

             const currentBuyButton = searchResultDisplay.querySelector(`.buy-button-large[data-symbol="${symbol}"]`);

             if (currentBuyButton) currentBuyButton.disabled = false;

             updateBuySellButtonStates(symbol);

         });

}

function handleSellClick(event) {

    const button = event.target.closest('button');

     if (!button) return;

    const symbol = button.dataset.symbol;

    const priceString = button.dataset.price;

    const price = parseFloat(priceString);

     if (isNaN(price) || price <= 0) {

         showNotification(`Cannot sell ${symbol}: Price unavailable or invalid.`, "error"); return;

     }

    const quantityOwned = userHoldings[symbol]?.quantity || 0;

     if (quantityOwned <= 0) {

         showNotification(`You do not own any shares of ${symbol}.`, "info"); return;

     }

    const quantityString = prompt(`Enter quantity of ${symbol} to sell at ~${formatCurrency(price)} (You own ${quantityOwned.toFixed(quantityOwned % 1 === 0 ? 0 : 2)}):`);

    if (quantityString === null) return;

    const quantityToSell = parseInt(quantityString.trim(), 10);

    if (isNaN(quantityToSell) || quantityToSell <= 0) {

        showNotification("Invalid quantity. Please enter a positive whole number.", "error"); return;

    }

     if (quantityToSell > quantityOwned) {

         showNotification(`You cannot sell more ${symbol} than you own (${quantityOwned.toFixed(quantityOwned % 1 === 0 ? 0 : 2)}).`, "error"); return;

     }

     button.disabled = true;

     recordTransactionFirebase(symbol, 'sell', quantityToSell, price)

          .finally(() => {

             const currentSellButton = searchResultDisplay.querySelector(`.sell-button-large[data-symbol="${symbol}"]`);

             if (currentSellButton) currentSellButton.disabled = false;

             updateBuySellButtonStates(symbol);

          });

}

 async function handleRefresh(button, refreshFunction) {

     if (!button) return;

     button.disabled = true;

     const icon = button.querySelector('.icon');

     if (icon) icon.classList.add('spin');

     try {

         await refreshFunction(true);

         showNotification("Data refreshed.", "info", 1500);

     } catch (error) {

         showNotification("Failed to refresh data.", "error");

     } finally {

         button.disabled = false;

         if (icon) icon.classList.remove('spin');

     }

 }

if (logoutButton && auth) {

    logoutButton.addEventListener("click", () => {

        signOut(auth).catch((error) => {

            showNotification("Error signing out.", "error");

        });

    });

}

if (searchButton) {

    searchButton.addEventListener('click', () => handleSearch());

}

if (stockSearchInput) {

    stockSearchInput.addEventListener('keypress', (event) => {

        if (event.key === 'Enter') {

            event.preventDefault();

            handleSearch();

        }

    });

}

if (userSearchButton) {

    userSearchButton.addEventListener('click', () => handleUserSearch());

}

if (userSearchInput) {

    userSearchInput.addEventListener('keypress', (event) => {

        if (event.key === 'Enter') {

            event.preventDefault();

            handleUserSearch();

        }

    });

}

if (toggleConversationsButton && conversationsTab) {

    toggleConversationsButton.addEventListener('click', () => {

        conversationsTab.classList.toggle('collapsed');

        toggleConversationsButton.textContent = conversationsTab.classList.contains('collapsed') ? '▼' : '▲';

        if (!conversationsTab.classList.contains('collapsed') && chatMessagesContainer) {

             chatMessagesContainer.scrollTop = chatMessagesContainer.scrollHeight;

        }

    });

}

if (sendChatButton) {

    sendChatButton.addEventListener('click', sendChatMessage);

}

if (chatInput) {

    chatInput.addEventListener('keypress', (event) => {

        if (event.key === 'Enter') {

            event.preventDefault();

            sendChatMessage();

        }

    });

}

(function ensureSpinAnimation() { // IIFE to ensure spin animation is defined

    if (document.getElementById('spin-animation-style')) return;

    let spinAnimationDefined = false;

    for (const sheet of document.styleSheets) {

        try {

            if (sheet.cssRules) {

                for (const rule of sheet.cssRules) {

                    if (rule instanceof CSSKeyframesRule && rule.name === 'spin') {

                        spinAnimationDefined = true;

                        break;

                    }

                }

            }

        } catch (e) { /\* Silently ignore CORS errors for external stylesheets \*/ }

        if (spinAnimationDefined) break;

    }

    if (!spinAnimationDefined) {

        const styleSheet = document.createElement("style");

        styleSheet.id = 'spin-animation-style';

        styleSheet.type = "text/css";

        styleSheet.innerText = `

            @keyframes spin {

                0% { transform: rotate(0deg); }

                100% { transform: rotate(360deg); }

            }

            .spin {

                animation: spin 1s linear infinite;

            }

        `;

        document.head.appendChild(styleSheet);

    }

})();

## login.js

import { initializeApp } from "https://www.gstatic.com/firebasejs/10.7.2/firebase-app.js";

import {

    getAuth,

    signInWithEmailAndPassword,

    createUserWithEmailAndPassword,

    onAuthStateChanged,

    sendPasswordResetEmail

} from "https://www.gstatic.com/firebasejs/10.7.2/firebase-auth.js";

import {

    getFirestore,

    doc,

    setDoc,

    getDocs,

    collection,

    query,

    where

} from "https://www.gstatic.com/firebasejs/10.7.2/firebase-firestore.js";

const firebaseConfig = {

    apiKey: "AIzaSyAdKXFPMjYamBbNAG2z1Kj8Cp8HjMFcrYA",

    authDomain: "shiftstock-9ba92.firebaseapp.com",

    projectId: "shiftstock-9ba92",

    storageBucket: "shiftstock-9ba92.appspot.com",

    messagingSenderId: "637859468099",

    appId: "1:637859468099:web:26d7984a3296f6b9bb15f6",

    measurementId: "G-EE5FEY81E6"

};

let isProcessingAuth = false;

let app;

let auth;

let db;

try {

    app = initializeApp(firebaseConfig);

    auth = getAuth(app);

    db = getFirestore(app);

} catch (error) {

    try { displayMessage("Could not connect to authentication service.", 'error'); } catch (e) {}

}

let loginForm, signupForm, loginBtn, signupBtn, formTitle, switchLink, forgotPasswordLink, messageArea, loader;

function initializeDOMElements() {

    loginForm = document.getElementById('login-form');

    signupForm = document.getElementById('signup-form');

    loginBtn = document.getElementById('login-btn');

    signupBtn = document.getElementById('signup-btn');

    formTitle = document.getElementById('form-title');

    switchLink = document.getElementById('switch-link');

    forgotPasswordLink = document.getElementById('forgot-password-link');

    messageArea = document.getElementById("message-area");

    loader = document.getElementById('loader');

}

function showLoader() {

    if (loader) loader.style.display = 'block';

    if (loginBtn) loginBtn.disabled = true;

    if (signupBtn) signupBtn.disabled = true;

}

function hideLoader() {

    if (loader) loader.style.display = 'none';

    if (loginBtn) loginBtn.disabled = false;

    if (signupBtn) signupBtn.disabled = false;

}

function displayMessage(message, type = 'info') {

    if (messageArea) {

        messageArea.textContent = message;

        messageArea.className = 'message-area';

        if (type === 'error' || type === 'success') {

            messageArea.classList.add(type);

        }

        messageArea.classList.add('visible');

    }

}

function clearMessages() {

    if (messageArea) messageArea.classList.remove('visible');

}

function getFirebaseErrorMessage(errorCode) {

    switch (errorCode) {

        case "auth/invalid-email": return "Invalid email format.";

        case "auth/user-disabled": return "This account has been disabled.";

        case "auth/invalid-credential": return "Incorrect email or password.";

        case "auth/user-not-found": return "No account found with this email.";

        case "auth/wrong-password": return "Incorrect password.";

        case "auth/email-already-in-use": return "This email is already registered. Please log in or use a different email.";

        case "auth/weak-password": return "Password is too weak (min. 6 characters).";

        case "auth/operation-not-allowed": return "Email/password accounts are not enabled.";

        case "auth/missing-password": return "Please enter your password.";

        case "auth/network-request-failed": return "Network error. Please check your connection.";

        case "auth/missing-email": return "Please enter your email address.";

        default:

            return "An unexpected error occurred. Please try again.";

    }

}

async function isUsernameUnique(username) {

    if (!db) return false;

    const lowerCaseUsername = username.toLowerCase();

    try {

        const usersRef = collection(db, "users");

        const q = query(usersRef, where("username", "==", lowerCaseUsername));

        const querySnapshot = await getDocs(q);

        return querySnapshot.empty;

    } catch (error) {

        displayMessage("Error checking username availability. Please try again.", 'error');

        return false;

    }

}

function generateProfilePictureUrl(username) {

    const initial = username ? username.charAt(0).toUpperCase() : '?';

    const size = 30;

    const bgColor = 'eee';

    const textColor = '333';

    return `https://placehold.co/${size}x${size}/${bgColor}/${textColor}?text=${initial}`;

}

function looksLikeEmail(text) {

    return /\S+@\S+\.\S+/.test(text);

}

function setupAuthObserver() {

    if (auth) {

        onAuthStateChanged(auth, (user) => {

            if (user) {

                if (!isProcessingAuth && !window.location.pathname.includes('dashboard.html')) {

                    window.location.href = "dashboard.html";

                }

            } else {

                isProcessingAuth = false;

                if (!window.location.pathname.includes('dashboard.html')) {

                    showLoginForm();

                }

                hideLoader();

            }

        });

    } else {

        hideLoader();

    }

}

function showLoginForm() {

    if (loginForm && signupForm && formTitle && forgotPasswordLink && switchLink) {

        loginForm.style.display = "block";

        signupForm.style.display = "none";

        formTitle.textContent = "Login";

        forgotPasswordLink.style.display = "block";

        switchLink.innerHTML = '<a href="javascript:void(0);" onclick="toggleForm()">Don\'t have an account? Sign Up</a>';

    }

}

function showSignupForm() {

     if (loginForm && signupForm && formTitle && forgotPasswordLink && switchLink) {

        loginForm.style.display = "none";

        signupForm.style.display = "block";

        formTitle.textContent = "Sign Up";

        forgotPasswordLink.style.display = "none";

        switchLink.innerHTML = '<a href="javascript:void(0);" onclick="toggleForm()">Already have an account? Log in</a>';

     }

}

function setupEventListeners() {

    if (!loginForm || !signupForm) {

        return;

    }

    if (auth) {

        loginForm.addEventListener("submit", (event) => {

            event.preventDefault();

            clearMessages();

            showLoader();

            isProcessingAuth = true;

            const email = document.getElementById("email").value.trim();

            const password = document.getElementById("password").value;

            if (!email || !password) {

                displayMessage("Please enter both email and password.", 'error');

                hideLoader();

                isProcessingAuth = false;

                return;

            }

            signInWithEmailAndPassword(auth, email, password)

                .then((userCredential) => {

                    window.location.href = "dashboard.html";

                })

                .catch((error) => {

                    const friendlyMessage = getFirebaseErrorMessage(error.code);

                    displayMessage(friendlyMessage, 'error');

                    hideLoader();

                    isProcessingAuth = false;

                });

        });

    } else {

        loginForm.addEventListener("submit", (e) => {

            e.preventDefault();

            displayMessage("Authentication service not available.", 'error');

        });

    }

    if (auth && db) {

        signupForm.addEventListener("submit", async (event) => {

            event.preventDefault();

            clearMessages();

            showLoader();

            isProcessingAuth = true;

            const usernameInput = document.getElementById("signup-username");

            const username = usernameInput.value.trim();

            const email = document.getElementById("signup-email").value.trim();

            const password = document.getElementById("signup-password").value;

            const passwordConfirmation = document.getElementById("signup-confirm-password").value;

            if (!username || !email || !password || !passwordConfirmation || password !== passwordConfirmation || password.length < 6 || looksLikeEmail(username)) {

                 if (!username || !email || !password || !passwordConfirmation) displayMessage("Please fill in all required fields.", 'error');

                 else if (password !== passwordConfirmation) displayMessage("Passwords do not match.", 'error');

                 else if (password.length < 6) displayMessage("Password must be at least 6 characters long.", 'error');

                 else if (looksLikeEmail(username)) displayMessage("Please enter a valid username, not an email address.", 'error');

                 hideLoader();

                 isProcessingAuth = false;

                 return;

            }

            try {

                const isUnique = await isUsernameUnique(username);

                if (!isUnique) {

                    displayMessage(`Username "${username}" is already taken. Please choose another.`, 'error');

                    hideLoader();

                    isProcessingAuth = false;

                    return;

                }

                const userCredential = await createUserWithEmailAndPassword(auth, email, password);

                const user = userCredential.user;

                const profilePictureUrl = generateProfilePictureUrl(username);

                const userData = {

                    username: username.toLowerCase(),

                    originalUsername: username,

                    email: user.email,

                    profilePictureUrl: profilePictureUrl,

                    createdAt: new Date(),

                    savedStocks: [],

                    transactions: [],

                    following: [],

                    profitLossAmount: 0,

                    profitLossPercentage: 0,

                    followerCount: 0

                };

                const userDocRef = doc(db, "users", user.uid);

                await setDoc(userDocRef, userData);

                isProcessingAuth = false;

                window.location.href = "dashboard.html";

            } catch (error) {

                const friendlyMessage = getFirebaseErrorMessage(error.code);

                displayMessage(friendlyMessage || `Sign-up failed: ${error.message}`, 'error');

                hideLoader();

                isProcessingAuth = false;

            }

        });

    } else {

        signupForm.addEventListener("submit", (event) => {

            event.preventDefault();

            displayMessage("Authentication or database service unavailable. Cannot sign up.", 'error');

        });

    }

}

window.toggleForm = () => {

    clearMessages();

    hideLoader();

    if (!loginForm || !signupForm) {

        return;

    }

    if (loginForm.style.display === "none") {

        showLoginForm();

    } else {

        showSignupForm();

    }

};

window.handleForgotPassword = () => {

    clearMessages();

    const emailInput = document.getElementById("email");

    let email = emailInput ? emailInput.value.trim() : '';

    if (!email) {

         email = prompt("Please enter your email address to reset your password:");

         if (email === null) return;

         email = email.trim();

    }

    if (!email) { displayMessage("Please enter your email address.", 'error'); return; }

    if (!looksLikeEmail(email)) { displayMessage("Please enter a valid email address format.", 'error'); return; }

    if (auth) {

        showLoader();

        sendPasswordResetEmail(auth, email)

            .then(() => {

                hideLoader();

                displayMessage("Password reset email sent! Please check your inbox (and spam folder).", 'success');

                if (loginForm && loginForm.style.display === "none") { toggleForm(); }

            })

            .catch((error) => {

                hideLoader();

                const friendlyMessage = getFirebaseErrorMessage(error.code);

                displayMessage(`Could not send reset email: ${friendlyMessage}`, 'error');

            });

    } else {

         displayMessage("Authentication service not available.", "error");

    }

};

document.addEventListener('DOMContentLoaded', () => {

    initializeDOMElements();

    setupEventListeners();

    setupAuthObserver();

    const header = document.getElementById('mainHeader');

    if (header) {

        header.style.top = '0';

    }

});

## profile.js

import { initializeApp } from "https://www.gstatic.com/firebasejs/10.7.2/firebase-app.js";

import { getAuth, onAuthStateChanged, signOut } from "https://www.gstatic.com/firebasejs/10.7.2/firebase-auth.js";

import { getFirestore, doc, getDoc } from "https://www.gstatic.com/firebasejs/10.7.2/firebase-firestore.js";

const firebaseConfig = {

    apiKey: "AIzaSyAdKXFPMjYamBbNAG2z1Kj8Cp8HjMFcrYA",

    authDomain: "shiftstock-9ba92.firebaseapp.com",

    projectId: "shiftstock-9ba92",

    storageBucket: "shiftstock-9ba92.appspot.com",

    messagingSenderId: "637859468099",

    appId: "1:637859468099:web:26d7984a3296f6b9bb15f6",

    measurementId: "G-EE5FEY81E6"

};

let app, auth, db;

try {

    app = initializeApp(firebaseConfig);

    auth = getAuth(app);

    db = getFirestore(app);

} catch (error) {

    displayProfileNotification("Error connecting to services. Profile data may not load.", "error");

}

const userGreetingHeader = document.getElementById("user-greeting");

const userProfilePictureHeader = document.getElementById("user-profile-picture");

const logoutButton = document.getElementById("logout-button");

const profilePictureDisplay = document.getElementById("profile-picture-display");

const profileUsernameSpan = document.getElementById("profile-username");

const profileEmailSpan = document.getElementById("profile-email");

const profileMemberSinceSpan = document.getElementById("profile-member-since");

const profileBioTextSpan = document.getElementById("profile-bio-text"); // Added bio display element

const profilePortfolioCountSpan = document.getElementById("profile-portfolio-count");

const profileFollowersSpan = document.getElementById("profile-followers");

const profileTransactionsCountSpan = document.getElementById("profile-transactions-count");

const profileTotalValueSpan = document.getElementById("profile-total-value");

const profileTotalPLSpan = document.getElementById("profile-total-pl");

const profilePortfolioList = document.getElementById("profile-portfolio-list");

const profileTransactionsList = document.getElementById("profile-transactions-list");

const tabButtons = document.querySelectorAll(".tab-button");

const tabContents = document.querySelectorAll(".tab-content");

const notificationAreaProfile = document.getElementById("notification-area-profile");

let currentUser = null;

let currentUserData = null;

let userHoldings = {};

function displayProfileNotification(message, type = 'info', duration = 5000) {

    if (!notificationAreaProfile) return;

    const notification = document.createElement('div');

    notification.classList.add('notification-message', type);

    notification.textContent = message;

    notificationAreaProfile.appendChild(notification);

    setTimeout(() => {

        notification.style.opacity = '0';

        notification.addEventListener('transitionend', () => {

             if (notification.parentNode) {

                notification.parentNode.removeChild(notification);

            }

        }, { once: true });

    }, duration);

}

function formatCurrency(value) {

    const num = typeof value === 'string' ? parseFloat(value) : value;

    if (typeof num !== 'number' || isNaN(num)) return 'N/A';

    return `$${num.toFixed(2)}`;

}

function formatPercentage(value) {

     const num = typeof value === 'string' ? parseFloat(value) : value;

     if (typeof num !== 'number' || isNaN(num)) return 'N/A';

     const sign = num > 0 ? '+' : '';

     return `${sign}${num.toFixed(2)}%`;

}

function getChangeClass(value) {

    const num = typeof value === 'string' ? parseFloat(value) : value;

    if (typeof num !== 'number' || isNaN(num)) return '';

    return num >= 0 ? 'positive' : 'negative';

}

function processTransactions(transactions) {

    const tempHoldings = {};

    if (!transactions) return {};

    const sortedTransactions = [...transactions].sort((a, b) => {

        const dateA = a.timestamp instanceof Date ? a.timestamp.getTime() : (typeof a.timestamp?.toDate === 'function' ? a.timestamp.toDate().getTime() : 0);

        const dateB = b.timestamp instanceof Date ? b.timestamp.getTime() : (typeof b.timestamp?.toDate === 'function' ? b.timestamp.toDate().getTime() : 0);

        return dateA - dateB;

    });

    sortedTransactions.forEach(transaction => {

        const { symbol, type, quantity, price } = transaction;

        if (!symbol || (type !== 'buy' && type !== 'sell') || !(quantity > 0) || !(price > 0)) return;

        const s = symbol.toUpperCase();

        if (!tempHoldings[s]) tempHoldings[s] = { quantity: 0, totalCost: 0 };

        if (type === 'buy') {

            tempHoldings[s].quantity += quantity;

            tempHoldings[s].totalCost += quantity \* price;

        } else if (type === 'sell') {

             const quantityToSell = quantity;

             const currentQuantity = tempHoldings[s].quantity;

             const currentTotalCost = tempHoldings[s].totalCost;

             if (currentQuantity >= quantityToSell) {

                 const proportionSold = quantityToSell / currentQuantity;

                 tempHoldings[s].totalCost -= currentTotalCost \* proportionSold;

                 tempHoldings[s].quantity -= quantityToSell;

                 if (tempHoldings[s].quantity < 1e-9) {

                     tempHoldings[s].quantity = 0;

                     tempHoldings[s].totalCost = 0;

                 }

             } else {

                 tempHoldings[s].quantity = 0;

                 tempHoldings[s].totalCost = 0;

             }

        }

    });

    const processedHoldings = {};

    for (const symbol in tempHoldings) {

        if (tempHoldings[symbol].quantity > 0) {

            processedHoldings[symbol] = {

                quantity: tempHoldings[symbol].quantity,

                totalCost: tempHoldings[symbol].totalCost,

                averagePrice: tempHoldings[symbol].quantity > 0 ? tempHoldings[symbol].totalCost / tempHoldings[symbol].quantity : 0

            };

        }

    }

    return processedHoldings;

}

async function loadAndDisplayUserProfile() {

    if (!currentUser || !db) {

         if (profileUsernameSpan) profileUsernameSpan.textContent = "N/A";

         if (profileEmailSpan) profileEmailSpan.textContent = "N/A";

         if (profileBioTextSpan) profileBioTextSpan.textContent = "Bio not available.";

         if (profilePictureDisplay) profilePictureDisplay.src = 'https://placehold.co/150x150/eee/ccc?text=?';

         if (profileMemberSinceSpan) profileMemberSinceSpan.textContent = "N/A";

         if (profileTotalValueSpan) profileTotalValueSpan.textContent = "N/A";

         if (profileTotalPLSpan) { profileTotalPLSpan.textContent = "N/A"; profileTotalPLSpan.className = 'summary-value'; }

         if (profileFollowersSpan) profileFollowersSpan.textContent = "0";

         if (profilePortfolioCountSpan) profilePortfolioCountSpan.textContent = "0";

         if (profileTransactionsCountSpan) profileTransactionsCountSpan.textContent = "0";

         if (profilePortfolioList) profilePortfolioList.innerHTML = '<p class="info-message">Please log in to see your portfolio.</p>';

         if (profileTransactionsList) profileTransactionsList.innerHTML = '<p class="info-message">Please log in to see your transactions.</p>';

        return;

    }

    const userDocRef = doc(db, "users", currentUser.uid);

    if (profileUsernameSpan) profileUsernameSpan.textContent = "Loading...";

    if (profileEmailSpan) profileEmailSpan.textContent = `Email: Loading... (Private)`;

    if (profileBioTextSpan) profileBioTextSpan.textContent = "Loading bio...";

    if (profilePictureDisplay) profilePictureDisplay.src = currentUser.photoURL || 'https://placehold.co/150x150/eee/ccc?text=Loading';

    if (profileMemberSinceSpan) profileMemberSinceSpan.textContent = "Member Since: Loading...";

    if (profileTotalValueSpan) profileTotalValueSpan.textContent = "Loading...";

    if (profileTotalPLSpan) { profileTotalPLSpan.textContent = "Loading..."; profileTotalPLSpan.className = 'summary-value'; }

    if (profileFollowersSpan) profileFollowersSpan.textContent = "Loading...";

    if (profilePortfolioCountSpan) profilePortfolioCountSpan.textContent = "Loading...";

    if (profileTransactionsCountSpan) profileTransactionsCountSpan.textContent = "Loading...";

    if (profilePortfolioList) profilePortfolioList.innerHTML = '<p class="info-message">Loading portfolio details...</p>';

    if (profileTransactionsList) profileTransactionsList.innerHTML = '<p class="info-message">Loading transaction history...</p>';

    try {

        const docSnap = await getDoc(userDocRef);

        if (docSnap.exists()) {

            currentUserData = docSnap.data();

            userHoldings = processTransactions(currentUserData.transactions);

            const usernameToDisplay = currentUserData.originalUsername || currentUserData.username || "User";

            if (profileUsernameSpan) profileUsernameSpan.textContent = usernameToDisplay;

            if (profileEmailSpan) profileEmailSpan.textContent = `Email: ${currentUserData.email || "N/A"} (Private)`;

            if (profileBioTextSpan) profileBioTextSpan.textContent = currentUserData.bio || "No bio provided.";

            if (profilePictureDisplay) {

                profilePictureDisplay.src = currentUserData.profilePictureUrl || currentUser.photoURL || 'https://placehold.co/150x150/eee/ccc?text=?';

                profilePictureDisplay.alt = `${usernameToDisplay}'s Profile Picture`;

            }

            profilePictureDisplay.onerror = () => {

                 profilePictureDisplay.src = 'https://placehold.co/150x150/eee/ccc?text=Error';

                 profilePictureDisplay.alt = "Error loading picture";

            };

            if (profileMemberSinceSpan && currentUser.metadata?.creationTime) {

                 try {

                     const creationDate = new Date(currentUser.metadata.creationTime);

                     profileMemberSinceSpan.textContent = `Member Since: ${creationDate.toLocaleDateString()}`;

                 } catch (e) {

                     profileMemberSinceSpan.textContent = "Member Since: Date unavailable";

                 }

            } else if (profileMemberSinceSpan) {

                 profileMemberSinceSpan.textContent = "Member Since: Date unavailable";

            }

            const portfolioSymbols = Object.keys(userHoldings);

            const transactionCount = currentUserData.transactions?.length || 0;

            const followerCount = currentUserData.followerCount !== undefined ? currentUserData.followerCount : 0;

            if (profilePortfolioCountSpan) profilePortfolioCountSpan.textContent = portfolioSymbols.length;

            if (profileFollowersSpan) profileFollowersSpan.textContent = followerCount;

            if (profileTransactionsCountSpan) profileTransactionsCountSpan.textContent = transactionCount;

            if (portfolioSymbols.length > 0) {

                 await displayProfilePortfolio(userHoldings);

            } else {

                 if (profileTotalValueSpan) profileTotalValueSpan.textContent = formatCurrency(0);

                 if (profileTotalPLSpan) { profileTotalPLSpan.textContent = `${formatCurrency(0)} (${formatPercentage(0)})`; profileTotalPLSpan.className = 'summary-value positive'; }

                 if (profilePortfolioList) profilePortfolioList.innerHTML = '<p class="info-message">Your portfolio is empty.</p>';

            }

            if (transactionCount > 0) {

                 displayTransactionLog(currentUserData.transactions);

            } else {

                 if (profileTransactionsList) profileTransactionsList.innerHTML = '<p class="info-message">No transactions recorded yet.</p>';

            }

        } else {

            displayProfileNotification("Could not load all profile details from database.", "error");

             const usernameToDisplay = currentUser.displayName || currentUser.email?.split('@')[0] || "User";

             if (profileUsernameSpan) profileUsernameSpan.textContent = usernameToDisplay;

             if (profileEmailSpan) profileEmailSpan.textContent = `Email: ${currentUser.email || "N/A"} (Private)`;

             if (profileBioTextSpan) profileBioTextSpan.textContent = "Bio not found.";

             if (profilePictureDisplay) profilePictureDisplay.src = currentUser.photoURL || 'https://placehold.co/150x150/eee/ccc?text=?';

             profilePictureDisplay.alt = `${usernameToDisplay}'s Profile Picture`;

             profilePictureDisplay.onerror = () => {

                 profilePictureDisplay.src = 'https://placehold.co/150x150/eee/ccc?text=Error';

                 profilePictureDisplay.alt = "Error loading picture";

             };

             if (profileMemberSinceSpan && currentUser.metadata?.creationTime) {

                 try {

                     const creationDate = new Date(currentUser.metadata.creationTime);

                     profileMemberSinceSpan.textContent = `Member Since: ${creationDate.toLocaleDateString()}`;

                 } catch (e) {

                     profileMemberSinceSpan.textContent = "Member Since: Date unavailable";

                 }

             } else if (profileMemberSinceSpan) {

                 profileMemberSinceSpan.textContent = "Member Since: Date unavailable";

             }

             if (profilePortfolioCountSpan) profilePortfolioCountSpan.textContent = "0";

             if (profileFollowersSpan) profileFollowersSpan.textContent = "0";

             if (profileTransactionsCountSpan) profileTransactionsCountSpan.textContent = "0";

             if (profileTotalValueSpan) profileTotalValueSpan.textContent = formatCurrency(0);

             if (profileTotalPLSpan) { profileTotalPLSpan.textContent = `${formatCurrency(0)} (${formatPercentage(0)})`; profileTotalPLSpan.className = 'summary-value positive'; }

             if (profilePortfolioList) profilePortfolioList.innerHTML = '<p class="info-message">Could not load portfolio details.</p>';

             if (profileTransactionsList) profileTransactionsList.innerHTML = '<p class="info-message">Could not load transaction history.</p>';

        }

    } catch (error) {

        displayProfileNotification("Error loading profile details.", "error");

         if (profileUsernameSpan) profileUsernameSpan.textContent = "Error";

         if (profileEmailSpan) profileEmailSpan.textContent = `Email: Error`;

         if (profileBioTextSpan) profileBioTextSpan.textContent = "Error loading bio.";

         if (profilePictureDisplay) { profilePictureDisplay.src = 'https://placehold.co/150x150/eee/ccc?text=Error'; profilePictureDisplay.alt = "Error loading picture"; }

         if (profileMemberSinceSpan) profileMemberSinceSpan.textContent = "Member Since: Error";

         if (profilePortfolioCountSpan) profilePortfolioCountSpan.textContent = "Error";

         if (profileFollowersSpan) profileFollowersSpan.textContent = "Error";

         if (profileTransactionsCountSpan) profileTransactionsCountSpan.textContent = "Error";

         if (profileTotalValueSpan) profileTotalValueSpan.textContent = "Error";

         if (profileTotalPLSpan) { profileTotalPLSpan.textContent = "Error"; profileTotalPLSpan.className = 'summary-value negative'; }

         if (profilePortfolioList) profilePortfolioList.innerHTML = '<p class="error-message">Error loading portfolio details.</p>';

         if (profileTransactionsList) profileTransactionsList.innerHTML = '<p class="error-message">Error loading transaction history.</p>';

    }

    updateHeaderUI();

}

async function displayProfilePortfolio(holdings) {

    if (!profilePortfolioList || !profileTotalValueSpan || !profileTotalPLSpan) return;

    profilePortfolioList.innerHTML = '';

    const symbols = Object.keys(holdings);

    if (symbols.length === 0) {

        profilePortfolioList.innerHTML = '<p class="info-message">Your portfolio is empty.</p>';

         if (profileTotalValueSpan) profileTotalValueSpan.textContent = formatCurrency(0);

         if (profileTotalPLSpan) { profileTotalPLSpan.textContent = `${formatCurrency(0)} (${formatPercentage(0)})`; profileTotalPLSpan.className = 'summary-value positive'; }

        return;

    }

    const fetchPromises = symbols.map(symbol => fetchStockData(symbol));

    const results = await Promise.allSettled(fetchPromises);

    let totalPortfolioValue = 0;

    let totalCostBasis = 0;

    results.forEach(result => {

        if (result.status === 'fulfilled' && result.value?.ticker) {

            const data = result.value;

            const symbol = data.ticker;

            const holding = holdings[symbol];

            if (holding?.quantity > 0) {

                const portfolioItem = document.createElement('div');

                portfolioItem.classList.add('portfolio-item');

                const logoUrl = data.logo || `https://placehold.co/48x48/eee/ccc?text=${symbol[0]}`;

                const companyName = data.name || symbol;

                const currentPrice = typeof data.c === 'number' ? data.c : NaN;

                const quantityOwned = holding.quantity;

                const averageBuyPrice = holding.averagePrice;

                const currentValue = (typeof currentPrice === 'number') ? currentPrice \* quantityOwned : NaN;

                const costBasis = holding.totalCost;

                const profitLoss = (!isNaN(currentValue) && !isNaN(costBasis)) ? currentValue - costBasis : NaN;

                const profitLossPercent = (!isNaN(profitLoss) && costBasis > 0) ? (profitLoss / costBasis) \* 100 : NaN;

                const plClass = getChangeClass(profitLoss);

                 if (!isNaN(currentValue)) totalPortfolioValue += currentValue;

                 if (!isNaN(costBasis)) totalCostBasis += costBasis;

                portfolioItem.innerHTML = `

                    <div class="portfolio-item-header">

                        <img class="portfolio-stock-logo" src="${logoUrl}" alt="${companyName} logo" onerror="this.src='https://placehold.co/48x48/eee/ccc?text=${symbol[0]}'; this.onerror=null;">

                        <div class="portfolio-stock-details">

                            <span class="portfolio-symbol">${symbol}</span>

                            <span class="portfolio-company-name">${companyName}</span>

                        </div>

                    </div>

                    <div class="portfolio-item-stats">

                        <div class="portfolio-item-stat">Qty: <strong>${quantityOwned.toFixed(quantityOwned % 1 === 0 ? 0 : 2)}</strong></div>

                        <div class="portfolio-item-stat">Avg Cost: <strong>${formatCurrency(averageBuyPrice)}</strong></div>

                        <div class="portfolio-item-stat">Current Price: <strong>${formatCurrency(currentPrice)}</strong></div>

                    </div>

                     <div class="portfolio-item-stats">

                         <div class="portfolio-item-stat">Value: <strong>${formatCurrency(currentValue)}</strong></div>

                         <div class="portfolio-item-stat portfolio-item-pl ${plClass}">P/L: <span>${formatCurrency(profitLoss)} (${formatPercentage(profitLossPercent)})</span></div>

                     </div>

                `;

                profilePortfolioList.appendChild(portfolioItem);

            }

        } else {

             const symbolWithError = symbols.find(sym => result.reason?.message?.includes(sym) || result.reason?.detail?.includes(sym));

             if(symbolWithError && holdings[symbolWithError]?.quantity > 0) {

                 const errorItem = document.createElement('div');

                 errorItem.classList.add('portfolio-item', 'error-message');

                 errorItem.textContent = `Could not load data for ${symbolWithError} (${holdings[symbolWithError].quantity.toFixed(holdings[symbolWithError].quantity % 1 === 0 ? 0 : 2)} shares).`;

                 profilePortfolioList.appendChild(errorItem);

                 if (!isNaN(holdings[symbolWithError].totalCost)) totalCostBasis += holdings[symbolWithError].totalCost;

             }

        }

    });

     const totalProfitLoss = totalPortfolioValue - totalCostBasis;

     const totalProfitLossPercent = totalCostBasis > 0 ? (totalProfitLoss / totalCostBasis) \* 100 : 0;

     if (profileTotalValueSpan) profileTotalValueSpan.textContent = formatCurrency(totalPortfolioValue);

     if (profileTotalPLSpan) {

         profileTotalPLSpan.textContent = `${formatCurrency(totalProfitLoss)} (${formatPercentage(totalProfitLossPercent)})`;

         profileTotalPLSpan.className = `summary-value ${getChangeClass(totalProfitLoss)}`;

     }

     if (profilePortfolioList.children.length === 0 && symbols.length > 0) {

         profilePortfolioList.innerHTML = '<p class="error-message">Could not load data for your portfolio holdings.</p>';

     } else if (profilePortfolioList.children.length === 0) {

          profilePortfolioList.innerHTML = '<p class="info-message">Your portfolio is empty.</p>';

     }

}

function displayTransactionLog(transactions) {

    if (!profileTransactionsList) return;

    profileTransactionsList.innerHTML = '';

    if (!transactions || transactions.length === 0) {

        profileTransactionsList.innerHTML = '<p class="info-message">No transactions recorded yet.</p>';

        return;

    }

    const sortedTransactions = [...transactions].sort((a, b) => {

        const dateA = a.timestamp instanceof Date ? a.timestamp.getTime() : (typeof a.timestamp?.toDate === 'function' ? a.timestamp.toDate().getTime() : 0);

        const dateB = b.timestamp instanceof Date ? b.timestamp.getTime() : (typeof b.timestamp?.toDate === 'function' ? b.timestamp.toDate().getTime() : 0);

        return dateB - dateA;

    });

    sortedTransactions.forEach(transaction => {

        const { symbol, type, quantity, price, timestamp } = transaction;

        if (!symbol || !type || !(quantity > 0) || !(price > 0)) return;

        const transactionItem = document.createElement('div');

        transactionItem.classList.add('transaction-item');

        const typeClass = type === 'buy' ? 'buy' : 'sell';

        let formattedDate = 'Date unavailable';

        if (timestamp instanceof Date) {

             formattedDate = timestamp.toLocaleString();

        } else if (typeof timestamp?.toDate === 'function') {

             try { formattedDate = timestamp.toDate().toLocaleString(); } catch(e) { /\* Error formatting date \*/ }

        } else if (typeof timestamp === 'number') {

             try { formattedDate = new Date(timestamp \* 1000).toLocaleString(); } catch(e) { /\* Error formatting date \*/ }

        }

        transactionItem.innerHTML = `

            <div class="transaction-details">

                <div class="transaction-type-symbol">

                    <span class="${typeClass}">${type.toUpperCase()}</span> ${symbol.toUpperCase()}

                </div>

                <div class="transaction-info">

                    Qty: <strong>${quantity.toFixed(quantity % 1 === 0 ? 0 : 2)}</strong> at <strong>${formatCurrency(price)}</strong>

                </div>

            </div>

            <div class="transaction-date">${formattedDate}</div>

        `;

        profileTransactionsList.appendChild(transactionItem);

    });

    if (profileTransactionsList.children.length === 0 && transactions.length > 0) {

         profileTransactionsList.innerHTML = '<p class="error-message">Could not display transaction history.</p>';

    } else if (profileTransactionsList.children.length === 0) {

         profileTransactionsList.innerHTML = '<p class="info-message">No transactions recorded yet.</p>';

    }

}

function updateHeaderUI() {

    const usernameToDisplay = currentUserData?.originalUsername || currentUserData?.username || currentUser?.displayName || currentUser?.email?.split('@')[0] || "User";

    const profilePictureUrl = currentUserData?.profilePictureUrl || currentUser?.photoURL || 'https://placehold.co/30x30/eee/ccc?text=?';

    const headerGreeting = document.getElementById("user-greeting");

    const headerProfilePic = document.getElementById("user-profile-picture");

    if (headerGreeting) {

        headerGreeting.textContent = `Hello, ${usernameToDisplay}!`;

    }

    if (headerProfilePic) {

        headerProfilePic.src = profilePictureUrl;

        headerProfilePic.alt = `${usernameToDisplay}'s Profile Picture`;

         headerProfilePic.onerror = () => {

            headerProfilePic.src = 'https://placehold.co/30x30/eee/ccc?text=!';

            headerProfilePic.alt = "Error loading picture";

         };

    }

}

if (auth) {

    onAuthStateChanged(auth, (user) => {

        if (user) {

            currentUser = user;

            loadAndDisplayUserProfile();

        } else {

            currentUser = null;

            currentUserData = null;

            userHoldings = {};

            loadAndDisplayUserProfile(); // This will set N/A states

            window.location.href = 'login.html';

        }

    });

} else {

    displayProfileNotification("Authentication service not available.", "error");

    if (userGreetingHeader) userGreetingHeader.textContent = "Error";

    if (userProfilePictureHeader) userProfilePictureHeader.src = 'https://placehold.co/30x30/eee/ccc?text=!';

     loadAndDisplayUserProfile();

}

if (logoutButton) {

    logoutButton.addEventListener("click", () => {

        if (auth) {

            signOut(auth).catch((error) => {

                displayProfileNotification("Error signing out.", "error");

            });

        }

    });

}

tabButtons.forEach(button => {

    button.addEventListener('click', () => {

        const targetTab = button.dataset.tab;

        tabButtons.forEach(btn => btn.classList.remove('active'));

        tabContents.forEach(content => content.classList.remove('active'));

        button.classList.add('active');

        document.getElementById(`${targetTab}-tab`).classList.add('active');

    });

});

async function fetchStockData(symbol) {

     try {

         const response = await fetch(`http://127.0.0.1:8000/get-company-data?symbol=${symbol}`);

         if (!response.ok) {

             // const errorDetail = await response.text();

             return { ticker: symbol, error: true, message: `Could not load data (${response.status})` };

         }

         const data = await response.json();

          if (!data || typeof data !== 'object' || !data.ticker) {

               return { ticker: symbol, error: true, message: `Invalid data format` };

          }

         return data;

     } catch (error) {

         return { ticker: symbol, error: true, message: `Network error` };

     }

}

## settings.js

import { initializeApp } from "https://www.gstatic.com/firebasejs/10.7.2/firebase-app.js";

import {

    getAuth,

    onAuthStateChanged,

    signOut,

    updateProfile,

    reauthenticateWithCredential,

    EmailAuthProvider,

    updatePassword,

    deleteUser

} from "https://www.gstatic.com/firebasejs/10.7.2/firebase-auth.js";

import {

    getFirestore,

    doc,

    getDoc,

    updateDoc,

    collection,

    query,

    where,

    getDocs

} from "https://www.gstatic.com/firebasejs/10.7.2/firebase-firestore.js";

const firebaseConfig = {

    apiKey: "AIzaSyAdKXFPMjYamBbNAG2z1Kj8Cp8HjMFcrYA",

    authDomain: "shiftstock-9ba92.firebaseapp.com",

    projectId: "shiftstock-9ba92",

    storageBucket: "shiftstock-9ba92.appspot.com",

    messagingSenderId: "637859468099",

    appId: "1:637859468099:web:26d7984a3296f6b9bb15f6",

    measurementId: "G-EE5FEY81E6"

};

let app, auth, db;

try {

    app = initializeApp(firebaseConfig);

    auth = getAuth(app);

    db = getFirestore(app);

} catch (error) {

    displaySettingsNotification("Error connecting to services. Some features may not work.", "error");

}

const userGreetingHeader = document.getElementById("user-greeting");

const userProfilePictureHeader = document.getElementById("user-profile-picture");

const logoutButton = document.getElementById("logout-button");

const profileSettingsForm = document.getElementById("profile-settings-form");

const usernameInput = document.getElementById("username");

const emailInput = document.getElementById("email");

const profilePictureUrlInput = document.getElementById("profilePictureUrl");

const profilePicturePreview = document.getElementById("profile-picture-preview");

const bioTextarea = document.getElementById("bio"); // Added bio textarea

const saveProfileButton = document.getElementById("save-profile-button");

const passwordChangeForm = document.getElementById("password-change-form");

const currentPasswordInput = document.getElementById("currentPassword");

const newPasswordInput = document.getElementById("newPassword");

const confirmNewPasswordInput = document.getElementById("confirmNewPassword");

const changePasswordButton = document.getElementById("change-password-button");

const deleteAccountButton = document.getElementById("delete-account-button");

const notificationAreaSettings = document.getElementById("notification-area-settings");

let currentUser = null;

let currentUserData = null;

function displaySettingsNotification(message, type = 'info', duration = 5000) {

    if (!notificationAreaSettings) return;

    const notification = document.createElement('div');

    notification.classList.add('notification-message', type);

    notification.textContent = message;

    notificationAreaSettings.appendChild(notification);

    setTimeout(() => {

        notification.style.opacity = '0';

        notification.addEventListener('transitionend', () => {

             if (notification.parentNode) {

                notification.parentNode.removeChild(notification);

            }

        }, { once: true });

    }, duration);

}

function setButtonLoading(button, loadingText = "Saving...") {

    if (button) {

        button.disabled = true;

        button.dataset.originalText = button.textContent;

        button.textContent = loadingText;

    }

}

function resetButtonLoading(button) {

    if (button && button.dataset.originalText) {

        button.disabled = false;

        button.textContent = button.dataset.originalText;

    }

}

async function isUsernameUnique(username) {

    if (!db) {

        return false;

    }

    const lowerCaseUsername = username.toLowerCase();

    try {

        const usersRef = collection(db, "users");

        const q = query(usersRef, where("username", "==", lowerCaseUsername));

        const querySnapshot = await getDocs(q);

        return querySnapshot.empty;

    } catch (error) {

        displaySettingsNotification("Error checking username availability. Please try again.", 'error');

        return false;

    }

}

if (auth) {

    onAuthStateChanged(auth, async (user) => {

        if (user) {

            currentUser = user;

            await loadUserProfile();

            updateHeaderUI();

        } else {

            currentUser = null;

            currentUserData = null;

            window.location.href = 'login.html';

        }

    });

} else {

    displaySettingsNotification("Authentication service not available.", "error");

    if (userGreetingHeader) userGreetingHeader.textContent = "Error";

    if (userProfilePictureHeader) userProfilePictureHeader.src = 'https://placehold.co/30x30/eee/ccc?text=!';

}

async function loadUserProfile() {

    if (!currentUser || !db) return;

    if (emailInput) emailInput.value = currentUser.email || "";

    const userDocRef = doc(db, "users", currentUser.uid);

    try {

        const docSnap = await getDoc(userDocRef);

        if (docSnap.exists()) {

            currentUserData = docSnap.data();

            if (usernameInput) usernameInput.value = currentUserData.originalUsername || currentUserData.username || "";

            if (profilePictureUrlInput) profilePictureUrlInput.value = currentUserData.profilePictureUrl || "";

            if (profilePicturePreview) profilePicturePreview.src = currentUserData.profilePictureUrl || 'https://placehold.co/100x100/eee/ccc?text=?';

            if (bioTextarea) bioTextarea.value = currentUserData.bio || ""; // Load bio

        } else {

            displaySettingsNotification("Could not load all profile details.", "error");

            if (usernameInput) usernameInput.value = currentUser.displayName || "";

            if (profilePicturePreview) profilePicturePreview.src = 'https://placehold.co/100x100/eee/ccc?text=?';

            if (bioTextarea) bioTextarea.value = "";

        }

    } catch (error) {

        displaySettingsNotification("Error loading profile details.", "error");

        if (usernameInput) usernameInput.value = currentUser.displayName || "";

        if (profilePicturePreview) profilePicturePreview.src = 'https://placehold.co/100x100/eee/ccc?text=?';

        if (bioTextarea) bioTextarea.value = "";

    }

}

function updateHeaderUI() {

    if (currentUserData && userGreetingHeader) {

        userGreetingHeader.textContent = `Hello, ${currentUserData.originalUsername || currentUserData.username || "User"}!`;

    } else if (currentUser && userGreetingHeader) {

        userGreetingHeader.textContent = `Hello, ${currentUser.displayName || currentUser.email.split('@')[0]}!`;

    }

    if (currentUserData && userProfilePictureHeader) {

        userProfilePictureHeader.src = currentUserData.profilePictureUrl || 'https://placehold.co/30x30/eee/ccc?text=?';

    } else if (currentUser && userProfilePictureHeader) {

         userProfilePictureHeader.src = currentUser.photoURL || 'https://placehold.co/30x30/eee/ccc?text=?';

    }

}

if (logoutButton) {

    logoutButton.addEventListener("click", () => {

        if (auth) {

            signOut(auth).catch((error) => {

                displaySettingsNotification("Error signing out.", "error");

            });

        }

    });

}

if (profilePictureUrlInput && profilePicturePreview) {

    profilePictureUrlInput.addEventListener('input', () => {

        const newUrl = profilePictureUrlInput.value.trim();

        if (newUrl) {

            profilePicturePreview.src = newUrl;

        } else {

            profilePicturePreview.src = currentUserData?.profilePictureUrl || 'https://placehold.co/100x100/eee/ccc?text=?';

        }

    });

    profilePicturePreview.onerror = () => {

        profilePicturePreview.src = 'https://placehold.co/100x100/eee/ccc?text=Invalid';

    };

}

if (profileSettingsForm) {

    profileSettingsForm.addEventListener("submit", async (event) => {

        event.preventDefault();

        if (!currentUser || !db) {

            displaySettingsNotification("Not connected. Please try again.", "error");

            return;

        }

        const newUsername = usernameInput.value.trim();

        const newProfilePictureUrl = profilePictureUrlInput.value.trim();

        const newBio = bioTextarea.value.trim(); // Get bio content

        const currentUsername = currentUserData?.originalUsername || currentUserData?.username || '';

        if (!newUsername) {

            displaySettingsNotification("Username cannot be empty.", "error");

            return;

        }

        if (newUsername.toLowerCase() !== currentUsername.toLowerCase()) {

             setButtonLoading(saveProfileButton, "Checking username...");

             const isUnique = await isUsernameUnique(newUsername);

             if (!isUnique) {

                 displaySettingsNotification(`Username "${newUsername}" is already taken. Please choose another.`, 'error');

                 resetButtonLoading(saveProfileButton);

                 return;

             }

        }

        setButtonLoading(saveProfileButton, "Saving...");

        try {

            await updateProfile(currentUser, {

                displayName: newUsername, // Auth profile only stores displayName and photoURL

                photoURL: newProfilePictureUrl

            });

            const userDocRef = doc(db, "users", currentUser.uid);

            const updates = {

                originalUsername: newUsername,

                username: newUsername.toLowerCase(),

                profilePictureUrl: newProfilePictureUrl,

                bio: newBio // Add bio to Firestore updates

            };

            await updateDoc(userDocRef, updates);

            await loadUserProfile(); // Reload data to reflect changes, including the new bio

            updateHeaderUI();

            displaySettingsNotification("Profile updated successfully!", "success");

        } catch (error) {

            displaySettingsNotification(`Error updating profile: ${error.message}`, "error");

        } finally {

            resetButtonLoading(saveProfileButton);

        }

    });

}

if (passwordChangeForm) {

    passwordChangeForm.addEventListener("submit", async (event) => {

        event.preventDefault();

        if (!currentUser) {

            displaySettingsNotification("Not authenticated.", "error");

            return;

        }

        const currentPass = currentPasswordInput.value;

        const newPass = newPasswordInput.value;

        const confirmNewPass = confirmNewPasswordInput.value;

        if (newPass.length < 6) {

            displaySettingsNotification("New password must be at least 6 characters long.", "error");

            return;

        }

        if (newPass !== confirmNewPass) {

            displaySettingsNotification("New passwords do not match.", "error");

            return;

        }

        setButtonLoading(changePasswordButton, "Changing...");

        try {

            const credential = EmailAuthProvider.credential(currentUser.email, currentPass);

            await reauthenticateWithCredential(currentUser, credential);

            await updatePassword(currentUser, newPass);

            displaySettingsNotification("Password changed successfully!", "success");

            passwordChangeForm.reset();

        } catch (error) {

            let friendlyMessage = "Error changing password.";

            if (error.code === 'auth/wrong-password') {

                friendlyMessage = "Incorrect current password. Please try again.";

            } else if (error.code === 'auth/too-many-requests') {

                friendlyMessage = "Too many attempts. Please try again later.";

            } else if (error.code === 'auth/requires-recent-login') {

                friendlyMessage = "This operation is sensitive and requires recent authentication. Please log out and log back in before trying again.";

            } else {

                friendlyMessage = `Error: ${error.message}`;

            }

            displaySettingsNotification(friendlyMessage, "error");

        } finally {

            resetButtonLoading(changePasswordButton);

        }

    });

}

if (deleteAccountButton) {

    deleteAccountButton.addEventListener("click", async () => {

        if (!currentUser) {

            displaySettingsNotification("Not authenticated.", "error");

            return;

        }

        const confirmation = prompt("This action is irreversible. You will lose all your data.\nType your email address to confirm account deletion:");

        if (confirmation === null) {

            return;

        }

        if (confirmation.toLowerCase() !== currentUser.email.toLowerCase()) {

            displaySettingsNotification("Email confirmation does not match. Account deletion cancelled.", "info");

            return;

        }

        const reauthPassword = prompt("For security, please re-enter your password to delete your account:");

        if (reauthPassword === null) {

             displaySettingsNotification("Password not provided. Account deletion cancelled.", "info");

            return;

        }

        setButtonLoading(deleteAccountButton, "Deleting...");

        try {

            const credential = EmailAuthProvider.credential(currentUser.email, reauthPassword);

            await reauthenticateWithCredential(currentUser, credential);

            await deleteUser(currentUser);

            // Note: Deleting Firestore document for the user might be desired here too.

            // await deleteDoc(doc(db, "users", currentUser.uid));

            displaySettingsNotification("Account deleted successfully. You will be logged out.", "success", 7000);

            setTimeout(() => {

                window.location.href = "login.html";

            }, 7000);

        } catch (error) {

             let friendlyMessage = "Error deleting account.";

            if (error.code === 'auth/wrong-password') {

                friendlyMessage = "Incorrect password. Account deletion failed.";

            } else if (error.code === 'auth/too-many-requests') {

                friendlyMessage = "Too many attempts. Please try again later.";

            } else if (error.code === 'auth/requires-recent-login') {

                friendlyMessage = "This operation is sensitive and requires recent authentication. Please log out and log back in before trying again.";

            } else {

                friendlyMessage = `Error: ${error.message}`;

            }

            displaySettingsNotification(friendlyMessage, "error");

            resetButtonLoading(deleteAccountButton);

        }

    });

}

## user\_profile.js

import { initializeApp } from "https://www.gstatic.com/firebasejs/10.7.2/firebase-app.js";

import { getAuth, onAuthStateChanged, signOut } from "https://www.gstatic.com/firebasejs/10.7.2/firebase-auth.js";

import { getFirestore, doc, getDoc, updateDoc, arrayUnion, arrayRemove, increment } from "https://www.gstatic.com/firebasejs/10.7.2/firebase-firestore.js";

const firebaseConfig = {

    apiKey: "AIzaSyAdKXFPMjYamBbNAG2z1Kj8Cp8HjMFcrYA",

    authDomain: "shiftstock-9ba92.firebaseapp.com",

    projectId: "shiftstock-9ba92",

    storageBucket: "shiftstock-9ba92.appspot.com",

    messagingSenderId: "637859468099",

    appId: "1:637859468099:web:26d7984a3296f6f9bb15f6",

    measurementId: "G-EE5FEY81E6"

};

let app, auth, db;

try {

    app = initializeApp(firebaseConfig);

    auth = getAuth(app);

    db = getFirestore(app);

} catch (error) {

    displayNotification("Error connecting to services. User profile data may not load.", "error");

}

const userGreetingHeader = document.getElementById("user-greeting");

const userProfilePictureHeader = document.getElementById("user-profile-picture");

const logoutButton = document.getElementById("logout-button");

const userProfilePictureDisplay = document.getElementById("user-profile-picture-display");

const userProfileUsernameSpan = document.getElementById("user-profile-username");

const userProfileMemberSinceSpan = document.getElementById("user-profile-member-since");

const userProfilePLSpan = document.getElementById("user-profile-pl");

const userProfilePortfolioCountSpan = document.getElementById("user-profile-portfolio-count");

const userProfileFollowersSpan = document.getElementById("user-profile-followers");

const userProfileTransactionsCountSpan = document.getElementById("user-profile-transactions-count");

const userProfileTotalValueSpan = document.getElementById("user-profile-total-value");

const userProfileTotalPLSummarySpan = document.getElementById("user-profile-total-pl-summary");

const userProfilePortfolioList = document.getElementById("user-profile-portfolio-list");

const userProfileTransactionsList = document.getElementById("user-profile-transactions-list");

const tabButtons = document.querySelectorAll(".content-tabs .tab-button");

const tabContents = document.querySelectorAll(".profile-content-section .tab-content");

const followUserButton = document.getElementById("follow-user-button");

const notificationArea = document.getElementById("notification-area-user-profile");

const userProfileBioTextSpan = document.getElementById("user-profile-bio-text");

let currentUser = null;

let displayedUserId = null;

let displayedUserData = null;

let currentUserFollowing = new Set();

function displayNotification(message, type = 'info', duration = 5000) {

    if (!notificationArea) return;

    const notification = document.createElement('div');

    notification.classList.add('notification-message', type);

    notification.textContent = message;

    notificationArea.appendChild(notification);

    setTimeout(() => {

        notification.style.opacity = '0';

        notification.addEventListener('transitionend', () => {

             if (notification.parentNode) {

                notification.parentNode.removeChild(notification);

            }

        }, { once: true });

    }, duration);

}

function formatCurrency(value) {

    const num = typeof value === 'string' ? parseFloat(value) : value;

    if (typeof num !== 'number' || isNaN(num)) return 'N/A';

    return `$${num.toFixed(2)}`;

}

function formatPercentage(value) {

     const num = typeof value === 'string' ? parseFloat(value) : value;

     if (typeof num !== 'number' || isNaN(num)) return 'N/A';

     const sign = num > 0 ? '+' : '';

     return `${sign}${num.toFixed(2)}%`;

}

function getChangeClass(value) {

    const num = typeof value === 'string' ? parseFloat(value) : value;

    if (typeof num !== 'number' || isNaN(num)) return '';

    return num >= 0 ? 'positive' : 'negative';

}

// Processes transactions for the displayed user to get detailed holdings (qty, cost, avgPrice)

function processDisplayedUserTransactionsAndHoldings(transactions) {

    const tempHoldings = {};

    if (!transactions) return {};

    const sortedTransactions = [...transactions].sort((a, b) => {

        const dateA = a.timestamp instanceof Date ? a.timestamp.getTime() : (typeof a.timestamp?.toDate === 'function' ? a.timestamp.toDate().getTime() : 0);

        const dateB = b.timestamp instanceof Date ? b.timestamp.getTime() : (typeof b.timestamp?.toDate === 'function' ? b.timestamp.toDate().getTime() : 0);

        return dateA - dateB;

    });

    sortedTransactions.forEach(transaction => {

        const { symbol, type, quantity, price } = transaction; // Assuming price is stored with transaction

        if (!symbol || (type !== 'buy' && type !== 'sell') || !(quantity > 0) || !(price > 0)) return;

        const s = symbol.toUpperCase();

        if (!tempHoldings[s]) tempHoldings[s] = { quantity: 0, totalCost: 0 };

        if (type === 'buy') {

            tempHoldings[s].quantity += quantity;

            tempHoldings[s].totalCost += quantity \* price;

        } else if (type === 'sell') {

             const quantityToSell = quantity;

             const currentQuantity = tempHoldings[s].quantity;

             const currentTotalCost = tempHoldings[s].totalCost;

             if (currentQuantity >= quantityToSell) {

                 const proportionSold = quantityToSell / currentQuantity;

                 tempHoldings[s].totalCost -= currentTotalCost \* proportionSold;

                 tempHoldings[s].quantity -= quantityToSell;

                 if (tempHoldings[s].quantity < 1e-9) {

                     tempHoldings[s].quantity = 0;

                     tempHoldings[s].totalCost = 0;

                 }

             } else {

                 tempHoldings[s].quantity = 0;

                 tempHoldings[s].totalCost = 0;

             }

        }

    });

    const processedHoldings = {};

    for (const symbol in tempHoldings) {

        if (tempHoldings[symbol].quantity > 0) {

            processedHoldings[symbol] = {

                quantity: tempHoldings[symbol].quantity,

                totalCost: tempHoldings[symbol].totalCost,

                averagePrice: tempHoldings[symbol].quantity > 0 ? tempHoldings[symbol].totalCost / tempHoldings[symbol].quantity : 0

            };

        }

    }

    return processedHoldings;

}

function updateHeaderUI(loggedInUser) {

    const headerGreeting = document.getElementById("user-greeting");

    const headerProfilePic = document.getElementById("user-profile-picture");

    if (loggedInUser && headerGreeting) {

        const usernameToDisplay = loggedInUser.displayName || loggedInUser.email?.split('@')[0] || "User";

        const profilePictureUrl = loggedInUser.photoURL || 'https://placehold.co/30x30/eee/ccc?text=?';

        headerGreeting.textContent = `Hello, ${usernameToDisplay}!`;

         if (headerProfilePic) {

            headerProfilePic.src = profilePictureUrl;

            headerProfilePic.alt = `${usernameToDisplay}'s Profile Picture`;

             headerProfilePic.onerror = () => {

                headerProfilePic.src = 'https://placehold.co/30x30/eee/ccc?text=!';

                headerProfilePic.alt = "Error loading picture";

             };

        }

    } else {

         if (headerGreeting) headerGreeting.textContent = "Guest";

         if (headerProfilePic) headerProfilePic.src = 'https://placehold.co/30x30/eee/ccc?text=?';

    }

}

if (auth) {

    onAuthStateChanged(auth, async (user) => {

        currentUser = user;

        if (user) {

            await loadCurrentUserFollowing();

            loadAndDisplayUserProfile(getQueryParameter('userId'));

            updateHeaderUI(user);

        } else {

            updateHeaderUI(null);

            loadAndDisplayUserProfile(getQueryParameter('userId')); // Load profile even if no one is logged in

        }

    });

} else {

    displayNotification("Authentication service not available. Cannot load user profile.", "error");

    const headerGreeting = document.getElementById("user-greeting");

    const headerProfilePic = document.getElementById("user-profile-picture");

    if (headerGreeting) headerGreeting.textContent = "Error";

    if (headerProfilePic) headerProfilePic.src = 'https://placehold.co/30x30/eee/ccc?text=!';

}

async function loadAndDisplayUserProfile(userIdToLoad) {

     if (!db) {

         displayNotification("Database connection error. Cannot load profile.", "error");

         setUserProfileErrorState("Database Error");

         return;

     }

    if (!userIdToLoad) {

        if (currentUser) {

            window.location.href = `profile.html`; // Redirect to own profile if logged in and no specific user requested

        } else {

             displayNotification("No user specified to display.", "error"); // If not logged in and no user in URL

             setUserProfileErrorState("No User Specified");

        }

        return;

    }

    if (currentUser && userIdToLoad === currentUser.uid) {

        window.location.href = `profile.html`; // Redirect to own profile page if trying to view self via user\_profile.html

        return;

    }

    displayedUserId = userIdToLoad;

    setUserProfileLoadingState();

    const userDocRef = doc(db, "users", displayedUserId);

    try {

        const docSnap = await getDoc(userDocRef);

        if (docSnap.exists()) {

            displayedUserData = docSnap.data();

            const usernameToDisplay = displayedUserData.originalUsername || displayedUserData.username || "User";

            if (userProfileUsernameSpan) userProfileUsernameSpan.textContent = usernameToDisplay;

            const profilePictureUrl = displayedUserData.profilePictureUrl || `https://placehold.co/200x200/eee/ccc?text=${usernameToDisplay.charAt(0).toUpperCase()}`;

            if (userProfilePictureDisplay) {

                userProfilePictureDisplay.src = profilePictureUrl;

                userProfilePictureDisplay.alt = `${usernameToDisplay}'s Profile Picture`;

                userProfilePictureDisplay.onerror = () => {

                     userProfilePictureDisplay.src = 'https://placehold.co/200x200/eee/ccc?text=Error';

                     userProfilePictureDisplay.alt = "Error loading picture";

                };

            }

            if (userProfileBioTextSpan) userProfileBioTextSpan.textContent = displayedUserData.bio || "No bio provided.";

            let memberSinceText = "Member Since: Date unavailable";

             if (displayedUserData.createdAt instanceof Date) {

                  memberSinceText = `Member Since: ${displayedUserData.createdAt.toLocaleDateString()}`;

             } else if (typeof displayedUserData.createdAt?.toDate === 'function') {

                  try { memberSinceText = `Member Since: ${displayedUserData.createdAt.toDate().toLocaleDateString()}`; }

                  catch(e) { /\* Error formatting date \*/ }

             }

            if (userProfileMemberSinceSpan) userProfileMemberSinceSpan.textContent = memberSinceText;

            // P/L in bio will be updated by displayUserProfilePortfolio

            const transactionCount = displayedUserData.transactions?.length || 0;

            const followerCount = displayedUserData.followerCount !== undefined ? displayedUserData.followerCount : 0;

            const detailedHoldings = processDisplayedUserTransactionsAndHoldings(displayedUserData.transactions);

            const portfolioStockCount = Object.keys(detailedHoldings).length;

            if (userProfilePortfolioCountSpan) userProfilePortfolioCountSpan.textContent = portfolioStockCount;

            if (userProfileFollowersSpan) userProfileFollowersSpan.textContent = followerCount;

            if (userProfileTransactionsCountSpan) userProfileTransactionsCountSpan.textContent = transactionCount;

            await displayUserProfilePortfolio(detailedHoldings); // Pass detailed holdings

            displayUserProfileTransactionLog(displayedUserData.transactions);

        } else {

            displayNotification("User profile not found.", "error");

            setUserProfileErrorState("User Not Found");

        }

    } catch (error) {

        displayNotification("Error loading user profile details.", "error");

        setUserProfileErrorState("Error Loading Profile");

    } finally {

         if (currentUser && followUserButton && displayedUserId) {

              followUserButton.dataset.userId = displayedUserId;

              updateFollowButtonState(displayedUserId, currentUserFollowing.has(displayedUserId));

              followUserButton.style.display = 'inline-block';

         } else if (followUserButton) {

             followUserButton.style.display = 'none';

         }

    }

}

function setUserProfileLoadingState() {

     if (userProfileUsernameSpan) userProfileUsernameSpan.textContent = "Loading...";

     if (userProfilePictureDisplay) userProfilePictureDisplay.src = 'https://placehold.co/200x200/eee/ccc?text=Loading';

     if (userProfilePictureDisplay) userProfilePictureDisplay.alt = "Loading...";

     if (userProfileMemberSinceSpan) userProfileMemberSinceSpan.textContent = "Member Since: Loading...";

     if (userProfilePLSpan) { userProfilePLSpan.textContent = "Total P/L: Loading..."; userProfilePLSpan.className = 'profile-bio'; }

     if (userProfilePortfolioCountSpan) userProfilePortfolioCountSpan.textContent = "...";

     if (userProfileFollowersSpan) userProfileFollowersSpan.textContent = "...";

     if (userProfileTransactionsCountSpan) userProfileTransactionsCountSpan.textContent = "...";

     if (userProfileTotalValueSpan) userProfileTotalValueSpan.textContent = "Loading...";

     if (userProfileTotalPLSummarySpan) { userProfileTotalPLSummarySpan.textContent = "Loading..."; userProfileTotalPLSummarySpan.className = 'summary-value'; }

     if (userProfilePortfolioList) userProfilePortfolioList.innerHTML = '<p class="info-message">Loading portfolio details...</p>';

     if (userProfileTransactionsList) userProfileTransactionsList.innerHTML = '<p class="info-message">Loading recent transactions...</p>';

     if (followUserButton) { followUserButton.disabled = true; followUserButton.textContent = "Loading..."; }

}

function setUserProfileErrorState(message = "Could not load profile.") {

     if (userProfileUsernameSpan) userProfileUsernameSpan.textContent = `Error: ${message}`;

     if (userProfilePictureDisplay) userProfilePictureDisplay.src = 'https://placehold.co/200x200/eee/ccc?text=Error';

     if (userProfilePictureDisplay) userProfilePictureDisplay.alt = "Error";

     if (userProfileMemberSinceSpan) userProfileMemberSinceSpan.textContent = "Member Since: Error";

     if (userProfilePLSpan) { userProfilePLSpan.textContent = "Total P/L: Error"; userProfilePLSpan.className = 'profile-bio negative'; }

     if (userProfilePortfolioCountSpan) userProfilePortfolioCountSpan.textContent = "-";

     if (userProfileFollowersSpan) userProfileFollowersSpan.textContent = "-";

     if (userProfileTransactionsCountSpan) userProfileTransactionsCountSpan.textContent = "-";

     if (userProfileTotalValueSpan) userProfileTotalValueSpan.textContent = "Error";

     if (userProfileTotalPLSummarySpan) { userProfileTotalPLSummarySpan.textContent = "Error"; userProfileTotalPLSummarySpan.className = 'summary-value negative'; }

     if (userProfilePortfolioList) userProfilePortfolioList.innerHTML = `<p class="error-message">${message}</p>`;

     if (userProfileTransactionsList) userProfileTransactionsList.innerHTML = `<p class="error-message">${message}</p>`;

     if (followUserButton) { followUserButton.disabled = true; followUserButton.textContent = "Error"; }

}

function getQueryParameter(name) {

    const urlParams = new URLSearchParams(window.location.search);

    return urlParams.get(name);

}

async function displayUserProfilePortfolio(detailedHoldings) { // Renamed parameter for clarity

    if (!userProfilePortfolioList || !userProfileTotalValueSpan || !userProfileTotalPLSummarySpan) return;

    userProfilePortfolioList.innerHTML = '';

    const symbols = Object.keys(detailedHoldings);

    if (symbols.length === 0) {

        userProfilePortfolioList.innerHTML = '<p class="info-message">This user\'s portfolio is empty or private.</p>';

        if (userProfileTotalValueSpan) userProfileTotalValueSpan.textContent = formatCurrency(0);

        if (userProfileTotalPLSummarySpan) {

            userProfileTotalPLSummarySpan.textContent = `${formatCurrency(0)} (${formatPercentage(0)})`;

            userProfileTotalPLSummarySpan.className = 'summary-value positive';

        }

        if (userProfilePLSpan) { // Also update bio P/L

            userProfilePLSpan.textContent = `Total P/L: ${formatCurrency(0)} (${formatPercentage(0)})`;

            userProfilePLSpan.className = 'profile-bio positive';

        }

        return;

    }

    const fetchPromises = symbols.map(symbol => fetchStockData(symbol));

    const results = await Promise.allSettled(fetchPromises);

    let overallTotalPortfolioValue = 0;

    let overallTotalCostBasis = 0;

    let hasAddedItem = false;

    results.forEach(result => {

        if (result.status === 'fulfilled' && result.value?.ticker) {

            const stockData = result.value;

            const symbol = stockData.ticker;

            const holdingDetails = detailedHoldings[symbol]; // Get qty, totalCost, avgPrice

            if (holdingDetails && holdingDetails.quantity > 0) {

                const portfolioItem = document.createElement('div');

                portfolioItem.classList.add('portfolio-item');

                const logoUrl = stockData.logo || `https://placehold.co/48x48/eee/ccc?text=${symbol[0]}`;

                const companyName = stockData.name || symbol;

                const currentPrice = typeof stockData.c === 'number' ? stockData.c : NaN;

                const quantityOwned = holdingDetails.quantity;

                const averageBuyPrice = holdingDetails.averagePrice;

                const itemCostBasis = holdingDetails.totalCost;

                const itemCurrentValue = (!isNaN(currentPrice)) ? currentPrice \* quantityOwned : NaN;

                const itemProfitLoss = (!isNaN(itemCurrentValue) && !isNaN(itemCostBasis)) ? itemCurrentValue - itemCostBasis : NaN;

                const itemProfitLossPercent = (!isNaN(itemProfitLoss) && itemCostBasis > 0) ? (itemProfitLoss / itemCostBasis) \* 100 : NaN;

                const itemPlClass = getChangeClass(itemProfitLoss);

                portfolioItem.innerHTML = `

                    <div class="portfolio-item-header">

                        <img class="portfolio-stock-logo" src="${logoUrl}" alt="${companyName} logo" onerror="this.src='https://placehold.co/48x48/eee/ccc?text=${symbol[0]}'; this.onerror=null;">

                        <div class="portfolio-stock-details">

                            <span class="portfolio-symbol">${symbol}</span>

                            <span class="portfolio-company-name">${companyName}</span>

                        </div>

                    </div>

                    <div class="portfolio-item-stats">

                        <div class="portfolio-item-stat">Qty: <strong>${quantityOwned.toFixed(quantityOwned % 1 === 0 ? 0 : 2)}</strong></div>

                        <div class="portfolio-item-stat">Avg Cost: <strong>${formatCurrency(averageBuyPrice)}</strong></div>

                        <div class="portfolio-item-stat">Current Price: <strong>${formatCurrency(currentPrice)}</strong></div>

                    </div>

                     <div class="portfolio-item-stats">

                         <div class="portfolio-item-stat">Value: <strong>${formatCurrency(itemCurrentValue)}</strong></div>

                         <div class="portfolio-item-stat portfolio-item-pl ${itemPlClass}">P/L: <span>${formatCurrency(itemProfitLoss)} (${formatPercentage(itemProfitLossPercent)})</span></div>

                     </div>

                `;

                userProfilePortfolioList.appendChild(portfolioItem);

                hasAddedItem = true;

                if (!isNaN(itemCurrentValue)) overallTotalPortfolioValue += itemCurrentValue;

                if (!isNaN(itemCostBasis)) overallTotalCostBasis += itemCostBasis;

            }

        } else {

             const symbolWithError = symbols.find(sym => result.reason?.message?.includes(sym) || result.reason?.detail?.includes(sym));

             if(symbolWithError && detailedHoldings[symbolWithError]?.quantity > 0) {

                 const errorItem = document.createElement('div');

                 errorItem.classList.add('portfolio-item', 'error-message');

                 errorItem.textContent = `Could not load market data for ${symbolWithError}.`;

                 userProfilePortfolioList.appendChild(errorItem);

                 hasAddedItem = true; // Still count it as an attempt to display

                 // Add its cost basis if available, current value will be NaN or 0

                 if (detailedHoldings[symbolWithError] && !isNaN(detailedHoldings[symbolWithError].totalCost)) {

                    overallTotalCostBasis += detailedHoldings[symbolWithError].totalCost;

                 }

             }

        }

    });

    const overallTotalPLAmount = overallTotalPortfolioValue - overallTotalCostBasis;

    const overallTotalPLPercent = overallTotalCostBasis > 0 ? (overallTotalPLAmount / overallTotalCostBasis) \* 100 : 0;

    if (userProfileTotalValueSpan) userProfileTotalValueSpan.textContent = formatCurrency(overallTotalPortfolioValue);

    if (userProfileTotalPLSummarySpan) {

        userProfileTotalPLSummarySpan.textContent = `${formatCurrency(overallTotalPLAmount)} (${formatPercentage(overallTotalPLPercent)})`;

        userProfileTotalPLSummarySpan.className = `summary-value ${getChangeClass(overallTotalPLAmount)}`;

    }

    if (userProfilePLSpan) { // Update P/L in bio section

        userProfilePLSpan.textContent = `Total P/L: ${formatCurrency(overallTotalPLAmount)} (${formatPercentage(overallTotalPLPercent)})`;

        userProfilePLSpan.className = `profile-bio ${getChangeClass(overallTotalPLAmount)}`;

    }

     if (!hasAddedItem && symbols.length > 0) {

         userProfilePortfolioList.innerHTML = '<p class="error-message">Could not load data for this user\'s portfolio holdings.</p>';

     } else if (!hasAddedItem) { // This implies symbols.length was 0

          userProfilePortfolioList.innerHTML = '<p class="info-message">This user\'s portfolio is empty or private.</p>';

     }

}

function displayUserProfileTransactionLog(transactions) {

    if (!userProfileTransactionsList) return;

    userProfileTransactionsList.innerHTML = '';

    if (!transactions || transactions.length === 0) {

        userProfileTransactionsList.innerHTML = '<p class="info-message">No transactions recorded yet or transactions are private.</p>';

        return;

    }

    const sortedTransactions = [...transactions].sort((a, b) => {

        const dateA = a.timestamp instanceof Date ? a.timestamp.getTime() : (typeof a.timestamp?.toDate === 'function' ? a.timestamp.toDate().getTime() : 0);

        const dateB = b.timestamp instanceof Date ? b.timestamp.getTime() : (typeof b.timestamp?.toDate === 'function' ? b.timestamp.toDate().getTime() : 0);

        return dateB - dateA;

    });

    const recentTransactions = sortedTransactions.slice(0, 10);

    recentTransactions.forEach(transaction => {

        const { symbol, type, quantity, price, timestamp } = transaction;

        if (!symbol || !type || !(quantity > 0) || !(price > 0)) return;

        const transactionItem = document.createElement('div');

        transactionItem.classList.add('transaction-item');

        const typeClass = type === 'buy' ? 'buy' : 'sell';

        let formattedDate = 'Date unavailable';

        if (timestamp instanceof Date) {

             formattedDate = timestamp.toLocaleString();

        } else if (typeof timestamp?.toDate === 'function') {

             try { formattedDate = timestamp.toDate().toLocaleString(); } catch(e) { /\* Error formatting date \*/ }

        } else if (typeof timestamp === 'number') {

             try { formattedDate = new Date(timestamp \* 1000).toLocaleString(); } catch(e) { /\* Error formatting date \*/ }

        }

        transactionItem.innerHTML = `

            <div class="transaction-details">

                <div class="transaction-type-symbol">

                    <span class="${typeClass}">${type.toUpperCase()}</span> ${symbol.toUpperCase()}

                </div>

                <div class="transaction-info">

                    Qty: <strong>${quantity.toFixed(quantity % 1 === 0 ? 0 : 2)}</strong> at <strong>${formatCurrency(price)}</strong>

                </div>

            </div>

            <div class="transaction-date">${formattedDate}</div>

        `;

        userProfileTransactionsList.appendChild(transactionItem);

    });

    if (userProfileTransactionsList.children.length === 0 && transactions.length > 0) {

         userProfileTransactionsList.innerHTML = '<p class="info-message">Transaction history is private or could not be loaded.</p>';

    } else if (userProfileTransactionsList.children.length === 0) {

         userProfileTransactionsList.innerHTML = '<p class="info-message">No transactions recorded yet or transactions are private.</p>';

    } else if (transactions.length > recentTransactions.length) {

        const moreMessage = document.createElement('p');

        moreMessage.classList.add('info-message');

        moreMessage.textContent = `Showing ${recentTransactions.length} most recent trades.`;

        userProfileTransactionsList.appendChild(moreMessage);

    }

}

async function loadCurrentUserFollowing() {

    if (!currentUser || !db) {

        currentUserFollowing.clear();

        return;

    }

    const userDocRef = doc(db, "users", currentUser.uid);

    try {

        const docSnap = await getDoc(userDocRef);

        if (docSnap.exists()) {

            const data = docSnap.data();

            currentUserFollowing = new Set(data.following || []);

        } else {

            currentUserFollowing.clear();

        }

    } catch (error) {

        currentUserFollowing.clear();

    }

}

function updateFollowButtonState(targetUserId, isFollowing) {

    if (!followUserButton || followUserButton.dataset.userId !== targetUserId) return;

    followUserButton.textContent = isFollowing ? 'Following' : 'Follow';

    followUserButton.classList.toggle('following', isFollowing);

    followUserButton.disabled = false;

}

if (logoutButton) {

    logoutButton.addEventListener("click", () => {

        if (auth) {

            signOut(auth).catch((error) => {

                displayNotification("Error signing out.", "error");

            });

        }

    });

}

tabButtons.forEach(button => {

    button.addEventListener('click', () => {

        const targetTab = button.dataset.tab;

        tabButtons.forEach(btn => btn.classList.remove('active'));

        tabContents.forEach(content => content.classList.remove('active'));

        button.classList.add('active');

        document.getElementById(`${targetTab}-tab`).classList.add('active');

    });

});

if (followUserButton) {

    followUserButton.addEventListener('click', async () => {

        if (!currentUser) {

            displayNotification("You must be logged in to follow users.", "info");

            return;

        }

        if (!displayedUserId || displayedUserId === currentUser.uid) {

             return;

        }

        followUserButton.disabled = true;

        const isCurrentlyFollowing = currentUserFollowing.has(displayedUserId);

        try {

            const currentUserDocRef = doc(db, "users", currentUser.uid);

            const targetUserDocRef = doc(db, "users", displayedUserId);

            if (isCurrentlyFollowing) {

                await updateDoc(currentUserDocRef, { following: arrayRemove(displayedUserId) });

                await updateDoc(targetUserDocRef, { followerCount: increment(-1) });

                currentUserFollowing.delete(displayedUserId);

                displayNotification("User unfollowed.", "info", 1500);

            } else {

                await updateDoc(currentUserDocRef, { following: arrayUnion(displayedUserId) });

                 await updateDoc(targetUserDocRef, { followerCount: increment(1) });

                currentUserFollowing.add(displayedUserId);

                displayNotification("User followed.", "success", 1500);

            }

             if (userProfileFollowersSpan) {

                 const currentCount = parseInt(userProfileFollowersSpan.textContent, 10);

                 if (!isNaN(currentCount)) {

                     userProfileFollowersSpan.textContent = isCurrentlyFollowing ? currentCount - 1 : currentCount + 1;

                 } else {

                     loadAndDisplayUserProfile(displayedUserId); // Fallback to reload if current count is not a number

                 }

             }

        } catch (error) {

            displayNotification(`Failed to ${isCurrentlyFollowing ? 'unfollow' : 'follow'} user.`, "error");

            await loadCurrentUserFollowing(); // Resync following state on error

            await loadAndDisplayUserProfile(displayedUserId); // Resync full profile data on error

        } finally {

            updateFollowButtonState(displayedUserId, currentUserFollowing.has(displayedUserId));

        }

    });

}

async function fetchStockData(symbol) {

     try {

         const response = await fetch(`http://127.0.0.1:8000/get-company-data?symbol=${symbol}`);

         if (!response.ok) {

             // const errorDetail = await response.text(); // Potentially remove for production

             return { ticker: symbol, error: true, message: `Could not load data (${response.status})` };

         }

         const data = await response.json();

          if (!data || typeof data !== 'object' || !data.ticker) {

               return { ticker: symbol, error: true, message: `Invalid data format` };

          }

         return data;

     } catch (error) {

         return { ticker: symbol, error: true, message: `Network error` };

     }

}

## dashboard.html

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Dashboard - INVESTA</title>

    <link href="https://fonts.googleapis.com/css2?family=Inconsolata:wght@400;700&display=swap" rel="stylesheet">

    <link rel="stylesheet" href="https://fonts.googleapis.com/css2?family=Material+Symbols+Outlined:opsz,wght,FILL,GRAD@20..48,100..700,0..1,-50..200" />

    <link rel="stylesheet" href="dashboard.css">

</head>

<body>

    <header>

        <nav>

            <div class="logo">

                <a href="dashboard.html">INVESTA</a>

             </div>

             <div class="header-right">

                 <div class="nav-links">

                     <ul>

                         <li><a href="dashboard.html" class="active">Dashboard</a></li>

                         <li><a href="profile.html">My Profile</a></li>

                         <li><a href="settings.html">Settings</a></li>

                     </ul>

                 </div>

                 <div class="user-info">

                     <div id="profile-picture-container">

                        <img id="user-profile-picture" src="https://placehold.co/30x30/eee/ccc?text=?" alt="Profile Picture">

                     </div>

                     <div class="user-text-details">

                         <span id="user-greeting">Loading...</span>

                         </div>

                    <button id="logout-button">Logout</button>

                 </div>

             </div>

        </nav>

    </header>

    <div class="dashboard-layout">

        <aside id="left-panel">

            <div class="market-status-container left-panel-section">

                 <div class="market-status-indicator" id="market-status-indicator">

                    <div id="market-status-dot"></div>

                    <span id="market-status-text">Checking Status...</span>

                 </div>

            </div>

            <div class="saved-stocks-container left-panel-section">

                <div class="section-header">

                     <h3>My Watchlist</h3>

                     <button class="refresh-button" id="refresh-watchlist-btn" title="Refresh Watchlist">

                         <span class="icon material-symbols-outlined">refresh</span>

                     </button>

                 </div>

                <div id="saved-stock-list" class="scrollable-list">

                    <p class="info-message">Loading watchlist...</p>

                </div>

            </div>

            <div class="top-stocks-container left-panel-section">

                 <div class="section-header">

                     <h3>Market Movers</h3>

                      <button class="refresh-button" id="refresh-movers-btn" title="Refresh Market Movers">

                          <span class="icon material-symbols-outlined">refresh</span>

                      </button>

                 </div>

                 <div class="scrollable-list" id="stock-list">

                     <p class="info-message">Loading market data...</p>

                 </div>

            </div>

        </aside>

        <main class="right-content">

            <div class="search-container">

                <input type="text" id="stock-search-input" placeholder="Search stock symbol (e.g., AAPL, MSFT)">

                <button id="search-button">Search</button>

            </div>

            <div id="search-result-display">

                 <p class="info-message">Search for a stock or select one from the lists.</p>

            </div>

            <div id="portfolio-container">

                 <div class="portfolio-header">

                     <h3>My Portfolio</h3>

                      <button class="refresh-button" id="refresh-portfolio-btn" title="Refresh Portfolio">

                           <span class="icon material-symbols-outlined">refresh</span>

                       </button>

                 </div>

                 <div class="portfolio-summary">

                     <div id="total-portfolio-value">Value: <span>Calculating...</span></div>

                     <div id="total-cost-basis">Cost: <span>Calculating...</span></div>

                     <div id="total-profit-loss">P/L: <span class="value">Calculating...</span></div>

                 </div>

                 <div id="portfolio-list" class="scrollable-list">

                    <p class="info-message">Loading portfolio...</p>

                </div>

            </div>

             <div id="market-news-container">

                  <div class="news-header">

                      <h3>Market News</h3>

                       <button class="refresh-button" id="refresh-news-btn" title="Refresh News">

                           <span class="icon material-symbols-outlined">refresh</span>

                       </button>

                  </div>

                 <div id="news-list" class="news-list scrollable-list">

                     <p class="info-message">Loading news...</p>

                 </div>

             </div>

             <div id="users-directory-container">

                  <div class="users-directory-header">

                      <h3>Other Users</h3>

                  </div>

                  <div class="search-container users-search-container">

                      <input type="text" id="user-search-input" placeholder="Search for users...">

                     <button id="user-search-button">Search User</button>

                 </div>

                 <div id="user-search-results-display">

                     <p class="info-message">Search for a user by username.</p>

                 </div>

                 <div id="users-list" class="users-list scrollable-list">

                     <p class="info-message">Loading users...</p>

                 </div>

             </div>

        </main>

    </div>

    <div id="notification-area"></div>

    <div id="conversations-tab" class="conversations-tab collapsed">

        <div class="tab-header">

            <span>Global Chat</span> <button id="toggle-conversations" class="toggle-button">▼</button>

        </div>

        <div class="tab-content">

            <div id="chat-messages" class="chat-messages">

                <p class="info-message">Connecting to chat...</p>

            </div>

            <div class="chat-input-area">

                <input type="text" id="chat-input" placeholder="Type your message...">

                <button id="send-chat-button">Send</button>

            </div>

        </div>

    </div>

    <script src="https://www.gstatic.com/firebasejs/10.7.2/firebase-app.js" type="module"></script>

    <script src="https://www.gstatic.com/firebasejs/10.7.2/firebase-auth.js" type="module"></script>

    <script src="https://www.gstatic.com/firebasejs/10.7.2/firebase-firestore.js" type="module"></script>

    <script type="module" src="dashboard.js" defer></script>

</body>

</html>

## dashboard.css

\* {

    margin: 0;

    padding: 0;

    box-sizing: border-box;

}

body {

    font-family: 'Inconsolata', monospace;

    background-color: #f8f9fa;

    color: #333;

    overflow: hidden;

    height: 100vh;

    display: flex;

    flex-direction: column;

}

header {

    position: fixed;

    width: 100%;

    top: 0;

    left: 0;

    z-index: 1000;

    background-color: rgba(255, 255, 255, 0.95);

    backdrop-filter: blur(5px);

    padding: 10px 20px;

    border-bottom: 1px solid #eee;

    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.05);

    flex-shrink: 0;

}

nav {

    display: flex;

    justify-content: space-between;

    align-items: center;

    max-width: 1600px;

    margin: 0 auto;

}

.logo {

    font-size: 18px;

    font-weight: 700;

    margin-right: 20px;

}

.logo a {

    text-decoration: none;

    color: #333;

}

.header-right {

    display: flex;

    align-items: center;

    gap: 15px;

}

.user-info {

    display: flex;

    align-items: center;

    gap: 10px;

    padding: 4px 10px;

    border-left: 1px solid #eee;

}

#user-greeting {

     font-size: 14px;

     font-weight: 600;

     color: #333;

}

#profile-picture-container {

    border-radius: 50%;

    overflow: hidden;

    width: 30px;

    height: 30px;

    flex-shrink: 0;

    background-color: #eee;

     border: 1px solid #ccc;

}

.user-info img {

     width: 100%;

     height: 100%;

     object-fit: cover;

     display: block;

}

.user-text-details {

     display: flex;

     flex-direction: column;

     align-items: flex-start;

}

.user-text-details span {

     align-self: center;

}

#logout-button {

    background-color: transparent;

    color: #dc3545;

    border: 1px solid #dc3545;

    padding: 6px 12px;

    border-radius: 5px;

    cursor: pointer;

    font-size: 13px;

    font-weight: 600;

    transition: all 0.2s ease;

}

#logout-button:hover {

    background-color: #dc3545;

    color: white;

    transform: translateY(-1px);

}

.nav-links ul {

    list-style-type: none;

    display: flex;

    gap: 15px;

    margin: 0;

    padding: 0;

}

.nav-links ul li a {

    text-decoration: none;

    font-size: 14px;

    color: #333;

    font-weight: 600;

    padding: 8px 10px;

    border-radius: 5px;

    transition: background-color 0.2s ease, color 0.2s ease;

}

 .nav-links ul li a:hover,

 .nav-links ul li a.active {

    background-color: #e9ecef;

}

.dashboard-layout {

    display: flex;

    flex-grow: 1;

    margin-top: 55px;

    height: calc(100vh - 55px);

    overflow: hidden;

}

#left-panel {

    width: 280px;

    border-right: 1px solid #ddd;

    background-color: #fff;

    height: 100%;

    display: flex;

    flex-direction: column;

    padding: 0;

    flex-shrink: 0;

    overflow-y: auto;

}

.right-content {

    flex-grow: 1;

    padding: 20px;

    overflow-y: auto;

    height: 100%;

    background-color: #f8f9fa;

}

.left-panel-section {

    padding: 12px 15px;

    border-bottom: 1px solid #eee;

    display: flex;

    flex-direction: column;

}

.section-header {

     display: flex;

     justify-content: space-between;

     align-items: center;

     margin-bottom: 10px;

}

.section-header h3 {

    font-size: 14px;

    font-weight: 700;

    color: #333;

    margin: 0;

}

.refresh-button {

     background: none;

     border: none;

     padding: 2px;

     cursor: pointer;

     color: #6c757d;

     transition: color 0.2s, transform 0.2s;

     display: inline-flex;

     align-items: center;

     justify-content: center;

}

.refresh-button .icon {

    font-size: 18px;

}

.refresh-button:hover {

    color: #0056b3;

    transform: rotate(90deg);

}

.refresh-button:disabled {

    color: #adb5bd;

    cursor: not-allowed;

    transform: none;

}

.scrollable-list {

    flex-grow: 1;

    overflow-y: auto;

    min-height: 80px;

    scrollbar-width: thin;

    scrollbar-color: #ccc #f0f0f0;

}

.scrollable-list::-webkit-scrollbar { width: 6px; }

.scrollable-list::-webkit-scrollbar-track { background: #f0f0f0; border-radius: 3px; }

.scrollable-list::-webkit-scrollbar-thumb { background-color: #ccc; border-radius: 3px; }

.empty-list-message, .info-message, .error-message {

    font-size: 13px;

    color: #777;

    text-align: center;

    padding: 15px 10px;

}

.error-message {

    color: #dc3545;

    font-weight: bold;

}

.market-status-indicator {

    display: flex;

    align-items: center;

    font-size: 14px;

    color: #333;

}

#market-status-dot {

    width: 10px;

    height: 10px;

    border-radius: 50%;

    margin-right: 8px;

    flex-shrink: 0;

    border: 1px solid rgba(0,0,0,0.1);

}

#market-status-text {

    font-weight: bold;

}

.market-open #market-status-dot { background-color: #28a745; }

.market-closed #market-status-dot { background-color: #dc3545; }

.market-unknown #market-status-dot { background-color: #adb5bd; }

.saved-stocks-container { flex-grow: 1; min-height: 0; }

.top-stocks-container { flex-grow: 1; min-height: 0; border-bottom: none; }

.stock-row {

    display: flex;

    justify-content: space-between;

    align-items: center;

    padding: 8px 5px;

    gap: 8px;

    border-bottom: 1px solid #f0f0f0;

    cursor: pointer;

    transition: background-color 0.2s ease;

}

.stock-row:last-child { border-bottom: none; }

.stock-row:hover { background-color: #f1f3f5; }

.stock-info { display: flex; align-items: center; gap: 8px; flex-shrink: 1; min-width: 0; }

.stock-logo { width: 24px; height: 24px; border-radius: 50%; flex-shrink: 0; object-fit: contain; background-color: #eee; }

.stock-details { display: flex; flex-direction: column; gap: 1px; flex-shrink: 1; min-width: 0; }

.symbol-container { display: flex; align-items: center; gap: 4px; }

.symbol { font-weight: bold; font-size: 12px; white-space: nowrap; overflow: hidden; text-overflow: ellipsis; }

.company-name { font-size: 10px; color: #555; white-space: nowrap; overflow: hidden; text-overflow: ellipsis; }

.stock-price-info { display: flex; align-items: center; gap: 8px; flex-shrink: 0; text-align: right;}

.stock-price { font-weight: bold; font-size: 12px; color: black; }

.stock-change { font-weight: bold; font-size: 11px; padding: 2px 4px; border-radius: 3px; }

.positive { color: #198754; background-color: rgba(25, 135, 84, 0.1); }

.negative { color: #dc3545; background-color: rgba(220, 53, 69, 0.1); }

.save-remove-button {

    background: none;

    border: none;

    padding: 0;

    cursor: pointer;

    font-size: 18px;

    font-family: 'Material Symbols Outlined';

    font-weight: normal;

    font-style: normal;

    display: inline-flex;

    align-items: center;

    justify-content: center;

    line-height: 1;

    text-transform: none;

    letter-spacing: normal;

    word-wrap: normal;

    white-space: nowrap;

    direction: ltr;

    -webkit-font-smoothing: antialiased;

    color: #adb5bd;

    transition: color 0.2s ease, transform 0.1s ease;

    flex-shrink: 0;

    width: 24px;

    height: 24px;

}

.save-remove-button:hover {

    color: #ffc107;

    transform: scale(1.1);

}

.save-remove-button.saved {

    color: #ffc107;

    font-variation-settings: 'FILL' 1;

}

.save-remove-button::before {

    content: "star";

}

.search-container {

    display: flex;

    padding: 0 0 20px 0;

    gap: 10px;

    align-items: center;

    margin-bottom: 20px;

    border-bottom: 1px solid #ddd;

}

.users-search-container {

     padding: 0 0 15px 0;

     margin-bottom: 15px;

     border-bottom: 1px solid #eee;

}

#stock-search-input, #user-search-input, #chat-input {

    flex-grow: 1;

    padding: 10px 12px;

    border: 1px solid #ccc;

    border-radius: 5px;

    font-size: 14px;

    font-family: 'Inconsolata', monospace;

    transition: border-color 0.3s, box-shadow 0.3s;

}

#stock-search-input:focus, #user-search-input:focus, #chat-input:focus {

    outline: none;

    border-color: #0056b3;

    box-shadow: 0 0 5px rgba(0, 86, 179, 0.2);

}

#search-button, #user-search-button, #send-chat-button {

    padding: 10px 20px;

    background-color: #0056b3;

    color: white;

    border: none;

    border-radius: 5px;

    cursor: pointer;

    font-size: 14px;

    font-weight: bold;

    transition: background-color 0.3s ease;

}

#search-button:hover, #user-search-button:hover, #send-chat-button:hover {

    background-color: #00408d;

}

#search-result-display, #user-search-results-display {

    padding: 20px;

    border: 1px solid #ddd;

    border-radius: 8px;

    min-height: 100px;

    background-color: #fff;

    box-shadow: 0 2px 5px rgba(0,0,0,0.05);

    margin-bottom: 25px;

}

#user-search-results-display {

    margin-bottom: 15px;

}

.loading-message {

    font-size: 14px;

    color: #555;

    text-align: center;

    padding: 30px 10px;

}

.large-stock-display {

    display: flex;

    flex-direction: column;

    gap: 20px;

    padding: 25px;

    border-radius: 10px;

    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.08);

    border: 1px solid #e0e0e0;

    background-color: #fff;

}

.large-stock-display h2,

.large-stock-display .large-symbol,

.large-stock-display .large-price,

.large-stock-display .key-metrics strong {

    color: #111;

}

.company-header { display: flex; align-items: center; gap: 15px; padding-bottom: 15px; border-bottom: 1px solid #eee; }

.large-stock-logo { width: 50px; height: 50px; border-radius: 50%; object-fit: contain; background-color: #eee; }

.company-title { display: flex; flex-direction: column; flex-grow: 1; }

.title-line-1 { display: flex; align-items: center; gap: 10px; margin-bottom: 2px; }

.large-symbol { font-size: 22px; font-weight: bold; color: #333; }

.company-title .save-remove-button { font-size: 24px; width: 30px; height: 30px; }

.large-company-name { font-size: 16px; color: #555; }

.company-website { font-size: 13px; color: #0056b3; text-decoration: none; margin-top: 4px; word-break: break-all; }

.company-website:hover { text-decoration: underline; }

.price-info { display: flex; align-items: baseline; gap: 10px; flex-wrap: wrap; }

.large-price { font-size: 32px; font-weight: bold; color: #000; }

.large-price-change { font-size: 16px; font-weight: bold; padding: 3px 6px; border-radius: 4px; }

.key-metrics {

    display: grid;

    grid-template-columns: repeat(auto-fit, minmax(140px, 1fr));

    gap: 12px 15px;

    padding-top: 15px;

    border-top: 1px solid #eee;

}

.metric-item { font-size: 13px; color: #555; }

.metric-item span { display: block; margin-bottom: 2px; font-weight: 600; }

.metric-item strong { color: #333; font-size: 14px; }

.stock-actions-large { display: flex; gap: 15px; margin-top: 15px; justify-content: flex-start; padding-top: 15px; border-top: 1px solid #eee; }

.stock-actions-large .action-button {

    padding: 10px 20px;

    border-radius: 6px;

    cursor: pointer;

    font-size: 15px;

    font-weight: bold;

    border: none;

    transition: background-color 0.3s ease, transform 0.2s ease;

    display: inline-flex;

    align-items: center;

    gap: 5px;

}

.stock-actions-large .action-button:hover:not(:disabled) { transform: translateY(-2px); }

.stock-actions-large .action-button:disabled { opacity: 0.6; cursor: not-allowed; transform: none; background-color: #ccc !important; color: #666 !important; }

.buy-button-large { background-color: #28a745; color: white; }

.buy-button-large:hover:not(:disabled) { background-color: #218838; }

.sell-button-large { background-color: #dc3545; color: white; }

.sell-button-large:hover:not(:disabled) { background-color: #c82333; } /\* Corrected class name \*/

.buy-button-large::before { content: 'shopping\_cart'; font-family: 'Material Symbols Outlined'; font-size: 18px; }

.sell-button-large::before { content: 'sell'; font-family: 'Material Symbols Outlined'; font-size: 18px; }

#portfolio-container {

     padding: 20px;

     border: 1px solid #ddd;

     border-radius: 8px;

     background-color: #fff;

     box-shadow: 0 2px 5px rgba(0,0,0,0.05);

     margin-bottom: 25px;

}

 .portfolio-header {

     display: flex;

     justify-content: space-between;

     align-items: center;

     border-bottom: 1px solid #eee;

     padding-bottom: 10px;

     margin-bottom: 15px;

 }

 .portfolio-header h3 { font-size: 18px; color: #333; margin: 0; }

 .portfolio-header .refresh-button { margin-left: 10px; }

.portfolio-summary { display: flex; gap: 15px; margin-bottom: 15px; flex-wrap: wrap; font-size: 14px; }

#total-portfolio-value, #total-profit-loss, #total-cost-basis { font-weight: 600; color: #555; }

#total-portfolio-value span, #total-profit-loss span, #total-cost-basis span { font-weight: bold; color: #333; margin-left: 5px; }

#total-profit-loss .value.positive { color: #198754; }

.portfolio-profit-loss .negative { color: #dc3545; } /\* This seems to be a duplicate or mis-scoped rule \*/

.portfolio-item {

    display: flex;

    justify-content: space-between;

    align-items: center;

    padding: 12px 0;

    border-bottom: 1px solid #eee;

    gap: 15px;

    cursor: pointer;

    transition: background-color 0.2s ease;

}

.portfolio-item:last-child { border-bottom: none; }

.portfolio-item:hover { background-color: #f9f9f9; }

.portfolio-stock-info { display: flex; align-items: center; gap: 10px; flex-grow: 1; min-width: 0; }

.portfolio-stock-logo { width: 28px; height: 28px; border-radius: 50%; object-fit: contain; background-color: #eee; }

.portfolio-stock-details { display: flex; flex-direction: column; gap: 2px; overflow: hidden; }

.portfolio-symbol { font-weight: bold; font-size: 13px; color: #333; white-space: nowrap; overflow: hidden; text-overflow: ellipsis; }

.portfolio-company-name { font-size: 11px; color: #555; white-space: nowrap; overflow: hidden; text-overflow: ellipsis; }

.portfolio-value-info { display: flex; flex-direction: column; align-items: flex-end; gap: 4px; flex-shrink: 0; text-align: right; }

.portfolio-current-value strong, .portfolio-average-price strong { color: #333; font-weight: bold; margin-left: 4px; }

.portfolio-profit-loss { font-size: 13px; font-weight: bold; }

.portfolio-profit-loss span { margin-left: 4px; }

.portfolio-profit-loss .positive { color: #198754; }

.portfolio-profit-loss .negative { color: #dc3545; }

#market-news-container {

     padding: 20px;

     border: 1px solid #ddd;

     border-radius: 8px;

     background-color: #fff;

     box-shadow: 0 2px 5px rgba(0,0,0,0.05);

     margin-bottom: 25px;

}

 .news-header {

     display: flex;

     justify-content: space-between;

     align-items: center;

     border-bottom: 1px solid #eee;

     padding-bottom: 10px;

     margin-bottom: 15px;

 }

 .news-header h3 { font-size: 18px; color: #333; margin: 0; }

 .news-header .refresh-button { margin-left: 10px; }

 .news-list { max-height: 400px; overflow-y: auto; padding-right: 5px; }

 .news-item {

     border-bottom: 1px solid #f0f0f0;

     padding: 12px 0;

     display: flex;

     gap: 15px;

 }

 .news-item:last-child { border-bottom: none; }

 .news-image { width: 80px; height: 60px; object-fit: cover; border-radius: 4px; flex-shrink: 0; background-color: #eee; }

 .news-content { display: flex; flex-direction: column; gap: 4px; }

 .news-headline { font-size: 14px; font-weight: bold; color: #333; text-decoration: none; }

 .news-headline:hover { color: #0056b3; text-decoration: underline; }

 .news-summary { font-size: 12px; color: #555; line-height: 1.5; }

 .news-source-time { font-size: 11px; color: #888; }

#users-directory-container {

     padding: 20px;

     border: 1px solid #ddd;

     border-radius: 8px;

     background-color: #fff;

     box-shadow: 0 2px 5px rgba(0,0,0,0.05);

     margin-bottom: 25px;

}

.users-directory-header {

     display: flex;

     justify-content: space-between;

     align-items: center;

     border-bottom: 1px solid #eee;

     padding-bottom: 10px;

     margin-bottom: 15px;

}

.users-directory-header h3 { font-size: 18px; color: #333; margin: 0; }

.users-list { max-height: 400px; overflow-y: auto; padding-right: 5px; }

.user-directory-item {

    display: flex;

    justify-content: space-between;

    align-items: center;

    padding: 10px 0;

    border-bottom: 1px solid #eee;

    gap: 15px;

}

.user-directory-item:last-child { border-bottom: none; }

.user-directory-info { display: flex; align-items: center; gap: 10px; flex-grow: 1; min-width: 0; }

.user-directory-pfp { width: 32px; height: 32px; border-radius: 50%; object-fit: cover; flex-shrink: 0; background-color: #eee; }

.user-directory-details {

    display: flex;

    flex-direction: column;

    gap: 4px;

    flex-grow: 1;

    min-width: 0;

}

.user-directory-username { font-weight: bold; font-size: 14px; color: #333; white-space: nowrap; overflow: hidden; text-overflow: ellipsis; }

.user-directory-stats {

    display: flex;

    gap: 15px;

    font-size: 12px;

    color: #555;

    white-space: nowrap;

}

.user-pl.positive { color: #198754; }

.user-pl.negative { color: #dc3545; }

.user-directory-actions {

    display: flex;

    gap: 8px;

    align-items: center;

    flex-shrink: 0;

}

.follow-button {

    padding: 5px 12px;

    border-radius: 5px;

    cursor: pointer;

    font-size: 12px;

    font-weight: bold;

    border: 1px solid #0056b3;

    background-color: #0056b3;

    color: white;

    transition: background-color 0.2s ease, color 0.2s ease, transform 0.1s ease;

    flex-shrink: 0;

}

.follow-button:hover:not(:disabled) {

    background-color: #00408d;

    transform: translateY(-1px);

}

.follow-button.following {

    background-color: transparent;

    color: #0056b3;

    border-color: #0056b3;

}

.follow-button.following:hover:not(:disabled) {

    background-color: rgba(0, 86, 179, 0.1);

}

.follow-button:disabled {

    background-color: #ccc;

    border-color: #ccc;

    color: #666;

    cursor: not-allowed;

    transform: none;

}

.message-button {

    padding: 5px 12px;

    border-radius: 5px;

    cursor: pointer;

    font-size: 12px;

    font-weight: bold;

    border: 1px solid #0d6efd;

    background-color: #0d6efd;

    color: white;

    transition: background-color 0.2s ease, color 0.2s ease, transform 0.1s ease;

    flex-shrink: 0;

}

.message-button:hover:not(:disabled) {

    background-color: #0b5ed7;

    border-color: #0b5ed7;

    transform: translateY(-1px);

}

.message-button:disabled {

    background-color: #ccc;

    border-color: #ccc;

    color: #666;

    cursor: not-allowed;

    transform: none;

}

#notification-area {

    position: fixed;

    bottom: 20px;

    right: 20px;

    z-index: 1050;

    display: flex;

    flex-direction: column;

    gap: 10px;

    align-items: flex-end;

}

.notification {

    background-color: #333;

    color: white;

    padding: 12px 20px;

    border-radius: 6px;

    box-shadow: 0 4px 10px rgba(0, 0, 0, 0.2);

    opacity: 0;

    transform: translateX(100%);

    transition: opacity 0.5s ease, transform 0.5s ease;

    font-size: 14px;

    max-width: 300px;

}

.notification.show {

    opacity: 1;

    transform: translateX(0);

}

.notification.success { background-color: #28a745; }

.notification.error { background-color: #dc3545; }

.notification.info { background-color: #0d6efd; }

.conversations-tab {

    position: fixed;

    bottom: 20px;

    right: 20px;

    width: 300px;

    max-height: 400px;

    background-color: #fff;

    border: 1px solid #ddd;

    border-radius: 8px;

    box-shadow: 0 4px 12px rgba(0, 0, 0, 0.15);

    z-index: 1000;

    display: flex;

    flex-direction: column;

    overflow: hidden;

    transition: max-height 0.3s ease-out;

}

.conversations-tab.collapsed {

    max-height: 40px;

}

.tab-header {

    display: flex;

    justify-content: space-between;

    align-items: center;

    padding: 10px 15px;

    background-color: #0056b3;

    color: white;

    font-weight: bold;

    cursor: pointer;

    flex-shrink: 0;

}

.tab-header span {

    font-size: 14px;

}

.toggle-button {

    background: none;

    border: none;

    color: white;

    font-size: 18px;

    cursor: pointer;

    transition: transform 0.3s ease;

}

.conversations-tab.collapsed .toggle-button {

    transform: rotate(180deg);

}

.tab-content {

    flex-grow: 1;

    display: flex;

    flex-direction: column;

    overflow: hidden;

}

.chat-messages {

    flex-grow: 1;

    overflow-y: auto;

    padding: 15px;

    font-size: 13px;

    color: #333;

    display: flex;

    flex-direction: column;

    gap: 10px;

    scrollbar-width: thin;

    scrollbar-color: #ccc #f0f0f0;

}

.chat-messages::-webkit-scrollbar { width: 6px; }

.chat-messages::-webkit-scrollbar-track { background: #f0f0f0; border-radius: 3px; }

.chat-messages::-webkit-scrollbar-thumb { background-color: #ccc; border-radius: 3px; }

.chat-input-area {

    display: flex;

    padding: 10px 15px;

    border-top: 1px solid #eee;

    background-color: #f8f9fa;

    flex-shrink: 0;

    gap: 10px;

    align-items: center;

}

.chat-message {

    display: flex;

    flex-direction: column;

    max-width: 85%;

    word-break: break-word;

    border-radius: 8px;

    padding: 8px 12px;

}

.chat-message strong {

    font-size: 12px;

    margin-bottom: 2px;

}

.chat-message span {

    font-size: 13px;

}

.chat-message.sent {

    align-self: flex-end;

    background-color: #007bff;

    color: white;

}

.chat-message.received {

    align-self: flex-start;

    background-color: #e9ecef;

    color: #333;

}

.chat-message.status {

    align-self: center;

    font-style: italic;

    font-size: 11px;

    color: #6c757d;

    background-color: transparent;

    padding: 4px 8px;

}

@media (max-width: 992px) {

    #left-panel { width: 240px; }

    .large-symbol { font-size: 20px; }

    .large-price { font-size: 28px; }

    .large-price-change { font-size: 15px; }

    .stock-actions-large .action-button { font-size: 14px; padding: 8px 16px; }

    .portfolio-summary { font-size: 13px; }

    .conversations-tab { width: 250px; }

}

@media (max-width: 768px) {

    .dashboard-layout { flex-direction: column; height: auto; margin-top: 55px; }

    #left-panel {

        width: 100%;

        height: auto;

        border-right: none;

        border-bottom: 1px solid #ddd;

        max-height: 45vh;

        overflow-y: auto;

        order: 2;

    }

    .right-content {

        height: auto;

        padding: 15px;

        order: 1;

         overflow-y: visible;

    }

     .scrollable-list { max-height: 180px; }

     .key-metrics { grid-template-columns: 1fr 1fr; }

     .stock-actions-large { justify-content: center; }

     #user-email { display: none; }

     .search-container { padding-bottom: 15px; margin-bottom: 15px; }

     #search-result-display, #portfolio-container, #market-news-container, #users-directory-container, #user-search-results-display { padding: 15px; margin-bottom: 20px; }

     .portfolio-item { flex-direction: column; align-items: flex-start; gap: 8px; }

     .portfolio-value-info { align-items: flex-start; }

     .news-list, .users-list { max-height: 300px; }

     .news-item { flex-direction: column; }

     .news-image { width: 100%; height: 100px; margin-bottom: 10px; }

     .portfolio-header, .news-header, .section-header, .users-directory-header { flex-direction: column; align-items: flex-start; gap: 5px; }

     .portfolio-header .refresh-button, .news-header .refresh-button, .section-header .refresh-button { margin-left: 0; }

     .portfolio-summary { flex-direction: column; gap: 5px; align-items: flex-start; }

     .user-directory-item { flex-direction: column; align-items: flex-start; gap: 8px; }

     .user-directory-info { width: 100%; justify-content: space-between;}

     .user-directory-details { flex-grow: 1; }

     .user-directory-actions { width: 100%; justify-content: center; }

     .follow-button, .message-button { flex-grow: 1; text-align: center; justify-content: center;}

     .conversations-tab { width: calc(100% - 40px); right: 20px; left: 20px; bottom: 10px; }

}

@keyframes spin {

    0% { transform: rotate(0deg); }

    100% { transform: rotate(360deg); }

}

.spin {

    animation: spin 1s linear infinite;

}

## index.html

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>INVESTA - Track, Share, and Grow Your Portfolio</title>

    <link href="https://fonts.googleapis.com/css2?family=Inconsolata:wght@400;700&display=swap" rel="stylesheet">

    <script src="https://cdn.tailwindcss.com"></script>

    <style>

        \* {

            margin: 0;

            padding: 0;

            box-sizing: border-box;

        }

        body {

            font-family: 'Inconsolata', monospace;

            background-color: #ffffff;

            color: #333;

            line-height: 1.6;

        }

        header {

            position: fixed;

            width: 100%;

            top: 0;

            left: 0;

            z-index: 1000;

            background-color: rgba(255, 255, 255, 0.95);

            backdrop-filter: blur(5px);

            padding: 15px 25px;

            transition: top 0.3s ease-in-out;

            border-bottom: 1px solid #eee;

            box-shadow: 0 2px 4px rgba(0, 0, 0, 0.05);

        }

        nav {

            display: flex;

            justify-content: space-between;

            align-items: center;

            max-width: 1200px;

            margin: 0 auto;

        }

        .logo {

            font-size: 20px;

            font-weight: 700;

        }

        .logo a {

            text-decoration: none;

            color: #333;

        }

        .nav-links ul {

            list-style-type: none;

            display: flex;

            gap: 15px;

        }

        .nav-links ul li a {

            text-decoration: none;

            font-size: 14px;

            color: #333;

            font-weight: 700;

            padding: 8px 15px;

            border-radius: 6px;

            transition: background-color 0.3s, color 0.3s, transform 0.2s;

        }

        .nav-links ul li a:hover,

        .nav-links ul li a.active {

            background-color: #0056b3;

            color: white;

            transform: translateY(-2px);

        }

        .hero-section {

            background-color: #f8f9fa;

            text-align: center;

            padding: 150px 25px 100px;

            min-height: 100vh;

            display: flex;

            justify-content: center;

            align-items: center;

            flex-direction: column;

            background-image: radial-gradient(#e0e0e0 0.5px, transparent 0.5px);

            background-size: 15px 15px;

        }

        .hero-content {

            max-width: 700px;

            margin: 0 auto;

        }

        .hero-content h1 {

            font-size: 36px;

            font-weight: 700;

            margin-bottom: 20px;

            color: #212529;

            opacity: 0;

            transform: translateY(20px);

            transition: opacity 1s ease 0.5s, transform 1s ease 0.5s;

        }

        .hero-content p {

            font-size: 18px;

            margin-bottom: 30px;

            color: #555;

            opacity: 0;

            transform: translateY(20px);

            transition: opacity 1s ease 1s, transform 1s ease 1s;

        }

        .hero-section.visible .hero-content h1,

        .hero-section.visible .hero-content p {

            opacity: 1;

            transform: translateY(0);

        }

        .cta-button {

            display: inline-block;

            background-color: #0056b3;

            color: white;

            padding: 12px 30px;

            font-size: 16px;

            font-weight: bold;

            text-decoration: none;

            border-radius: 6px;

            transition: opacity 1s ease 1.5s, transform 1s ease 1.5s, background-color 0.3s, transform 0.2s; /\* Combined transitions \*/

            opacity: 0;

            transform: translateY(20px);

        }

        .hero-section.visible .cta-button {

             opacity: 1;

             transform: translateY(0);

        }

        .cta-button:hover {

            background-color: #00408d;

            transform: translateY(-2px) scale(1.02);

        }

        @media (max-width: 768px) {

            header {

                padding: 10px 15px;

            }

            .nav-links {

                display: none;

            }

            .hero-content h1 {

                font-size: 28px;

            }

            .hero-content p {

                font-size: 16px;

            }

            .cta-button {

                padding: 10px 25px;

                font-size: 14px;

            }

        }

    </style>

</head>

<body>

    <header id="mainHeader">

        <nav>

            <div class="logo">

                <a href="#">INVESTA</a>

            </div>

            <div class="nav-links">

                <ul>

                    <li><a href="#">Contact</a></li>

                    <li><a href="./login.html">Login / Sign Up</a></li>

                </ul>

            </div>

        </nav>

    </header>

    <section class="hero-section" id="hero">

        <div class="hero-content">

            <h1>INVESTA: Your Social Portfolio Hub</h1>

            <p>Effortlessly manage your investments, connect with other investors, and discover new strategies. Take control of your financial journey with INVESTA.</p>

            <a href="./login.html" class="cta-button">Start Managing Your Portfolio</a>

        </div>

    </section>

    <script>

        document.addEventListener('DOMContentLoaded', function() {

            const heroSection = document.getElementById('hero');

            const observerOptions = {

                root: null,

                rootMargin: '0px',

                threshold: 0.1

            };

            const observerCallback = (entries, observer) => {

                entries.forEach(entry => {

                    if (entry.isIntersecting) {

                        entry.target.classList.add('visible');

                    }

                });

            };

            const heroObserver = new IntersectionObserver(observerCallback, observerOptions);

            if (heroSection) {

                heroObserver.observe(heroSection);

            }

        });

    </script>

</body>

</html>

## login.html

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>INVESTA - Login / Sign Up</title>

    <link href="https://fonts.googleapis.com/css2?family=Inconsolata:wght@400;700&display=swap" rel="stylesheet">

    <script src="https://cdn.tailwindcss.com"></script>

    <style>

        \* { margin: 0; padding: 0; box-sizing: border-box; }

        body {

            font-family: 'Inconsolata', monospace;

            background-color: #f8f9fa;

            color: #333;

            line-height: 1.6;

        }

        header {

            position: fixed; width: 100%; top: 0;

            left: 0; z-index: 1000;

            background-color: rgba(255, 255, 255, 0.95); backdrop-filter: blur(5px);

            padding: 15px 25px; transition: top 0.3s ease-in-out;

            border-bottom: 1px solid #eee; box-shadow: 0 2px 4px rgba(0, 0, 0, 0.05);

        }

        nav {

            display: flex; justify-content: space-between; align-items: center;

            max-width: 1200px; margin: 0 auto;

        }

        .logo { font-size: 20px; font-weight: 700; }

        .logo a { text-decoration: none; color: #333; }

        .nav-links ul { list-style-type: none; display: flex; gap: 15px; }

        .nav-links ul li a {

            text-decoration: none; font-size: 14px; color: #333; font-weight: 700;

            padding: 8px 15px; border-radius: 6px;

            transition: background-color 0.3s, color 0.3s, transform 0.2s;

        }

        .nav-links ul li a:hover,

        .nav-links ul li a.active {

            background-color: #0056b3; color: white; transform: translateY(-2px);

        }

        .login-section {

            padding: 120px 15px 60px;

            display: flex;

            justify-content: center;

            align-items: center;

            min-height: 100vh;

            background-color: #f4f4f4;

        }

        .login-container {

            background-color: white;

            padding: 30px 35px;

            border-radius: 8px;

            box-shadow: 0px 5px 15px rgba(0, 0, 0, 0.1);

            width: 100%;

            max-width: 400px;

            text-align: center;

        }

        .login-container h2 {

            font-size: 24px;

            margin-bottom: 25px;

            color: #333;

        }

        .form-group {

            margin-bottom: 18px;

            text-align: left;

        }

        label {

            display: block;

            font-size: 14px;

            margin-bottom: 6px;

            font-weight: 600;

            color: #555;

        }

        input[type="email"],

        input[type="password"],

        input[type="text"] {

            width: 100%;

            padding: 12px 15px;

            border: 1px solid #ccc;

            border-radius: 5px;

            font-size: 14px;

            font-family: 'Inconsolata', monospace;

            transition: border-color 0.3s, box-shadow 0.3s;

        }

        input[type="email"]:focus,

        input[type="password"]:focus,

        input[type="text"]:focus {

            outline: none;

            border-color: #0056b3;

            box-shadow: 0 0 5px rgba(0, 86, 179, 0.2);

        }

        button.cta-btn {

            width: 100%;

            padding: 12px;

            background-color: #0056b3;

            color: white;

            font-size: 16px;

            font-weight: bold;

            border: none;

            border-radius: 5px;

            cursor: pointer;

            transition: background-color 0.3s, transform 0.2s;

            margin-top: 10px;

        }

        button.cta-btn:hover:not(:disabled) {

            background-color: #00408d;

            transform: translateY(-2px);

        }

         button.cta-btn:disabled {

            background-color: #cccccc;

            cursor: not-allowed;

            transform: none;

        }

        .switch-link-container,

        .forgot-password {

            text-align: center;

            margin-top: 18px;

            font-size: 14px;

        }

        .switch-link-container a,

        .forgot-password a {

            color: #0056b3;

            text-decoration: none;

            font-weight: 600;

            cursor: pointer;

        }

        .switch-link-container a:hover,

        .forgot-password a:hover {

            text-decoration: underline;

        }

        .message-area {

            color: #333;

            background-color: #e9ecef;

            border: 1px solid #ced4da;

            font-size: 14px;

            margin-top: 15px;

            padding: 10px 15px;

            border-radius: 5px;

            text-align: center;

            display: none;

            transition: opacity 0.3s;

        }

        .message-area.error {

             color: #721c24;

             background-color: #f8d7da;

             border-color: #f5c6cb;

        }

         .message-area.success {

             color: #0f5132;

             background-color: #d1e7dd;

             border-color: #badbcc;

         }

        .message-area.visible {

            display: block;

            opacity: 1;

        }

        .loader {

            border: 4px solid #f3f3f3;

            border-top: 4px solid #0056b3;

            border-radius: 50%;

            width: 25px;

            height: 25px;

            animation: spin 1s linear infinite;

            margin: 15px auto;

            display: none;

        }

        @keyframes spin {

            0% { transform: rotate(0deg); }

            100% { transform: rotate(360deg); }

        }

        @media (max-width: 768px) {

             header { padding: 10px 15px; }

            .nav-links { display: none; }

             .login-container { padding: 20px; max-width: 90%; }

             .login-container h2 { font-size: 20px; }

        }

    </style>

    <script src="https://www.gstatic.com/firebasejs/10.7.2/firebase-app.js" type="module"></script>

    <script src="https://www.gstatic.com/firebasejs/10.7.2/firebase-auth.js" type="module"></script>

    <script src="https://www.gstatic.com/firebasejs/10.7.2/firebase-firestore.js" type="module"></script>

    </head>

<body>

    <header id="mainHeader">

        <nav>

            <div class="logo"><a href="./index.html">INVESTA</a></div>

            <div class="nav-links">

                 <ul>

                    <li><a href="#">Contact</a></li>

                    <li><a href="./login.html" class="active">Login / Sign Up</a></li>

                </ul>

            </div>

        </nav>

    </header>

    <section id="login-signup" class="login-section">

        <div class="login-container">

            <h2 id="form-title">Login</h2>

            <div id="loader" class="loader"></div>

            <div id="message-area" class="message-area"></div>

            <form id="login-form">

                <div class="form-group">

                    <label for="email">Email:</label>

                    <input type="email" id="email" name="email" required autocomplete="email">

                </div>

                <div class="form-group">

                    <label for="password">Password:</label>

                    <input type="password" id="password" name="password" required autocomplete="current-password">

                </div>

                <div class="form-group">

                    <button type="submit" id="login-btn" class="cta-btn">Login</button>

                </div>

            </form>

            <form id="signup-form" style="display: none;">

                 <div class="form-group">

                    <label for="signup-username">Username:</label>

                    <input type="text" id="signup-username" name="signup-username" required autocomplete="username">

                </div>

                <div class="form-group">

                    <label for="signup-email">Email:</label>

                    <input type="email" id="signup-email" name="signup-email" required autocomplete="email">

                </div>

                <div class="form-group">

                    <label for="signup-password">Password (min. 6 characters):</label>

                    <input type="password" id="signup-password" name="signup-password" required autocomplete="new-password">

                </div>

                <div class="form-group">

                    <label for="signup-confirm-password">Confirm Password:</label>

                    <input type="password" id="signup-confirm-password" name="signup-confirm-password" required autocomplete="new-password">

                </div>

                 <div class="form-group">

                    <button type="submit" id="signup-btn" class="cta-btn">Sign Up</button>

                </div>

            </form>

            <p id="switch-link" class="switch-link-container">

                <a href="javascript:void(0);" onclick="toggleForm()">Don't have an account? Sign Up</a>

            </p>

            <p id="forgot-password-link" class="forgot-password">

                <a href="javascript:void(0);" onclick="handleForgotPassword()">Forgot Password?</a>

            </p>

        </div>

    </section>

    <script type="module" src="login.js"></script>

</body>

</html>

## profile.css

\* {

    margin: 0;

    padding: 0;

    box-sizing: border-box;

}

body {

    font-family: 'Inconsolata', monospace;

    background-color: #fafafa;

    color: #262626;

    line-height: 1.5;

    display: flex;

    flex-direction: column;

    min-height: 100vh;

    overflow-y: auto;

}

.profile-container {

    max-width: 935px;

    margin: 30px auto;

    padding: 0 20px;

    flex-grow: 1;

    display: flex;

    flex-direction: column;

    gap: 44px;

    margin-top: 75px;

}

.profile-header-section {

    display: flex;

    align-items: flex-start;

    gap: 80px;

    padding: 20px 0;

    border-bottom: 1px solid #dbdbdb;

}

.profile-picture-area {

    flex-shrink: 0;

    width: 150px;

    display: flex;

    justify-content: center;

    align-items: center;

}

#profile-picture-display {

    width: 150px;

    height: 150px;

    border-radius: 50%;

    object-fit: cover;

    border: 1px solid #dbdbdb;

    background-color: #eee;

}

.profile-info-area {

    flex-grow: 1;

    display: flex;

    flex-direction: column;

    gap: 12px;

}

.username-and-edit {

    display: flex;

    align-items: center;

    gap: 20px;

    flex-wrap: wrap;

}

.username-and-edit h1 {

    font-size: 28px;

    font-weight: 300;

    margin: 0;

    color: #262626;

}

.btn-small {

    padding: 5px 15px;

    font-size: 14px;

    border-radius: 4px;

    border: 1px solid #dbdbdb;

    background-color: transparent;

    color: #262626;

    font-weight: 600;

    text-decoration: none;

}

.btn-small:hover {

    background-color: #efefef;

}

.profile-stats {

    display: flex;

    gap: 40px;

    list-style: none;

    padding: 0;

    margin: 0;

    flex-wrap: wrap;

    font-size: 16px;

}

.stat-item {

    display: flex;

    align-items: center;

    color: #262626;

}

.stat-number {

    font-weight: 600;

    margin-right: 5px;

}

.stat-label {

    font-weight: 400;

    color: #8e8e8e;

}

.profile-bio {

    font-size: 15px;

    color: #262626;

    display: flex;

    flex-direction: column;

    gap: 5px;

}

.profile-bio p { margin: 0; }

.profile-content-section {

    display: flex;

    flex-direction: column;

}

.content-tabs {

    display: flex;

    justify-content: center;

    gap: 60px;

    border-top: 1px solid #dbdbdb;

    margin-bottom: 0;

    margin-top: 20px;

}

.tab-button {

    background: none;

    border: none;

    padding: 15px 0;

    font-size: 12px;

    font-weight: 600;

    color: #8e8e8e;

    cursor: pointer;

    text-transform: uppercase;

    letter-spacing: 1px;

    border-top: 2px solid transparent;

    transition: color 0.2s ease, border-top-color 0.2s ease;

    margin-bottom: -1px;

}

.tab-button:hover {

    color: #262626;

}

.tab-button.active {

    color: #262626;

    border-top-color: #262626;

}

.tab-content {

    display: none;

    padding-top: 20px;

}

.tab-content.active {

    display: block;

}

.tab-content h2 {

    font-size: 20px;

    font-weight: 600;

    color: #34495e;

    margin-bottom: 15px;

    text-align: center;

    display: none;

}

.portfolio-summary-details {

    display: flex;

    justify-content: center;

    gap: 30px;

    margin-bottom: 20px;

    padding-bottom: 20px;

    border-bottom: 1px solid #eee;

    flex-wrap: wrap;

}

.summary-item {

    display: flex;

    flex-direction: column;

    align-items: center;

    font-size: 15px;

}

.summary-label {

    font-weight: 600;

    color: #555;

    margin-bottom: 3px;

}

.summary-value {

    color: #333;

    font-weight: bold;

}

.summary-value.positive { color: #198754; }

.summary-value.negative { color: #dc3545; }

.content-list {

    display: grid;

    grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));

    gap: 20px;

    padding: 0;

    list-style: none;

}

.content-list .info-message,

.content-list .error-message {

    grid-column: 1 / -1;

    text-align: center;

    padding: 30px;

    color: #777;

    font-size: 14px;

}

.content-list .error-message { color: #dc3545; font-weight: bold; }

.portfolio-item {

    background-color: #fff;

    border: 1px solid #eee;

    border-radius: 8px;

    padding: 15px;

    display: flex;

    flex-direction: column;

    gap: 10px;

    box-shadow: 0 2px 5px rgba(0,0,0,0.05);

}

.portfolio-item-header {

    display: flex;

    align-items: center;

    gap: 10px;

    border-bottom: 1px dashed #eee;

    padding-bottom: 8px;

}

.portfolio-stock-logo {

    width: 35px;

    height: 35px;

    border-radius: 50%;

    object-fit: contain;

    background-color: #eee;

}

.portfolio-stock-details {

    flex-grow: 1;

    display: flex;

    flex-direction: column;

}

.portfolio-symbol {

    font-weight: bold;

    font-size: 14px;

    color: #333;

}

.portfolio-company-name {

    font-size: 12px;

    color: #555;

}

.portfolio-item-stats {

    display: flex;

    justify-content: space-between;

    font-size: 13px;

    color: #555;

}

.portfolio-item-stat strong {

    color: #333;

    margin-left: 5px;

}

.portfolio-item-pl.positive { color: #198754; font-weight: bold;}

.portfolio-item-pl.negative { color: #dc3545; font-weight: bold;}

.transaction-item {

    background-color: #fff;

    border: 1px solid #eee;

    border-radius: 8px;

    padding: 15px;

    display: flex;

    justify-content: space-between;

    align-items: center;

    gap: 15px;

    box-shadow: 0 2px 5px rgba(0,0,0,0.05);

    font-size: 14px;

}

.transaction-details {

    flex-grow: 1;

    display: flex;

    flex-direction: column;

    gap: 5px;

}

.transaction-type-symbol {

    font-weight: bold;

    font-size: 15px;

}

.transaction-type-symbol .buy { color: #198754; }

.transaction-type-symbol .sell { color: #dc3545; }

.transaction-info {

    color: #555;

}

.transaction-info strong { color: #333; }

.transaction-date {

    font-size: 12px;

    color: #888;

    text-align: right;

    flex-shrink: 0;

}

#notification-area-profile {

    margin-bottom: 20px;

}

#notification-area-profile .notification-message {

    padding: 12px 18px;

    border-radius: 6px;

    font-size: 14px;

    margin-bottom: 10px;

    text-align: center;

    opacity: 0;

    transform: translateY(-10px);

    animation: fadeInSlideDown 0.5s forwards;

}

@keyframes fadeInSlideDown {

    to {

        opacity: 1;

        transform: translateY(0);

    }

}

.notification-message.success {

    background-color: #d1e7dd;

    color: #0f5132;

    border: 1px solid #badbcc;

}

.notification-message.error {

    background-color: #f8d7da;

    color: #721c24;

    border: 1px solid #f5c6cb;

}

.notification-message.info {

    background-color: #cff4fc;

    color: #055160;

    border: 1px solid #b6effb;

}

@media (max-width: 768px) {

    .profile-container {

        margin: 20px 15px;

        padding: 0 15px;

        gap: 20px;

        margin-top: 75px;

    }

    .profile-header-section {

        flex-direction: column;

        align-items: center;

        gap: 20px;

        padding: 15px 0;

    }

    .profile-picture-area { width: 120px; }

    #profile-picture-display { width: 120px; height: 120px; }

    .profile-info-area {

        align-items: center;

        gap: 15px;

    }

    .username-and-edit {

        flex-direction: column;

        gap: 10px;

    }

    .username-and-edit h1 { font-size: 24px; }

    .btn-small { width: 100%; text-align: center; }

    .profile-stats {

        gap: 20px;

        justify-content: center;

    }

    .profile-bio {

        align-items: center;

        font-size: 14px;

    }

    .profile-content-section { padding-top: 10px; }

    .content-tabs {

        gap: 20px;

        margin-top: 15px;

    }

    .tab-button { padding: 10px 0; font-size: 11px; }

    .tab-content h2 { font-size: 18px; margin-bottom: 10px; }

    .portfolio-summary-details {

        flex-direction: column;

        gap: 15px;

        padding-bottom: 15px;

        margin-bottom: 15px;

    }

     .summary-item { align-items: flex-start; width: 100%; }

     .summary-value { text-align: left; }

    .content-list {

        grid-template-columns: 1fr;

        gap: 15px;

    }

    .portfolio-item { padding: 12px; gap: 8px; }

    .portfolio-item-header { gap: 8px; padding-bottom: 6px; }

    .portfolio-stock-logo { width: 30px; height: 30px; }

    .portfolio-symbol { font-size: 13px; }

    .portfolio-company-name { font-size: 11px; }

    .portfolio-item-stats { font-size: 12px; }

    .transaction-item { padding: 12px; gap: 10px; font-size: 13px; }

    .transaction-details { gap: 3px; }

    .transaction-type-symbol { font-size: 14px; }

    .transaction-info { font-size: 13px; }

    .transaction-date { font-size: 10px; }

}

@media (max-width: 480px) {

    .profile-container h1 { font-size: 20px; }

    .profile-card h2 { font-size: 16px; }

    .profile-picture-area { width: 100px; }

    #profile-picture-display { width: 100px; height: 100px; }

    .username-and-edit h1 { font-size: 20px; }

    .profile-stats { gap: 15px; font-size: 14px; }

    .profile-bio { font-size: 13px; }

    .content-tabs { gap: 15px; }

    .tab-button { font-size: 10px; }

}

## profile.html

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>My Profile - INVESTA</title>

    <link href="https://fonts.googleapis.com/css2?family=Inconsolata:wght@400;700&display=swap" rel="stylesheet">

    <link rel="stylesheet" href="https://fonts.googleapis.com/css2?family=Material+Symbols+Outlined:opsz,wght,FILL,GRAD@20..48,100..700,0..1,-50..200" />

    <link rel="stylesheet" href="dashboard.css">

    <link rel="stylesheet" href="profile.css">

</head>

<body>

    <header>

        <nav>

            <div class="logo">

                <a href="dashboard.html">INVESTA</a>

            </div>

             <div class="header-right">

                 <div class="nav-links">

                     <ul>

                         <li><a href="dashboard.html">Dashboard</a></li>

                         <li><a href="profile.html" class="active">My Profile</a></li>

                         <li><a href="settings.html">Settings</a></li>

                     </ul>

                 </div>

                 <div class="user-info">

                     <div id="profile-picture-container">

                        <img id="user-profile-picture" src="https://placehold.co/30x30/eee/ccc?text=?" alt="Profile Picture">

                     </div>

                     <div class="user-text-details">

                         <span id="user-greeting">Loading...</span>

                         </div>

                    <button id="logout-button">Logout</button>

                 </div>

             </div>

        </nav>

    </header>

    <main class="profile-container">

        <div id="notification-area-profile"></div>

        <section class="profile-header-section">

            <div class="profile-picture-area">

                <img id="profile-picture-display" src="https://placehold.co/150x150/eee/ccc?text=?" alt="Profile Picture">

            </div>

            <div class="profile-info-area">

                <div class="username-and-edit">

                    <h1 id="profile-username">Loading...</h1>

                    <a href="settings.html" class="btn btn-secondary btn-small">Edit Profile</a>

                </div>

                <div class="profile-stats">

                    <div class="stat-item">

                        <span class="stat-number" id="profile-portfolio-count">0</span>

                        <span class="stat-label">Stocks</span>

                    </div>

                    <div class="stat-item">

                        <span class="stat-number" id="profile-followers">0</span>

                        <span class="stat-label">Followers</span>

                    </div>

                    <div class="stat-item">

                        <span class="stat-number" id="profile-transactions-count">0</span>

                        <span class="stat-label">Trades</span>

                    </div>

                </div>

                <div class="profile-bio">

                    <p id="profile-email">Email: Loading... (Private)</p>

                    <p id="profile-member-since">Member Since: Loading...</p>

                </div>

            </div>

        </section>

        <section id="profile-bio-section" style="text-align: center;">

            <p id="profile-bio-text">Loading bio...</p>

        </section>

        <section class="profile-content-section">

             <div class="content-tabs">

                 <button class="tab-button active" data-tab="portfolio">Portfolio</button>

                 <button class="tab-button" data-tab="transactions">Transactions</button>

             </div>

             <div id="portfolio-tab" class="tab-content active">

                 <div class="portfolio-summary-details">

                     <div class="summary-item">

                         <span class="summary-label">Total Value:</span>

                         <span class="summary-value" id="profile-total-value">Loading...</span>

                     </div>

                     <div class="summary-item">

                         <span class="summary-label">Total P/L:</span>

                         <span class="summary-value" id="profile-total-pl">Loading...</span>

                     </div>

                 </div>

                  <div id="profile-portfolio-list" class="content-list">

                     <p class="info-message">Loading portfolio details...</p>

                 </div>

             </div>

             <div id="transactions-tab" class="tab-content">

                  <div id="profile-transactions-list" class="content-list">

                     <p class="info-message">Loading transaction history...</p>

                 </div>

             </div>

        </section>

    </main>

    <script src="https://www.gstatic.com/firebasejs/10.7.2/firebase-app.js" type="module"></script>

    <script src="https://www.gstatic.com/firebasejs/10.7.2/firebase-auth.js" type="module"></script>

    <script src="https://www.gstatic.com/firebasejs/10.7.2/firebase-firestore.js" type="module"></script>

    <script type="module" src="profile.js" defer></script>

</body>

</html>

## settings.css

\* {

    margin: 0;

    padding: 0;

    box-sizing: border-box;

}

body {

    font-family: 'Inconsolata', monospace;

    background-color: #f4f7f6;

    color: #333;

    line-height: 1.6;

    display: flex;

    flex-direction: column;

    min-height: 100vh;

    overflow-y: auto;

}

.settings-container {

    max-width: 800px;

    margin: 75px auto 30px auto;

    padding: 0 25px;

    flex-grow: 1;

}

.settings-container h1 {

    font-size: 28px;

    font-weight: 700;

    color: #2c3e50;

    margin-bottom: 30px;

    text-align: center;

    border-bottom: 1px solid #eee;

    padding-bottom: 15px;

}

.settings-card {

    background-color: #fff;

    padding: 25px 30px;

    border-radius: 8px;

    box-shadow: 0 3px 10px rgba(0, 0, 0, 0.08);

    margin-bottom: 30px;

}

.settings-card:last-child {

    margin-bottom: 0;

}

.settings-card h2 {

    font-size: 20px;

    font-weight: 600;

    color: #34495e;

    margin-bottom: 20px;

    padding-bottom: 10px;

    border-bottom: 1px solid #ecf0f1;

}

.form-group {

    margin-bottom: 20px;

}

.form-group label {

    display: block;

    font-size: 14px;

    font-weight: 600;

    color: #555;

    margin-bottom: 8px;

}

.form-group input[type="text"],

.form-group input[type="email"],

.form-group input[type="password"],

.form-group input[type="url"],

.form-group select {

    width: 100%;

    padding: 12px 15px;

    border: 1px solid #ced4da;

    border-radius: 6px;

    font-size: 14px;

    font-family: 'Inconsolata', monospace;

    transition: border-color 0.2s ease, box-shadow 0.2s ease;

    background-color: #fdfdfd;

}

.form-group input:read-only,

.form-group input:disabled {

    background-color: #e9ecef;

    cursor: not-allowed;

    opacity: 0.7;

}

.form-group input:focus,

.form-group select:focus {

    outline: none;

    border-color: #0056b3;

    box-shadow: 0 0 0 0.2rem rgba(0, 86, 179, 0.25);

}

.form-group small {

    display: block;

    font-size: 12px;

    color: #6c757d;

    margin-top: 5px;

}

.checkbox-group {

    display: flex;

    align-items: center;

    gap: 10px;

}

.checkbox-group input[type="checkbox"] {

    width: auto;

    margin-right: 5px;

    accent-color: #0056b3;

    transform: scale(1.1);

}

.checkbox-group label {

    margin-bottom: 0;

    font-weight: normal;

    font-size: 14px;

    color: #333;

}

.profile-picture-preview-container {

    margin-top: 15px;

    margin-bottom: 20px;

    display: flex;

    justify-content: center;

}

#profile-picture-preview {

    width: 100px;

    height: 100px;

    border-radius: 50%;

    object-fit: cover;

    border: 2px solid #e9ecef;

    background-color: #f8f9fa;

}

.btn {

    padding: 10px 20px;

    font-size: 14px;

    font-weight: 600;

    border-radius: 6px;

    cursor: pointer;

    border: none;

    transition: background-color 0.2s ease, transform 0.1s ease;

    display: inline-flex;

    align-items: center;

    justify-content: center;

    gap: 8px;

}

.btn:hover:not(:disabled) {

    transform: translateY(-1px);

}

.btn:disabled {

    opacity: 0.65;

    cursor: not-allowed;

}

.btn-primary {

    background-color: #0056b3;

    color: white;

}

.btn-primary:hover:not(:disabled) {

    background-color: #00408d;

}

.btn-secondary {

    background-color: #6c757d;

    color: white;

}

.btn-secondary:hover:not(:disabled) {

    background-color: #5a6268;

}

.btn-danger {

    background-color: #dc3545;

    color: white;

}

.btn-danger:hover:not(:disabled) {

    background-color: #c82333;

}

.danger-zone p {

    font-size: 14px;

    color: #555;

    margin-bottom: 15px;

}

.danger-zone .btn-danger {

    width: 100%;

    margin-top: 10px;

}

#notification-area-settings {

    margin-bottom: 20px;

}

#notification-area-settings .notification-message {

    padding: 12px 18px;

    border-radius: 6px;

    font-size: 14px;

    margin-bottom: 10px;

    text-align: center;

    opacity: 0;

    transform: translateY(-10px);

    animation: fadeInSlideDown 0.5s forwards;

}

@keyframes fadeInSlideDown {

    to {

        opacity: 1;

        transform: translateY(0);

    }

}

.notification-message.success {

    background-color: #d1e7dd;

    color: #0f5132;

    border: 1px solid #badbcc;

}

.notification-message.error {

    background-color: #f8d7da;

    color: #721c24;

    border: 1px solid #f5c6cb;

}

.notification-message.info {

    background-color: #cff4fc;

    color: #055160;

    border: 1px solid #b6effb;

}

@media (max-width: 768px) {

    .settings-container {

        margin: 75px 15px 20px 15px;

        padding: 0 15px;

    }

    .settings-container h1 {

        font-size: 24px;

        margin-bottom: 20px;

    }

    .settings-card {

        padding: 20px;

    }

    .settings-card h2 {

        font-size: 18px;

    }

    .form-group input[type="text"],

    .form-group input[type="email"],

    .form-group input[type="password"],

    .form-group input[type="url"],

    .form-group select {

        padding: 10px 12px;

    }

    .btn {

        width: 100%;

        padding: 12px;

    }

}

## settings.html

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Settings - INVESTA</title>

    <link href="https://fonts.googleapis.com/css2?family=Inconsolata:wght@400;700&display=swap" rel="stylesheet">

    <link rel="stylesheet" href="https://fonts.googleapis.com/css2?family=Material+Symbols+Outlined:opsz,wght,FILL,GRAD@20..48,100..700,0..1,-50..200" />

    <link rel="stylesheet" href="dashboard.css">

    <link rel="stylesheet" href="settings.css">

</head>

<body>

    <header>

        <nav>

            <div class="logo">

                <a href="dashboard.html">INVESTA</a>

            </div>

            <div class="header-right">

                <div class="nav-links">

                    <ul>

                        <li><a href="dashboard.html">Dashboard</a></li>

                        <li><a href="profile.html">My Profile</a></li>

                        <li><a href="settings.html" class="active">Settings</a></li>

                    </ul>

                </div>

                <div class="user-info">

                    <div id="profile-picture-container">

                       <img id="user-profile-picture" src="https://placehold.co/30x30/eee/ccc?text=?" alt="Profile Picture">

                    </div>

                    <div class="user-text-details">

                        <span id="user-greeting">Loading...</span>

                    </div>

                   <button id="logout-button">Logout</button>

                </div>

            </div>

        </nav>

    </header>

    <main class="settings-container">

        <h1>Account Settings</h1>

        <div id="notification-area-settings"></div>

        <section class="settings-card">

            <h2>Profile Information</h2>

            <form id="profile-settings-form">

                <div class="form-group">

                    <label for="username">Username:</label>

                    <input type="text" id="username" name="username" required>

                </div>

                <div class="form-group">

                    <label for="email">Email:</label>

                    <input type="email" id="email" name="email" readonly disabled>

                    <small>Email address cannot be changed.</small>

                </div>

                <div class="form-group">

                    <label for="profilePictureUrl">Profile Picture URL:</label>

                    <input type="url" id="profilePictureUrl" name="profilePictureUrl" placeholder="https://example.com/image.png">

                    <small>Enter a publicly accessible URL for your profile image.</small>

                </div>

                <div class="profile-picture-preview-container">

                    <img id="profile-picture-preview" src="https://placehold.co/100x100/eee/ccc?text=?" alt="Profile Picture Preview">

                </div>

                <div class="form-group">

                    <label for="bio">Bio:</label>

                    <textarea id="bio" name="bio" rows="4" placeholder="Tell us a little about yourself... (max 200 characters)" maxlength="200"></textarea>

                    <small>A short bio to display on your profile.</small>

                </div>

                <button type="submit" class="btn btn-primary" id="save-profile-button">Save Profile Changes</button>

            </form>

        </section>

        <section class="settings-card">

            <h2>Change Password</h2>

            <form id="password-change-form">

                <div class="form-group">

                    <label for="currentPassword">Current Password:</label>

                    <input type="password" id="currentPassword" name="currentPassword" required>

                </div>

                <div class="form-group">

                    <label for="newPassword">New Password (min. 6 characters):</label>

                    <input type="password" id="newPassword" name="newPassword" required minlength="6">

                </div>

                <div class="form-group">

                    <label for="confirmNewPassword">Confirm New Password:</label>

                    <input type="password" id="confirmNewPassword" name="confirmNewPassword" required>

                </div>

                <button type="submit" class="btn btn-primary" id="change-password-button">Change Password</button>

            </form>

        </section>

        <section class="settings-card danger-zone">

            <h2>Danger Zone</h2>

            <p>Be careful, these actions are irreversible.</p>

            <button class="btn btn-danger" id="delete-account-button">Delete My Account</button>

        </section>

    </main>

    <script src="https://www.gstatic.com/firebasejs/10.7.2/firebase-app.js" type="module"></script>

    <script src="https://www.gstatic.com/firebasejs/10.7.2/firebase-auth.js" type="module"></script>

    <script src="https://www.gstatic.com/firebasejs/10.7.2/firebase-firestore.js" type="module"></script>

    <script type="module" src="settings.js" defer></script>

</body>

</html>

## styles.css

\* {

    margin: 0;

    padding: 0;

    box-sizing: border-box;

}

body {

    font-family: 'Inconsolata', monospace;

    background-color: #f4f7f6;

    color: #333;

    line-height: 1.6;

    display: flex;

    flex-direction: column;

    min-height: 100vh;

}

header {

    position: sticky;

    top: 0;

    width: 100%;

    z-index: 1000;

    background-color: rgba(255, 255, 255, 0.98);

    backdrop-filter: blur(5px);

    padding: 10px 25px;

    border-bottom: 1px solid #dde;

    box-shadow: 0 1px 3px rgba(0, 0, 0, 0.05);

}

nav {

    display: flex;

    justify-content: space-between;

    align-items: center;

    max-width: 1200px;

    margin: 0 auto;

}

.logo {

    font-size: 20px;

    font-weight: 700;

}

.logo a {

    text-decoration: none;

    color: #2c3e50;

}

.header-right {

    display: flex;

    align-items: center;

    gap: 20px;

}

.nav-links ul {

    list-style-type: none;

    display: flex;

    gap: 15px;

    margin: 0;

    padding: 0;

}

.nav-links ul li a {

    text-decoration: none;

    font-size: 14px;

    color: #333;

    font-weight: 600;

    padding: 8px 12px;

    border-radius: 6px;

    transition: background-color 0.2s ease, color 0.2s ease;

}

.nav-links ul li a:hover,

.nav-links ul li a.active {

    background-color: #0056b3;

    color: white;

}

.user-info {

    display: flex;

    align-items: center;

    gap: 10px;

    padding-left: 15px;

    border-left: 1px solid #eee;

}

#user-greeting-header {

     font-size: 14px;

     font-weight: 600;

     color: #333;

}

#profile-picture-container-header {

    border-radius: 50%;

    overflow: hidden;

    width: 32px;

    height: 32px;

    flex-shrink: 0;

    background-color: #e9ecef;

    border: 1px solid #ced4da;

}

#user-profile-picture-header {

     width: 100%;

     height: 100%;

     object-fit: cover;

     display: block;

}

#logout-button {

    background-color: transparent;

    color: #c82333;

    border: 1px solid #c82333;

    padding: 7px 14px;

    border-radius: 6px;

    cursor: pointer;

    font-size: 13px;

    font-weight: 600;

    transition: all 0.2s ease;

}

#logout-button:hover {

    background-color: #c82333;

    color: white;

    transform: translateY(-1px);

}

.settings-container {

    max-width: 800px;

    margin: 30px auto;

    padding: 20px 25px;

    flex-grow: 1;

}

.settings-container h1 {

    font-size: 28px;

    font-weight: 700;

    color: #2c3e50;

    margin-bottom: 30px;

    text-align: center;

    border-bottom: 1px solid #eee;

    padding-bottom: 15px;

}

.settings-card {

    background-color: #fff;

    padding: 25px 30px;

    border-radius: 8px;

    box-shadow: 0 3px 10px rgba(0, 0, 0, 0.08);

    margin-bottom: 30px;

}

.settings-card:last-child {

    margin-bottom: 0;

}

.settings-card h2 {

    font-size: 20px;

    font-weight: 600;

    color: #34495e;

    margin-bottom: 20px;

    padding-bottom: 10px;

    border-bottom: 1px solid #ecf0f1;

}

.form-group {

    margin-bottom: 20px;

}

.form-group label {

    display: block;

    font-size: 14px;

    font-weight: 600;

    color: #555;

    margin-bottom: 8px;

}

.form-group input[type="text"],

.form-group input[type="email"],

.form-group input[type="password"],

.form-group input[type="url"],

.form-group select {

    width: 100%;

    padding: 12px 15px;

    border: 1px solid #ced4da;

    border-radius: 6px;

    font-size: 14px;

    font-family: 'Inconsolata', monospace;

    transition: border-color 0.2s ease, box-shadow 0.2s ease;

    background-color: #fdfdfd;

}

.form-group input:read-only,

.form-group input:disabled {

    background-color: #e9ecef;

    cursor: not-allowed;

    opacity: 0.7;

}

.form-group input:focus,

.form-group select:focus {

    outline: none;

    border-color: #0056b3;

    box-shadow: 0 0 0 0.2rem rgba(0, 86, 179, 0.25);

}

.form-group small {

    display: block;

    font-size: 12px;

    color: #6c757d;

    margin-top: 5px;

}

.checkbox-group {

    display: flex;

    align-items: center;

    gap: 10px;

}

.checkbox-group input[type="checkbox"] {

    width: auto;

    margin-right: 5px;

    accent-color: #0056b3;

    transform: scale(1.1);

}

.checkbox-group label {

    margin-bottom: 0;

    font-weight: normal;

    font-size: 14px;

    color: #333;

}

.profile-picture-preview-container {

    margin-top: 15px;

    margin-bottom: 20px;

    display: flex;

    justify-content: center;

}

#profile-picture-preview {

    width: 100px;

    height: 100px;

    border-radius: 50%;

    object-fit: cover;

    border: 2px solid #e9ecef;

    background-color: #f8f9fa;

}

.btn {

    padding: 10px 20px;

    font-size: 14px;

    font-weight: 600;

    border-radius: 6px;

    cursor: pointer;

    border: none;

    transition: background-color 0.2s ease, transform 0.1s ease;

    display: inline-flex;

    align-items: center;

    justify-content: center;

    gap: 8px;

}

.btn:hover:not(:disabled) {

    transform: translateY(-1px);

}

.btn:disabled {

    opacity: 0.65;

    cursor: not-allowed;

}

.btn-primary {

    background-color: #0056b3;

    color: white;

}

.btn-primary:hover:not(:disabled) {

    background-color: #00408d;

}

.btn-secondary {

    background-color: #6c757d;

    color: white;

}

.btn-secondary:hover:not(:disabled) {

    background-color: #5a6268;

}

.btn-danger {

    background-color: #dc3545;

    color: white;

}

.btn-danger:hover:not(:disabled) {

    background-color: #c82333;

}

.danger-zone p {

    font-size: 14px;

    color: #555;

    margin-bottom: 15px;

}

.danger-zone .btn-danger {

    width: 100%;

    margin-top: 10px;

}

#notification-area-settings {

    margin-bottom: 20px;

}

#notification-area-settings .notification-message {

    padding: 12px 18px;

    border-radius: 6px;

    font-size: 14px;

    margin-bottom: 10px;

    text-align: center;

    opacity: 0;

    transform: translateY(-10px);

    animation: fadeInSlideDown 0.5s forwards;

}

@keyframes fadeInSlideDown {

    to {

        opacity: 1;

        transform: translateY(0);

    }

}

.notification-message.success {

    background-color: #d1e7dd;

    color: #0f5132;

    border: 1px solid #badbcc;

}

.notification-message.error {

    background-color: #f8d7da;

    color: #721c24;

    border: 1px solid #f5c6cb;

}

.notification-message.info {

    background-color: #cff4fc;

    color: #055160;

    border: 1px solid #b6effb;

}

@media (max-width: 768px) {

    header { padding: 10px 15px; }

    .nav-links { display: none; }

    .user-info { padding-left: 0; border-left: none; }

    .settings-container {

        margin: 20px 15px;

        padding: 15px;

    }

    .settings-container h1 {

        font-size: 24px;

        margin-bottom: 20px;

    }

    .settings-card {

        padding: 20px;

    }

    .settings-card h2 {

        font-size: 18px;

    }

    .form-group input[type="text"],

    .form-group input[type="email"],

    .form-group input[type="password"],

    .form-group input[type="url"],

    .form-group select {

        padding: 10px 12px;

    }

    .btn {

        width: 100%;

        padding: 12px;

    }

}

@media (max-width: 480px) {

    .settings-container h1 { font-size: 22px; }

    .settings-card h2 { font-size: 17px; }

    .form-group label { font-size: 13px; }

    .form-group input, .form-group select { font-size: 13px; }

}

## user\_profile.css

#user-profile-picture-display {

     width: 150px;

     height: 150px;

     border-radius: 50%;

     object-fit: cover;

     border: 1px solid #dbdbdb;

     background-color: #eee;

}

.username-and-actions {

    display: flex;

    align-items: center;

    gap: 20px;

    flex-wrap: wrap;

}

.username-and-actions h1 {

    font-size: 28px;

    font-weight: 300;

    margin: 0;

    color: #262626;

}

.user-actions {

    display: flex;

    gap: 10px;

    align-items: center;

}

#user-profile-member-since {

     font-size: 15px;

     color: #262626;

}

#user-profile-pl {

    font-size: 15px;

    color: #262626;

    font-weight: bold;

}

#user-profile-pl.positive { color: #198754; }

#user-profile-pl.negative { color: #dc3545; }

#user-profile-portfolio-summary {

    display: flex;

    justify-content: center;

    gap: 30px;

    margin-bottom: 20px;

    padding-bottom: 20px;

    border-bottom: 1px solid #eee;

    flex-wrap: wrap;

}

.follow-button.following { /\* This style is also in dashboard.css, ensure consistency or remove if redundant \*/

    background-color: #e0e0e0;

    color: #555;

    border-color: #d0d0d0;

}

.follow-button:hover:not(:disabled).following { /\* This style is also in dashboard.css \*/

    background-color: #d5d5d5;

    transform: translateY(-1px);

}

#notification-area-user-profile { /\* This ID is present in user\_profile.html \*/

    margin-bottom: 20px;

}

#notification-area-user-profile .notification-message { /\* Used by displayNotification in user\_profile.js \*/

    padding: 12px 18px;

    border-radius: 6px;

    font-size: 14px;

    margin-bottom: 10px;

    text-align: center;

    opacity: 0;

    transform: translateY(-10px);

    animation: fadeInSlideDown 0.5s forwards; /\* Assumes fadeInSlideDown is defined, likely in profile.css or dashboard.css \*/

}

/\* Notification type styles (.success, .error, .info) are assumed to be inherited or defined elsewhere (e.g., profile.css) \*/

@media (max-width: 768px) {

     #user-profile-picture-display {

          width: 120px;

          height: 120px;

     }

    .username-and-actions {

        flex-direction: column;

        gap: 10px;

    }

    .username-and-actions h1 { font-size: 24px; }

    .user-actions { width: 100%; justify-content: center; gap: 8px; }

    .user-actions .btn-small { flex-grow: 1; text-align: center; }

}

@media (max-width: 480px) {

    #user-profile-picture-display {

        width: 100px;

        height: 100px;

    }

}

## user\_profile.html

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>User Profile - INVESTA</title>

    <link href="https://fonts.googleapis.com/css2?family=Inconsolata:wght@400;700&display=swap" rel="stylesheet">

    <link rel="stylesheet" href="https://fonts.googleapis.com/css2?family=Material+Symbols+Outlined:opsz,wght,FILL,GRAD@20..48,100..700,0..1,-50..200" />

    <link rel="stylesheet" href="dashboard.css">

    <link rel="stylesheet" href="profile.css">

    <link rel="stylesheet" href="user\_profile.css">

</head>

<body>

    <header>

        <nav>

            <div class="logo">

                <a href="dashboard.html">INVESTA</a>

            </div>

             <div class="header-right">

                 <div class="nav-links">

                     <ul>

                         <li><a href="dashboard.html">Dashboard</a></li>

                         <li><a href="profile.html">My Profile</a></li>

                         <li><a href="settings.html">Settings</a></li>

                     </ul>

                 </div>

                 <div class="user-info">

                     <div id="profile-picture-container">

                        <img id="user-profile-picture" src="https://placehold.co/30x30/eee/ccc?text=?" alt="Profile Picture">

                     </div>

                     <div class="user-text-details">

                         <span id="user-greeting">Loading...</span>

                     </div>

                    <button id="logout-button">Logout</button>

                 </div>

             </div>

        </nav>

    </header>

    <main class="profile-container">

        <div id="notification-area-user-profile"></div>

        <section class="profile-header-section">

            <div class="profile-picture-area">

                <img id="user-profile-picture-display" src="https://placehold.co/200x200/eee/ccc?text=?" alt="User Profile Picture">

            </div>

            <div class="profile-info-area">

                <div class="username-and-actions">

                    <h1 id="user-profile-username">Loading...</h1>

                    <div class="user-actions">

                         <button class="btn btn-secondary btn-small follow-button" id="follow-user-button" data-user-id="">Follow</button>

                    </div>

                </div>

                <div class="profile-stats">

                    <div class="stat-item">

                        <span class="stat-number" id="user-profile-portfolio-count">0</span>

                        <span class="stat-label">Stocks</span>

                    </div>

                    <div class="stat-item">

                        <span class="stat-number" id="user-profile-followers">0</span>

                        <span class="stat-label">Followers</span>

                    </div>

                    <div class="stat-item">

                        <span class="stat-number" id="user-profile-transactions-count">0</span>

                        <span class="stat-label">Trades</span>

                    </div>

                </div>

                <div class="profile-bio">

                    <p id="user-profile-member-since">Member Since: Loading...</p>

                    <p id="user-profile-pl" class="profile-bio-item">Total P/L: Loading...</p>

                </div>

            </div>

        </section>

        <section id="user-profile-bio-section" style="text-align: center;">

            <p id="user-profile-bio-text">Loading bio...</p>

        </section>

        <section class="profile-content-section">

             <div class="content-tabs">

                 <button class="tab-button active" data-tab="portfolio">Portfolio</button>

                 <button class="tab-button" data-tab="transactions">Recent Trades</button>

             </div>

             <div id="portfolio-tab" class="tab-content active">

                 <div class="portfolio-summary-details" id="user-profile-portfolio-summary">

                     <div class="summary-item">

                         <span class="summary-label">Total Value:</span>

                         <span class="summary-value" id="user-profile-total-value">Loading...</span>

                     </div>

                     <div class="summary-item">

                         <span class="summary-label">Total P/L:</span>

                         <span class="summary-value" id="user-profile-total-pl-summary">Loading...</span>

                     </div>

                 </div>

                 <div id="user-profile-portfolio-list" class="content-list">

                     <p class="info-message">Loading portfolio details... (Portfolio may be private)</p>

                 </div>

             </div>

             <div id="transactions-tab" class="tab-content">

                  <div id="user-profile-transactions-list" class="content-list">

                     <p class="info-message">Loading recent transactions...</p>

                 </div>

             </div>

        </section>

    </main>

    <script src="https://www.gstatic.com/firebasejs/10.7.2/firebase-app.js" type="module"></script>

    <script src="https://www.gstatic.com/firebasejs/10.7.2/firebase-auth.js" type="module"></script>

    <script src="https://www.gstatic.com/firebasejs/10.7.2/firebase-firestore.js" type="module"></script>

    <script type="module" src="user\_profile.js" defer></script>

</body>

</html>