



קרית החינוך השש שנתית
ע"ש חיים בר-לב
מקיף ח
סמל מוסד 441287



ספר פרויקט WhatsBot



שם התלמיד: רועי רחמים

תז של התלמיד:

שם בית הספר ועיר: מקיף עירוני ח ראשון לציון

שם המנחה: יעקב שוצמן

שם הפרויקט: WhatsBot

מועד הגשה: יוני 2020

תוכן עניינים

1	ספר פרויקט
3	מבוא
6	מבנה / ארכיטקטורה של הפרויקט
8	מדריך למשתמש
27	בסיס הנתונים
29	מדריך למפתח
44	סיכום אישי / רפלקציה
46	ביבליוגרפיה

מבוא

ספר הפרויקט שלי מכיל בתוכו את כל ההסברים והפרטים בנוגע לפרויקט הגמר שלי במגמת סייבר.

הפרויקט שלי הוא תוכנה הנקראת WhatsBot. בחרתי בשם זה מפני שהתוכנה שלי היא בעצם Bot שעושה אוטומציה (פועל באופן אוטומטי) ב-WhatsApp Web.

המטרה העיקרית של תוכנה זו היא תחזוקה של חשבון WhatsApp למטרות פרסום בפלטפורמה זו על ידי שליחת הודעות באופן אוטומטי לחלוטין על ידי כמה פעולות עיקריות (ואוטומטיות):

1. הבוט מחפש קישורים המובילים להצטרפות לקבוצות WhatsApp נוספות אשר נמצאים בהיסטוריית ההודעות שהוא נחשף אליהן.
2. הבוט מצטרף לקישורים שמצא בפעולה הראשונה בזה אחר זה ובכך מרחיב את מאגר הקבוצות שלו. על ידי כך יותר ויותר משתמשי WhatsApp נחשפים לחשבון זה וכך הקהל אשר מקבל את הודעות הפרסום מחשבון זה גדל וגדל וההודעות שהחשבון שולח מפורסמות ביותר ויותר קבוצות.
3. הבוט מחלץ את רשימת הצ'אטים שהוא נמצא בהם ונותן למשתמש אופציה לבחור באילו צ'אטים הודעות הפרסום שלו יישלחו. שלב זה יכול להיות מאוד קריטי אם המשתמש רוצה לפרסם הודעות בקבוצות ספציפיות ולא בקבוצות שלא קשורות לנושא הפרסומת, לדוגמא, אם המשתמש רוצה לפרסם בית בקבוצות הקשורות לנושא הנדל"ן, הוא לא בהכרח ירצה לפרסם את ההודעה שלו בקבוצה שקשורה לנושאי פנאי ותרבות ולכן בחלק מהמקרים הוא ירצה לשלוט יותר בפרסום ההודעות על ידי שליחת ההודעות רק בצ'אטים שהוא מצא לנכון לפרסם בהם את ההודעה שלו.
4. הפעולה המרכזית והעיקרית ביותר של הבוט היא שליחת ההודעות באופן אוטומטי לכל הקבוצות. הבוט שולח את הודעת הפרסום שהמשתמש ניסח לכל הצ'אטים שהמשתמש בחר בהם. בנוסף לכך, הבוט יכול להוסיף גם קובץ כגון תמונה או סרטון שיישלחו ביחד עם ההודעה בהודעה אחת לפי רצון המשתמש ולהשתמש באימוג'ים בהודעה. הבוט שולח הודעה בקבוצה אחת ועובר לקבוצה הבאה רק לאחר שההודעה נשלחה במלואה מפני שהוא פועל יותר מידי מהר ואם לא יהיו לו כמה רגעים בהם לא ישלח הודעות לאף קבוצה, ה-WhatsApp עלול לקרוס מפני שקצב שליחת ההודעות הוא מהיר מדי.

מהן הסיבות לבחירת נושא זה לפרוייקט?

החלטתי לפתח את תוכנה זו משום שהייתי חבר בקבוצת WhatsApp שהיו בה הרבה משתתפים, ויום אחד נשלחה שם הודעה ממספר טלפון מסוים אשר פרסמה מוצר כלשהו. בסוף הודעה זו נכתב גם כי הודעה זו נשלחת גם בעוד 500 קבוצות WhatsApp פרט לקבוצה הזו. התעניינתי בנושא ולכן פניתי למשתמש ששלח את הודעה זו. כשאר דיברתי איתו הוא שלח לי את האפליקציה בה הוא משתמש, ולאחר שימוש קצר באפליקציה זו ראיתי שהיא די איטית ולא שימושית מפני שהאפשרויות בה ממש מוגבלות, מה שהופך אותה לכמעט בלתי ניתנת לשימוש.

לאחר בדיקת וסקירת המצב הקיים בשוק בכל הקשור לנושא הפרסום ב-WhatsApp ראיתי שזוהי האפליקציה היחידה שעוסקת בכך ולכן רציתי לפתח תוכנה שתעלה על אפליקציה זו בביצועיה, ביעילותה, במהירותה ובכמות ואיכות האפשרויות שיש בתוכה אשר זמינות למשתמש ובכך לחדש משמעותית את כל נושא הפרסום דרך ה-WhatsApp.

בנוסף לכך, תמיד רציתי להרחיב את הידע שלי בכל הקשור לפיתוח תוכנות העוסקות בהפיכת תהליכים מסוימים לאוטומטיים וזאת הייתה הזדמנות טובה בשבילי ללמוד את נושא זה. העניין הרב שמצאתי בנושא זה גם היווה גורם משמעותי למוטיבציה הגבוהה שהייתה לי לפתח את התוכנה ולעשות את הפרוייקט.

הבעיה איתה התמודדתי במהלך פיתוח התוכנה ופתרונה

הבעיה הגדולה ביותר שחוויתי במהלך פיתוח התוכנה היא העובדה כי chromedriver, דפדפן standalone של google chrome, לא תמך בכתיבת תווים שלא נמצאים ב-ASCII Table, דבר שלא אפשר שליחת אימוג'ים ותווים בשפה העברית בהודעות הפרסום והקשה עליי במשך הרבה זמן.

כעבור חודשיים בהם ניסיתי למצוא פתרון לבעיה בכל יום, מצאתי פתרון יצירתי לבעיה זו – במקום לתת לדפדפן לכתוב את ההודעה בעצמו, הוספתי קטע קוד בתחילת פונקציית השליחה של ההודעות אשר מעתיק את הודעת המשתמש למקלדת, וקטע קוד נוסף באמצע פונקציה זו אשר מדביק את ההודעה בתיבת הצ'אט של WhatsApp. כלומר, תיקנתי את הבעיה על ידי כך שיצרתי מעין לחיצה אנושית על CTRL+C, CTRL+V, ובכך הדפדפן רק הדביק את ההודעות ולא "כתב" אותן בעצמו.

הבעיה אשר עלתה מהמחקר המקדים שערכתי ופירטתי עליו במעלה העמוד

הקודם:

לאחר שראיתי שאין אפליקציה או תוכנה ראויה לשימוש בתחום הפרסום ב-WhatsApp הבנתי שיש באפשרותי וביכולותיי לפתח תוכנה יותר מתקדמת, אך לא ידעתי מאיפה להתחיל מפני שהייתי זר לתחום זה. לאחר שקראתי קצת על הנושא של אוטומציה הבנתי שהדרך הקלה והיעילה ביותר לפתח את תוכנה זו היא באמצעות שימוש ב-API של WhatsApp.

כשהתחלתי לכתוב את הקוד גיליתי להפתעתי שאין מודולים או ספריות העוסקים בתקשורת מול WhatsApp באמצעות API. לאחר מחקר מעמיק נוסף הבנתי ש-WhatsApp לא מפרסמים את ה-API שלהם ולכן יש לחשוב על פתרון אחר.

מבנה / ארכיטקטורה של הפרויקט

הפתרון לבעיה אשר עלתה מהמחקר המקדים שערכתי:

הפתרון שמצאתי לבעיה המפורטת בעמוד הקודם הוא לעשות אוטומציה לדפדפן האינטרנט. ישנה ספריית python העוסקת בתחום זה ונקראת "selenium". באמצעות ספרייה זו ניתן לדמות input המגיע מבן אדם המשתמש במחשב (לחיצה על העכבר, לחיצה על מקשים במקלדת, שימוש בגלגלת העכבר וכדומה) באמצעות התנהלות מול קוד ה-HTML של דף האינטרנט (במקרה שלנו – קוד ה-HTML של אתר WhatsApp Web) על ידי שימוש בפונקציות המוגדרות בספרייה זו.

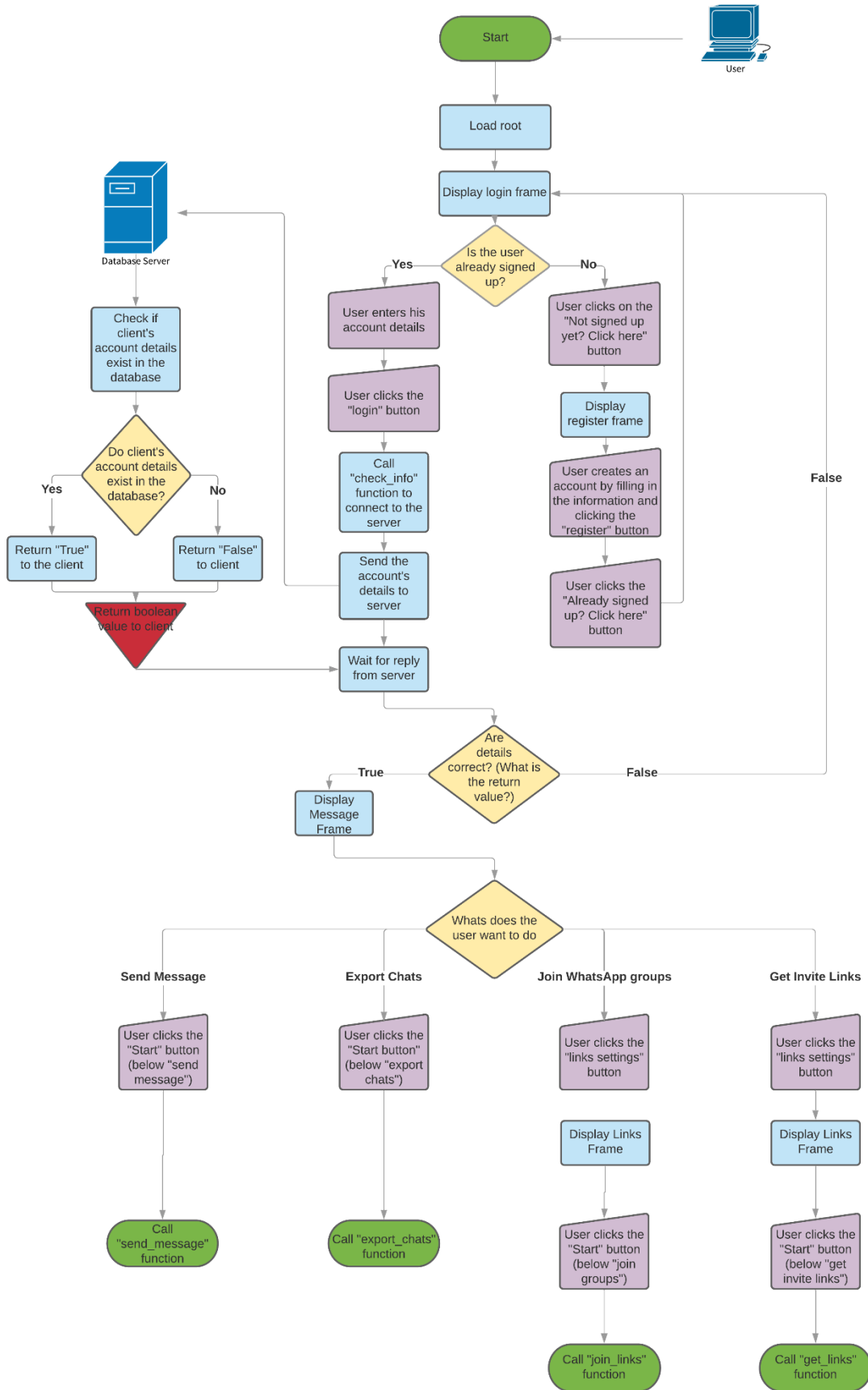
לאחר מכן תכננתי מה אני רוצה שיקרה ועל איזה כפתורים מסוימים אני צריך ללחוץ כדי להגיע למצב בו מה שחשבתי עליו יקרה. כך, בדרך זו, הצלחתי לפתח את התוכנה שלי, שהיא בעצם בוט (Bot) ל-WhatsApp, כלומר עושה מספר פעולות מסוימות באופן אוטומטי ב-WhatsApp Web.

מבנה התוכנה:

מבנה התוכנה מתואר בתרשים שלפנינו (בעמוד הבא).

בתרשים זה ניתן לראות את המעברים בין מסכי הממשק הגרפי בתוכנה, הקשרים בין כל היחידות בתוכנה, והקלטים והפלטרים של התוכנה.

בנוסף לכך, ניתן גם ללמוד על מבנה התוכנה ועל אופן ומטרות השימוש בה, ואף לוודא כי אנו משתמשים בתוכנה בצורה נכונה.



מדריך למשתמש

בפרק זה נלמד על התוכנה וכיצד להפעיל אותה ולהשתמש בה.

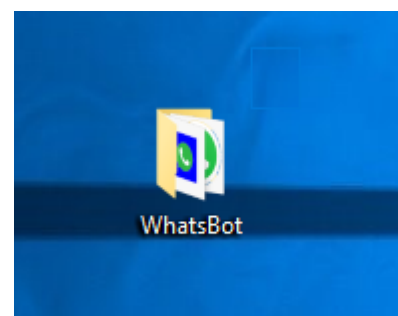
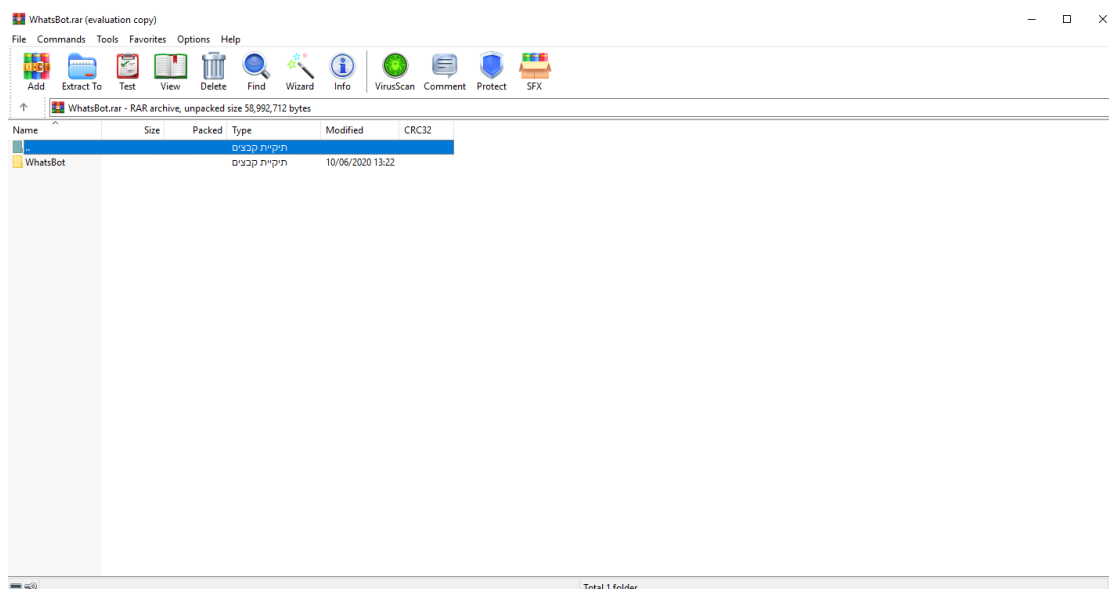
שלב 1 - הורדה:

תחילה, יש להוריד את קובץ ה-winrar (.rar) של התוכנה למחשב. קובץ זה נקרא "WhatsBot.rar"

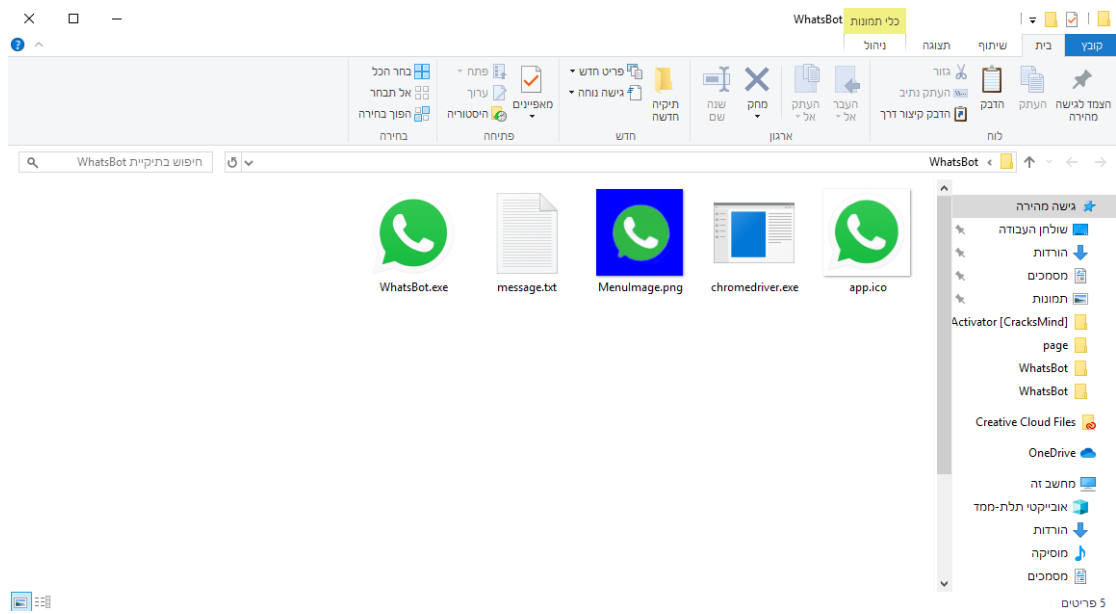
^ WhatsBot.rar

שלב 2 – התקנה:

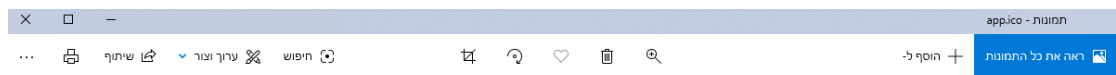
לאחר מכן, פתח את קובץ זה וחלץ ממנו את התיקייה "WhatsBot" למקום כלשהו במחשב שלך אשר אתה רוצה שהתוכנה תהיה שם. בדוגמא זו התיקייה מחולצת לשולחן העבודה:



פתח את התיקיה שחילצת. כעת תוכל לראות חמישה קבצים בתוך תיקייה זו:

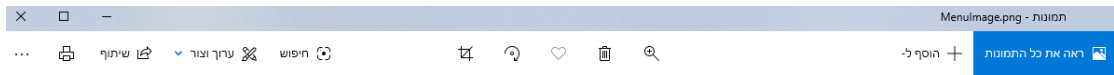


• app.ico – תמונה שנמצאת בתוך התוכנה.

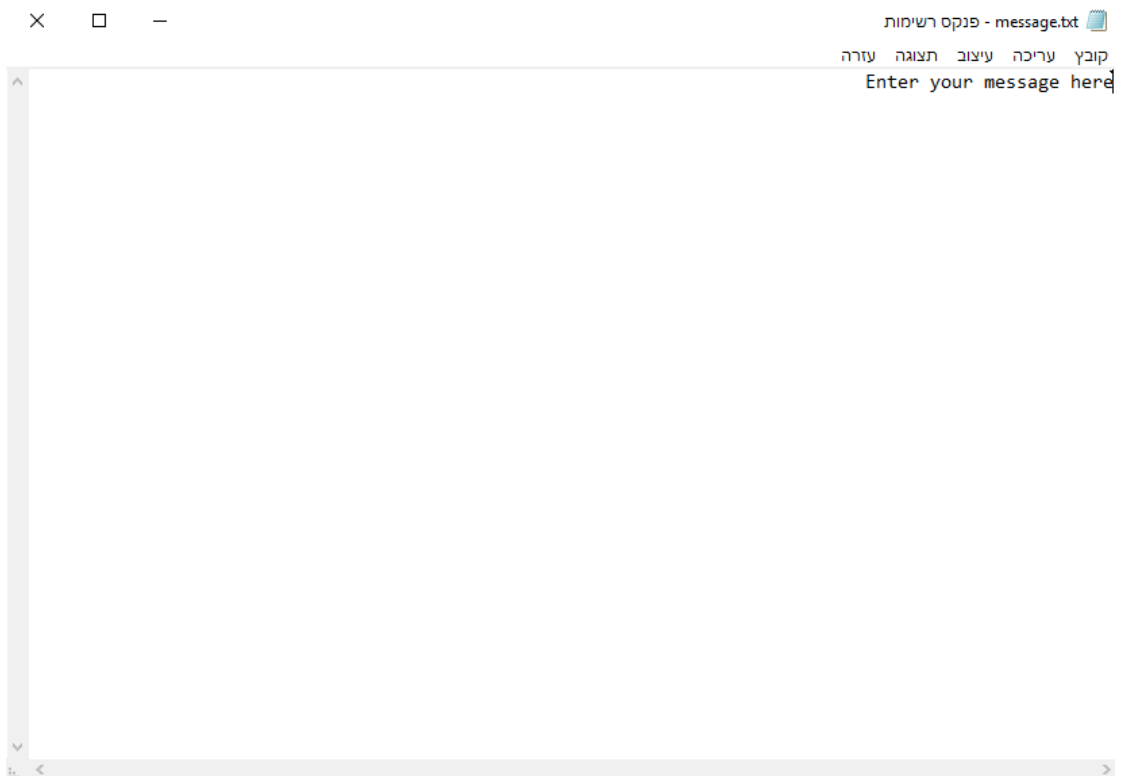


הטלגרם Windows

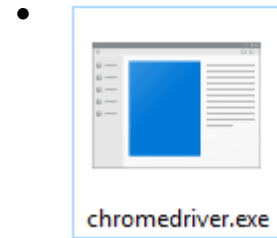
• MenuImage.png – תמונה שנמצאת בתוך התוכנה.



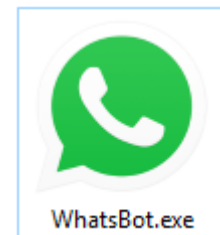
• message.txt – קובץ הטקסט בו יש לשמור את ההודעה שאתה רוצה לשלוח.



- chromedriver.exe – Driver של Google Chrome אשר התוכנה עושה בו שימוש בשביל להיכנס לאינטרנט ולממש את כל הפעולות בצורה אוטומטית.



- WhatsBot.exe – קובץ ההרצה של התוכנה.



***** שים לב:** אם תשנה את תוכנו או שמו של כל אחד מהקבצים הללו, התוכנה לא תפעל כראוי. לכן, איו לבצע שינויים בקבצים אלה הנלווים לתוכנה ולא בקובץ ההרצה של התוכנה עצמה.

שלב 3 – פתיחת / הפעלת התוכנה:

לחץ על קובץ ההרצה (WhatsBot.exe) בשביל לפתוח את התוכנה. לאחר שפתחת את התוכנה יתקבל בפניך המסך הבא:

WhatsBot - Login

WhatsBot

Your exclusive WhatsApp Bot

WhatsBot is an automation bot for WhatsApp, that was founded by Roy A.K.A Boomboom1.
This bot has a couple of unique and useful features:

- Send messages to WhatsApp chats and advertise your business!
- Automatically join and leave WhatsApp groups
- Export all your WhatsApp chats' names into .txt file (This feature is also used for sending the messages)
- Use emojis in messages
- Use images, videos and files in messages

Contact for support & information:
roybwa123@gmail.com

Username

Password

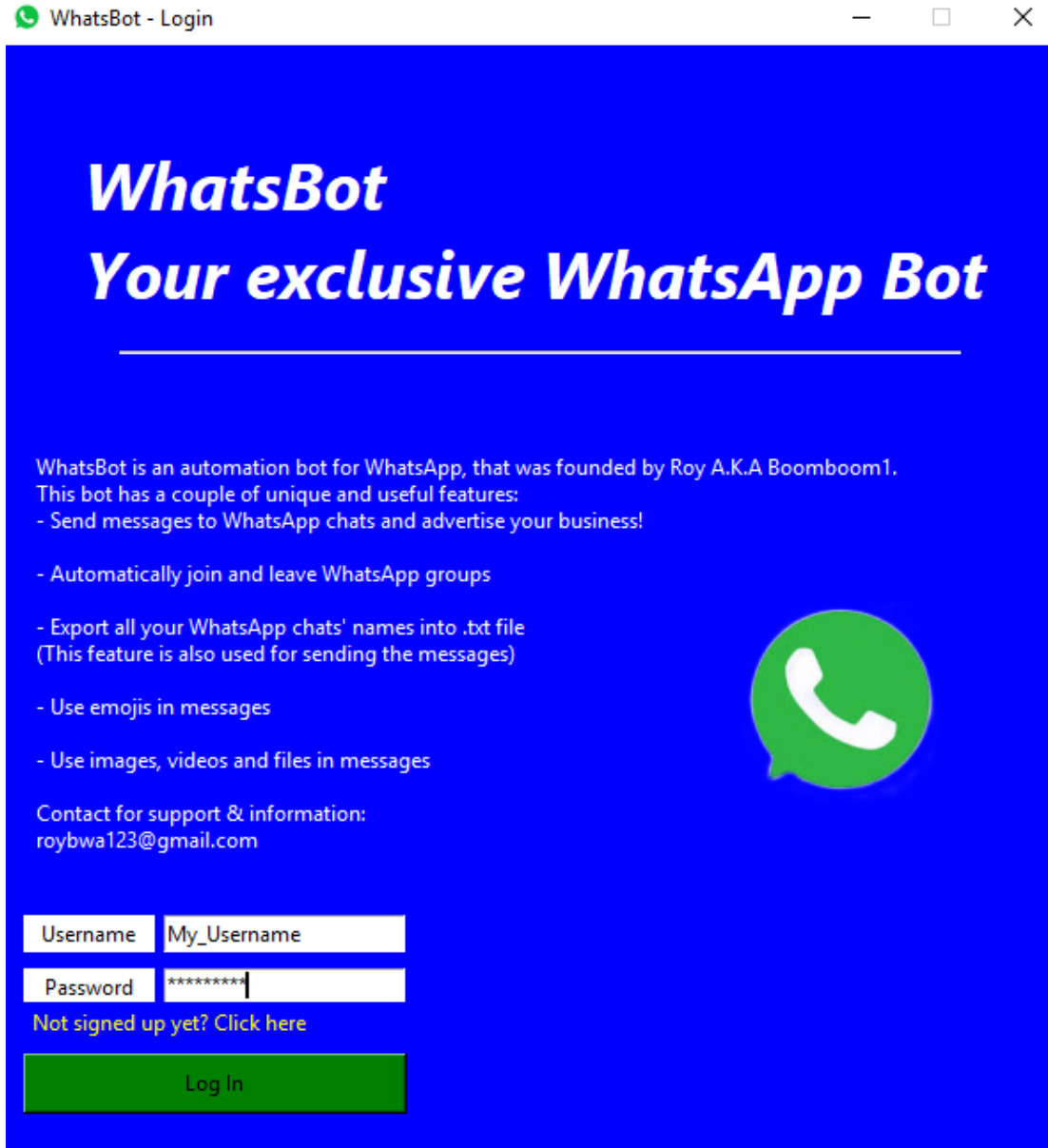
[Not signed up yet? Click here](#)

Log In

מסך זה הוא מסך ההתחברות למשתמש שלך.

שלב 4 – התחברות / יצירת חשבון:

אם יש לך חשבון, אנא הזין את פרטיו לתוך השדות "username" ו-"password" ולחץ על הכפתור "Log In":



WhatsBot - Login

WhatsBot

Your exclusive WhatsApp Bot

WhatsBot is an automation bot for WhatsApp, that was founded by Roy A.K.A Boomboom1.
This bot has a couple of unique and useful features:

- Send messages to WhatsApp chats and advertise your business!
- Automatically join and leave WhatsApp groups
- Export all your WhatsApp chats' names into .txt file (This feature is also used for sending the messages)
- Use emojis in messages
- Use images, videos and files in messages

Contact for support & information:
roybwa123@gmail.com

Username

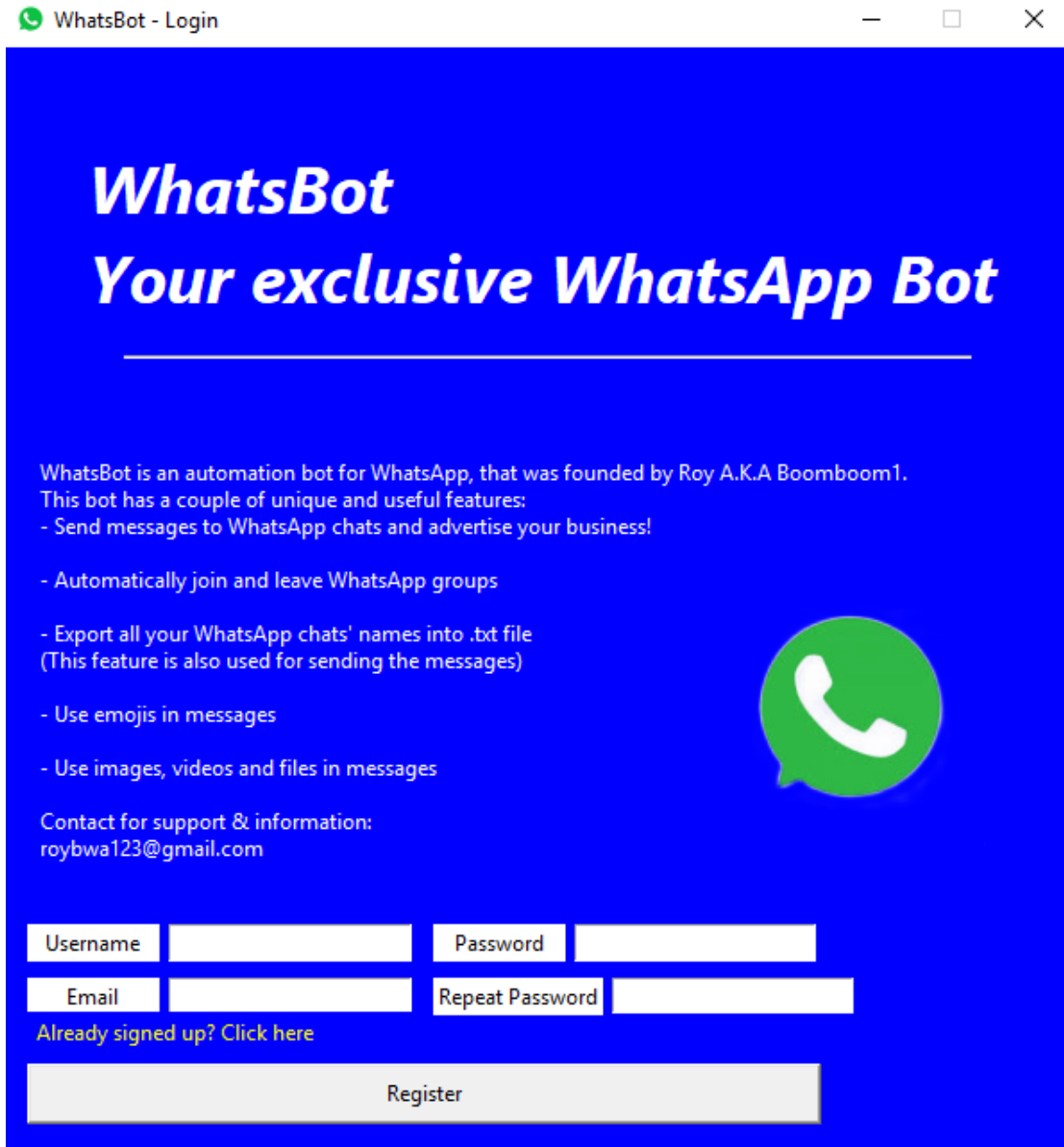
Password

[Not signed up yet? Click here](#)

אם אין לך חשבון, אנא לחץ על הכפתור "Not signed up yet? Click here":

Not signed up yet? Click here

לאחר שלחצת על כפתור זה יתקבל בפניך המסך הבא:



WhatsBot - Login

WhatsBot

Your exclusive WhatsApp Bot

WhatsBot is an automation bot for WhatsApp, that was founded by Roy A.K.A Boomboom1.
This bot has a couple of unique and useful features:

- Send messages to WhatsApp chats and advertise your business!
- Automatically join and leave WhatsApp groups
- Export all your WhatsApp chats' names into .txt file (This feature is also used for sending the messages)
- Use emojis in messages
- Use images, videos and files in messages

Contact for support & information:
roybwa123@gmail.com

Username Password

Email Repeat Password

[Already signed up? Click here](#)

אנא צור משתמש כך:

- הזן את שם המשתמש אשר אתה רוצה להשתמש בו בתהליך ההתחברות לתוכנה בתוך השדה: "Username".
- הזן את כתובת הדוא"ל שלך בתוך השדה: "Email".
- הזן את הסיסמא אשר אתה רוצה להשתמש בה בתהליך ההתחברות לתוכנה בתוך השדה: "Password".
- הזן שוב פעם את סיסמא זו, הפעם בשדה: "Repeat Password".

לאחר מכן, לחץ על הכפתור "Register":

WhatsBot
Your exclusive WhatsApp Bot

WhatsBot is an automation bot for WhatsApp, that was founded by Roy A.K.A Boomboom1.
This bot has a couple of unique and useful features:

- Send messages to WhatsApp chats and advertise your business!
- Automatically join and leave WhatsApp groups
- Export all your WhatsApp chats' names into .txt file (This feature is also used for sending the messages)
- Use emojis in messages
- Use images, videos and files in messages

Contact for support & information:
roybwa123@gmail.com

Username Password

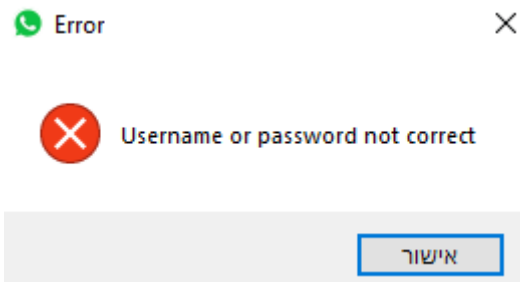
Email Repeat Password

[Already signed up? Click here](#)

לאחר שיצרת חשבון, לחץ על הכפתור "Already signed up? Click here" כדי לחזור למסך ההתחברות. לאחר מכן, התחבר לפי ההוראות אשר הופיעו קודם:

Already signed up? Click here

לאחר שהזנת את פרטי המשתמש שלך במסך ההתחברות ולחצת על הכפתור "Log In",
אנא חכה לאחת מההודעות הבאות ולחץ על "אישור" באיזה מן ההודעות שתופיע:



הודעה זו אומרת כי חלק מהפרטים שהזנת או כל הפרטים שהזנת שגויים, ואין לך הרשאה
להתחבר לתוכנה. לכן, תהליך ההתחברות נכשל.

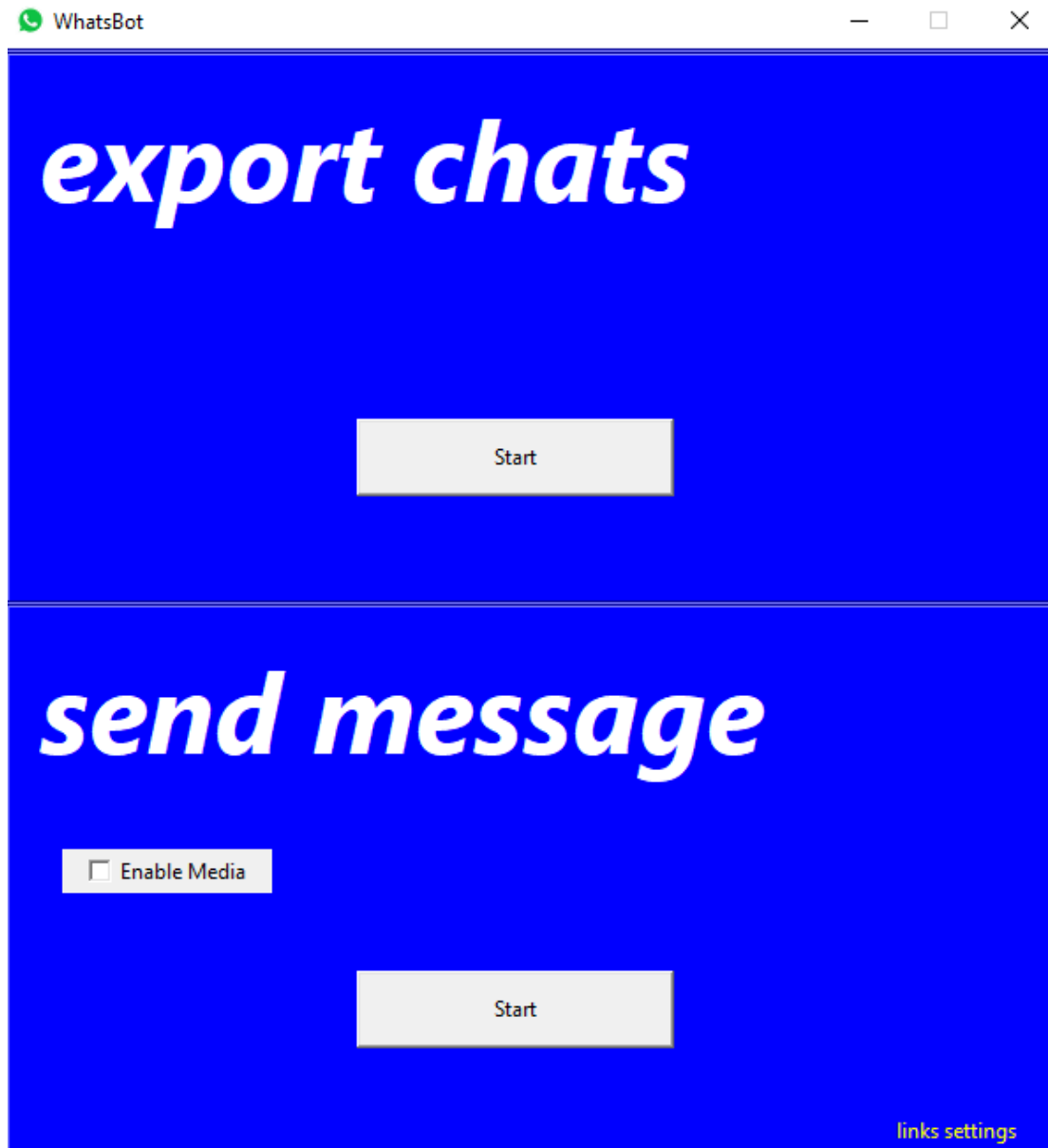


הודעה זו אומרת כי הפרטים שהזנת נכונים ותהליך ההתחברות הסתיים בהצלחה.

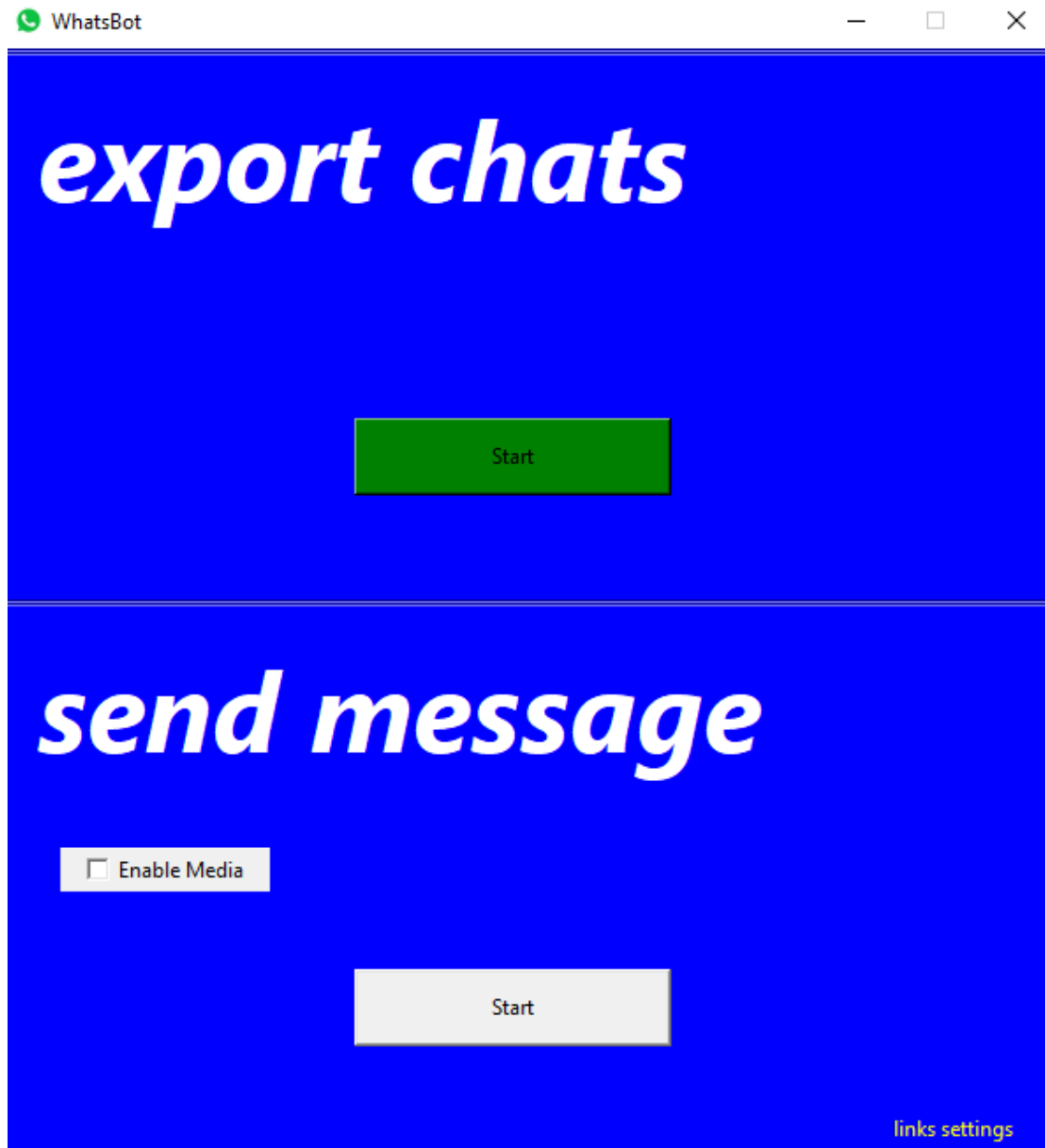
שלב 5 – שימוש בתוכנה:

לאחר שהתחברת בהצלחה לתוכנה ולחצת על "אישור", יופיע בפניך מסך האפשרויות הראשון של התוכנה, בו שתי אפשרויות:

1. "export chats" - ייצוא כל שמות הצ'אטים של חשבון ה-WhatsApp שלך לקובץ טקסט הנקרא "chats.exe".
2. "send message" – שליחת הודעה לכל הצ'אטים אשר יש להם אזכור בקובץ "chats.txt".

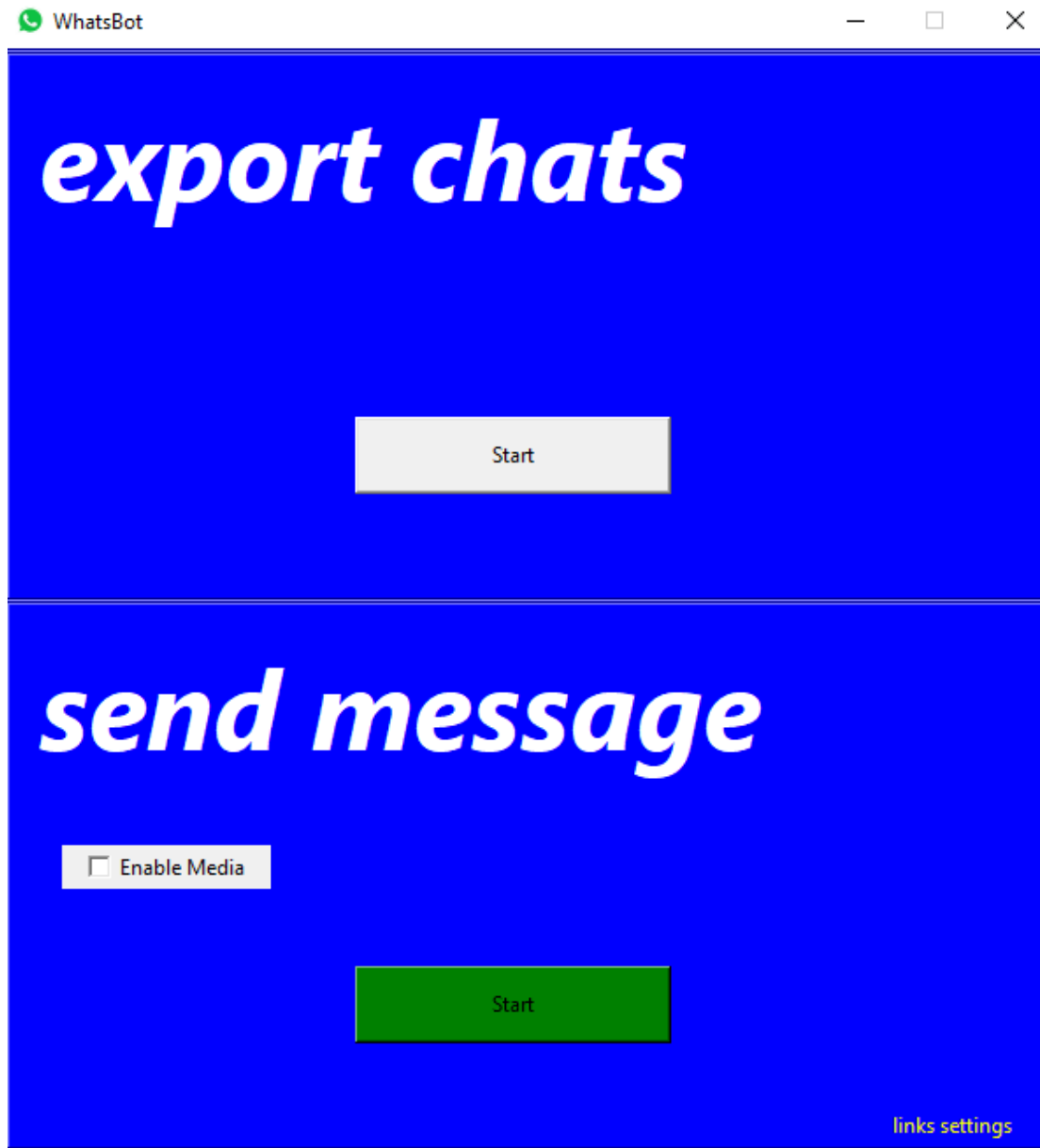


אם הינך מעוניין לשלוח הודעה, הרי שהתוכנה צריכה לדעת באילו צ'אטים אתה רוצה לשלוח את התוכנה. לכן, לחץ על הכפתור "Start" אשר נמצא מתחת לכיתוב "export chats":

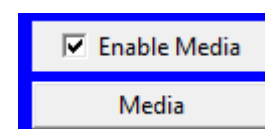


לאחר שהתוכנית הסתיימה, יתווסף לתיקיית התוכנה קובץ טקסט הנקרא "chats.txt". קובץ זה כולל את שמות כל הצ'אטים שחשבון ה-WhatsApp שלך משתמש בהם. אם ברצונך להסיר חלק מהצ'אטים הקיימים ובכך לא לשלוח בהם הודעה, פתח את קובץ הטקסט, מחק את צ'אטים אלה, ושמור את הקובץ בעת יציאתך ממנו.

כעת, לאחר שהצ'אטים בהם אתה רוצה לשלוח את ההודעה נמצאים בקובץ "chats.txt", ניתן לשלוח את ההודעה אשר נמצאת בקובץ "message.txt" בכל אחד מצ'אטים אלה על ידי לחיצה על הכפתור "Start" אשר נמצא מתחת לכיתוב "send message":



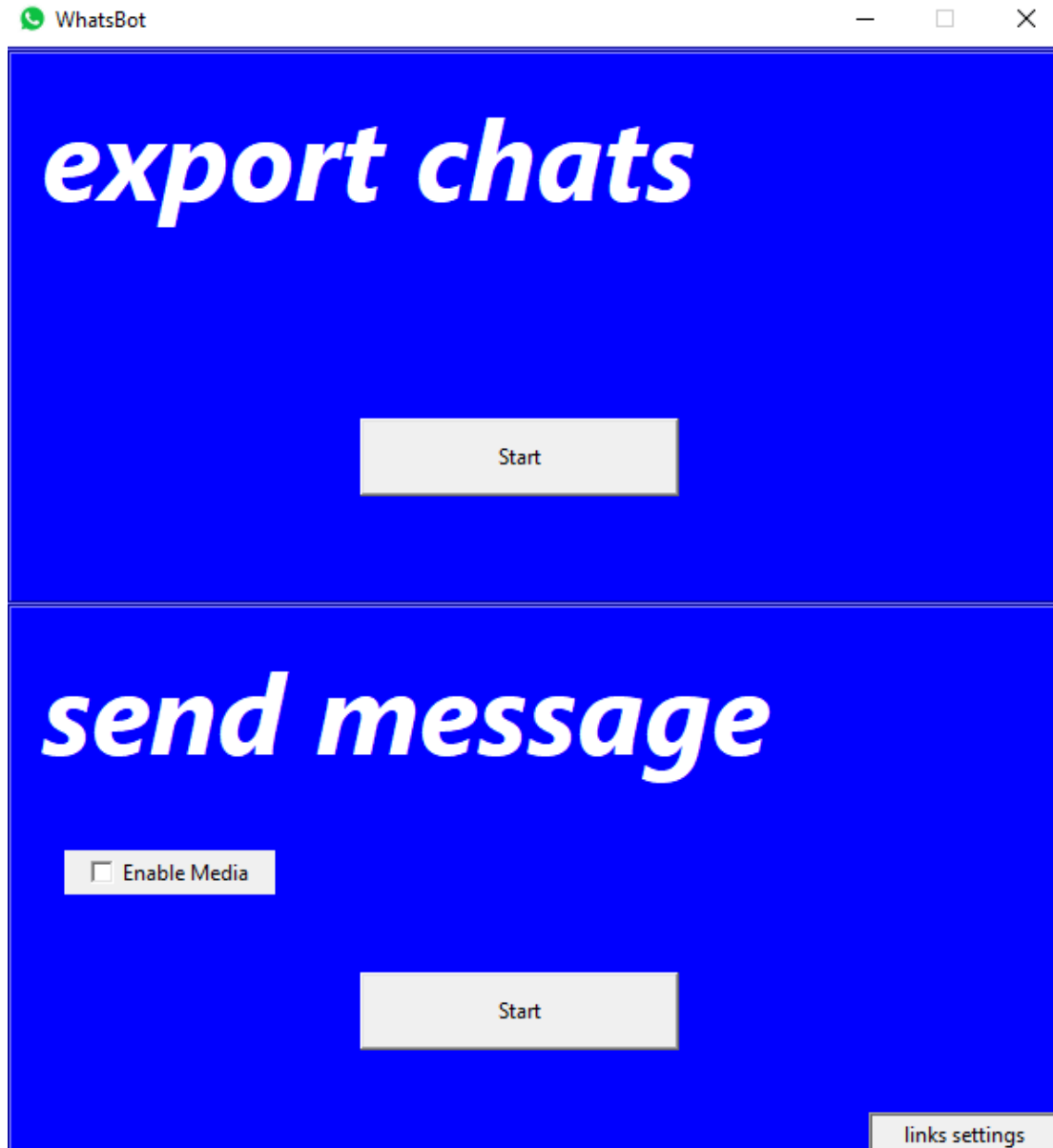
אם ברצונך להוסיף גם קובץ מדיה, סמן את השדה "Enable Media" בווי על ידי לחיצה עליו עם המקש השמאלי של העכבר. לאחר מכן תוכל להבחין בכפתור הנקרא "Media":



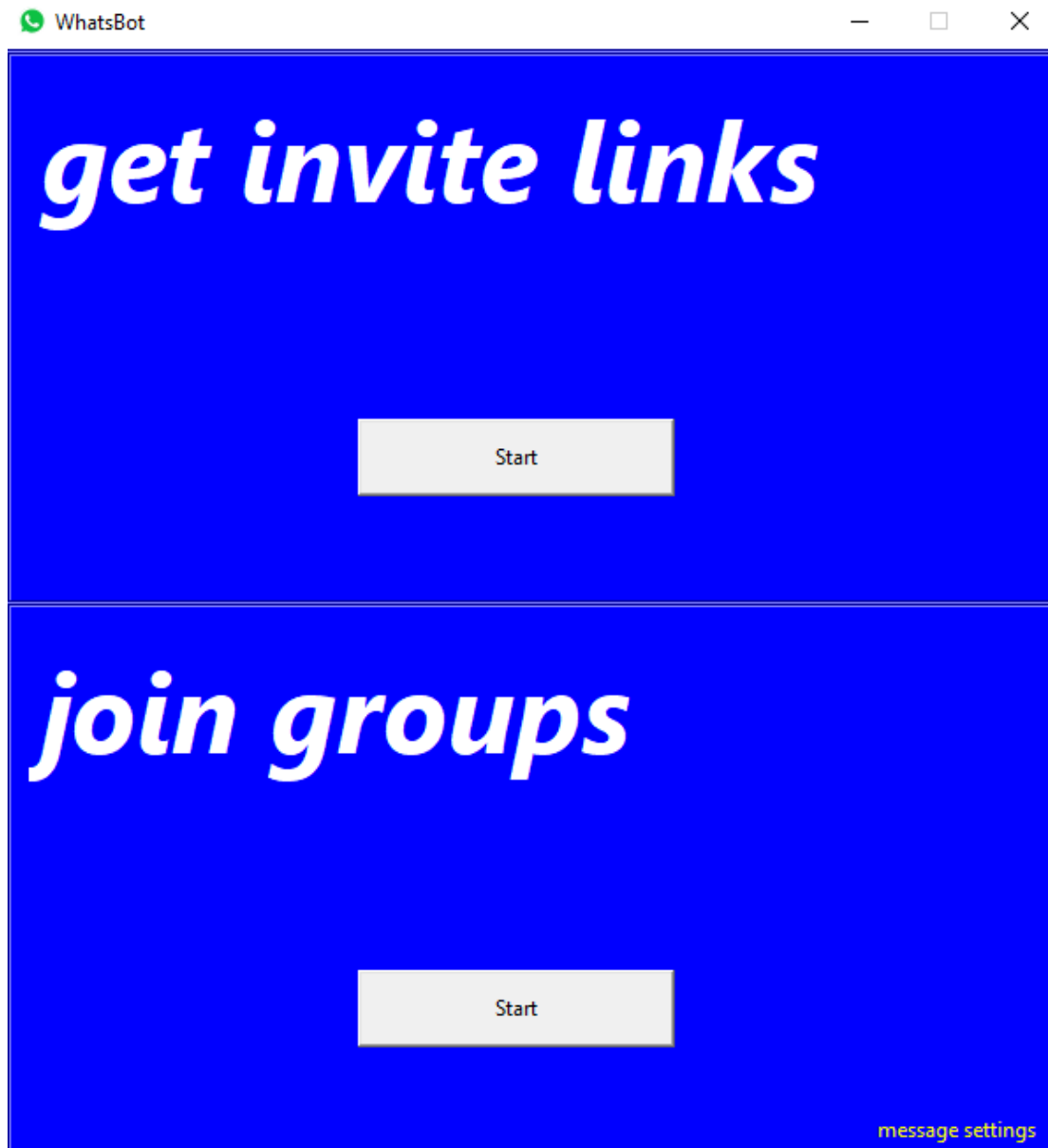
לחץ על כפתור זה ובחר את מיקום הקובץ אותו אתה רוצה לשלוח ביחד עם ההודעה. לבסוף, בדומה להסבר במעלה עמוד זה, לחץ על הכפתור "Start" אשר נמצא מתחת לכיתוב "send message".

בתוכנה ישנה גם אפשרות להצטרף לקבוצות WhatsApp על ידי שימוש בקישורי הצטרפות לקבוצות אלה.

כדי להשתמש באפשרות זו, אנא לחץ על הכפתור "links settings", הממוקם בצד ימין בתחתית מסך האפשרויות הראשון של התוכנה:

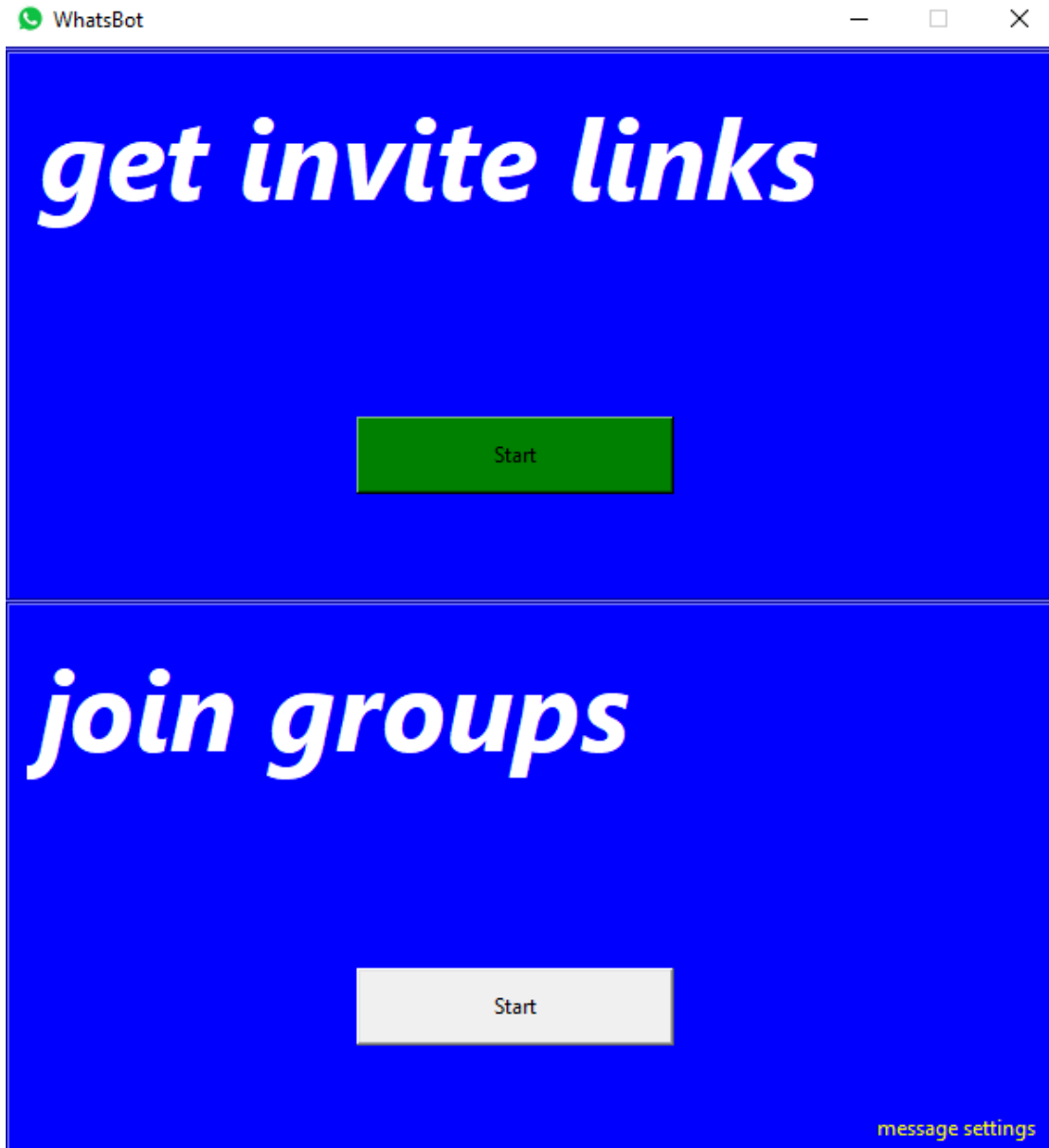


לאחר מכן, יתקבל בפניך מסך האפשרויות השני של התוכנה:



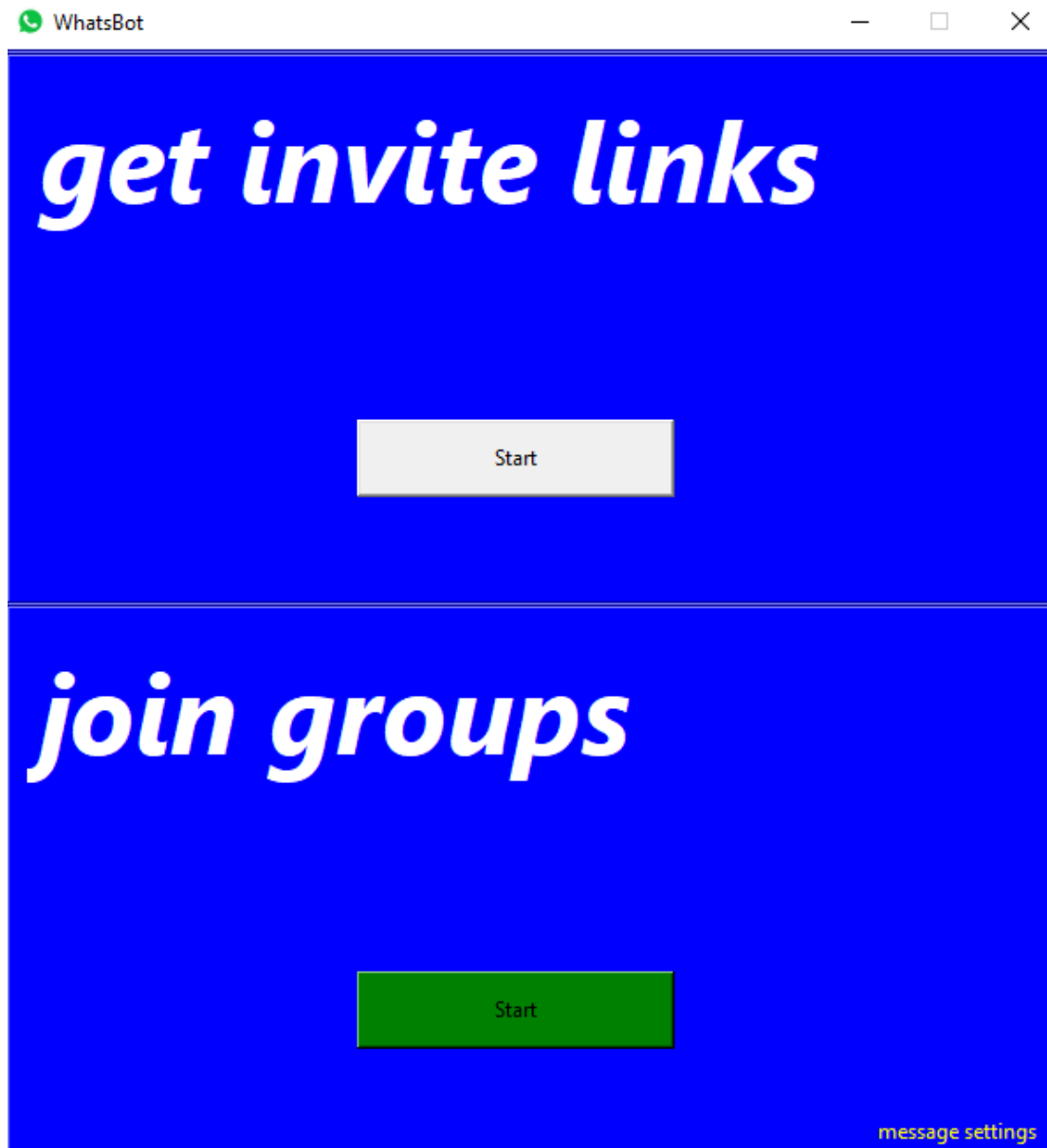
בדומה לשתי הפונקציות שבמסך הקודם, כדי להצטרף לקבוצות WhatsApp באמצעות קישורי הצטרפות יש צורך בחיפוש ואיסוף קישורים אלה. התוכנה מחפשת את הקישורים בהיסטוריית ההודעות שלך (שם ניתן לחפש כל הודעה שאי פעם שלחת או קיבלת) ו"אוספת" אותם בזה אחר זה.

כדי להפעיל את פעולה זו, אנא לחץ על הכפתור "Start" אשר נמצא מתחת לכיתוב "get invite links":

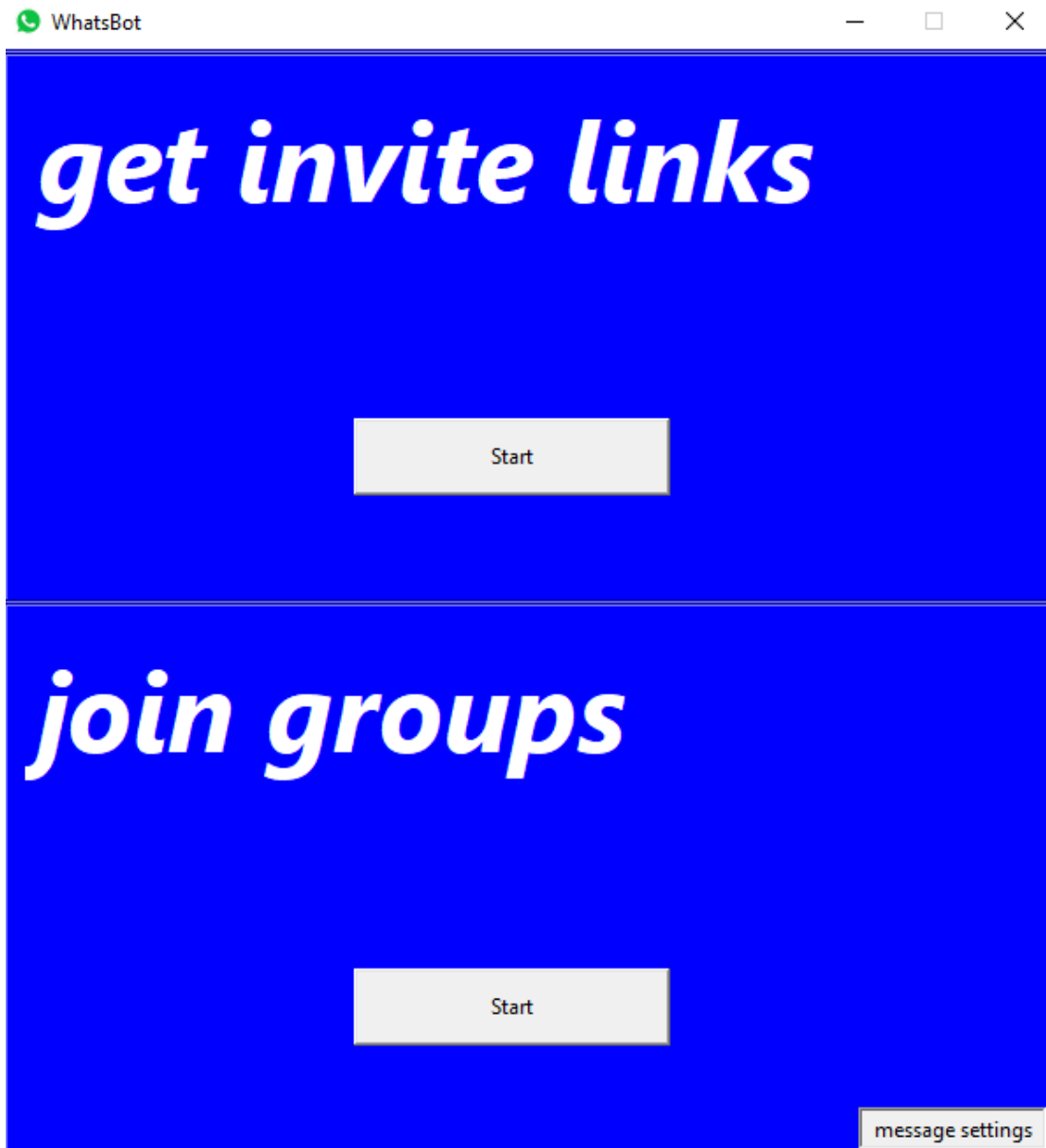


לאחר שהתוכנית הסתיימה, יתווסף לתיקיית התוכנה קובץ הנקרא "get_links.db". קובץ זה מאחסן בתוכו את קישורי ההצטרפות לקבוצות ה-WhatsApp שהתוכנה שלך אספה באמצעות הפעולה הקודמת.

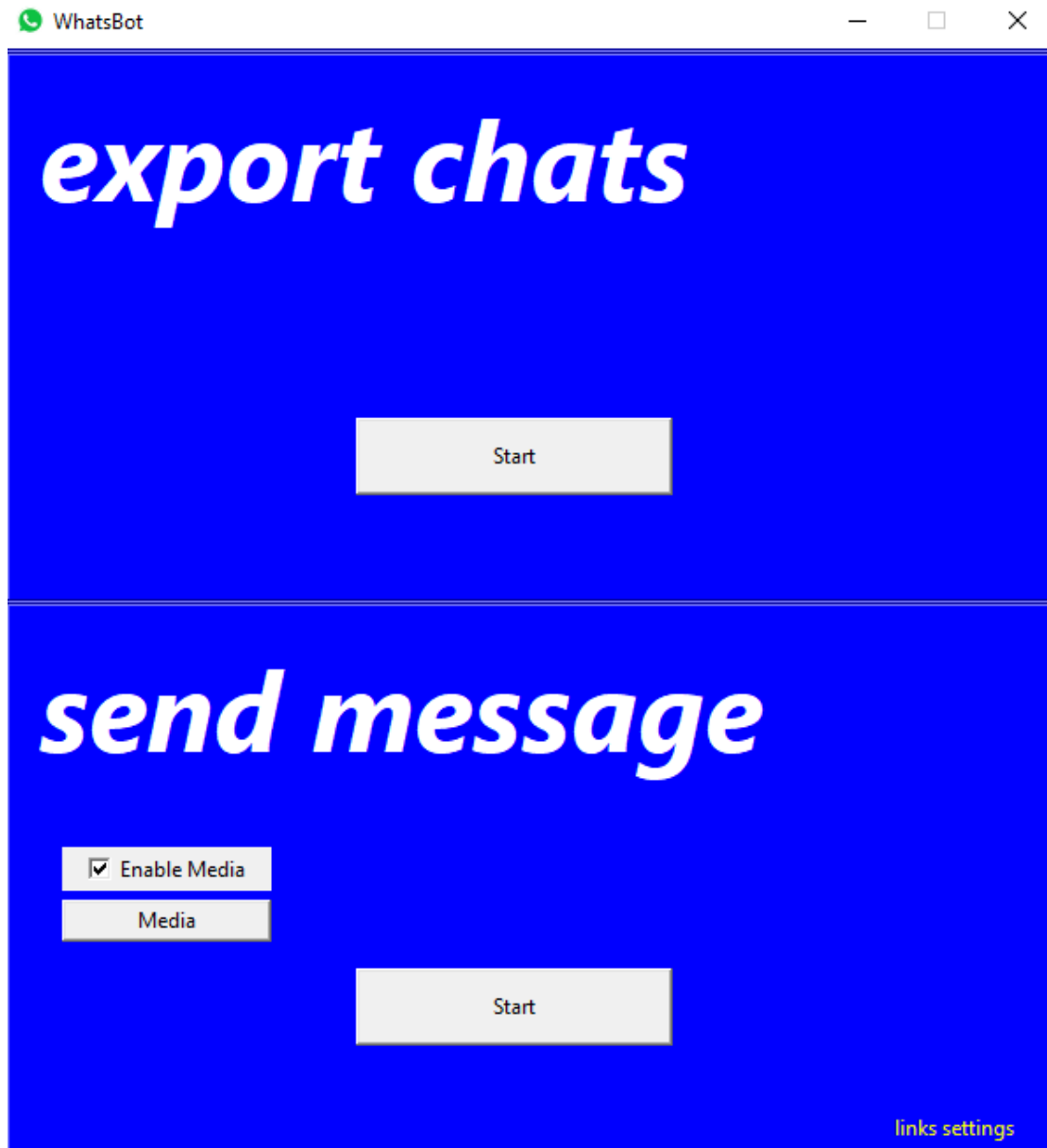
כעת, לאחר שלתוכנה יש מאגר קישורים של קבוצות WhatsApp, ניתן להצטרף לקבוצות אלה על ידי לחיצה על הכפתור "Start" אשר נמצא מתחת לכיתוב "join groups":



אם ברצונך לחזור למסך הקודם, אנא לחץ על הכפתור "message settings", הממוקם בצד ימין בתחתית מסך האפשרויות השני של התוכנה:



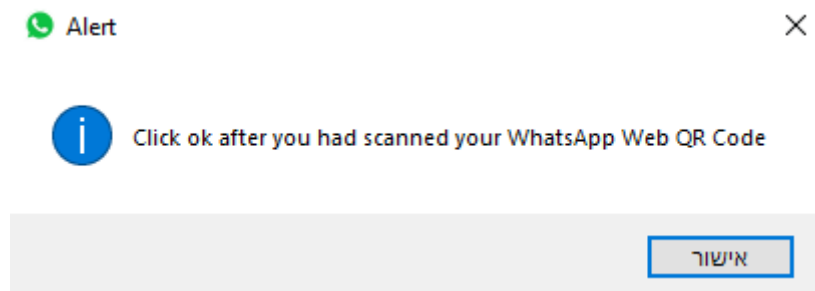
לאחר מכן, יתקבל בפניך בחזרה מסך האפשרויות הראשון של התוכנה:



שים לב:

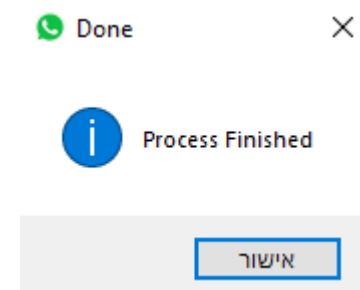
בתחילת כל אחת מפעולות אלה, תתבקש לסרוק את הברקוד אשר יופיע לאחר שדפדפן האינטרנט (שנפתח בעצמו) ינווט בעצמו לאתר של WhatsApp Web למטרת התחברות ל-WhatsApp מדפדפן זה.

לכן, בתחילת כל פעולה תופיע בפניך ההודעה הבאה:



כפי שכתוב בהודעה זו, אנא לחץ "אישור" על הודעה זו וסגור אותה **אחרי** שסרקת את הברקוד והתחברת בהצלחה ל-WhatsApp Web דרך הדפדפן שנפתח עבורך לאחר הפעלת הפעולה.

בנוסף לכך, בסוף כל פעולה תתקבל ההודעה הבאה, אשר אומרת כי הפעולה הסתיימה:



בדרך זו תוכל לדעת כשהפעולה שהפעלת אכן הסתיימה. סגור את הודעה זו על ידי לחיצה על "אישור" והמשך להשתמש בתוכנה לפי רצונך.

בסיס הנתונים

בסיסי הנתונים בתוכנה בשני הצדדים (Server Side & Client Side) ממומשים על ידי ספריית ה-Python: sqlite3, אשר עוסקת בניהול Database במודל טבלאי, כלומר באמצעות שימוש בטבלאות SQL.

בסיס הנתונים בצד השרת (Server Side):

בסיס הנתונים בצד השרת מכיל בתוכו את פרטי המשתמשים בתוכנה. התפקיד של בסיס נתונים זה הוא בדיקת הפרטים שמזינים המשתמשים בתוכנה כשהם מתחברים אליה ושליחת עדכונים והודעות למשתמשים דרך ה-Email במידת הצורך.

שמו של בסיס נתונים זה הוא "server_database.db" ויש בו טבלה אחת הנקראת "users". כל עמודה ועמודה בטבלה זו הינה חובה למילוי:

- ❖ username – עמודה זו מאחסנת בתוכה את שמות המשתמשים של כל חשבונותיהם של המשתמשים בתוכנה לצורך בדיקת שם המשתמש המוזן בתהליך ההתחברות (טיפוס מחרוזת - TEXT). כאשר משתמש מנסה להתחבר לתוכנה, הוא מתבקש להזין את שם המשתמש שלו ביחד עם הסיסמא שלו (עליה נפרט בהמשך). אם שם המשתמש שלו זהה לשם המשתמש אשר נמצא בשדה זה, תיבדק גם הסיסמא שהוא הזין במהלך התחברותו.
 - ❖ password – עמודה זו מאחסנת בתוכה את ה-hash (עליו פורט גם בחלק "מבנה / ארכיטקטורה של הפרויקט") של סיסמאות חשבונותיהם של המשתמשים בתוכנה לצורך בדיקת סיסמת המשתמש המוזנת בתהליך ההתחברות (טיפוס מחרוזת - TEXT). אם הגענו לשלב בדיקה זה, הרי שאנו כבר יודעים ששם המשתמש המוזן הוא נכון. לכן, גם הסיסמא תיבדק על ידי הפעלת פונקציית hash על הסיסמא המוזנת, ופונקציה המשווה את hash זה ל-hash השמור בעמודה זו. אם ישנה התאמה, המשתמש רשאי להתחבר לתוכנה.
- ***חשוב לציין כי שימוש ב-hash מהווה שכבת הגנה נוספת וכמעט בלתי אפשרית לחדירה מאחר שלא ניתן להחזיר את הסיסמא לצורתה ולתוכנה המקורי ובנוסף לכך פונקציה זו מחזירה ערכים שונים בכל קריאה אשר היא מקבלת (אם משתמשים בה כמה פעמים על אותו ערך היא תחזיר ערך שונה בכל פעם, אך עדיין תוכל להשוות אותו ל-hash אחר של הסיסמא ולוודא את תקינותה של הסיסמא). בכך, פונקציה זו מחזקת בצורה משמעותית את רמת האבטחה בצד השרת בכל הנוגע לסיסמאות המשתמשים.

❖ email – עמודה זו מאחסנת בתוכה את כתובות ה-Email של כל חשבונותיהם של משתמשי התוכנה (טיפוס **מחרוזת - TEXT**) למטרת שליחת עדכונים והודעות למשתמשים דרך ה-Email במידת הצורך.

בסיס הנתונים בצד הלקוח (Client Side):

בסיס הנתונים בצד הלקוח מכיל בתוכו קישורי הצטרפות לקבוצות WhatsApp שונות אשר הבוט "אסף" בפעולה ספציפית המשמשת לחיפוש קישורים אלה (ששמה הוא "find_link" והיא נקראת על ידי הפונקציה "get_links").

שמו של בסיס נתונים זה הוא "get_links.db" ויש בו טבלה אחת הנקראת "links", שיש בה עמודה אחת אשר הינה חובה למילוי:

❖ link – עמודה זו מאחסנת בתוכה את כל קישורי ההצטרפות לקבוצות ה-WhatsApp. שימושה העיקרי של עמודה זו הוא שליפת קישורי ההצטרפות בזה אחר זה כאשר הבוט מצטרף לקבוצות WhatsApp באמצעות הפעולה "join_links". בנוסף לכך, כאשר מופעלת הפונקציה "find_link" נעשה שימוש בעמודה זו בכדי לבדוק האם קישור מסוים כבר נמצא בתוך עמודה זו, ובכך למנוע כפילויות של קישורים.

מדריך למפתח

התוכנה "WhatsBot.py" נכתבה בשפת התכנות python 3.6.6 דרך תוכנת PyCharm.

ניתן לחלק את קובץ התוכנה "WhatsBot.py" לשלוש יחידות:

יחידה 1:

ביחידה זו ישנן פונקציות העוסקות בתהליך האוטומציה (Automation) בדפדפן האינטרנט או קשורות לנושא זה.

ביחידה זו נכנסות הפונקציות הבאות:

```
def wait_until_message_is_sent(driver):
    """
    This function is used to wait until there is "√" or "√√" sign at the
    bottom right of the message that has just been
    sent to the WhatsApp chat

    :param driver: selenium.webdriver object (of "chromedriver.exe"
    WebDriver). This is a constant variable
    :return: None
    """

    x2, y2 = 1567, 253

    is_clock_symbol = find_element_by_position(x2, y2, driver)

    while is_clock_symbol.get_attribute('data-icon') == 'status-time':
        pass

def find_element_by_position(x, y, driver):
    """
    This function is used to search an HTML element by it's position
    (location on the screen)

    :param x: width coordination
    :param y: height coordination
    :param driver: selenium.webdriver object (of "chromedriver.exe"
    WebDriver). This is a constant variable
    :return: HTML element at the position of (x, y)
    """

    return driver.execute_script('return document.elementFromPoint({0},
    {1});'.format(x, y))

# -----
# -----
# UPDATED TO 9.6.2020
def join_links(driver):
    """
    This function is used to join WhatsApp groups by opening invite links to
    those groups in the "driver" (desc)

    :param driver: selenium.webdriver object (of "chromedriver.exe"
    WebDriver). This is a constant variable
    :return: None
    """
```

```

get_links_db = sqlite3.connect('get_links.db')
cursor = get_links_db.cursor()
links = cursor.execute('SELECT * FROM links;').fetchall()

if links:
    for link in links:
        driver.get(link[0])

        is_loaded = wait_until_element_is_loaded(10,
'visibility_of_element_located', 'ID', 'action-button', driver)
        if is_loaded:
            join_chat = driver.find_element_by_id('action-button')
            join_chat.click()

            is_loaded = wait_until_element_is_loaded(10,
'visibility_of_element_located', 'CLASS_NAME', '_8ibw', driver)
            if is_loaded:
                join_chat =
driver.find_element_by_class_name('_8ibw').find_element_by_class_name('_36or
')
                join_chat.click()

                is_loaded = wait_until_element_is_loaded(10,
'visibility_of_all_elements_located', 'CLASS_NAME',
'_2LPYs',
driver)
                if is_loaded:
                    join_chat =
driver.find_element_by_class_name('_2LPYs').find_elements_by_xpath('.//*')
                    if len(join_chat) == 2: # if len(join_chat)=2 -->
the link is good and you can join the group
                        join_chat = join_chat[1]
                        join_chat.click()
                        time.sleep(2)

# -----
# UPDATED TO 9.6.2020
def find_link(element, cursor, get_links_db, driver):
    """
    This function finds all the invite links in the current opened chat and
    adds them to the "get_links.db"
    database by adding them to "get_links_db" object (that is described
    below)

    :param element: HTML element of the current opened chat
    :param cursor: cursor of the "get_links_db" object (that is described
    below). This is a constant variable
    :param get_links_db: sqlite3 object. Describes the "get_links.db"
    database (which stores the invite links in it).
    This is a constant variable. This is a constant variable
    :param driver: selenium.webdriver object (of "chromedriver.exe"
    WebDriver). This is a constant variable
    :return: None
    """

    try:
        WebDriverWait(element,
10).until(EC.presence_of_element_located((By.XPATH, '//a[@href]')))
        is_loaded = True
    except:
        is_loaded = False

    if is_loaded:
        hrefs = driver.find_elements_by_xpath('//a[@href]')
        for href in hrefs:

```

```

        link = href.get_attribute('href')
        if re.search('^https://chat.whatsapp.com', link):
            is_already_in_database = cursor.execute('SELECT link FROM
links WHERE link="{}";'.format(
                link)).fetchone()
            if not is_already_in_database:
                cursor.execute('INSERT INTO links(link) VALUES("{}");'
                .format(link))
                get_links_db.commit()

def scroll_and_find_link(x, y, scroll_box, cursor, get_links_db, driver):
    """
    This function clicks the first (from top) visible chat on the screen,
    and then calls the "find_link" function
    (the above function) afterwards. After the "find_link" function ends,
    this function also scrolls down the
    "scroll_box" (that is described below) by one chat's height (by it's
    HTML element's height - 72 pixels)

    :param x: 1550 - width coordination of the first (from top) visible chat
    on the screen. This is a constant variable
    :param y: 312 - height coordination of the first (from top) visible chat
    on the screen. This is a constant variable
    :param scroll_box: HTML element tab that contains all the chats that
    messages that include invite links to WhatsApp
    groups were sent in them. This is a constant variable
    :param cursor: cursor of the "get_links_db" variable (that is described
    below)
    :param get_links_db: sqlite3 object. Describes the "get_links.db"
    database (which stores the invite links in it).
    This is a constant variable.
    :param driver: selenium.webdriver object (of "chromedriver.exe"
    WebDriver). This is a constant variable
    :return: None
    """

    chat = find_element_by_position(x, y, driver)
    chat.click()
    find_link(chat, cursor, get_links_db, driver)
    driver.execute_script('arguments[0].scrollBy(0, 72)', scroll_box)

def get_links(timer, driver):
    """
    This function is used to get invite links to WhatsApp groups.

    :param timer: 60 - time for this function to work (in seconds). This is
    a constant variable
    :param driver: selenium.webdriver object (of "chromedriver.exe"
    WebDriver). This is a constant variable
    :return: None
    """

    get_links_db = sqlite3.connect('get_links.db')
    cursor = get_links_db.cursor()
    cursor.execute('CREATE TABLE IF NOT EXISTS links(link TEXT);')

    search_name =
driver.find_element_by_xpath('//*[@@id="side"]/div[1]/div/label/div/div[2]')
    search_name.send_keys('https://chat.whatsapp.com')
    time.sleep(1)

    is_loaded = wait_until_element_is_loaded(10,
'visibility_of_element_located', 'CLASS_NAME', '-GlrD', driver)
    if is_loaded:
        scroll_box = driver.find_element_by_id('pane-side')

```

```

x, y = 1550, 312
is_bottom_of_page = False
start_time = time.time()
current_time = time.time()
while current_time - start_time <= timer and is_bottom_of_page is
False:
    scroll_and_find_link(x, y, scroll_box, cursor, get_links_db,
driver)
    is_bottom_of_page = driver.execute_script(
        'return (arguments[0].scrollTop >=
(arguments[0].scrollHeight - arguments[0].offsetHeight));', scroll_box)
    current_time = time.time()

    if is_bottom_of_page is True:
        for i in range(9):
            chat = find_element_by_position(x, y + (72 * (i + 1)),
driver)
            find_link(chat, cursor, get_links_db, driver)

get_links_db.close()

# -----
# -----

# UPDATED TO 8.6.2020
def send_message(driver, send_media):
    """
    This function sends the message that is stored "message.txt" text file
    to all the chats that are stored in the
    "chats.txt" text file. If the "send_media" variable (that is described
    below) is True, this function will also add
    media file (that was chosen by the user) to the message

    :param driver: selenium.webdriver object (of "chromedriver.exe"
WebDriver). This is a constant variable
    :param send_media: boolean value. if the user has added media file and
he wants to send it with the text message -
        it's value is True. Else - it's value is False
    :return: None
    """

    global media_path

    message = codecs.open('message.txt', 'r', 'utf-8')
    pyperclip.copy(message.read())
    message.close()

    chats_file = codecs.open('chats.txt', 'r', 'utf-8')
    chats = chats_file.readlines()

    search_name =
driver.find_element_by_xpath('//*[@id="side"]/div[1]/div/label/div/div[2]')

    x, y = 1550, 312
    for chat_name in chats:
        search_name.send_keys(chat_name)
        time.sleep(0.5)

        is_loaded = wait_until_element_is_loaded(10,
'visibility_of_element_located', 'CLASS_NAME', '_210SC', driver)
        if is_loaded:
            chat_element = find_element_by_position(x, y, driver)
            chat_element.click()
            time.sleep(1)

            is_loaded = wait_until_element_is_loaded(10,
'presence_of_element_located', 'CLASS_NAME', '3FRCZ', driver)

```



```

        if is_loaded:
            text_box = driver.find_elements_by_class_name('_3FRCZ')[-1]
            text_box.send_keys(Keys.CONTROL, 'v')
            time.sleep(1)

            if send_media:
                add_file =
driver.find_element_by_xpath('//*[@id="main"]/header/div[3]/div/div[2]/div')
                add_file.click()

                is_loaded = wait_until_element_is_loaded(10,
'presence_of_element_located', 'CSS_SELECTOR',
'input[type="file"]', driver)
                if is_loaded:
                    file =
driver.find_element_by_css_selector('input[type="file"')
                    file.send_keys(media_path)

                    is_loaded = wait_until_element_is_loaded(10,
'visibility_of_element_located',
'CLASS_NAME', '_3y5oW', driver)
                    if is_loaded:
                        send_button =
driver.find_element_by_class_name('_3y5oW')
                        send_button.click()

                        time.sleep(2)
                        wait_until_message_is_sent(driver)
                else:
                    text_box.send_keys(Keys.RETURN)
                    wait_until_message_is_sent(driver)

            search_name.clear()

# -----
# -----
# UPDATED TO 8.6.2020
def find_title_of_chat(element, driver):
    """
    this function finds the name of the first (from top) visible chat on the
    screen

    :param element: HTML element of the first (from top) visible chat on the
    screen
    :param driver: selenium.webdriver object (of "chromedriver.exe"
    WebDriver). This is a constant variable
    :return: title - variable (str) that it's value is the name of the first
    (from top) visible chat on the screen
    """

    title = element.get_attribute('title')
    if len(title) == 0 or title is None:
        element.click()
        while len(title) == 0 or title is None:
            title =
driver.find_element_by_xpath('//*[@id="main"]/header/div[2]/div/div/span').g
et_attribute('title')
            if len(title) != 0 or title is not None:
                break
        else:
            title = driver.find_element_by_xpath(
'//*[@id="main"]/header/div[2]/div/div/span/span').get_attribute('title')
    return title

```

```

def scroll_and_find_title(x, y, scroll_box, driver):
    """
    This function finds the first (from top) visible chat on the screen, and
    then calls the "find_title_of_chat"
    function (the above function) afterwards. After the "find_title_of_chat"
    function ends, this function also scrolls
    down the "scroll_box" (that is described below) by one chat's height (by
    it's HTML element's height - 72 pixels).

    :param x: 1550 - width coordination of the name of the first (from top)
    visible chat on the screen. This is a
        constant variable
    :param y: 240 - height coordination of the name of the first (from top)
    visible chat on the screen. This is a
        constant variable
    :param scroll_box: HTML element tab that contains all the chats that
    your WhatsApp account is part of. This is a
        constant variable
    :param driver: selenium.webdriver object (of "chromedriver.exe"
    WebDriver). This is a constant variable
    :return: title - variable (str) that it's value is the name of the first
    (from top) visible chat on the screen
    """

    chat = find_element_by_position(x, y, driver)
    title = find_title_of_chat(chat, driver)
    driver.execute_script('arguments[0].scrollBy(0, 72)', scroll_box)
    return title

def export_chats(driver):
    """
    This function exports all the the chats that your WhatsApp account is
    part of to a utf-8 text (.txt) file called
    "chats.txt"

    :param driver: selenium.webdriver object (of "chromedriver.exe"
    WebDriver). This is a constant variable
    :return: None
    """

    scroll_box = driver.find_element_by_id('pane-side')

    x, y = 1550, 240
    chats = codecs.open('chats.txt', 'w', 'utf-8')

    is_bottom_of_page = False
    while is_bottom_of_page is False:
        title = scroll_and_find_title(x, y, scroll_box, driver)
        chats.write(title + '\n')
        is_bottom_of_page = driver.execute_script(
            'return (arguments[0].scrollTop >= (arguments[0].scrollHeight -
            arguments[0].offsetHeight));', scroll_box)

    for i in range(9):
        chat = find_element_by_position(x, y + (72 * (i + 1)), driver)
        title = find_title_of_chat(chat, driver)
        chats.write(title + '\n')

    chats.close()

# -----
def open_driver(function_name):
    """

```

```

    This function opens the "chromedriver.exe" WebDriber and stores it's
    object in "driver". After that, it calls the
    function is stored in the "function_name" variable (that is described
    below)

    :param function_name: variable (str) that stores the name of the
    function
        ("export_chats" / "send_message" / "get_links" / "join_links")
    that is connected to the "Start" button
        (in the GUI) the user clicked on
    :return: None
    """

global is_checked, media_path, valid_images_formats

disable_secondary_buttons()

send_message_if_error_in_media = True

if function_name == 'send_message':
    is_media_enabled = is_checked.get()
    is_available_path = False
    send_media = False

    if is_media_enabled:
        file_format = media_path.split('/')[-1].split('.')[1]
        for image_format in valid_images_formats:
            if file_format == image_format:
                is_available_path = True
                send_media = True
            if is_available_path is False:
                send_message_if_error_in_media =
messagebox.askyesno('Warning', 'The file that you try to send is not an
image. Do you still want to send the message without the image?',
icon='warning')

    if send_message_if_error_in_media:
        driver = webdriver.Chrome(executable_path='chromedriver')
        driver.maximize_window()
        driver.page_source.encode('utf-8')
        driver.get('https://web.whatsapp.com/')

        messagebox.showinfo('Alert', 'Click ok after you had scanned your
WhatsApp Web QR Code')

        is_loaded = wait_until_element_is_loaded(10,
'visibility_of_element_located', 'CLASS_NAME', '_1DzHI', driver)

        if is_loaded:
            time.sleep(1)

            if function_name == 'export_chats':
                export_chats(driver)
            elif function_name == 'send_message':
                send_message(driver, send_media)
            elif function_name == 'get_links':
                get_links(60, driver)
            elif function_name == 'join_links':
                join_links(driver)
            time.sleep(1)

        driver.quit()

messagebox.showinfo('Done', 'Process Finished')
enable_secondary_buttons()
root.focus_force()

```

```

# -----
def check_if_can_start_driver_thread():
    """
    This function checks if there is already a running thread of the
    "open_driver" function before starting another one.
    If there is no running thread of the "open_driver" function - it returns
    True. Else - it returns False. This
    function is used in order to avoid the user from spam clicking the
    "Start" buttons (and maybe crash the software by
    doing that).

    :return: boolean value (True / False)
    """

    global driver_threads

    if len(driver_threads) == 0:
        return True

    elif not driver_threads[-1].is_alive():
        driver_threads.remove(driver_threads[0])
        return True

    return False

def create_driver_thread(function_name):
    """
    This function creates and starts a thread of the "open_driver" function

    :param function_name: variable (str) that stores the name of the
    function
        ("export_chats" / "send_message" / "get_links" / "join_links")
    that is connected to the "Start" button
        (in the GUI) the user clicked on

    :return: None
    """

    global driver_threads

    new_thread = threading.Thread(target=open_driver, args=(function_name,
    ))

    can_start_thread = check_if_can_start_driver_thread()

    if can_start_thread:
        driver_threads.append(new_thread)
        new_thread.start()
    else:
        messagebox.showerror('Error', 'The bot is currently in use. Please
        wait until it ends')

```

יחידה 2:

ביחידה זו ישנן פונקציות העוסקות בתקשורת מול השרת או קשורות לנושא זה.

ביחידה זו נכנסות הפונקציות הבאות:

```
def check_if_data_is_ascii(data):
    """
    This function checks if the "data" variable (that is described below) is
    in the ASCII table. if it is - this
    function returns True. Else, it returns False

    :param data: variable (str) that contains some or all of the data that
    the client wants to send to the server
    :return: boolean value (True / False)
    """

    try:
        data.encode('ascii')
        return True
    except UnicodeEncodeError:
        return False

def send_data(data, encrypter, client_socket):
    """
    This function sends data to the server

    :param data: variable (str) that contains some or all of the data that
    the client wants to send to the server
    :param encrypter: Crypto.Cipher.PKCS1_OAEP object. This object encrypts
    the data in RSA encryption with the
        public RSA key that the server had generated for the client
    :param client_socket: socket.socket object. This is an endpoint of a
    conversation
    :return: if the "check_if_data_is_ascii" function returns False when its
    called - return False. Else - return None
    """

    data_type = type(data)

    if data_type is list or data_type is tuple:
        encrypted_data = []
        for string in data:
            if not check_if_data_is_ascii(string):
                return False
            encrypted_data.append(encrypter.encrypt(codecs.encode(string,
'rot-13').encode('ascii')))
        client_socket.send(pickle.dumps(encrypted_data))

    else:
        if not check_if_data_is_ascii(data):
            return False
        encrypted_data = encrypter.encrypt(codecs.encode(data, 'rot-
13').encode('ascii'))
        client_socket.send(pickle.dumps(encrypted_data))

    time.sleep(0.1)

def receive_data(data, decrypter):
    """
    This function receives data from the server

    :param data: variable (str) that contains the last 1024 bytes (or less)
    of the data that the client has received
    from the server
    """
```

```

    :param decrypter: Crypto.Cipher.PKCS1_OAEP object. This object decrypts
    the RSA encrypted data (that was encrypted
    by the server with the public RSA key that the client had
    generated for the server) with the private RSA key
    that the client had generated
    :return: decrypted_data - variable (str / list / tuple) that contains
    the decryption of "data". It's type is
    determined by the type of "data" ("data" and "decrypted_data"
    are ALWAYS the same type)
    """

    data_type = type(data)
    if data_type is list or data_type is tuple:
        decrypted_data = []
        for string in data:
            decrypted_data.append(codecs.decode(decrypter.decrypt(string).decode('ascii'), 'rot-13'))
    else:
        decrypted_data = codecs.decode(decrypter.decrypt(data).decode('ascii'), 'rot-13')
    return decrypted_data

def check_info(login_or_register):
    """
    This function connects the client to the server when the client logs in
    or registers, in order to validate his
    details (login) or add his details to the server's database (register)

    :param login_or_register: variable (str). It's value is 'login' when the
    client clicks on the "Log In" button
    and 'register' when the client clicks on the "Register" button
    :return: if the value that is stored in the "data_is_ascii" variable
    after calling the "send_data" function is False
    - return 'data is not ascii' (str). if "data" is 'valid
    account' - return True (boolean). if "data" is
    'invalid account' - return False (boolean)
    """

    global root

    client_key_pair = RSA.generate(1024) # Private key generated by the
client
    client_public_key = client_key_pair.publickey() # Public key generated
by the client

    decrypter = PKCS1_OAEP.new(client_key_pair)

    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

    try:
        client_socket.connect(('localhost', 6666))
        connected = True
    except:
        #print("Server is not up. Please come back later. (1)")
        connected = False

    server_has_client_public_key = False
    while connected:
        try:
            data = pickle.loads(client_socket.recv(1024))
            if server_has_client_public_key:
                data = receive_data(data, decrypter)
            #print("Data received from server: {}".format(data))
        except:
            #print('Server is not up. Please come back later. (2)')

```

```

        break

    if data[0] == 'welcome':
        data.remove('welcome')
        server_public_key = RSA.import_key(data[0]) # Public key
generated by the server
        encrypter = PKCS1_OAEP.new(server_public_key)

        client_public_key_str =
client_public_key.export_key(format='PEM')
        chunks = len(client_public_key_str)
        chunk_size = 86

        send_data('client_public_key', encrypter, client_socket)
        for i in range(0, chunks, chunk_size):
            send_data(client_public_key_str[i: i +
chunk_size].decode('ascii'), encrypter, client_socket)
        send_data('end_of_client_public_key', encrypter, client_socket)

        server_has_client_public_key = True

    elif data == 'login_or_register':
        if login_or_register == 'login':
            username = LoginFrame_UsernameEntry.get()
            password = LoginFrame_PasswordEntry.get()

            data_is_ascii = send_data(['login', '{}'.format(username),
'{}'.format(password)], encrypter, client_socket) # We need the return
value of 'send_data' in this case because the client can control the data
sent to this server in this command
            if data_is_ascii is False:
                return 'data is not ascii'
            elif login_or_register == 'register':
                return True

        elif data == 'valid account':
            return True
        elif data == 'invalid account':
            return False

    client_socket.close()

def login():
    """
    This function is in charge of the login process of the user to the
software

    :return: None
    """

    is_valid = check_info('login')
    if is_valid is True:
        switch_frames(1)
        root.title('WhatsBot')
        root.unbind('<Return>')
        messagebox.showinfo('Success', 'Welcome back!')
    elif is_valid is False:
        messagebox.showerror('Error', 'Username or password not correct')
    elif is_valid is None:
        messagebox.showerror('Error', 'Server is not up. Please come back
later.')
    elif is_valid == 'data is not ascii':
        messagebox.showerror('Error', 'Only characters in English, numbers
and symbols are supported')

def register():

```

```

"""
    This function is in charge of the registration process of the user to
    the software

    :return: None
    """

    is_valid = check_info('register')
    if is_valid is True:
        messagebox.showinfo('Success', 'You will be able to log in once your
account is verified by the system')
    elif is_valid is False:
        pass
    elif is_valid is None:
        pass
    elif is_valid == 'data is not ascii':
        pass

def check_if_can_start_login_or_register_thread():
    """
    This function checks if there are already running threads of the "login"
    / "register" functions before starting
    another one. If there is no running thread of the "login" function AND
    there is no running thread of the "register"
    function - it returns True. Else - it returns False. This function is
    used in order to avoid the user from spam
    clicking the "Log In" and the "Register" buttons (and maybe crash the
    software by doing that).

    :return: boolean value (True / False)
    """

    global login_and_register_threads

    if len(login_and_register_threads) == 0:
        return True

    elif not login_and_register_threads[-1].is_alive():
        login_and_register_threads.remove(login_and_register_threads[0])
        return True

    return False

def create_login_thread(event):
    """
    This function creates and starts a thread of the "login" function

    :param event: the user left clicks on the "Log In" button / the user
taps on the 'Enter' key in the keyword when the
    "LoginFrame" object (tkinter.LabelFrame object) is shown in the
    GUI

    :return: None
    """

    global login_and_register_threads

    new_thread = threading.Thread(target=login)

    can_start_thread = check_if_can_start_login_or_register_thread()

    if can_start_thread:
        login_and_register_threads.append(new_thread)
        new_thread.start()
    else:
        messagebox.showerror('Error', 'The bot is currently in use. Please
wait until it ends')

```



```

def create_register_thread(event):
    """
    This function creates and starts a thread of the "register" function

    :param event: the user left clicks on the "Register" button / the user
    taps on the 'Enter' key in the keyword when
    the "RegisterFrame" object (tkinter.LabelFrame object) is
    shown in the GUI
    :return: None
    """

    global login_and_register_threads

    new_thread = threading.Thread(target=register)

    can_start_thread = check_if_can_start_login_or_register_thread()

    if can_start_thread:
        login_and_register_threads.append(new_thread)
        new_thread.start()
    else:
        messagebox.showerror('Error', 'The bot is currently in use. Please
        wait until it ends')

```

יחידה 3:

ביחידה זו ישנן פונקציות העוסקות בתהליך בממשק הגרפי (GUI) של התוכנה או קשורות לנושא זה.

ביחידה זו נכנסות הפונקציות הבאות:

```

def switch_frames(number):
    """
    This function switches between the GUI's frames

    :param number: variable (int) of the index of the frame that needs be
    shown in the GUI in the "frames_list" list
    (global variable)
    :return: None
    """

    global frames_list
    frames_list[number].tkraise()

def on_enter(event, button):
    """
    This function changes the color of the "button" object (that is
    described below) to green when the user hovers over
    it with his mouse

    :param event: the user hovers over "button"
    :param button: tkinter object of the button that the user hovers over
    :return: None
    """

    button['background'] = 'green'

```

```

def on_leave(event, button):
    """
    This function changes the color of the "button" object (that is
    described below) to white when the user no longer
    hovers over it with his mouse

    :param event: the user no longer hovers over "button"
    :param button: tkinter object of the button that the user no longer
    hovers over
    :return: None
    """

    button['background'] = 'SystemButtonFace'

def disable_secondary_buttons():
    """
    This function disables all the secondary buttons (that are in the global
    list typed variable -
    "secondary_buttons_list") once a thread of the "open_driver" function
    starts

    :return: None
    """

    global secondary_buttons_list

    for button in secondary_buttons_list:
        button.config(state='disabled')

def enable_secondary_buttons():
    """
    This function enables all the secondary buttons (that are in the global
    list typed variable -
    "secondary_buttons_list") once a thread of the "open_driver" function
    ends

    :return: None
    """

    global secondary_buttons_list

    for button in secondary_buttons_list:
        button.config(state='normal')

def choose_image():
    """
    This function opens a filedialog window, lets the user choose media
    file, and finally stores it's path in the
    variable "media_path" (global variable)

    :return: None
    """

    global media_path

    media_path =
filedialog.askopenfilename(initialdir=os.path.dirname(os.path.realpath(__file__)))

def show_media_button():
    """
    This function creates tkinter.button object once the tkinter.checkbutton
    object "SendFrame_Checkbutton" (global)
    is checked on

```

```
:return: None
"""

global is_checked, SendFrame_MediaButton, media_path

if is_checked.get():
    SendFrame_MediaButton.place(relx=0.05, rely=0.54, relheight=0.079,
relwidth=0.202)
    SendFrame_MediaButton.configure(text='''Media''',
command=choose_image)

elif not is_checked.get():
    SendFrame_MediaButton.place_forget()
    media_path = None
```

סיכום אישי / רפלקציה

העבודה על הפרויקט הייתה אמנם מאוד קשה ומאתגרת, אך היא הייתה גם ממש מעניינת ומלהיבה עבורי, משום שאני באמת התעניינתי בתחום זה והיה לי כיף להמשיך את הפרויקט בכל פעם ולחפש עוד ועוד דרכים יעילות ואפקטיביות יותר לביצוע פעולות ותהליכים הקשורים לתוכנה, וכן גם להרחיב את הידע שלי כמה שיותר בנושאי מדעי המחשב והסייבר בכלל ובנושא זה בפרט.

קיבלתי עוד ידע ועוד ניסיון במהלך העבודה על פרויקט זה אשר אני בטוח שישמשו אותי בעתיד הקרוב ובעתיד הרחוק מפני שאני רוצה לעסוק בנושא הסייבר בעתיד ובזכות פרויקט זה והידע והניסיון שרכשתי במהלכו אני אדע איך להתנהל בפרויקטים ומשימות אחרות שינתנו לי בהמשך החיים.

הכלים שאני לוקח איתי להמשך הם היכולת לנהל זמן בצורה נכונה כאשר יש לי פרויקט חשוב או משימה משמעותית אותה אני עושה במשך שנה. כלי נוסף שאקח איתי להמשך הוא ניהול סדר עדיפויות נכון בכל הנוגע לפרויקט בסדר גודל כזה אשר אפשר "להיאבד" בו בקלות ולהוסיף הרבה דברים קטנים אך פחות משמעותיים במקום שיפור ויעול הרעיון המרכזי של הפרויקט קודם לכן עד למקסימום האפשרי. בנוסף לכך, רכשתי את היכולת להתמודד עם קשיים ובעיות בקנה מידה גדול שקשה למצוא להם פתרון בזה הרגע או תוך זמן קצר. כלים נוספים שרכשתי הם היכולת ללמוד חומרים ונושאים חדשים לגמרי לבדי בצורה טובה ומהירה ובכל הקשור לשימוש והכרת ספריות שונות ב-Python ובכלל ובתהליכים שונים הקשורים למחשב, לרשתות ולאינטרנט.

עמדו בפניי קשיים ואתגרים רבים במהלך עבודתי, כגון הקושי להציג תמונה ללא רקע באמצעות tkinter - הספרייה בה מימשתי את הממשק הגרפי (GUI) שלי, שליחת אימוג'ים ב-WhatsApp, שליחת קבצים כמו תמונות וסרטונים ב-WhatsApp ועוד. קשיים אלה היו ממש קשים לפתירה אך ראיתי בכל אחד ואחד מהם אתגר ולא הפסקתי לחשוב על פתרונות לאתגרים אלה עד שפתרתי אותם, דבר שלקח לפעמים גם כמה חודשים לבצע וגרם לי לעבור תהליך של למידת נושאים אלה בצורה מעמיקה מאוד הכוללת גם ביצוע ניסיונות שונים לתקן את הבעיה (שנכשלו פעם אחר פעם), עד שהגעתי לרגע בו הצלחתי לפתור את הבעיה ולצלוח את האתגר אשר התמודדתי איתו.

המסקנות שלי מפרויקט זה הן שיש ביכולותינו ללמוד נושאים קשים וגדולים לבד אם אנו רוצים בזה ודבקים במטרה זו. מסקנה נוספת שהגעתי אליה כתוצאה מעבודה זו היא שאנחנו לפעמים מסוגלים לעשות משהו יותר גדול ומאתגר ממה שאנחנו חושבים שאנחנו יכולים, מפני שבתחילת עבודתי על פרויקט זה לא ידעתי שאני יכול ליצור ולפתח פרויקט ברמה של פרויקט זה בשלב זה של חיי, ואילו ככל שהתקדמתי והגעתי למטרות שהצבתי לעצמי הבנתי

שאני יכול לפתח משהו הרבה יותר גדול ומשמעותי ממה שחשבתי שאני יכול לפתח בהתחלה.

אם הייתי מתחיל לעשות את הפרויקט היום הייתי מתחיל להתעסק בכל הנושא של הממשק הגרפי של התוכנה רק לאחר שהייתי מסיים לכתוב את כל הקוד שלה במלואו ומוודא שסיימתי לעבוד על כל החלק הביצועי של התוכנה (התוכן שלה מעבר לממשק הגרפי).

כמו כן, אני בטוח שאם הייתי מתחיל לעשות את פרויקט זה היום הייתי מסיים אותו תוך הרבה פחות זמן ובצורה יותר יעילה, מפני שכיום הידע והניסיון שלי רחבים מאלה שהיו לי לפני שנה ואני יודע להשתמש ביותר כלים ומבין הרבה יותר טוב איך להתחיל לעבוד על משימות ופרויקטים כמו פרויקט זה ולהתנהל בהם.

ביבליוגרפיה

- codecs - documentation*. (2020, June 13). Retrieved from Python:
<https://docs.python.org/3.6/library/codecs.html>
- Fredrik Lundh, A. C. (2015). *PIL / Pillow - documentation*. Retrieved from
<https://pillow.readthedocs.io/en/3.0.x/>
- os - documentation*. (2020, June 13). Retrieved from Python:
<https://docs.python.org/3.6/library/os.html>
- passlib - documentation*. (2020, May 12). Retrieved from PassLib:
<https://passlib.readthedocs.io/en/stable/>
- pickle - documentation*. (2020, June 13). Retrieved from Python:
<https://docs.python.org/3.6/library/pickle.html>
- pycryptodome - documentation*. (2004, January). Retrieved from PyCryptodome:
<https://pycryptodome.readthedocs.io/en/latest>
- python 3.6 - documentation*. (2020, June 13). Retrieved from Python:
<https://docs.python.org/3.6/>
- re (Regular expression operations) - documentation*. (2020, June 13). Retrieved from Python: <https://docs.python.org/3.6/library/re.html>
- Selenium - documentation*. (2011). Retrieved from
<https://www.selenium.dev/selenium/docs/api/py/api.html>
- socket - documentation*. (2020, June 13). Retrieved from Python:
<https://docs.python.org/3.6/library/socket.html>
- sqlite3 - documentation*. (2020, June 13). Retrieved from Python:
<https://docs.python.org/3.6/library/sqlite3.html>
- Sweigart, A. (2014). *pyperclip - documentation*. Retrieved from
<https://pyperclip.readthedocs.io/en/latest/>
- threading - documentation*. (2020, June 13). Retrieved from Python:
<https://docs.python.org/3.6/library/threading.html>
- time - documentation*. (2009, February 14). Retrieved from Python:
<https://docs.python.org/3.0/library/time.html>
- tkinter - documentation*. (2020, June 13). Retrieved from python:
<https://docs.python.org/3.6/library/tkinter.html>