**Overseer**

שם: אילן דוברומילסקי

ת.ז: 325070324

שם מנחה: רוברט פרידמן

בית ספר: תיכון חדרה חדרה

גרסה: 1.0V

תאריך: 08-Jun-2020



**היסטוריית גרסאות המסמך**

|  |  |  |
| --- | --- | --- |
| **תאריך** | גרסה | **תקציר השינויים** |
| 08-Jun-2020 | 1.0V | הגשה מלאה ראשונה |
|  |  |  |
|  |  |  |
|  |  |  |

**תוכן עניינים**

תוכן

[הקדמה 4](#_Toc42531152)

[1.1 מטרה 4](#_Toc42531153)

[1.2 המוצר 4](#_Toc42531154)

[1.3 הגדרות 4](#_Toc42531155)

[תיאור כללי 5](#_Toc42531156)

[2.1 פונקציונליות 5](#_Toc42531157)

[2.2 קהל היעד 5](#_Toc42531158)

[2.3 אילוצים עיקריים 5](#_Toc42531159)

[2.4 הנחות ותלויות 5](#_Toc42531160)

[**2.5** סיקור מצב השוק כיום (אופציונלי) 5](#_Toc42531161)

[דרישות מפורטות 6](#_Toc42531162)

[3.1 דרישות פונקציונליות 6](#_Toc42531163)

[3.2 דרישות של ממשקים חיצוניים 6](#_Toc42531164)

[3.3 דרישות לא פונקציונליות 7](#_Toc42531165)

[3.3.1 דרישות ביצועים (performance) 7](#_Toc42531166)

[3.3.2 דרישות מהימנות (reliability) 7](#_Toc42531167)

[3.3.3 דרישות זמינות (Availability) 7](#_Toc42531168)

[3.3.4 דרישות אבטחה (security) 7](#_Toc42531169)

[3.3.5 דרישות תחזוקה (maintainability) 7](#_Toc42531170)

[3.4 דרישות בסיס נתונים 7](#_Toc42531171)

[ארכיטקטורת המערכת 9](#_Toc42531172)

[4.1 מבט על 9](#_Toc42531173)

[4.2 פירוט רכיבי המערכת 9](#_Toc42531174)

[4.3 דיון בנושא העיצוב הנבחר 9](#_Toc42531175)

[עיצוב נתונים ופרוטוקולים 10](#_Toc42531176)

[ממשק משתמש 11](#_Toc42531177)

# הקדמה

החלק הזה ישמש להצגה כללית של מסמך האפיון - הוא נועד לאפשר לקורא להבין במה מדובר - מה זה בכלל הפרויקט הזה ומה מופיע בהמשך המסמך.

## 1.1 מטרה

מטרת המסמך הינו הצגה כללית של הפרויקט, כיצד הוא עובד ולמה הוא מיועד.

## 1.2 המוצר

*שם מוצר: Overseer*

*מה המוצר עושה: המוצר מייצר דוחות המסכמים פרטים על תעבורת הרשת בפרקי זמן או כמות המוגדרים על ידי המשתמש. בנוסף, יכול להציג סיכום של כל הדוחות.*

יתרונות: דוחות המוצר מוצגים בצורה קריא המאפשר ניתוח והבנה פשוטה ומהירה. המוצר קל לשימוש. המוצר הוא בעל רמת דיוק גבוהה.

## 1.3 הגדרות

שרת: השרת הוא ה"מוח" של המערכת. הוא דואג לניתוח ויצירת הדוחות. לצידו פועל האתר.

אתר: מציג את הדוחות הקיימים למשתמש.

סוכן: נמצא על מחשב העובדים. אחראי על קליטת וסיכום התעבורה ושליחתו לשרת.

פקטה: יחידת מידע העוברת ברשת.

DB(database) – קובץ המכיל json של מידע דרוש לפעולות בין הרצות.

# תיאור כללי

## 2.1 פונקציונליות

שרת:

* מריץ את האתר.
* מנתח את החבילות שנשלחות אליו מכל הסוכנים.
* מייצר דוח.

אתר:

* מציג את הדוחות.
* מייצר דוח - סיכום דוחות.

סוכן:

* מסניף פקטות היוצאות ונכנסות למחשב.
* עושה סיכום של כל פקטה על פי שדות קבועות מראש.
* שולח לשרת את לניתוח כללי את הפקטה במסוכמת.

## 2.2 קהל היעד

המוצר מיועד לכל ארגון המעוניין לפקח/להשיג מידע על תעוברת הרשת וכל עובד בפרט.

דרוש מעט ידע על רשתות. המשתמש מריץ את השרת – ביכולתו לבחור פורט אליו הסוכנים יפנו, כל כמה זמן ייצר דוח ובאיזה אופן. סוכן – פורט אליו יפנו הסוכנים בשרת, כתובת השרת.

על פי הגדרות אלו, המשתמש צריך לדעת למצוא את כתובת אינטרנט של השרת, ולדעת באיזה פורט לבחור כך שלא יפגע בתוכנות אחרות.

## 2.3 אילוצים עיקריים

בשביל שהמוצר יעבוד כמו שצריך, דרושים לפחות שתי מחשבים – אחד שיריץ את השרת, ואחד שיריץ את הסוכן.

על המחשבים דרושה השפה פייתון גרסת 3.7.4 לפחות עם הספריות os, socket, json, time, threading, subprocess, sys, requests, re, scapy, flask. לשרת צריך להיות מספר פנוי של פורטים ככמות הסוכנים +1.

## 2.4 הנחות ותלויות

המוצר עובד בעיקרו מעל הרשת. הוא נרשם ב windows אך אני מאמין שיכול לעבוד מעל כל המערכת הפעלה התומכת באילוצים.

## **2.5** סיקור מצב השוק כיום (אופציונלי)

https://www.solarwinds.com/netflow-traffic-analyzer/use-cases/network-traffic-analysis

wireshark

# דרישות מפורטות

## 3.1 דרישות פונקציונליות

**יצירת דוח**

זה הפיצ'ר המרכזי של המערכת, יצירת דוחות המפרטות על התעבורה במחשבים וברשת.  
יצירת הדוח נעשת בשרת, הפונקציה מקבלת מספר מוגדר (או לפי זמן) של מידע ומשתמשת ב template מוכן. הפונקציה ממלאת מידע בtemplate, ואת התוצר שומרת כדוח.

**יצירת סיכום**

הסיכום הוא תוספת לפרוייקט. מטרתו היא להקל אף יותר למשתמש ולייצר דוח מיוחד המסכם את המידע בכל הדוחות שנאספו כבר.   
הסיכום נשען על בסיס הנתונים. האתר מייצר את הסיכום. הוא עושה זאת על ידי קריאת בסיס הנתונים וכמו הדוח שימוש בtemplate ומילואו. התוצר נשלח ישירות למשתמש ואינו נשמר בשום מקום.  
שגיאות שיכולות להתרחש: שגאית קריאה מבסיס הנתונים. במקרה כזה, יוחזר למשתמש הודעה מתאימה.  
לא מספיק מידע ליצירת דוח סיכום. גם פה תוחזר הודעה מתאימה.

**שמירת נתונים**

שמירת הנתונים הוא חלק בעייתי יחסית. יש לוודאה שהנתונים שנצברו תקינים, ושלא תהיה בעיה בשמירתם והוספם לבסיס הנתונים. בסיס הנתונים שלי, לשם נוחיות, נשמר כ JSON. כך הקריאה והפיכתו למשתנה שמיש, ושמירתו נעשים במהירות ונוחיות. בחרתי לשמור את המידע ישר אחרי יצירת כל דוח. כך ניתן לקטלג את המידע בקלות – לפי דוח, וגם העדכון נעשה כמה שיותר כדי למנוע איבוד מידע.

**multithreading**

ה multithreading הוא הפיצ'ר המאפשר את הפרוייקט. כך השרת יכול לתקשר עם כמה סוכנים בו זמנית, שיפור ביצועים, והקטנת זמני חישוב. השרת לא יתקע במקרה חריג, אלה רק thread וכך המערכת יכולה להמשיך ולתפקד.  
הפיצ'ר הזה לא מקבלת פלט, וגם לא יוצר פלט, אך הוא עוזר רבות בטיפול שגיאות, והעבודה הכללית של הפרוייקט.

## 3.2 דרישות של ממשקים חיצוניים

**3.2.1 ממשקי משתמש** (למשל: GUI)

האתר משמש כממשק משתמש.  
כקלט הוא מקבל נתיב לתקיה בה מוכלים הדוחות להצגה.  
המשתמש מבצע שאילתת HTTP פשוטה (דרך הדפדפן כנראה) והאתר יחזיר את הדוחות/עמוד מתאים.  
כאשר מתרחשת שגיאה באתר, הוא ישנה ערכים בעמוד האומרות שהתרחשה שגיאה.

**3.2.2 ממשקי חומרה**

מחשב שמסוגל להריץ python עם הספריות os, socket, json, time, threading, subprocess, sys, requests, re, scapy, flask.

**3.2.3 ממשקי תוכנה** (אם התכנה מחולקת לכמה תוכנות שונות שצריכות לתקשר בניהן)

הפרויקט מחולק ל 3 תוכנות. הסוכן, השרת והאתר. הסוכן מדבר עם השרת, השרת עם האתר והסוכן, והאתר עם האינטרנט.

ראשית השרת צריך להעלות, וביחד איתו הוא יתחיל את האתר. לאחר מכן הסוכן מקבל כקלט את כתובת השרת ויפתח שיחה איתו. הסוכן משם ישלח מידע לשרת, השרת ייצר דוחות, והאתר מציג אותם.

אם שגיאה תתרחש אצל הסוכן הוא יעצור. שגיאה בשרת תסגור את השיחה בינו לסוכן. שגיאה באתר יחזיר עמוד מותאם.

**3.2.4 ממשקי תקשורת (למשל תקשורת בין שר**ת ללקוחות)

התקשורת בין הסוכן והשרת מתנהלת בפרוטוקול JSON, האתר ב HTTP.

## 3.3 דרישות לא פונקציונליות

*דרישות מהמערכת שלא מתבטאות בפיצ'ר ספציפי או בתהליך ספציפי שמתרחש במערכת אבל משפיעות על אופן עיצובה ומימושה, לדוגמא:*

### 3.3.1 דרישות ביצועים (performance)

קוד השרת בנוי ברובו מ threading. החילוק משפר ביצועים

### 3.3.2 דרישות מהימנות (reliability)

**לאורך הקוד יש הקפדה על שמירת נתונים בזמנים אופטימלים, דאגה על כך שהקוד לא יקרוס משגיאות, ודאגה ל**multithreading **כך שלא יתקעו.**

### 3.3.3 דרישות זמינות (Availability)

**כל עוד השרת עומד, הסוכנים יכולים להתחבר.**

### 3.3.4 דרישות אבטחה (security)

### 3.3.5 דרישות תחזוקה (maintainability)

**לפני תחילת ריצת התוכנית המשתמש יכול לקבוע הגדרות להתאמות אישיות, דבר שעוזר על תחזוקה נוחה יותר. בנוסף, במקרה של קריסה, יודפס הודעה עם מידע שיעזור לטיפול הבעייה.**

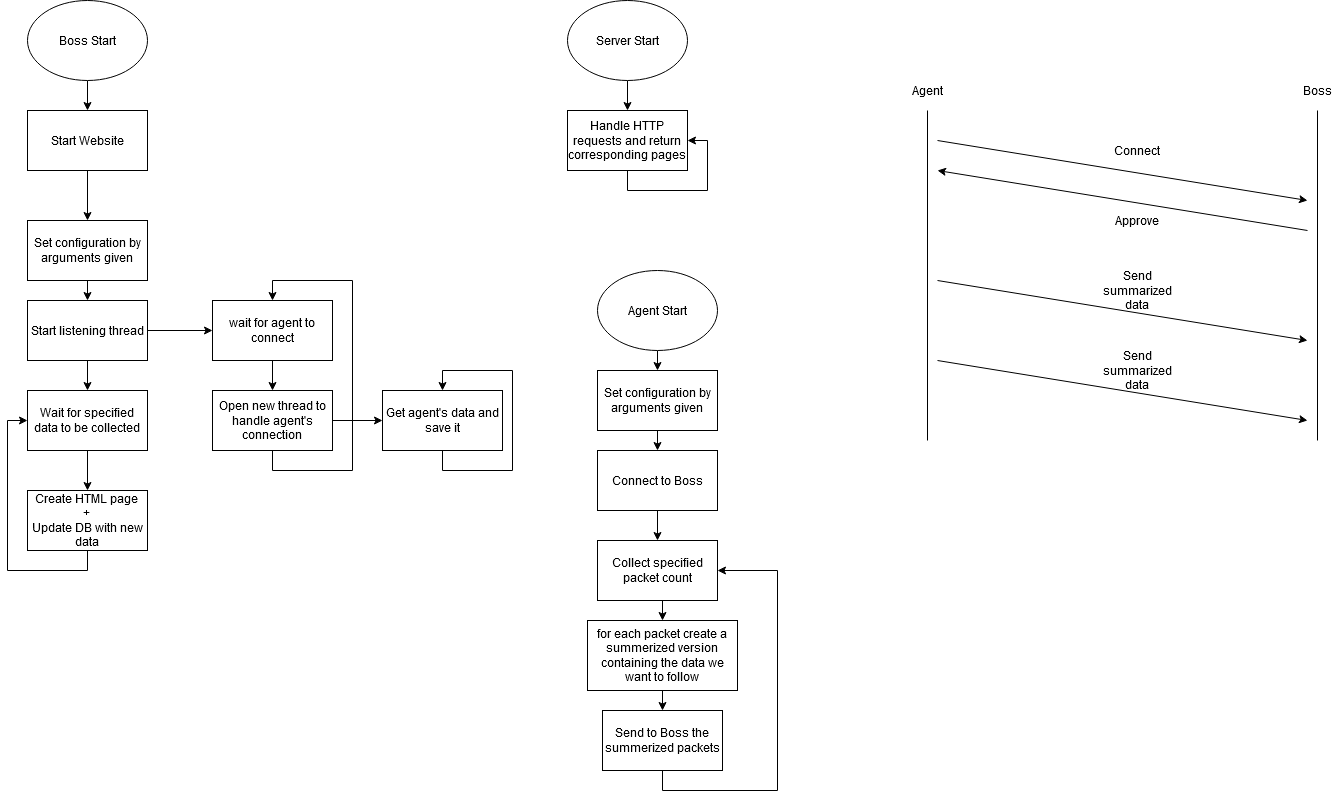
## 3.4 דרישות בסיס נתונים

*הנתונים הנכנסים לאחסון הם כל השדות הדרושות ליצור דוח. כך אם יש צורך בשחזור דוח, אפשר להעיר את הנתונים לפונקציה בשרת והוא ידע ליצר את הדוח בשנית.*

*הנתונים נשמרים כ json בקובץ מסוג .dat. אין חשיבות לסוג הקובץ, אלה רק למבנה התוכן שחשוב שהיה json שפייתון יכול לקרוא.*

*נפח האחסון הדרוש הוא תלוי זמן ריצה. ככל שהשרת רץ יותר זמן, כך ייצר יותר דוחות שישמרו ב DB. אומנם משקלו של הנתונים של כל דוח אינו הרבה, עם הזמן יגדל.*

*הקובץ עצמו אינו מוצפן בשום צורה ואף קריא יחסית.*

כלי ניתוח נוספים 

# ארכיטקטורת המערכת

## 4.1 מבט על

המערכת מחולקת לשלושה מודלים עיקריים.

השרת

הסוכן

האתר

השרת והאתר עובדים בצמידות על אותו מחשב. בעת הרצת השרת, האתר גם הוא עולה בצורה אוטומטית.

הסוכן, הנמצא על מחשבים אחרים (או אותו אחד), שולח הודעות לשרת.

פירקתי את המודלים כך כדי שכל רכיב מערכת יהיה תפקיד עיקרי. השרת אחראי על עיבוד מידע, הסוכן אחראי להעברתו לשרת, והאתר אחראי להציג את התוצאות למשתמש.

4.2 פירוט רכיבי המערכת  
**השרת**

השרת מתחיל את ריצתו על העלתו של האתר כ process נפרד ואז ממשיך בריצתו. (עשיתי זאת כדי שיהיה צורך להריץ רק קובץ אחד – השרת. זה בעיקר לנוחיות. אפשר להריץ את שניהים לכוד והפרוייקט יעבוד אותו הדבר).  
השרת פותח thread שירוץ לאורך שאר ההרצה. תפקידו הוא להאזין לסוכנים חדשים. ברגע שסוכן פונה לשרת, הוא מייצר class שמכיל בשדותיו ערכים המאפיינים את הלקוח ו thread שיטפל בצרכיו וממשיך לחכות לסוכנים נוספים.

thread הסוכן מחכה לקבלת סיכום התעבורה. לאחר שקיבל, מעדכן מספר משתנים לעיבוד עתידי ואז מחכה לסיכום התעבורה הבא. מטרת thread זה הוא פשוט לקלוט מידע ולאגור אותו.

בחזרה ל main – לאחר שפתח את הthread לקבלת סוכנים חדשים, נכנס ללולאה בה בוחר מתי לייצר דוח חדש (על פי הגדרות שקיבל בתחית הריצה כארגומנטים).

כאשר הגיע הזמן ליצור דוח, מדפיס את מספר החבילות הכולל שסוכמו עד כה ומתחיל לייצר את הדוח עצמו ושמירת נתונים חדשים ל DB.

**הסוכן**

הסוכן הינו מאוד נאיבי. מטרתו היחידה היא להסניף תעבורה ולסכם אותה.

הסוכן מתחיל לאגור פקטות (בשימוש של scapy) עד שמגיעה למספר ההוגדר לו (מועבר כארגומנט). כשהגיע לכמות הדרושה, מתחיל לעבור עליהם אחד אחד, ומייצר class בעל שדות מוגדרת מראש המכיל פרטים על הפקטה. לשם מילויי ה class יש פונקציה יחודית שמטרתה למלאות אותו. הסוכן מכין רשימה הבנוייה מהסיכומונים וכשהגיע לסוף, שולח את הרשימה לשרת ומתחיל את התהליך מהתחלה.

**האתר**

האתר הוא החלק היצוגי של הפרוייקט. הוא רשום בשימוש של flask. כאשר משתמש מתחבר לאתר, מוצג בפניו עמוד הבית. שם יכול לבחור להסתכל על דוחות שונים שהשרת ייצר. בנוסף, האתר מייצר סיכום דוחות בשימוש של ה DB שהשרת אגר.

## 4.3 דיון בנושא העיצוב הנבחר

היתרון בחלוקת הרכיבים לצורה הזו היא שמאוד קל להרחיב ולשנות כל רכיב. ניתן להוסיף שדות נוספים לכל פקטה בקלות, להוסיף עוד עמודים לאתר וניראות הדוחות שהשרת מציג. ניתן גם לשנות את הסוכן השרת והאתר לכל שפה רצויה והפרויקט ימשיך לעבוד כראוי כל עוד צורת המידע (פרוטוקול התקשורת) לא משתנה.

החסרון הוא שלפעמים יש קוד נוסף המכין הרחבה עתידית שלעולם לא ישומש. כלומר כתיבת קוד לא דרוש.

# עיצוב נתונים ופרוטוקולים

**פרוטוקול תקשורת**

בפרוייקט יש 2 פרוטוקולים ברשת.  
בין הסוכן והשרת:

הסוכן שולח לשרת רשימה בכל איבר הוא מילון המכיל 6 שדות, כל מילון מייצג סיכום של חבילה. המילון מכיל- שם תוכנה (מחרוזת), כתובת יעד (מחרוזת), מיקום גלובלי/מדינה (מחרוזת), האם החבילה נכנה או יצאה מהמכשיר (בוליאני), פורט בצד הסוכן (מספר) וגודל החבילה (מספר).

בין המשתמש לאתר:

שימוש סטנדרתי של פרוטוקל http- האתר שולח למשתמש את עמוד ה html.

**בסיס נתונים**

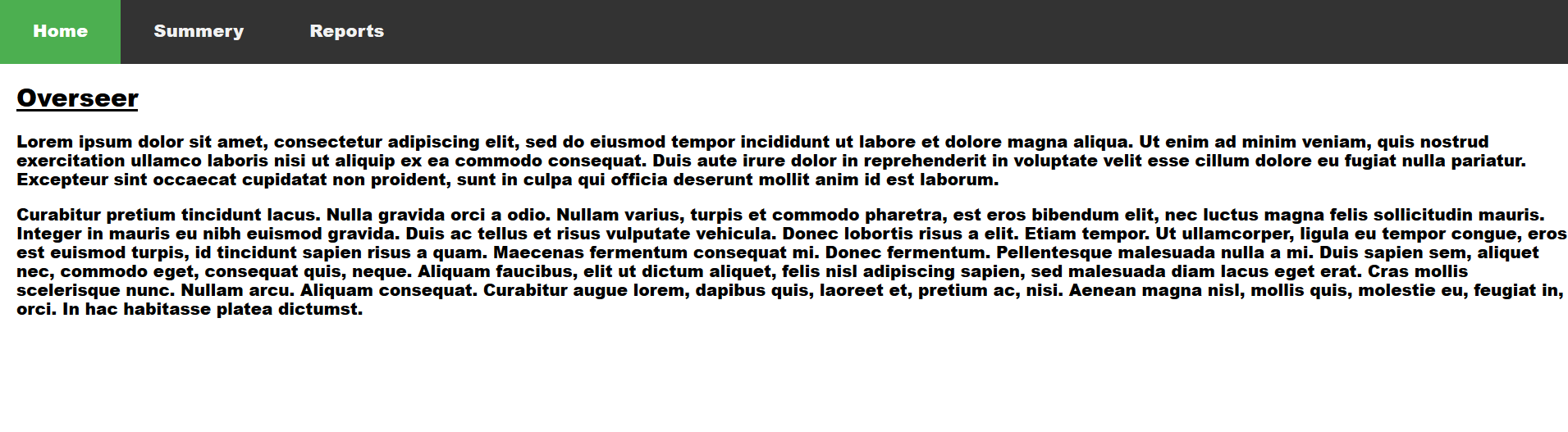
בסיס הנתונים בו אני משתמש הינו קובץ סטנדרתי בעל סיומת .dat – הקובץ שומר מחרוזת json של מילון, המכיל כמפתח את שם הדוח וכערך רשימה המכילה – מילון שמכיל כמפתח כתובת מכשיר וכערך את מספר בתים שקיבל, מילון שמכיל כמפתח כתובת מכשיר וכערך את מספר בתים ששלח, מילון שמכיל בכמפתח כתובת מכשיר וכערך התראות (כמו חיבור או התנתקות), מילון המכיל כמפתח מיקום גולובלי/שם מדינה וכערך את גודל המידע שנשלח לשם, מילון המכיל כערך את כתובת היעד וכערך גודל מידע, מילון המכיל כמפתח שם תוכנה וכערך גודל מידע, מילון המכיל כערך מספר פורט וכערך גודל מידע.

# ממשק משתמש

ממשק המשתמש בפרויקט שלי הוא למעשה האתר. לאתר יש 4 מסכים עיקריים.

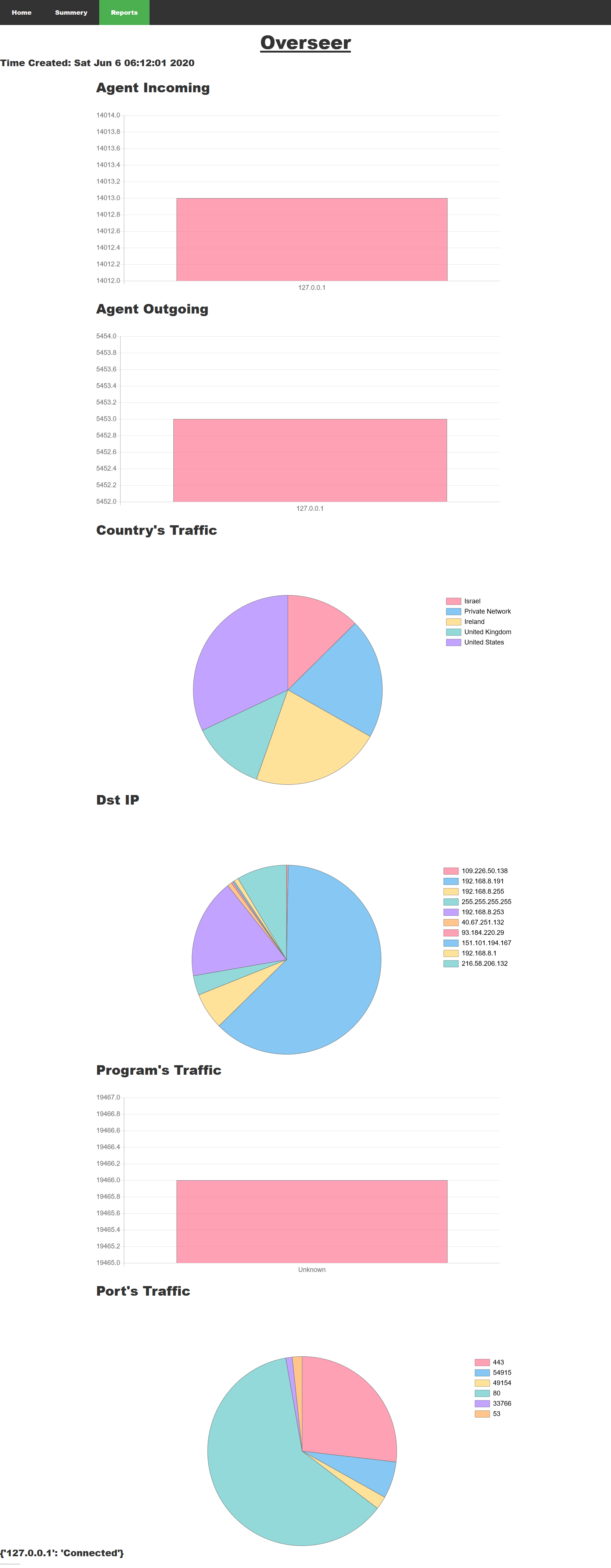
בכל מסך יש מעין tabs בחלקו העליון. הוא נועד למעבר בין חלונות נחויות.

בית



כך נראה מסך הבית. (כרגע יש פה גיבריש שמשמש כ filler. זה יוחלף בתיאור הפרויקט)

דוח

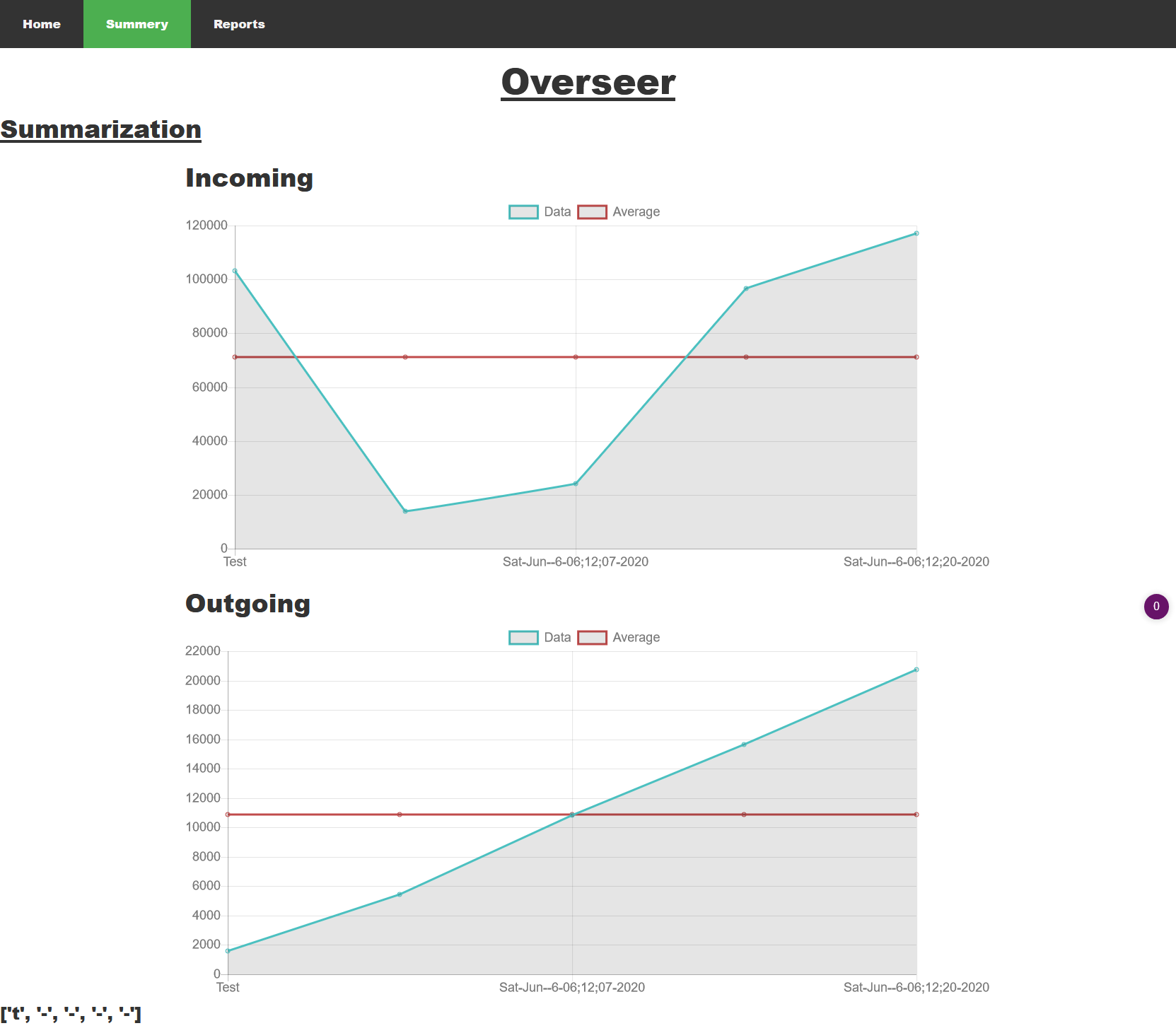


כך נראה הדוח שהפריקט מייצר.

רשימת כל הדוחות

במסך זה יופיעו כל הדוחות שיוצרו.

סיכום

מסך זה מציג סיכום בגרפים לכל הדוחות שיוצרו בעזרת ה DB.

# רפלקציה

נהניתי לעשות את הפרויקט. יצא לי להשתמש בהרבה כלים לצורך מימושו ואני גאה בתוצר הסופי.  
החלק הגראפי היה הכי מאתגר בשבילי כי לא היה לי ניסיון קודם בבניית אתרים ושימוש ב html, css ו javascript. לאורך הפרויקט חידדתי לעצמי נושאים בפיתון ואני מרגיש שלמדתי אפילו יותר לעומק אספקטים כמוlist comprehensions ושימוש בclass’ ו threading. לא הרגשתי שהיו לי קשיים משמעותיים אבל מדי פעם היו מכשולים שהייתי צריך לפתור או לעקוף. לאחר שסיימתי את מה שתכננתי לעשות ושמתי לב שיש לי עוד הרבה זמן, המשכתי להוסיף מעבר לתכנון שלי. סך הכל אני מרגיש שמיציתי את הפרויקט והוא נמצא במקום טוב ואין לי עוד פיצ'רים להוסיף.

# ביבליוגרפיה

<https://www.geeksforgeeks.org/comprehensions-in-python/>

<https://scapy.readthedocs.io/en/latest/>

<https://docs.python.org/3/library/threading.html>

<https://flask.palletsprojects.com/en/1.1.x/>

<https://realpython.com/primer-on-python-decorators/>