



TAKE PROFIT STOCK EXCHANGE – Almog Michael Hemo Python project

# TAKE PROFIT STOCK EXCHANGE



**Almog Michael Hemo Python Project**

**July 18, 2021**

File Name: "Take Profit stock exchange - Hebrew project proposal"

תוכן עיניינים

<b>1</b>	<b>מבוא</b>	<b>3</b>	<b>1</b>
1.1	הקדמה	3	
2.1	מטרת המסמך	3	
3.1	מטרת המערכת	3	
<b>2</b>	<b>ארכיטקטורת המערכת</b>	<b>4</b>	
1.2	סכימת מלבנים	4	
2.2	שפות תכנות, סביבת עבודה ובסיס הנתונים	7	
<b>3</b>	<b>תאור הקוד ובסיס הנתונים</b>	<b>10</b>	
1.3	פונקציות ומחלקות (Classes) ב- Client	10	
2.3	פונקציות ומחלקות (Classes) ב- Server	13	
3.3	תאור התחברות Client – Server	18	
3.4	קוד לדוגמא – תהליך רכישת מניה	19	
3.5	קוד לדוגמא – יצירת גרף	21	
3.6	סכימות בסיס הנתונים	22	
<b>4</b>	<b>ממשק המשתמש</b>	<b>26</b>	
1.4	דף רישום או כניסה למערכת	26	
4.2	דף רישום משתמש חדש – Sign Up	27	
4.3	דף כניסה למערכת למשתמש קיים – Sign In	28	
4.4	דף אודות – About Us	29	
4.5	דף דוחות – Reports	30	
6.4	דף מאזן החשבון – Balance	31	
4.7	דף תיק ההשקעות – Portfolio	32	
8.4	דף שוק המניות – Stock Market	33	
9.4	דף מדדים – Indexes	34	
01.4	דף יצירת פקודה – Create Order	35	
11.4	דף גרפים	36	
21.4	דף מידע נוסף – More Info	36	
<b>5</b>	<b>בבליוגרפיה</b>	<b>38</b>	



## 1 מבוא

### 1.1 הקדמה

פרויקט הגמר שבחברתי לעשות הינו פלטפורמה שמאפשרת ללמוד ולהתאמן במסחר במניות, בסביבה הדומה למערכות מסחר אמיתיות, אולם ללא צורך בהשקעה כספית. העולם הפיננסי, והמסחר במניות ומדדים בפרט, היה חדש עבורי וככל שקראתי והעמקתי את הידע התאורטי, הסתקרנתי יותר ורציתי להתנסות בעצמי במסחר מבלי להשקיע או לסכן כספים. מכאן החל הרעיון לפרויקט, להנגיש את הידע שצברתי ולספק חוויית לימוד תאורטית ומעשית לכלל.

הפלטפורמה מספקת מידע בסיסי, הגדרות, מושגי יסוד וחוויות מסחר קרובה למציאות לכל מי שמעוניין, היא מורכבת מצד שרת אשר משמש כזירת מסחר בין מוכרים וקונים, ומצד קליינט (מרובה) המאפשר למספר משתמשים לצפות במידע ולבצע מגוון פעולות בעזרת ממשק משתמש ידידותי.

### 1.2 מטרת המסמך

מטרת המסמך הינה לתאר את פרויקט הגמר מבחינת ההיבטים הטכנולוגיים, תכנותיים, ותאור פונקציונאלי של המערכת מצד משתמשי הקצה. וכמוכן להסביר את האתגרים ודרכי הפתרון והמימוש אשר ליוו את שלבי האפיון ושלבי פיתוח המערכת.

### 1.3 מטרת המערכת

מערכת "Take Profit Stock Exchange" הינה פלטפורמה המדמה מסחר אמיתי ונועדה ללמד ולאמן כל אדם אשר מעוניין להתנסות בחוויות המסחר מבלי לסכן כספים אמיתיים. המערכת תלמד את המושגים הרלוונטיים, כיצד לקרוא גרפים של מניות ומדדים, ותלמד כיצד לבצע פעולות מסחר במניות (סוגים שונים של קניה ומכירה).

המערכת מאפשרת למספר תלמידים/סוחרים בו זמנית, לסחור במניות ובמדדים שונים, ללא סיכון, אולם בסביבה המדמה סביבה אמיתית ועם נתוני מסחר אמיתיים.

לכל משתמש מותקן קליינט, מקומית במחשב שלו, אשר פועל מול הסרבר שמשמש כשרת הבורסה, כזירת המסחר המפגישה בין קונים למוכרים.

המשתמשים, לאחר שלב של אוטנטיקציה (אימות על ידי שם משתמש וסיסמא), יכולים לבצע את הפעולות הבאות בעזרת ממשק המשתמש:

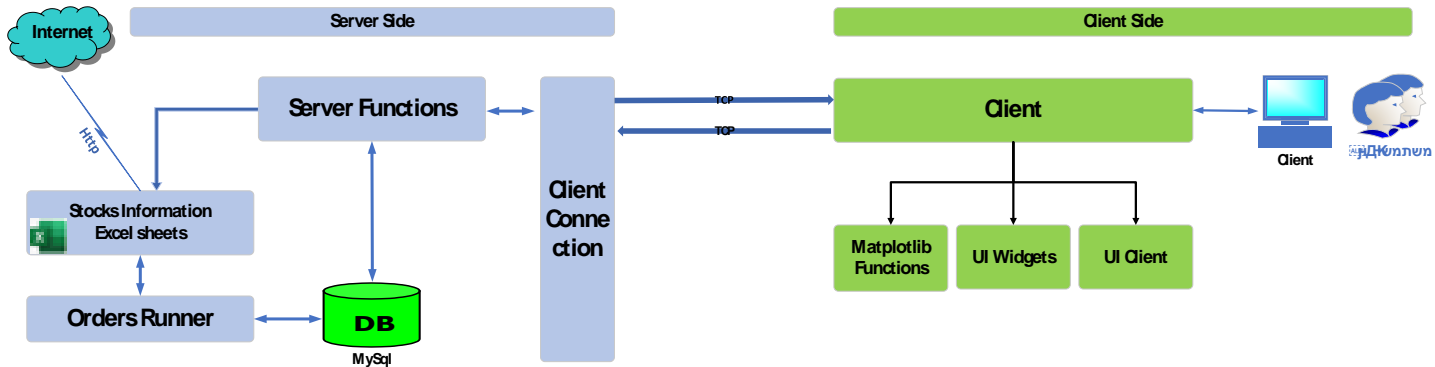
1. לצפות בנתוני המסחר של מניות ומדדים שונים.  
הנתונים כוללים את שער המניה, תאור החברה, מחיר פתיחה, מחיר נעילה אחרונה, מחזור המניות (כמה מניות נסחרות), volume (כמה כסף עובר בין הקונים למוכרים סה"כ בפרק זמן נתון), שינוי באחוזים, Price to earnings Ratio – P/E = מכפיל רווח ושם המדד בו המניה נסחרת
2. להזרים פקודות קניה ומכירה של מניות בשער מניה נוכחי (Market price) או לכשיגיע השער למחיר מבוקש (Limit orders)
3. לצפות בגרפים המתארים את ערכי המניה בציר הזמן
4. צפיה ביומן הפקודות שהוזרמו, ואפשרות לבטל את אלו שטרם בוצעו
5. צפיה ביתרות המסחר (מאזן החשבון), הרווחים/הפסדים
6. "מאמן מסחר" - אפשרות לקבל הסברים כתובים על מושגי מסחר ומידע על מדדים
7. אפשרות לחולל גרפים של רווח/הפסד ומאזן יתרות על פי היסטוריית המסחר של המשתמש



## 2 ארכיטקטורת המערכת

### 2.1 סכימת מלבנים

השרטוט הבא מציג את המודולים הלוגים בצד השרת ובצד הסרבר, מציג את בסיס הנתונים ואת קשרי הגומלין שבניהם:



להלן הסבר של כל מודול בנפרד:

#### 2.1.1 צד הקליינט – מודול UI Client

המודול הנ"ל אחראי על בניית כל העיצוב של ממשק המשתמש, של האובייקטים והרקעים ועל מיקומם בחלונות/מסכים של ממשק המשתמש. המודול נכתב בשפת Python בעזרת שימוש בספריית PyQt5 אשר מאפשרת עיצוב חלונות. כמוכן, השתמשתי ב- QtDesigner לטובת בניית הקוד העיצובי של המערכת.

#### 2.1.2 צד הקליינט – מודול UI Widgets

מודול זה כולל מספר מחלקות/Classes של Widgets אשר קובעים את ההגדרות של אלמנטים שונים בממשק המשתמש, את התכונות העיצוביות שלהם והפונקציות שיבצעו.

לדוגמא: `Stock_for_buy_Widget ()` הנה מחלקה/Class אשר מספקת תאור ומידעים על מניה מסוימת, ומאפשרת ביצוע פעולת קניה, הצגת גרפים והצגת מידע נוסף על מניה ספציפית:

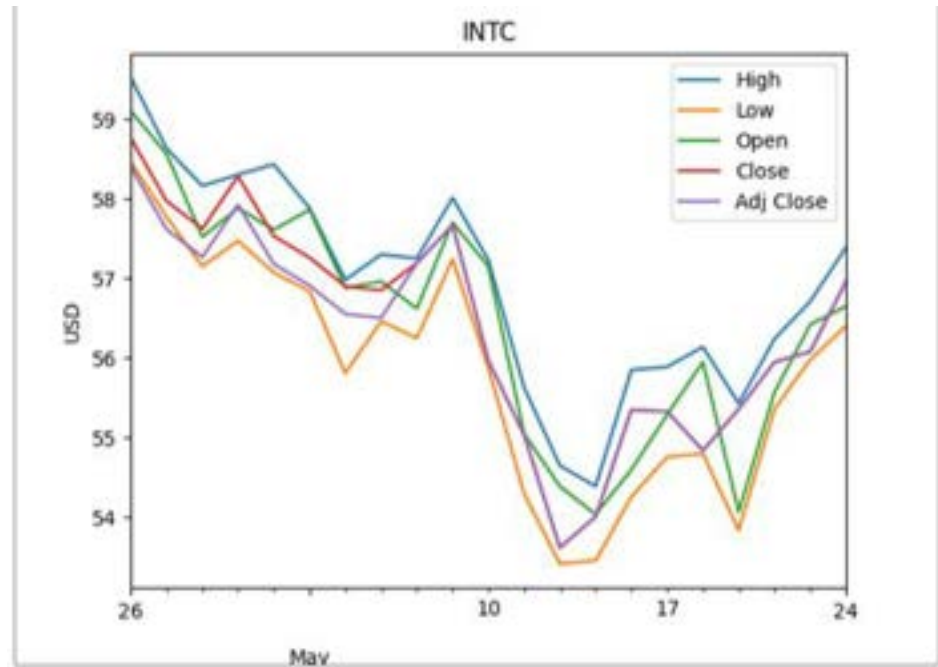




### 2.1.3 צד הקליינט – מודול Matplotlib Functions

מודול זה מכיל מערך של פונקציות אשר משתמשות בספרייה בשם Matplotlib לצורך קריאת ערכי המניות בציר הזמן (היסטוריה + הווה) והצגתם בצורה גרפית על גבי מערכת צירים.

לדוגמא, מספר גרפים המתארים מחירים שונים של המניה (בציר Y - מחירי פתיחה\סגירה\נמוך\גבוה) לאורך ציר הזמן (ציר X)



### 2.1.4 צד הקליינט – מודול Client

מודול ה-Client מנהל את התקשורת עם הסרבר על פי הבקשות\הפעולות שהמשתמש מבצע בממשק המשתמש, ומעדכן בחזרה את ממשק המשתמש בהתאם לשינויים וערכים מותאמים. כמובן, מודול ה-Client מנהל ומשתמש בשלושת המודולים הקודמים לצורך התאמת ממשק המשתמש בזמן אמת, על פי בחירת המשתמש והפעולות אותן הוא מבקש לבצע.



### 2.1.5 צד שרת – מודול Client Connection

מודול ה- Client Connection הינה מחלקה אשר מטפלת בכל פניות המשתמש. עבור כל פניה אפשרית, יש שם קוד ייחודי. וכאשר הוא מתקבל, מופעלת הפונקציה המתאימה לבקשת המשתמש.

### 2.1.6 צד שרת – מודול Server Functions

מודול ה- Server Functions הנו קובץ הכולל 14 פונקציות שונות אשר משמשות את ה- Client Connection & Orders\_Runner. הפונקציות הנ"ל מבצעות בין היתר פעולות כגון: חיפוש מניות, יצירת פקודות מסחר, קריאת מידע מבסיס הנתונים והצגת מידע.

### 2.1.7 צד שרת – מודול Orders Runner

מודול ה- Orders Runner הינה מחלקה אשר רצה/פועלת במקביל לתקשורת עם הלקוח, בוחנת אחת לדקה את היתכנות ביצוע הפקודות שממתינות לביצוע, על פי ההנחיות שהמשתמש הגדיר ועל פי שערי המניה בפועל. ובמידה ונמצאת התאמה, מבצעת את הפקודה. למחלקה 16 פונקציות שונות אשר מממשות פקודות מסחר שונות, ועוד 9 פונקציות לעדכון נתונים שונים במסד הנתונים במקרה של ביצוע חלקי של רכישה/מכירה (ביצוע חלקי – כאשר קיימת פקודה לרכישת X מניות במחיר מסוים, ובפועל היתה כמות קטנה יותר של מניות למכירה באותו המחיר. במקרה זה המערכת תבצע את הרכישה של המניות שהוצעו למכירה ותמשיך לנות לרכוש את יתרת המניות להשלמת X).

### 2.1.8 צד שרת – מודול Stocks Information – Excel Sheets

מודול ה- Stock Information Excel Sheets כולל ארבעה קבצים אשר מחולקים לפי אינדקסים (מדדים). בכל קובץ קיימות כל המניות שבאינדקס ונתונים קבועים ומשתנים עבור כל מניה ומניה. הקבצים משמשים את הסרבר לקבלת מידע מסחר עדכני על מניות בהן המשתמשים סוחרים.

### 2.1.9 צד שרת – Database

בסיס הנתונים, Access 2016, שומר מידע על המשתמשים, על היסטוריית המסחר שלהם ונתוני מניות ומדדים. בין הטבלאות קיימים קשרים שונים שנאכפים על מנת לשמור על ההתאמה בין הנתונים השונים. הסבר מורחב על בסיס הנתונים, הטבלאות והקשרים בפרק נפרד בהמשך.



## 2.2 שפות תכנות, סביבת עבודה ובסיס הנתונים

בפיתוח הפרויקט השתמשתי בשפות תכנות Python 3.7 וב SQL. מערכת ההפעלה אשר מותקנת במחשבים (קליינט וסרבר) עליהם עבדתי הינה: Microsoft Windows 10. השתמשתי ב- Python בסביבת Pycharm לצורך כתיבת צד השרת וצד הקליינט וכן ב- QtDesigner לעיצוב ממשק המשתמש (UI).

כמוכן, השתמשתי ב- Microsoft Access 2016 כבסיס הנתונים של המערכת אשר מכיל בין היתר את פרטי המשתמשים ואת נתוני המסחר שלהם, העדכניים וההיסטוריים. לצורך ייבוא נתוני מסחר אמיתיים (שערי מניות וכדומה) אשר ישמשו את המערכת ואת המשתמשים, השתמשתי ב Excel 365 אשר בו הוגדרו המניות כאובייקטים שמתעדכנים עם נתוני מסחר מהאינטרנט מ- FinanceAPI.

להלן רשימת שפות התכנות, סביבות העבודה ובסיס הנתונים שנבחרו למימוש המערכת:

שפות תכנות	שם	תאור	לשימוש ב
שפות תכנות	Python 3.7	שפת תכנות מהנפוצה בעולם, בעלת ספריות רבות שמאפשרות לי לפעול ברבדים שונים רק דרך השפה.	צד השרת וצד הלקוח נכתבו בשפה.
	SQL	שפת תכנות לניהול מסדי נתונים	בצד השרת אני משתמש בשפה כדי לקבל, להכניס ולעדכן מידע ממסד הנתונים
	VBA	שפת תכנות שמאפשרת פעילות על תוכנות office	לכל קובץ excel פונקציית VBA שמרעננת את המידע מהרשת כאשר נפתח הקובץ לשימוש.
סביבות עבודה	Pycharm	סביבת פיתוח לפיתוח תוכנות בעיקר בשפת פייתון	בסביבת פיתוח זו בניתי את צד הלקוח וצד השרת
	Qt Designer	כלי לתכנון ובניה של ממשקי GUI עם אובייקטים של Qt.	לעיצוב ה GUI התאים לצרכי והמרת קובץ ה UI לקובץ Python
מסד נתונים	Access 2016	תוכנה לניהול בסיס נתונים עם יכולות הגבלה, שמירה וקישור	שמירה של כל נתוני המשתמשים, ההודעות, הפקודות שנכנסו והמניות שבעלות המשתמשים בצד השרת
תוכנות צד שלישי	Excel 365	תוכנה ליצירת גיליונות אלקטרוניים והפעלת פעולות עליהם עם שפת תכנות של התוכנה	לצורך משיכת נתוני מסחר עדכניים מהאינטרנט באמצעות הגדרת רשומות כמניות
פרוטוקולים	TCP	Transmission Control Protocol – פרוטוקול תקשורת מבטיח העברה אמינה בשימוש פרוטוקול IP.	לתקשורת בין צד השרת והלקוחות

ספריות ושימושן בפרויקט

שם הספרייה	פעולותיה ושימושה בפרויקט
socket	מודול זה מאפשר ליצור sockets בין מחשבים לפי כתובות IP ברשת LAN ובכך לאפשר תקשורת בין מחשבים. השרת יוצר socket ובעזרתו מתקשר עם כל הלקוחות.
pickle	מודול זה מאפשר להמיר אובייקט של פייתון לתווים וכך לשלוח אותם באמצעות socket ולהמיר בחזרה לאובייקט בצד המקבל. השימוש בפרויקט הוא בעיקר בשליחת רשימות, דבר זה חוסך שליחה של כל איבר ברשימה בנפרד.
struct	בעזרת struct ניתן להמיר מידע לבייטים ולשלוח דרך socket. עם struct ניתן להגדיר את ההמרה בצורה הרצויה (שליחת קובץ בחלקים) ונעשה בו שימוש בפרויקט כדי לשלוח קבצים גדולים כמו קבצי csv של היסטוריית מניה.
sys	מודול sys מספק פונקציות ומשתנים המשמשים לתמרון חלקים שונים בסביבת זמן הריצה של Python. בצד הלקוח אני משתמש ב sys.argv, רשימת הארגומנטים של שורת הפקודה שהועברה לקוד ה-Python, על מנת ליצור אובייקט מסוג QApplication.
threading	המחלקה Thread ממודול threading ממנה יורשות כמה מחלקות בפרויקט מאפשרת ריצה במקביל של פעולות שונות וכך הפרויקט יכול לממש את פעולותיו בצורה מיטבית. לדוגמא, כדי שהתכנית תוכל למכור מניה של לקוח X ובאותו הזמן לשלוח הודעה ללקוח Y יש להשתמש ב Thread.
pyodbc	Pyodbc הוא מודול Python שמאפשר גישה למסדי נתונים של ODBC דרך קוד פייתון. באמצעות pyodbc, אני מחבר את קוד ה-Python למקור הנתונים עם מנהל התקן ODBC (Access במקרה של הפרויקט שלי). כך אני יכול לשמור מידע של המשתמשים שיישמר גם כשהתכנית נעצרת, לחפש מידע ספציפי במסד שרך הקוד ולהשתמש במידע קבוע שנמצא במסד הנתונים.
hashlib	מודול hashlib מכיל שיטות הצפנה שונות שניתן לבצע בפונקציות hash על מחרוזת. השימוש של הפונקציה הוא עבור הצפנת סיסמאות בעזרת MD5 ושמירתן מוצפנות במסד הנתונים.
os	מודול OS מספק פונקציות לאינטראקציה עם מערכת ההפעלה. בעזרת מודול זה אני יכול למחוק ממחשב השרת וממחשב הלקוח קבצים שנוצרו בעקבות הרצת התכנית ובשלב מסוים כבר אין בהם שימוש. לדוגמא, כאשר לקוח מבקש לצפות בגרף של מניה השרת קורא את היסטוריית המניה המבוקשת ויוצר קובץ csv עם נתונים ושולח אותו ללקוח שפותח את קובץ לתצוגה גרפית. לאחר השליחה של השרת אין צורך לשמור את הקובץ בשרת ולאחר פתיחת הגרף אין צורך בקובץ במחשב הלקוח לכן התכנית מוחקת את הקבצים לאחר סיום השימוש בהם.
datetime	מודול זה מאפשר ליצור אובייקט מסוג תאריך ולבצע פעולות שונות על אובייקטים מסוג תאריך. השימוש של המודול בפרויקט נמצא ביצירת גרפים בטווח תאריכים מסוים, בדיקה אם הבורסה פתוחה, תיעוד פקודות נכנסו למערכת ועוד.
pytz	ספרייה זו מאפשרת חישובי אזורי זמן מדויקים עם התחשבות בשעון קיץ וחורף. שימוש בספרייה נמצא בבדיקה אם הבורסות (NYSE ו-NASDAQ) שנמצאות





## TAKE PROFIT STOCK EXCHANGE – Almog Michael Hemo Python project

	באזור זמן של מזרח ארה"ב פתוחות או לא.
holidays	ספרייה זו מאפשרת גישה לתאריכי חגים עדכניים לפי מדינות בהן החגים מצוינים. שימוש בספרייה נמצא בבדיקה אם הבורסות פתוחות היות והן בורסות אמריקאיות שסגורות בזמני החגים שבארה"ב.
time	מודול time של Python מספק דרכים רבות לייצוג הזמן בקוד אך אני משתמש בו רק לצורך השהייה לזמן מוגדר של תכנית. לדוגמא, השרת, במקביל לפעולת התקשורת עם הלקוחות, מנסה לבצע את כל פקודות המסחר פעם בדקה ומחכה דקה בעזרת time.
xlwings	את המידע העדכני על כל המניות בתכנית שניתן לסחור בהן אני שומר בקבצי excel 365 שמאפשרים לי לייצר אובייקטים של מניות ולעדכן את המידע מהרשת. בקוד אני זקוק למידע הזה ואני קורא מקבצי excel בעזרת ספריית xlwings שהיא בעלת פעולות שמאפשרות לפתוח קבצים אלו ולקרוא מהם לפי שורות ועמודות ספציפיות.
pandas	Pandas היא חבילת Python שמשמשת לאיסוף וניתוח נתונים. בפרויקט אני משתמש בה רק בצד הלקוח כאשר יש לקרוא קובץ csv ולהציג את הנתונים בו בגרף. Pandas יכולת ליצור frames של מידע, אותם אני יכול להציג בגרף matplotlib.
pandas_datareader	שימושה דומה ל-Pandas אך לה גם היכולת לקרוא מידע מפלטפורמות שונות. שימושה הוא לקרוא מהרשת ("yahoo finance") מידע על היסטוריית מניה, המידע מועבר לקובץ שנשלח ללקוח עבור גרף.
PyQt5	PyQt5 הוא ערכת כלים שמאפשרת ליצור GUI בשפת. בעזרת הספרייה הזו בניתי את GUI של הפרויקט. PyQt מאפשרת ליצור חלונות עם אובייקטים ולקשר ביניהם לבין פעולות שהקוד מכיל.
PyQtGraph	PyQtGraph היא ספריית גרפיקה של PyQt שבעזרתה ייצרתי אובייקטים של גרפים והגדרתי בהם מאפיינים שונים. השימוש בספרייה נמצא בגרפים של balance, gains, losses.
matplotlib	Matplotlib היא ספרייה גרפית עבור Python. קוד matplotlib בנוי כך שמספר מועט של שורות קוד מספיקות כדי ליצור חלון עם גרפים לפי התאמות אישיות. יש שימוש בספרייה זו בפרויקט ביצירת גרפים של היסטוריית מניות בהם יש חשיבות לאפשרויות של מספר גרפים במערכת צירים אחת ואפשרויות הרחבה ותנועה של Matplotlib מספקת.



## 3 תאור הקוד ובסיס הנתונים

בפרק זה נסביר על הפונקציות והמחלקות (Classes) בקוד, נתאר את הקוד של הפונקציות העיקריות של המערכת ואת הסכימות המרכזיות בבסיס הנתונים.

### 3.1 פונקציות ומחלקות (Classes) ב- Client

#### ▪ קובץ TPclient\_plot

שם הפונקציה	תאור תפקיד הפונקציה
<code>def ask_for_stock_excel()</code>	שולחת לסרבר בקשה לקבל קובץ CSV עם כל הפרמטרים של מניה או מדד בין טווח תאריכים מסויים. ואחראית על קבלת המידע שחוזר מהסרבר (במידה והמידע רב, הוא יתקבל בחלקים והפונקציה תאחד אותם לקובץ אחד)
<code>def get_plot_parameters_list()</code>	מקבלת מהמשתמש UI רשימה בוליאנית של פרמטרים נבחרים של המניה, וממירה אותם לרשימה של שמות הפרמטרים שתועבר לפונקציה אשר תציג אותם בגרף
<code>def open_a_plot_func()</code>	פונה לקובץ הפרמטרים ומחלצת ממנו את הפרמטרים שנבחרו ברשימה, יוצרת חלון Matplotlib מציגה את הנתונים בצורת גרף
<code>def delete_a_file()</code>	מקבלת מיקום של קובץ CSV שהסתיים שימוש בפתחת גרף וכעת יש למחוק אותו מהמחשב. הפונקציה בודקת אם אכן הקובץ קיים ומוחקת אותו.

#### ▪ קובץ TPui

שם המחלקה	תאור תפקיד המחלקה
<code>class Ui_client()</code>	בונה את כל הדפים והחלונות בממשק המשתמש מבחינת העיצוב, צבעים רקעים, פונטים, גודל ומיקום.

#### ▪ קובץ TPui\_widgets

שם המחלקה	תאור תפקיד המחלקה
<code>class Stock_for_buy_Widget()</code>	בונה עבור כל מניה שהמשתמש מבקש, אובייקט הכולל אפשרות קניה, סוג קניה (orders type in drop down list), צפיה במידע נוסף על המניה ויצירת גרפים. האובייקט יצבע באדום או בירוק על פי שינוי ערך המניה העכשוי (ירידה/עליה בהתאמה)
<code>class Empty_Stock_Widget()</code>	במידה והמשתמש מחפש מניה לפי קריטריונים מסויימים ולא נמצאת מניה כזו, המחלקה תציג אובייקט שיוצג זאת בפני המשתמש



## TAKE PROFIT STOCK EXCHANGE – Almog Michael Hemo Python project

בונה עבור מניה מהפרוטפוליו של המשתמש אובייקט הכולל אפשרות מכירה, סוג מכירה (orders type in drop down list), צפיה במידע נוסף על המניה ויצירת גרפים. האובייקט יצבע באדום או בירוק על פי שינוי ערך המניה העכשוי (ירידה/עליה בהתאמה)	<code>class Owned_Stock_Widget()</code>
במידה ולמשתמש אין מניות בפרוטפוליו שלו, המחלקה תציג אובייקט שיצין זאת בפני המשתמש	<code>class Empty_Owned_Stock_Widget()</code>
בונה את החלון בעת לחיצה על More Info, מקבל את המידע הנדרש ומציג אותו למשתמש	<code>class Information_Widget(QWidget)</code>
בלחיצה על קניה/מכירה, יבנה חלון עם עיצוב ואפשרויות אשר תואמות את סוג הקניה/מכירה (order) שביקש המשתמש, שיאפשרו לבצע את הקניה או המכירה.	<code>class Order_Window(QWidget)</code>
בונה את התצוגה של רשימת ההזמנות הפתוחות, עם אפשרות ביטול, המגת מידע נוסף וגרפים.	<code>class Order_Widget()</code>
במידה ואין הזמנות פתוחות, יבנה את התצוגה אשר תיידע את המשתמש	<code>class No_Orders_Widget()</code>
מציג מידע וגרפים למשתמש לצורך למידה, במסך Reports, על פי בחירת המשתמש מהאפשרויות הזמינות ב Drop down	<code>class Report_Widget()</code>

### ▪ קובץ TPclient

שם המחלקה	תאור תפקיד המחלקה
<code>class Client(Ui_client)</code>	יורשת ממחלקת Ui_client ואחראית על כל התקשורת עם הסרבר ומענה לפעולות המשתמש
שמות הפונקציות	תאור תפקיד הפונקציה
<code>def checking_if_the_market_is_open(self)</code>	בודקת אם שוק המניות פתוח כעת על פי Timezone של ארה"ב, בליקחת חשבון סופי שבוע, חגים ושעות ביום. במידה והשוק סגור, תוצג הודעה מתאימה למשתמש
<code>def connect_to_server(self)</code>	יוצרת Socket ומתחברת ל IP ולפורט המתאים של הסרבר. מפעיל את הפונקציה הבאה:
<code>def connect_buttons_to_functions(self)</code>	מחברת את כל הלחצנים/פקדים בממשק המשתמש אל הפונקציות המתאימות להן
<code>def sign_in_button_func(self)</code>	שולחת לסרבר בקשת התחברות של משתמש קיים ומעבירה אותו לעמוד ההתחברות
<code>def sign_up_button_func(self)</code>	שולחת לסרבר בקשת רישום של משתמש חדש ומעבירה אותו לעמוד הרישום
<code>def sign_in_in_button_func(self)</code>	מבצעת ולידציה בסיסית שנתוני האוטנטיקציה (שם משתמש



## TAKE PROFIT STOCK EXCHANGE – Almog Michael Hemo Python project

וסיסמא) שהוזנו תקינים, מעבירה אותם לבדיקה בצד הסרבר ותציג הודעה תואמת על פי התשובה	
מבצעת ולידציה בסיסית שנתוני הרישום שהוזנו תקינים, מעבירה אותם לבדיקה בצד הסרבר ותציג הודעה תואמת על פי התשובה. לדוגמא אם הסרבר מצא ששם המשתמש כבר קיים במערכת.	<code>def sign_up_up_button_func(self)</code>
מבצעת רענון לנתונים של הטאב אליו המשתמש מבקש לעבור	<code>def tab_change_checking_func(self)</code>
מבקשת מהסרבר לקבל את נתוני המאזן (יתרה/רווח והפסד) העכשויים ומעדכנת זאת בגרפים שבעמוד Balance	<code>def refresh_balance_tab(self)</code>
מבקשת מהסרבר לקבל את הדוחות השמורים של המשתמש, בטור, ויוצרת Report widget לכל אחד מהם	<code>def refresh_reports_tab(self)</code>
מפעילה את 2 הפונקציות הבאות	<code>def refresh_portfolio_tab(self)</code>
מבקשת מהסרבר את ההזמנות הפתוחות של המשתמש, אחד אחד, ויוצרת Order widget לכל אחד מהם. במידה ואין למשתמש הזמנות פתוחות, הוא יצור No Orders widget	<code>def refresh_client_orders(self)</code>
מבקשת מהסרבר את המניות שרכש המשתמש, אחד אחד, ויוצרת Owned Stock widget לכל אחד מהם. במידה ואין למשתמש מניות בבעלותו, הוא יצור Empty Owned Stock widget	<code>def refresh_client_stocks(self)</code>
כשהמשתמש מבקש להוסיף לחשבון סכום כסף מסויים, הפונקציה הנ"ל מופעלת ומבקשת מהסרבר לבצע את ההוספה ולרענן את נתוני המאזן/יתרות והגרפים	<code>def add_money_func(self)</code>
מקבלת שם של מניה וטווח תאריכים, לוקחת את הפרמטרים המבוקשים (High/Low etc) מעמוד Stock Market ומפעילה את הפונקציה Ask for Stock Excel שנמצאת ב TPCClientPlot	<code>def ask_for_stock_excel(self,?,?,?)</code>
לאחר חיפוש בעמוד Stock Market, הפונקציה מקבלת את כל המניות שנמצאו ומייצרת עבור כל אחד Stock for Buy widget. במידה ולא נמצאה מניה מתאימה היא תיצור Empty Stock widget	<code>def getting_stocks_info_to_market_func(self,?)</code>
מעבירה את הפרמטרים של חיפוש ספציפי מהממשק לרשימה ומפעילה את פונקציה getting_stocks_info_to_market עם הפרמטרים המבוקשים	<code>def specific_search_in_market_func(self)</code>
מעבירה מממשק המשתמש את שם המניה המבוקשת לפונקציה getting_stocks_info_to_market	<code>def free_search_in_market_func(self)</code>
בודקת איזה סוג בקשה המשתמש שלח בעמוד Reports ושולחת אותה לסרבר ליצירת הדוח בבסיס הנתונים. ולאחר מכן מעדכנת את עמוד Reports	<code>def send_request(self)</code>



## 3.2 פונקציות ומחלקות (Classes) ב- Server

▪ קובץ Server

שם המחלקה	תאור תפקיד המחלקה
<code>class ServerListener (threading.Thread)</code>	המחלקה מאפשרת למפעיל הסרבר לעצור את התכנית בעזרת הכנסת המילה השמורה "kill". המחלקה יורשת מ Thread ובכך מאפשרת ריצה ברקע התכנית
שמות הפונקציות	תאור תפקיד הפונקציה
<code>def run(self)</code>	הפונקציה מחכה תמיד לקלט מתאים ממפעיל הסרבר, ולפי בקשתו סוגרת את התכנית
שם המחלקה	תאור תפקיד המחלקה
<code>class Server</code>	המחלקה יורשת מ Thread ויוצרת Socket לחיבור המשתמשים למערכת
שמות הפונקציות	תאור תפקיד הפונקציה
<code>def get_clients(self)</code>	הפונקציה מנסה לקבל את כתובת המשתמש שהתחבר ויוצרת לו אובייקט ClientConnection
שם המחלקה	תאור תפקיד המחלקה
<code>class ClientConnection</code>	מטפלת בכל בקשות המשתמש. המחלקה יורשת מ Thread ובכך מאפשרת למספר משתמשים לתקשר עם הסרבר במקביל.
שמות הפונקציות	תאור תפקיד הפונקציה
<code>def run(self)</code>	מקבלת את ההודעה הראשונה מהמשתמש ובודקת אם משתמש זה מבקש להתחבר או ליצור חשבון חדש. בהתאם לבקשתו, היא מפעילה את הפונקציה המתאימה במחלקה. במידה והתקשורת עם המשתמש קורסת, הפונקציה זורקת exception שמודיע על נפילה בתקשורת.
<code>def sign_in_client(self)</code>	מקבלת ממשתמש רשום את נתוני התחברות ובודקת את קיומם במסד הנתונים (כולל התאמה בין שם משתמש לסיסמא). ומחזירה הודעה מתאימה (התחברות הצליחה/נכשלה).
<code>def sign_up_client(self)</code>	מקבלת נתוני הרשמה של משתמש חדש, ומנסה ליצור עבורו משתמש חדש במערכת. הפעולה תצליח רק במידה ואין כבר נתוני התחברות זהים במסד הנתונים
<code>def client_handling(self)</code>	הפונקציה מאזינה כל הזמן לבקשת המשתמש המחובר, לפי בהקוד המוסכם ששלח, היא מפעילה פונקציה שתטפל בבקשתו.
<code>def refresh_balance_tab(self)</code>	מוציאה ממסד הנתונים את היסטוריית הנתונים הכספיים של המשתמש ושולחת לו אותם.
<code>def refresh_open_orders(self)</code>	מוציאה ממסד הנתונים את פקודות המסחר הפתוחות של המשתמש



## TAKE PROFIT STOCK EXCHANGE – Almog Michael Hemo Python project

ושולחת לו אותם.	
מוציאה ממסד הנתונים את רשימת המניות שבבעלות המשתמש ושולחת לו אותם.	<code>def refresh_own_stocks(self)</code>
מוציאה ממסד הנתונים את הדוחות וההודעות של המשתמש ושולחת לו אותם.	<code>def refresh_reports(self)</code>
מקבלת מהמשתמש סכום להוספה לחשבון ומעשכנת את מסד הנתונים בהתאם	<code>def add_money(self)</code>
הפונקציה מקבלת מידע על מניה מסוימת מהמשתמש, יוצרת קובץ אקסל עם היסטוריית הנתונים המבוקשת בעזרת הפונקציה <code>Create_Excel_Func</code> . לאחר מכן היא שולחת למשתמש את הקובץ שנוצר בחלקים ולבסוף מוחקת את הקובץ מהסרבר.	<code>def send_stock_excel(self)</code>
מקבלת תנאי חיפוש מניה ספציפים מהמשתמש, מוצאת מניות שתואמות לתנאי החיפוש בעזרת פונקציית עזר ושולחת את נתוני המניות שנמצאו בחזרה.	<code>def specific_searching(self)</code>
מקבלת שם של מניה או <code>Ticker Symbol</code> , מחפשת מניות מתאימות בעזרת פונקציית עזר ושולחת את נתוני המניות שנמצאו בחזרה.	<code>def free_searching(self)</code>
מקבלת <code>Ticker</code> של מניה ואת האינדקס בה הוא נמצא ומחזירה בעזרת פונקציית עזר מידע נרחב על המניה ועל החברה הציבורית שלה.	<code>def more_stock_info(self)</code>
מקבלת נתונים ליצירת פקודת מסחר, בודקת אם מדובר בפקודת קניה או מכירה ומפעילה פונקציית עזר מתאימה.	<code>def create_order(self)</code>
מקבלת מספר מפתח של פקודת קניה פתוחה ומוחקת אותה ממסד הנתונים, במידה והיא קיימת.	<code>def delete_buy_order(self)</code>
מקבלת מספר מפתח של פקודת מכירה פתוחה ומוחקת אותה ממסד הנתונים, במידה והיא קיימת. כמוכן היא מוחקת מטבלת המניות את הנתון כי המניה מוצעת למכירה.	<code>def delete_sell_order(self)</code>
יוצרת רשומה במסד הנתונים ובה הודעה שנוצרה על ידי הסרבר עבור המשתמש	<code>def create_report(self,?, ?, ?)</code>
הפונקציה מקבלת הוראה למחיקת דוח ומבצעת את המחיקה	<code>def delete_report(self)</code>
מקבלת בקשה מהמשתמש ליצירת <code>Report</code> , יוצרת <code>Report</code> מתאים בעזרת פונקציית עזר ומכניסה אותו למסד הנתונים.	<code>def create_request(self)</code>



## ▪ קובץ Server Functions

שמות הפונקציות	תאור תפקיד הפונקציה
<code>def checking_if_the_market_is_open_func()</code>	בודקת אם שוק המניות פתוח כעת על פי Timezone של ארה"ב, בליקחת חשבון סופי שבוע, חגים ושעות ביום. במידה והשוק סגור או פתוח, תחזיר True/False.
<code>def delete_a_file(?)</code>	מקבלת כתובת של קובץ במחשב הסרבר, בודקת את קיומו ומוחקת אותו.
<code>def create_excel_func(?)</code>	מקבלת Ticker של מניה וטווח תאריכים, קוראת מהאינטרנט את היסטוריית המניה בטווח המבוקש (ממקור Yahoo Finance) ושומרת את ההיסטוריה בקובץ CSV. לבסוף מחזירה את מיקום הקובץ.
<code>def specific_searching_in_market_func(?)</code>	מקבלת את תנאי החיפוש של מניות ובודקת אם קיימות מניות שעומדות בתנאי החיפוש בקבצי האקסל אשר שומרים את נתוני המניות. בפתיחת קובץ האקסל, הנתונים על המניות מתעדכנים מהאינטרנט. לבסוף סוגרת ושומרת את הקובץ ומחזירה רשימה עם המניות שנמצאו.
<code>def free_searching_in_market_func(?)</code>	מקבלת את שם המניה שמחפש המשתמש, בודקת אם המניה קיימת בקבצי האקסל אשר שומרים את נתוני המניות. בפתיחת קובץ האקסל, הנתונים על המניה מתעדכנים מהאינטרנט. לבסוף סוגרת ושומרת את הקובץ ומחזירה את המניה שנמצאה.
<code>def more_info_for_stock(?, ?)</code>	מקבלת את שם ה Ticker ואת האינדקס בו המניה נסחרת, ומחזירה את הנתונים המורחבים והעדכניים מקובץ האקסל.
<code>def description_fix(?)</code>	מקבלת הסבר בשורה אחת שלמה ומחלקת אותה למספר שורות כדי להתאים את התוכן לרוחב המתאים
<code>def industry_fix(?)</code>	מקבלת שם של התעשייה בשורה אחת שלמה ומחלקת אותה למספר שורות כדי להתאים את התוכן לרוחב המתאים
<code>def create_buy_order(?, ?)</code>	מקבלת נתונים ליצירת פקודת קניה ויוצרת רשומה במסד הנתונים לפקודה, בטבלה המתאימה.
<code>def create_sell_order(?, ?)</code>	מקבלת נתונים ליצירת פקודת מכירה ויוצרת רשומה במסד הנתונים לפקודה, בטבלה המתאימה.
<code>def searching_for_clients_orders(?, ?)</code>	מקבלת את מספר המפתח של המשתמש ומחזירה רשימה של כל הפקודות הפתוחות שהוא יצר.
<code>def searching_for_clients_stocks(?, ?)</code>	מקבלת את מספר המפתח של המשתמש ומחזירה רשימה של כל המניות שבבעלותו.



## TAKE PROFIT STOCK EXCHANGE – Almog Michael Hemo Python project

מקבלת את מספר המפתח של המשתמש ומחזירה רשימה של כל הדוחות וההודעות של המשתמש.	<code>def searching_for_clients_reports(?, ?)</code>
מקבלת את הבקשה של המשתמש ומחזירה את תוכן התשובה המתאימה	<code>def get_answer_to_client_request_func(?, ?, ?)</code>

### קובץ Orders execute

שם המחלקה	תאור תפקיד המחלקה
<code>class OrdersRunner(threading.Thread)</code>	המחלקה אחראית למימוש כל פקודות המסחר הפתוחות, תוך ריצה ברק' ללא הפרעה לפעולות הסרבר.
שמות הפונקציות	תאור תפקיד הפונקציה
<code>def run(self)</code>	מנסה לבצע את פקודות המסחר בהתאם לכל אינדקס אחת לדקה.
<code>def execute_orders(self, ?, ?)</code>	מקבלת שם טבלת פקודות ושם אינדקס. מוציאה מהטבלה את כל פקודות המסחר הפתוחות, פותחת ומעדכנת את קובץ האקסל המתאים ומבצעת את כל פקודות המסחר בהתאם לפונקציה המבוקשת לכל פקודה ופקודה.
<code>def searching_stock_price_in_excel_file(self, ?, ?)</code>	פונקציית עזר לפקודות המסחר אשר מחזירה את המחיר הנוכחי של המניה שביקשו
<code>def buy_market_func(self, ?, ?, ?)</code>	הפונקציות הללו מנסות לממש פקודות קניה פתוחות. כל פונקציה מתאימה לפקודה מסוג שונה, אך בבסיסן לכולן אותה מטרה. כל פונקציה מקבלת את נתוני הקנייה של פקודה פתוחה (מחיר המניה שהמשתמש מעוניין לשלם וכמות המניות שמעוניין לרכוש), את האינדקס שבו נמצאת המניה המבוקשת ואת אובייקט טבלת האקסל בה קיימים הנתונים העדכניים של המניה המבוקשת.  על מנת לבצע את הרכישה יש לבדוק אם הנתונים העדכניים של המניה עומדים בתנאים של הפקודה (לכל פקודה יכולות שונות) ובדיקת היתכנות הרכישה (בדיקה שלמשתמש יש מספיק כסף לביצוע הרכישה באותו רגע). בנוסף גם תיבדק האפשרות לרכישה מלאה או חלקית של המניות בפקודה לפי נתוני המשתמש ונתוני המסחר (לפי מספר המניות המבוקשות לקניה על ידי המשתמש ומספר המניות המוצעות למכירה). עבור כל תרחיש אפשרי יתעדכן בהתאם מסד הנתונים. יעודכנו ואף יימחקו פקודות המסחר, יעודכנו נתונים כספיים של המשתמש וכמובן יעודכנו המניות שבבעלותו.
<code>def buy_limit_func(self, ?, ?, ?):</code>	
<code>def buy_stop_loss_func(self, ?, ?, ?)</code>	
<code>def buy_stop_limit_func(self, ?, ?, ?)</code>	
<code>def buy_aon_func(self, ?, ?, ?)</code>	
<code>def buy_ioc_func(self, ?, ?, ?)</code>	
<code>def buy_buy_above_func(self, ?, ?, ?)</code>	
<code>def buy_fok_func(self, ?, ?, ?)</code>	
<code>def sell_market_func(self, ?, ?, ?)</code>	הפונקציות הללו מנסות לממש פקודות מכירה פתוחות. כל





## TAKE PROFIT STOCK EXCHANGE – Almog Michael Hemo Python project

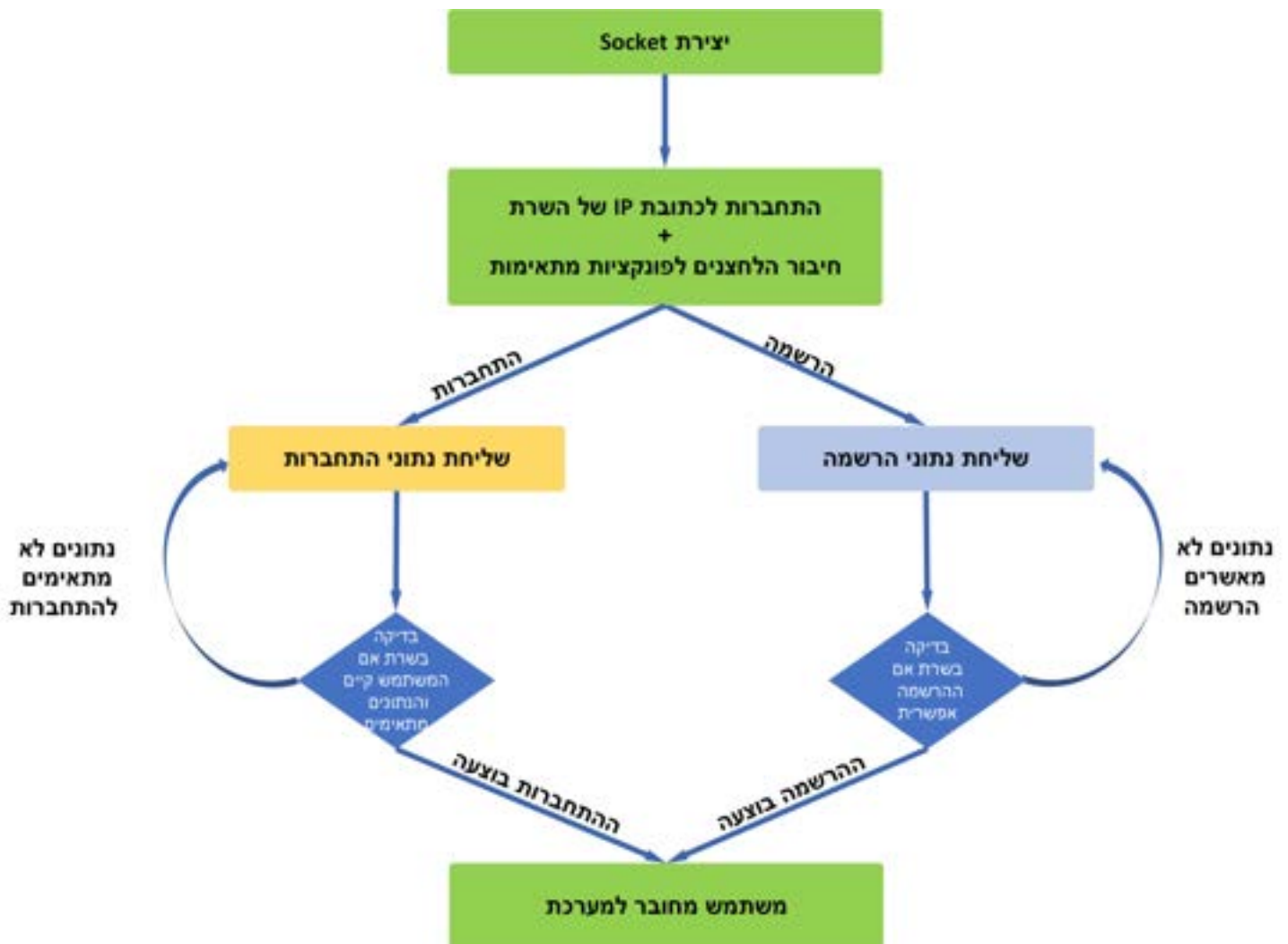
<p>פונקציה מתאימה לפקודה מסוג שונה, אך בבסיסן לכולן אותה מטרה. כל פונקציה מקבלת את נתוני המכירה של פקודה פתוחה (מחיר המניה שהמשתמש מעוניין למכור וכמות המניות שמעוניין למכור), את האינדקס שבו נמצאת המניה המבוקשת ואת אובייקט טבלת האקסל בה קיימים הנתונים העדכניים של המניה המבוקשת.</p> <p>על מנת לבצע את המכירה יש לבדוק אם הנתונים העדכניים של המניה עומדים בתנאים של הפקודה (לכל פקודה יכולות שונות). עבור כל תרחיש אפשרי יתעדכן בהתאם מסד הנתונים. יעודכנו ואף יימחקו פקודות המסחר, יעודכנו נתונים כספיים של המשתמש, ואף ימחקו המניות שמכר מרשימת המניות שבבעלותו. בנוסף, עבור ביצוע מכירה, התכנית תבדוק אם המשתמש הרוויח או הפסיד מהעסקה, ותעדכן בהתאם את גרף ה Gain/Losses</p>	<code>def sell_limit_func(self, ?, ?, ?)</code>
	<code>def sell_stop_loss_func(self, , ?, ?, ?)</code>
	<code>def sell_stop_limit_func(self, ?, ?, ?)</code>
	<code>def sell_aon_func(self, ?, ?, ?)</code>
	<code>def sell_ioc_func(self, ?, ?, ?)</code>
	<code>def sell_fok_func(self, ?, ?, ?)</code>
	<code>def sell_take_profit_func(self, ?, ?, ?)</code>
מוסיפה למסד הנתונים מניה שנרכשה בהצלחה	<code>def update_owned_stocks_table(self, ?, ?)</code>
מוחקת ממסד הנתונים מניה שנמכרה בהצלחה	<code>def delete_owned_stock(self, ?, ?)</code>
מעדכנת את מאזן החשבון של המשתמש בטבלת המשתמשים לאחר ביצוע עסקה	<code>def update_users_table(self, ?, ?)</code>
מעדכנת את מאזן החשבון של המשתמש בהיסטוריית הנתונים האישיים, לאחר ביצוע עסקה	<code>def update_balance_table(self, ?, ?)</code>
בודקת לאחר מכירת מניה אם העסקה היתה רווחית או לא בעזרת השוואה עם המחיר בו המניה נרכשה. בהתאם לכל מצב, מעדכנת את היסטוריית ההפסדים או הרווחים	<code>def update_gains_or_losses(self, ?, ? , ?)</code>
מוחקת לגמרי פקודה שבוצעה ואין בה שימוש יותר	<code>def delete_order(self, ?, ?)</code>
מעדכנת את מספר המניות לקניה בפקודה פתוחה, כאשר מתבצעת רכישה חלקית	<code>def update_order_stocks_amount(self, ?, ? , ?)</code>
מעדכנת את מספר המניות שבבעלות משתמש כאשר רק חלק מהמניות נמכרו בפקודה שהתבצעה.	<code>def update_owned_stock_amount_and_order(self, ?, ? , ?)</code>
מוחקת את הסימון על מניה בבעלות שהתבצע על חלקה פקודת מסחר לאחר שהפקודה התבצעה.	<code>def delete_sell_order_from_owned_stock(self, ?, ?)</code>



### 3.3 תאור התחברות Client – Server

תרשים זה מתאר את תהליך ההתחברות של הלקוח למערכת.

תחילה נוצר socket דרכו הוא יתקשר עם השרת, הוא מתחבר לשרת לפי כתובת ה-IP. מכאן המשתמש בוחר עם להירשם למערכת או להתחבר עם משתמש רשום. בשתי האפשרויות עליו להכניס פרטי התחברות/הרשמה ולקבל אישור מהשרת שהפרטים אפשריים ותואמים. כאשר ההרשמה/ההתחברות בוצעה הלקוח מחובר למערכת בתור המשתמש ויכול להתחיל לסחור.





### 3.4 קוד לדוגמא – תהליך רכישה מניה

הפונקציה הבאה אחראית על תהליך רכישה של מניה:  
היא מקבלת נתונים של פקודת רכישה מסוג Limit (כלומר רק כששער המניה יגיע בעתיד לשער אותו מוכן המשתמש לשלם עבור המניה, המערכת תבצע את הרכישה בפועל), בודקת אם התנאי מתקיים. במידה ותנאי הרכישה מתקיימים, הפונקציה בודקת כמה מהמניות שהמשתמש ביקש לרכוש, הוא אכן יכול לרכוש באותו רגע על פי היתרה שלו, ובהתאם מבצעת רכישה מלאה או חלקית ומעדכנת את נתוני המשתמש ואת הפקודה במידת הצורך.

```
def buy_limit_func(self, one_order, index, sht):
    stock_price = self.searcrhing_stock_price_in_excel_file(self, sht, one_order[4])
    # stock_for_buy_list = order_creator_key, ticker, index, today date, price, stocks_amount
    stock_for_buy_list = [one_order[2], one_order[4], index, str(datetime.date.today()), stock_price, one_order[5]]

    if stock_for_buy_list[4] <= one_order[6]:
        full_price = stock_for_buy_list[4] * one_order[5]
        self.cursor.execute("SELECT initial_amount FROM users_table WHERE user_key_number = ?", one_order[2])
        current_amount = self.cursor.fetchone()[0]

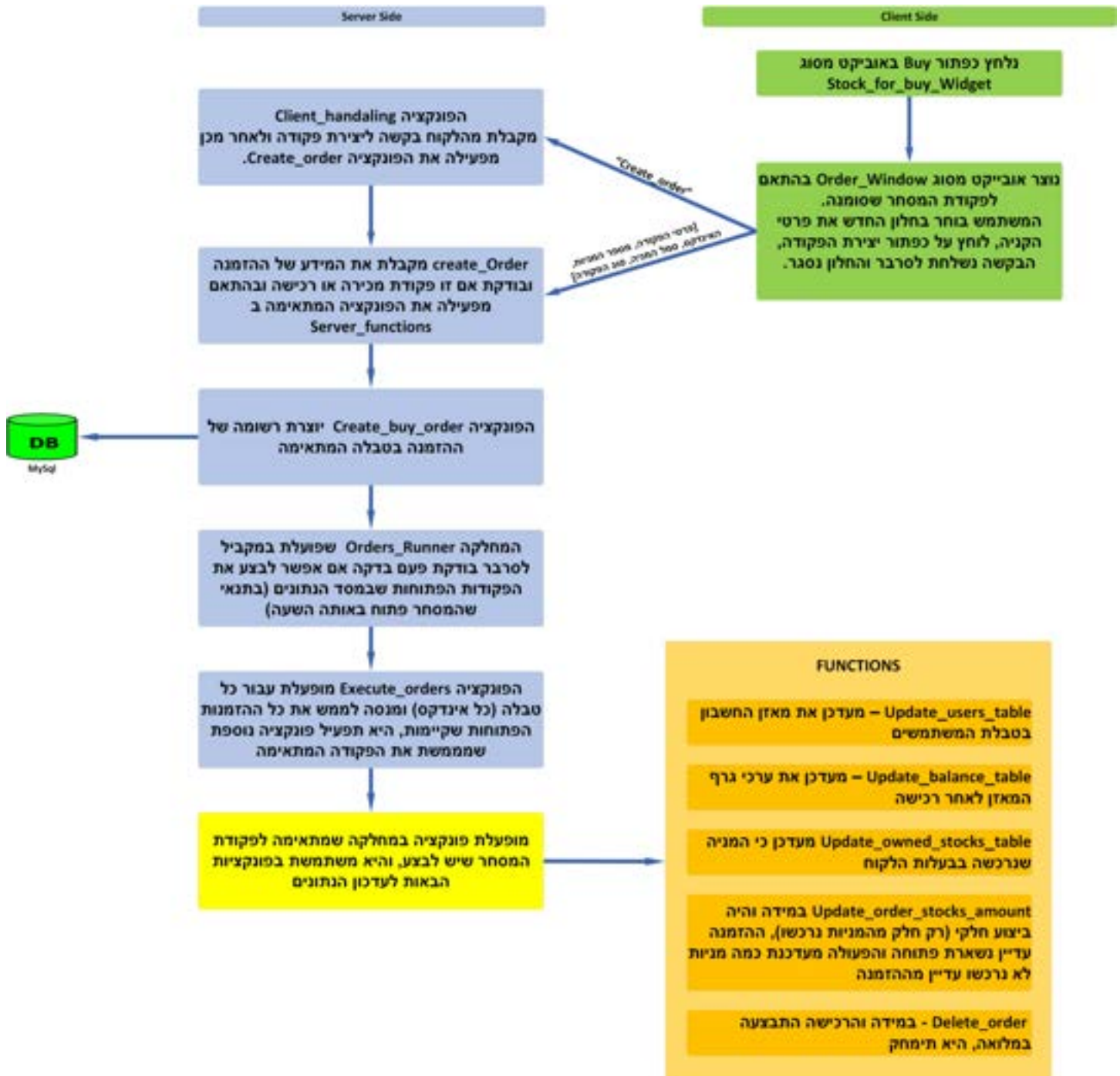
        if current_amount >= full_price:
            print("full transaction")
            self.update_users_table(current_amount - full_price, one_order[2])
            self.update_balance_table(current_amount - full_price, one_order[2])
            self.update_owned_stocks_table(stock_for_buy_list, index)
            self.delete_order(one_order[0], index)

        elif one_order[5] > 1:
            possible_amount = int(current_amount / stock_for_buy_list[4])
            if possible_amount != 0:
                stock_for_buy_list[5] = possible_amount
                full_price = stock_for_buy_list[4] * stock_for_buy_list[5]
                self.update_users_table(current_amount - full_price, one_order[2])
                self.update_balance_table(current_amount - full_price, one_order[2])
                self.update_owned_stocks_table(stock_for_buy_list, index)
                self.update_order_stocks_amount(one_order[0], index, one_order[5] - possible_amount)
```



## TAKE PROFIT STOCK EXCHANGE – Almog Michael Hemo Python project

להלן תרשים זרימה המתאר את תהליך הרכישה המלא אשר מתבצע במערכת. הקוד המתואר למעלה מממש את הרשום במלבן הצהוב אשר אחראי על לוגיקת הרכישה והפעלת פונקציות נוספות:





### 3.5 קוד לדוגמא – יצירת גרף

מטרת הפונקציה `ask_for_stock_excel` היא לקבל שם של מניה, טווח תאריכים ונתונים גרפים ולפתוח חלון `matplotlib` עם הגרף המבוקש. היא משתמשת בפונקציות עזר כמו `get_plot_parameters_list` כדי לקבל את נתוני הגרף כמחרוזת מהנתונים שסומנו ב UI והפונקציה `open_a_plot_func` שמקבלת את מיקום הקובץ עם היסטוריית המניה ויוצרת גרף מתאים.

בשלב הראשון מוגדרת הרשימה שתישלח לשרת, הפונקציה לא יודעת אם מפעיל הפונקציה הכניס תאריכים מסוג `Qdate` או מסוג `pydate` לכן ההמרה המתאימה נמצאת ב- `try` ו- `except`. לאחר מכן נשלחת הודעת בקשה לשרת עם ההודעה השמורה "stock\_excel". בשלב הזה השרת נענה לבקשת הלקוח ומחכה להודעה עם רשימת פרמטרים שהוא זקוק להם ליצירת קובץ עם היסטוריית המניה.

לאחר שהשרת קיבל את הנתונים הדרושים הוא יוצר קובץ `csv` עם היסטוריית המניה בטווח התאריכים המבוקש ושולח אותו ללקוח בחלקים מכיוון שמדובר בקובץ גדול ושליחה שלו בשלמותו עלולה לקחת זמן רב, לפגוע בתוכנו ואף לא להגיע ליעד.

הפונקציה מקבלת את החלקים ומרכיבה אותם לקובץ אחד בסדר הנכון. לבסוף היא קוראת לפונקציה `get_plot_parameters_list` שמקבלת אובייקטים מסוג `radio` ומחזירה מחרוזת עם הפרמטרים שיוצגו בגרף וגם קוראת לפונקציה `func_open_a_plot` שיוצרת את הגרף המתאים, מציגה אותו על המסך ומוחקת את הקובץ כאשר אין בו יותר שימוש (בעזרת הפונקציה `delete_a_file`).

```
def ask_for_stock_excel(client, ticker_name, start_date, end_date, plot_check_boxes):
    """
    Ask for excel file with the stock history
    """
    try:
        list_to_server = (ticker_name, start_date.toPyDate(), end_date)
    except:
        list_to_server = (ticker_name, start_date, end_date)
    client.send("stock_excel".encode())
    client.send(pickle.dumps(list_to_server))

    data = b""
    size = struct.calcsize('>Q')
    while len(data) < size:
        data += client.recv(1024)
    packed_size = data[:size]
    data = data[size:]
    actual_size = struct.unpack('>Q', packed_size)[0]
    while len(data) < actual_size:
        data += client.recv(1024)
    data = data[:actual_size]
    with open(PATH_EXCEL + ticker_name + '.csv', 'wb') as f:
        f.write(data)
    plot_parameters_list = get_plot_parameters_list(plot_check_boxes)

    open_a_plot_func(PATH_EXCEL + ticker_name + '.csv', ticker_name, plot_parameters_list)
```



## 3.6 סכימות בסיס הנתונים

להלן הטבלאות הרלוונטיות עבור כל סוג Index\מדד:

- **טבלת Users**

טבלה ששומרת את פרטי המשתמשים. לכל משתמש מספר מפתח ייחודי, שם משתמש, סיסמא שהוצפנה ב MD5, שם מלא, כתובת אימייל וסכום בחשבון.

users_table	
user_key_number	PK
username	
password	
fullname	
mail	
initial_amount	

- **טבלת balance\_history**

טבלה ששומרת עבור כל משתמש, לפי מספר המפתח שלו, את היסטוריית היתרות\רווח והפסד שלו.

balance_history_table	
user_key_number	PK
balance_y	
gains_y	
losses_y	

- **טבלת Reports**

טבלה ששומרת את ההודעות של כל משתמש. עבור כל הודעה, נשמר סוג ההודעה (גרף או טקסט), את התאריך בו נכתבה, הכותרת, התוכן ואת מספר המפתח של המשתמש.

reports_table	
report_number	PK
report_type	
sent_addressee_number	
article	
time	
content	



- **Nasdaq100\_orders טבלת**

הטבלה הנ"ל קיימת בנפרד עבור כל אינדקס\מדד שנתמך במערכת.  
הטבלה שומרת את כל ההזמנות\פקודות הפתוחות מכל המשתמשים שממתינות להתממשות.  
עבור כל הזמנה קיים מספר ייחודי חח"ע, תאריך הזמנה, סוג ההזמנה, מספר מפתח של המשתמש שיר את ההזמנה ופרמטרים נוספים של המניה ופקודת המסחר.

Nasdaq100_orders_table	
order_number	🔑
order_type	
order_creator_key	
order_date	
ticker	
stocks_number	
order_info	
owned_stock_number	

- **Nasdaq100\_owned\_stocks טבלת**

הטבלה הנ"ל קיימת בנפרד עבור כל אינדקס\מדד שנתמך במערכת.  
שומרת את כל המניות שבבעלות המשתמשים. עבור כל מניה נשמר תאריך הרכישה, מחיר הרכישה, מספר המניות, פקודת הרכישה ומספר מפתח של בעל המניה.

Nasdaq100_owned_stocks_table	
stock_number	🔑
owner_key	
ticker	
buy_date	
buy_price	
stock_amount	
sell_order	

- **Concepts\_explanations טבלת**

טבלה ששומרת מושגים מעולם המסחר ואת פירושם בכדי לאפשר למשתמשים ללמוד את התחום.  
הטבלה משתמש את עמוד Reports.

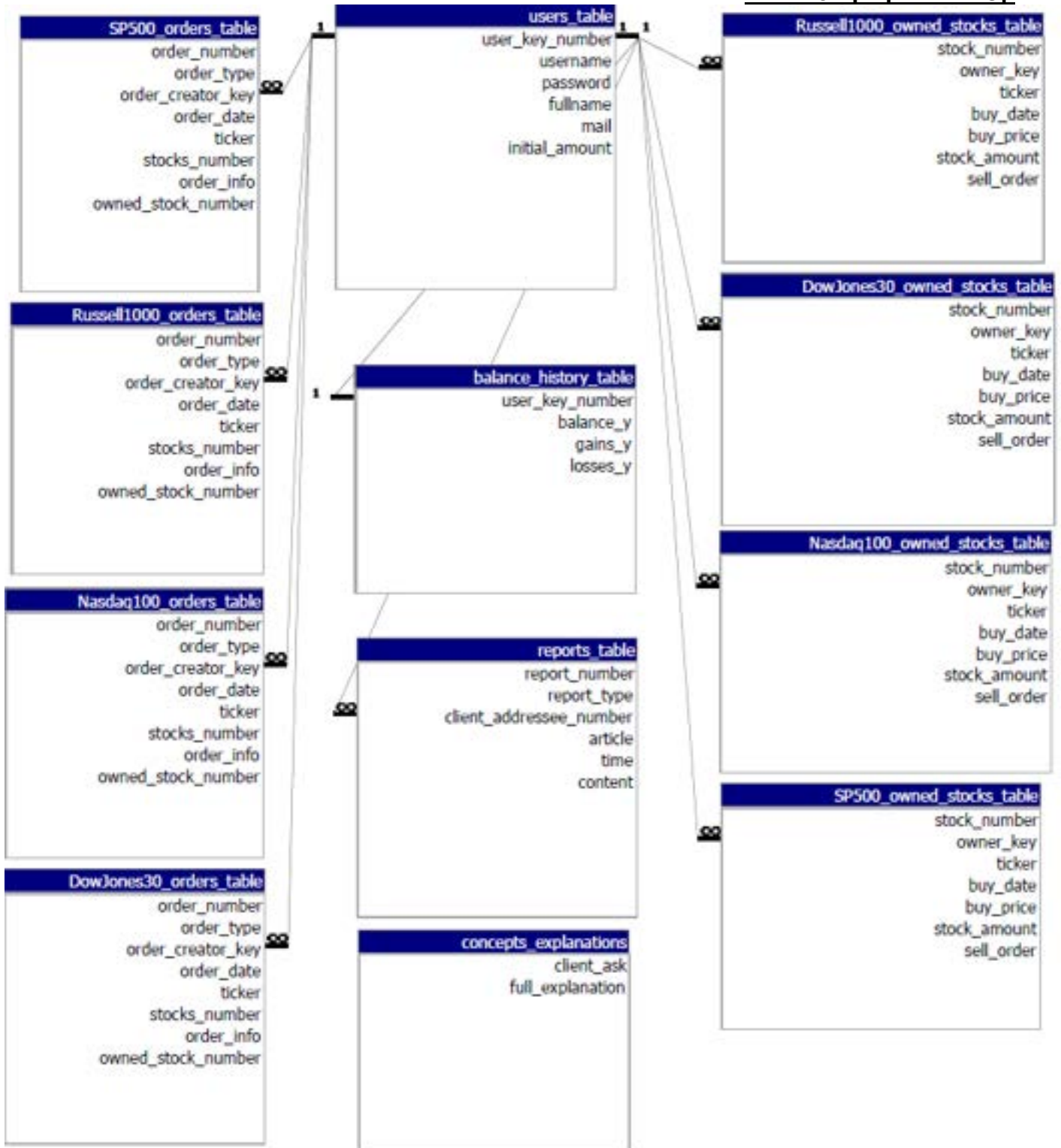
concepts_explanations	
client_ask	🔑
full_explanation	





## TAKE PROFIT STOCK EXCHANGE – Almog Michael Hemo Python project

### קשרי הגומלין בין הטבלאות:







**בין הטבלאות נאכפים קשרי גומלין אשר שומרים על תקינות הנתונים:**

- קשר בין טבלת User לטבלת Balance History :  
זהו קשר של יחיד ליחיד, עבור כל משתמש קיימת רשומה אחת ששומרת את היסטוריית היתרות שלו
- קשר בין טבלת User לטבלת Reports :  
זהו קשר של יחיד לרבים. עבור כל משתמש קיימות מספר הודעות.
- קשר בין טבלת User לטבלת Owned Stocks :  
קשר של יחיד לרבים. כל משתמש יכול להיות בעל מספר מניות.
- קשר בין טבלת User לטבלת Orders :  
קשר של יחיד לרבים. כל משתמש יכול להזין מספר פקודות/הזמנות לביצוע.



## 4 ממשק המשתמש

בפרק זה נתאר את המסכים העיקריים של המערכת המהווים ממשק למשתמשי הקצה. נתאר את התפריטים והכפתורים השונים ומה הם מבצעים.

### 4.1 דף רישום או כניסה למערכת

הדף הנ"ל מאפשר למשתמש רשום להגיע לדף הבא לביצוע Sign In – כניסה למערכת.  
וגם מאפשר למשתמש חדש להגיע לדף ההרשמה – Sign Up





## 4.2 דף רישום משתמש חדש – Sign Up

הדף הנ"ל מאפשר למשתמש חדש להירשם למערכת. המשתמש נדרש להזין פרטים אישיים (שם מלא וכתובת אימייל), סכום התחלתי למסחר, שם משתמש וסיסמא אשר תשמש אותו בהמשך לצורך אימות לאוטנטיקציה וכניסה למערכת - Sign In)

The screenshot shows a web application window titled "Take Profit". At the top center is a circular logo containing a bull. Below the logo, the form contains the following fields and labels:

- FULL NAME**: Input field with the text "almog hemo".
- MAIL ADDRESS**: Input field with the text "almoghemo@gmail.com".
- INITIAL AMOUNT**: Input field with the text "3500".
- USERNAME**: Input field with the text "Almog123".
- PASSWORD**: Input field with the text "almog4343@@".


At the bottom of the form is a large button labeled **SIGN UP**.



### 4.3 דף כניסה למערכת למשתמש קיים – Sign In

הדף הנ"ל מאפשר למשתמש רשום להיכנס לחשבון שלו במערכת על ידי הזנת שם המשתמש שלו וסיסמא. לאחר אימות אשר המערכת מבצעת, של התאמת הסיסמא לשם המשתמש ובדיקה ששם המשתמש קיים, יופנה המשתמש למסך הבית ויראה את נתוני המסחר שלו אשר נשמרים בבסיס הנתונים.

Take Profit



USERNAME

Almog123

PASSWORD

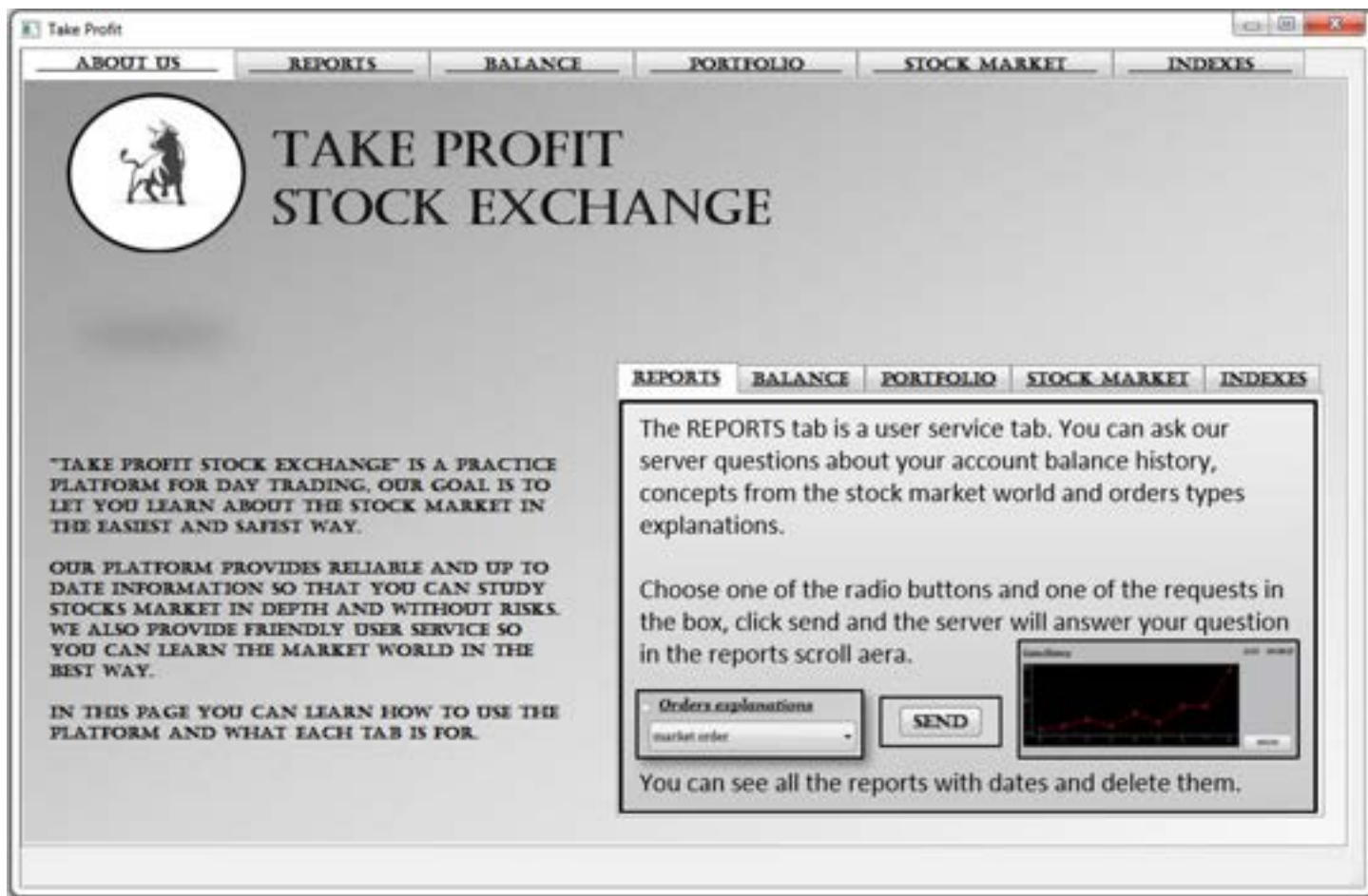
abcd4343@q!

SIGN IN



## 4.4 דף אודות – About Us

בדף About Us המשתמש יכול לקרוא הסבר על התוכנה והנחיות כיצד להשתמש בה.





## 4.5 דף דוחות - Reports

בדף Reports יכול המשתמש להפיק גרפים שונים המציגים את מידע המסחר שלו וכן יכול לצפות בהסברים כתובים של מושגי מסחר.





## 4.6 דף מאזן החשבון - Balance

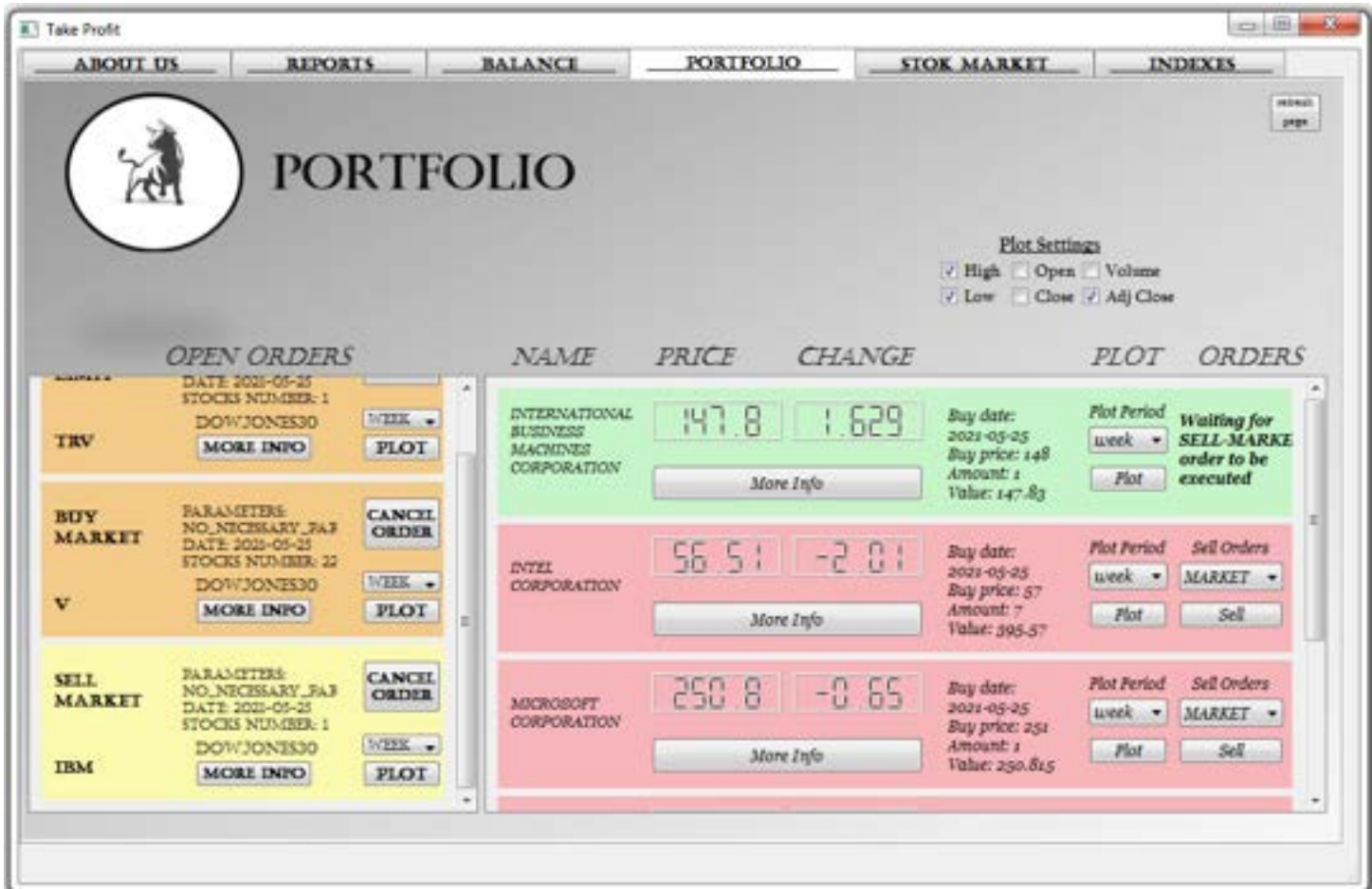
בדף זה יוכל משתמש קיים לצפות ביתרה שלו, ובסכום אותו הרוויח ו\או הפסיד. כמוכן, מוצגים גרפים המתארים את הרווח, ההפסד ויתרת החשבון בציר הזמן. בנוסף, המשתמש יכול להוסיף כסף למאזן שלו ממסך זה.





## 4.7 דף תיק ההשקעות - Portfolio

בדף Portfolio מוצגות למשתמש כל המניות שנמצאות כרגע בבעלותו (בצד ימין של המסך) וכן את כל ההזמנות שהזרים למערכת וטרם בוצעו (בצד שמאל של המסך).  
ברשימת ההזמנות הפתוחות, ניתן לבצע ביטול הזמנה, לצפות במידע נוסף על המניה ובגרף המניה.  
ברשימת המניות שבבעלותו, יוצג מחיר המניה, אחוז השינוי בשער המניה, ותאריך הרכישה, כמה מניות נרכשו, שם החברה ומידע נוסף. כמוכן, ה Widget של כל עסקה יצבע בירוק במידה והמניה עולה ברגע זה או באדום במידה והיא נמצאת בירידה.







## 4.8 דף שוק המניות - Stock Market

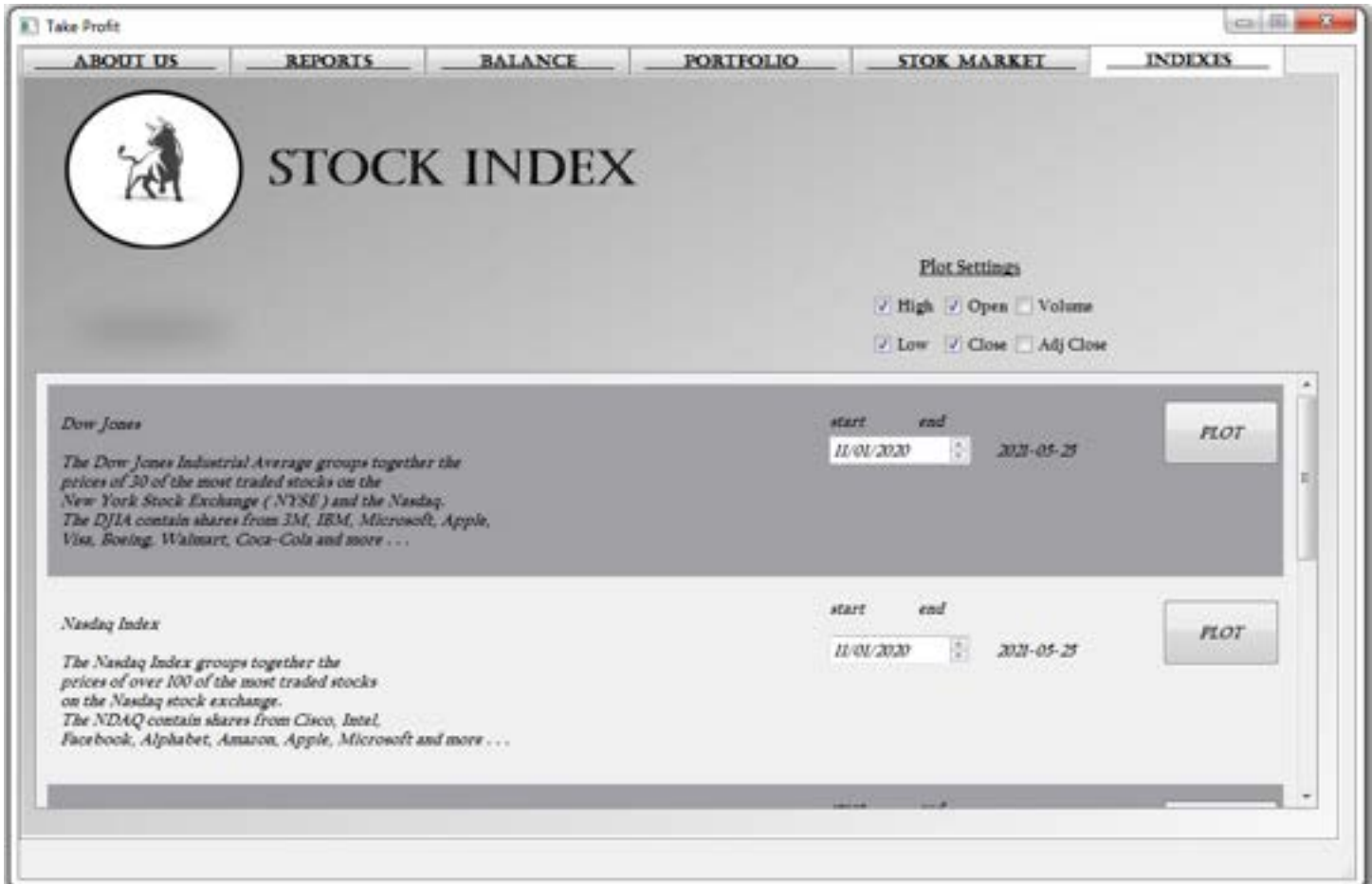
באמצעות דף זה יכול המשתמש לחפש ולהציג מניות מתוך רשימה ארוכה של מניות. המשתמש יכול לבצע חיפוש מניה על פי שם בחיפוש חופשי או מתוך רשימה מובנית (Drop down).  
כמוכן יכול המשתמש לסנן את תוצאות החיפוש על פי קריטריונים שונים כגון Market cap, Change Percentage and Price.  
לאחר ביצוע חיפוש וסינון, יוצגו המניות התואמות, המידע עליהן, גרף ואפשרות רכישת המניה.





## 4.9 דף מדדים - Indexes

דף המדדים \ Indexes מציג הסברים על כל מדד ומדד אשר המערכת מאפשרת לסחור במניות הכלולות בו וכן אפשרות להציג גרפים של המדדים בטווח תאריכים שבוחר המשתמש.





## 4.10 דף יצירת פקודה - Create Order

לאחר שמשתמש בוחר לרכוש מניה מסויימת, או למכור, הוא מופנה למסך Create Order לצורך הגדרת סוג הרכישה\מכירה.  
ניתן לבחור רכישה\מכירה במחיר Market (מחיר נוכחי), או להגדיר מחיר יעד, Limit, ואם וכאשר מחיר המניה יגיע למחיר שהוגדר, אז המערכת תבצע את הרכישה\מכירה.  
כמוכן, יש להגדיר את מספר המניות שמעוניינים לרכוש\למכור.

python

**LIMIT ORDER**

A LIMIT ORDER IS AN ORDER TO BUY OR SELL A STOCK AT A SPECIFIC PRICE OR BETTER

STOCK:  COMPANY

PRICE:

CHOOSE HOW MANY STOCKS YOU WANT TO BUY:

100

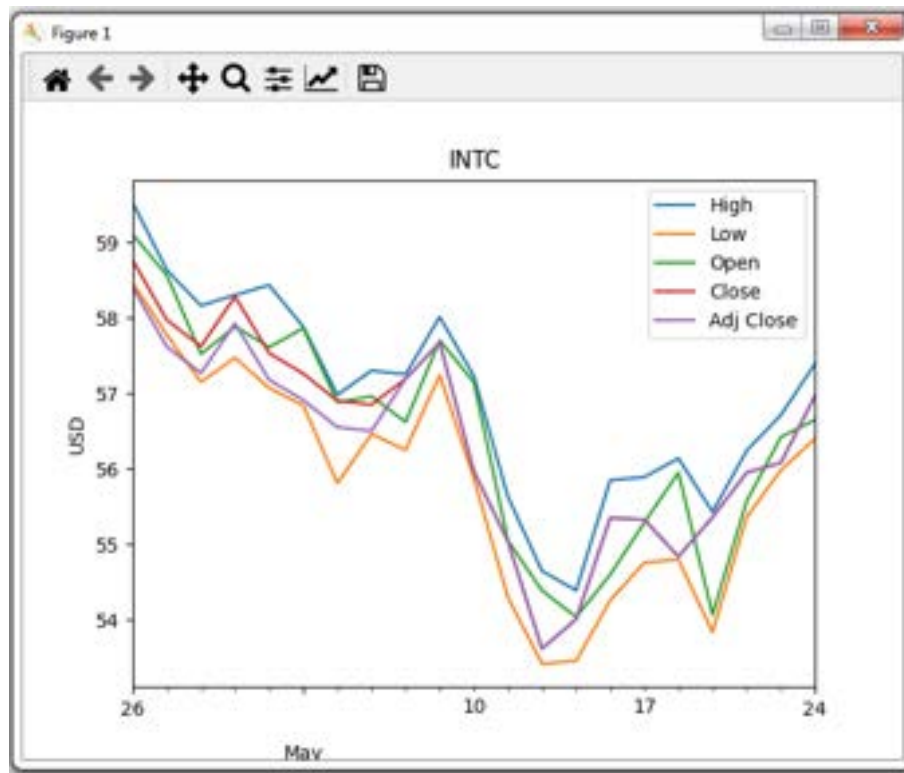
MAX LIMIT :

**CREATE ORDER**



## 4.11 דף גרפים

הגרפים של המניות והמדדים מספקים מידע חיוני עבור המשתמש. ניתן להציג מספר גרפים על גבי אותה מערכת צירים המתארים נתונים שונים על המניה, כל אחד בצבע שונה.



## 4.12 דף מידע נוסף - More Info



## TAKE PROFIT STOCK EXCHANGE – Almog Michael Hemo Python project

בדף זה מוצג מידע רב על כל מניה ומניה, הכולל את שם הכינוי של המנייה (Ticker), באיזה מדד היא נסחרת, מאיזו תעשייה, מספר עובדים בחברה, מחירים ותאור כללי.





## 5 בבליוגרפיה

להלן רשימת האתרים בהם השתמשתי לצורך מחקר, אפיון ומימוש הפרויקט:

- [Investopedia](#)
- [Stack Overflow](#)
- [Tutorials point](#)
- [Geeks for Geeks](#)
- [Matplotlib](#)
- [W3schools SQL](#)
- [Microsoft support](#)
- [PyQt5 Python 3 YouTube Tutorial](#)
- [Matplotlib YouTube Tutorial Series - Graphing in Python](#)
- [Python Programming for Finance YouTube Tutorial](#)
- [GitHub](#)
- [Xlwings](#)