**המחלקה להנדסת תוכנה**

שם הפרויקט:

אפליקציה לניטור רמות שירות בזמן אמת

Project Name:

Service Level Management Real time monitoring

ספר פרויקט

|  |  |
| --- | --- |
| שם הסטודנט: | יניב נחום |
| שם המנחה: | ד"ר אלי וינטראוב |
| תאריך ההגשה: | 15.05.2014 |

# 

**תודות**

ברצוני להודות למנחה האקדמי שאיתו שיתפתי פעולה על מנת לבצע את הפרויקט ד"ר אלי וינטראוב .

רעיון הפרויקט היה של המנחה ובנוסף על כך המנחה נתן הנחיות מדויקות שעזרו לי לבצע את הפרויקט בצורה הטובה ביותר ולהגישו בזמן.

# תוכן עניינים

[2. רשימות 5](#_Toc229729267)

[3. פתיחה 6](#_Toc229729268)

א. תקציר מנהלים..................................................................................6

ב.Executive Summary......................................................................7

ג.מילון מונחים......................................................................................8

ד.מבוא ...............................................................................................9

ה.מטרות יעדים ומדדים...........................................................................7

ו.סקירה ספרותית...................................................................................7

[4. תיאור מצב קיים 7](#_Toc229729268)

[5. ניתוח חלופות מערכתי 7](#_Toc229729270)

[6. דרישות המערכת(Software Requirements Document) .](#_Toc229729277) 7

[7. אפיון המערכת(Software specifications) .](#_Toc229729277) 9

א. מודל המערכת...................................................................................7

ב. אפיון פונקציונאלי..............................................................................7

ג. ביצועים עיקריים................................................................................7

ד. אילוצי סביבה....................................................................................7

ה. ניתוח חלופות טכנולוגיות.....................................................................7

[8. תיכון המערכתSoftware design) .(](#_Toc229729277) 7

א. ארכיטקטורת המערכת........................................................................7

ב. תיכון מפורט.....................................................................................7

ג. אלטרנטיבות לתיכון המערכת................................................................7

ד. תרשים בלוקים..................................................................................7

[9. תכנון הפרוייקט project planning) )](#_Toc229729277) 9

א. ניהול סיכונים....................................................................................7

ב. תוכנית עבודה....................................................................................7

[10. בדיקות והערכהSoftware testing and evaluation) ).](#_Toc229729277) .8

א. תוכנית בדיקות תוכנה.........................................................................7

ב. דווח בדיקות תוכנה.............................................................................7

ג. דוגמאות הפעלה מפורטות מקצה לקצה....................................................7

ד. ניתוח יעילות......................................................................................7

ה. אבטחת מידע......................................................................................7

[11. התוצר](#_Toc229729277) ..….7

[12. סיום.](#_Toc229729277) 7

א.סיכום ומסקנות....................................................................................7

ב. פיתוחיים עתידיים והמשך עבודה............................................................7

ג. ריכוז שינויים מדו"ח תכנון מפורט...........................................................7

ד. רשימת מקורות...................................................................................7

[13. מסמכי הרחבה](#_Toc229729277) 7

א. מסמך הרחבה א (SRD)- Software Requirements Document ...........7

ב. מסמך הרחבה ב (SDD)-.Software Design Description......................7

ג. מסמך הרחבה ג STD-........................................................................7

ד. מסמך הרחבה ד STR)-) … ...Software Test Report............................7

ה. מסמך הרחבה ה Software Project Management Plan(SPMP)-...........7

ו. מסמך הרחבה ו -הגדרת הפרויקט............................................................7

ז. מסמך הרחבה ז -מאמר באנגלית הנסקר במהלך הפרויקט..............................7

[נספחים..........](#_Toc229729277) …7

[נספח א – דפי משוב סקרי אמצע פרויקט](#_Toc229729277) 7

# רשימות

**רשימת איורים**

[איור 1: מסך פתיחה 18](#_Toc324256549)

[איור 2:מסך אפשרויות 19](#_Toc324256550)

[איור 3: Monitoring – לחיצה על Dashboards 19](#_Toc324256551)

[איור 4:Monitoring– לחיצה על Graphs 20](#_Toc324256552)

[איור 5: Graphs – בחירה של Threshold Types 20](#_Toc324256553)

[איור 6: Graphs – בחירה של Devices להשוואה 21](#_Toc324256554)

[איור 7: Reports –בחירה של Reports by Time 21](#_Toc324256555)

[איור 8: Reports – בחירה של Report Prediction 23](#_Toc324256556)

[איור 9: Management – לחיצה על User Management 23](#_Toc324256557)

[איור 10: Management – לחיצה SLA Manager 24](#_Toc324256558)

[איור 11: תרשים פונקציונאליDFD-0 25](#_Toc324256559)

[איור 12: תרשים פונקציונאליDFD-1 26](#_Toc324256560)

[איור 13: הארכיטקטורה במבנה של שכבות 31](#_Toc324256561)

[איור 14: תרשים בלוקים 35](#_Toc324256561)

**רשימת טבלאות**

טבלה 1: שקלול חלופות טכנולוגיות, שפות התכנות 18

טבלה 2:שקלול חלופות מערכתיות 19

טבלה 3: אמינות ואיכות הנתונים 19

טבלה 4:מתודות מחלקות ה- UI 20

טבלה 5: מתודות מחלקת StartScreen 20

טבלה 6: מתודות מחלקתMainMenu 21

טבלה 7: מתודות מחלקת Dashboard 21

טבלה 8: מתודות מחלקת Graphs 23

טבלה 9: מתודות מחלקת Reports by Time 23

טבלה 10:מתודות מחלקת Prediction Report 24

טבלה 11: מתודות מחלקת User Mangement 25

טבלה 12: מתודות מחלקת SLA Manager 26

טבלה 13: בדיקת כניסה ויציאה מאפליקציה 31

טבלה 14: בדיקת תפריטים 35

טבלה 15 בדיקת בסיס הנתונים 35

טבלה 15 הערכת סיכונים מעודכנת 35

# פתיחה

1. **תקציר מנהלים**

מטרת דו"ח זה הינה להציג סיכום הפרויקט העוסק במערכת לניטור רמות שירות בזמן אמת ,האפליקציה תשמש את המנהלים ואת קבוצת ה-IT לניטור השירותים הקריטיים בארגון .

עם התפתחות הטכנולוגיה ומורכבות המערכות יותר יותר חברות משקיעות בתוכנות לניטור התשתית והשירותים הקימיים בארגון או המתקבלים מספק חיצוני.

הפרויקט הינה אפליקציה המקבלת זרם נתונים לגבי השירותים הניתנים מהספק ותייצר דיווחים לגבי עמידה במדדי השירות שהוגדרו.

אחת הבעיות הנפוצות בניטור רמות שירות היא שהרבה חברות משתמשות בשיטה ידנית לניהול ודיווח על התחייבות רמות השירות וזו נחשבת להתנהלות יקרה ובעלת נטייה לשגיאות רבות.

באפליקציה זו ננטרל את השיטה הידנית לגמרי ואף נספק מספר יכולות אשר לא קיימות בחברות המספקות את הניטור האוטומאטי.

באפליקציה נתאים באופן אוטומאטי את סוגי החריגות לרכיב הנבדק כך שאם למשל ננטר דיסק קשיח נבדוק לגביו זמן חיפוש או מקום פנוי וזה לא רלוונטי לגבי רכיב רשת המספק קישור לאינטרנט.

מטרתו של הספר היא לפרט ולהציג את הבסיס הרעיוני של מימוש הפרויקט.

בגופו מכיל הספר:

* הצגת ארכיטקטורת המערכת המממשת צורה נוחה לתפעול המערכת "ניטור רמות שירות בזמן אמת".
* גיבוש ונימוק הפתרון הנבחר בהתאם להשוואות שנעשו בשלב המחקר(דו"ח תכנון ראשוני).
* תוכנית בדיקות כנספח ה-STD, המאגדת את תוצרי הפרויקט ומבטיחה את נכונותם ואת איכותם.
* הצגת ארכיטקטורת המערכת ותהליכים בסיסיים בנספח ה-SDD הכולל תיכון ראשוני של המערכת.
* הצגת התכנית לניהול הפרויקט כנספח SPMP הכולל לוח זמנים מעודכן וניתוח סיכונים מעודכן.

נכון לרגע כתיבת מסמך זה, הפרויקט נמצא אחרי שלבי מימוש האב טיפוס, מומשה ליבת המערכת , נבנו כול האובייקטים החיונים לפונקציונאליות המהווים ממשק בין כול שכבות המערכת ושלבי הבדיקות מגיעים לסיומם

מסמך זה מכיל כנספחים את המסמכים הבאים:

* מסמך SRD- Software Requirements Document

מסמך המתאר את הדרישות הפונקציונאליות והמערכתיות אשר בהן מערכת התוכנה צריכה לעמוד.

* מסמך SDD- Software Design Description

מסמך המתאר את תיכון המערכת וכולל את ארכיטקטורת התוכנה, רכיבי התוכנה השונים ופירוט ממשקיהם, תרשימי UML ו-Use Cases .

* מסמך STD- Software Test Documentation

תיאור תיכון תוכנית הבדיקות של המערכת, אשר תשמש כמדד איכותי לעמידה בדרישות המערכת.

* מסמך STR - Software Test Requirements

מסמך המפרט את דרישות התכנית לבדיקות.

* מסמך SPMP- Software Project Management Plan

מסמך זה מבצע מעקב אחר מטרות המערכת ושמירה על מסגרת לוחות הזמנים. כחלק ממסמך זה מבוצע גם ניהול סיכונים המתעדכן במקביל להליך פיתוח המערכת.

1. **Executive Summary**

The purpose of this manual is to show the final implementing of the application Service Level Management real time monitoring; the application will serve the managers and the IT group to monitor critical services in the organization.

With the development of the technology and the complexity of the systems more and more companies investing in applications of monitoring software to monitor the infrastructure and the services that exist in the organization or received from an external supplier.

The project is an application that receives a stream of data regarding the services that provided from the vendor and generate reports about the compliance with the defined service indices.

One common problem in monitoring service levels is that many companies are using a manual method for managing and reporting on service levels and this considered expensive and has a tendency to many errors.

In this application we will remove completely the manual method and will provide number of capabilities that not exist in the companies that provide automatic monitoring..

In the application we will automatically match the types of deviations component under test so if, for example we will monitor hard disk we will check form him seek time or free space and this deviations not relevant for network component which provide connectivity to the network.

The purpose of this document is to elaborate and present the ideal basics of this project implementation.

The manual includes:

* A presentation of the system's architecture that implements a convenient form to activate the application Service Level Management real time monitoring.
* Formulation and explanations of the chosen solutions respectively comparing it to what we made in the research section (First Report).
* Software test documentation as an extension document that merge the project's products and ensuring their correctness and their qualities.
* A presentation of the system architecture and the basic processes in the software design description extension document that Includes primal design for the system.
* Presenting for the software project management plan as an extension document that includes as updated schedule and an updated risk management.

1. **מילון מונחים**
2. **C# -** זוהי שפת תוכנה מונחית עצמים שתוכננה ופותחה על ידי מיקרוסופט בשנת 2000.
3. **Thread -** זהו הרצף הקטן ביותר של הוראות מתוכנתות שיכול להיות מנוהל באופן עצמאי ע"י מתזמן מערכת ההפעלה**.**
4. **Multithreading –** זהו מודל תיכנות וביצוע נרחב המאפשר למספר רצפים להתקיים בהקשר של תהליך אחד**.**
5. **Process -** הוא מופע של תכנית מחשב הוא יכול להכיל מספר Threads, הוא מכיל את קוד התכנית והפעילות השוטפת שלה.
6. **API - Application Programming Interface –** הוא כינוי מקובל לערכות של ספריות קוד, פקודות, פונקציות ופרוצדורות מן המוכן בהן יכולים המתכנתים לעשות שימוש ללא צורך לכתוב אותן בעצמם.
7. **GUI – Graphic User Interface –** ממשק משתמש – הוא החלק העיקרי והחשוףמבחינת המשתמש בתוכנה.
8. .**UML – Unified Modeling Language -** כלומר: שפת מידול מאוחדת. שפת מפרט תקני לעיצוב מכוון עצמים ,כול בעלי התפקידים בצוות פיתוח תוכנה אמורים להכיר את הדיאגרמות העיקריות והדבר מסייע להם לתקשר בינהם את מפרטי התוכנה.
9. **Use Cases –** (תרחיש שימוש)זוהי טכניקה לאיסוף וניתוח הדרישות הפונקציונאליות של מערכות או תת מערכות.

תרחיש שימוש מורכב מרצף אירועים אחד או יותר , המתאר כיצד המערכת מתקשרת עם משתמשים כדי להשיג יעד עיסקי או פונקציה מסויימת.

1. **DFD - Data Flow Diagram –** תרשים זרימת הנתונים.זהו כלי בהנדסת תוכנה המציג בצורה גרפית את תהליכי עיבוד הנתונים המתרחשים במערכת המידע
2. **SLA - Service Level Agreement -** הסכם רמת שירות , הוא חלק מחוזה שירות שבו שירות מוגדר באופן רישמי, בדרך כלל משתמשים בשם זה כדי להתייחס לזמני תגובה המוגדרים בחוזה.
3. **Threshold –** הנקודה שבה ,ישנם תנאים שאם יעלו על הנקודה הזאת המערכת תתריע.
4. **MTBF – Mean Time Before Failure -** זהו כלי מדידה שמראה עד כמה חומרה או תוכנה הם אמינים בדרך כלל לרכיבי חומרה זמן המדידה הוא במאות אלפי שעות, לדוגמא לדיסק קשיח יחושב זמן של 300,000 שעות.
5. **מבוא**

הפרויקט הינו פרויקט אקדמי הנעשה כחלק ממסלול לימודים לתואר ראשון בהנדסת תוכנה B.Sc במכללת אפקה להנדסה בתל אביב . הפרויקט עוסק בתחום הבקרה ומדידת רמות שירות .

ניטור רמות שירות בזמן אמת הינה אפליקציית בקרה מדידה והתרעה המספקת פיתרון לניטור רמת השירות המתקבלת מהספק.

כיום, מספר הכלים המספקים פתרון לניטור רמת השירות המתקבלת מספק השירות הוא דיי גדול ואף קיים במגוון תחומים. כלים אלו יודעים למנף את המידע מהסביבה המבצעית, מבינים כיצד מידע זה מנוצל במדידות ביצועים, ומשווים ביצוע כנגד התחייבות שנעשתה בין לקוח לספק. כתוצאה מכך, ארגון יכול לנהל באופן פרואקטיבי חוזים כגון הסכמי רמת שירות. בעזרת כלים אלו, יכול הארגון להפחית את כמות המאמץ הנדרש בדיווח על קשרים אלו ולשפר את שביעות רצון לקוחותיו.

האפליקציה שבוצעה בפרויקט זה בעלת יכולת התאמה לצורכי הלקוח ויכולת גמישות בהגדרת סעיפי הסכם השירות איתו. ביכולתה של האפליקציה להביא בזמן אמת עדכונים על מצב המערכת והתרעה על חריגות ביחס להסכם.

השימוש באפליקציה הוא פשוט ונוח למשתמש, למשתמש מוגדר שם משתמש וסיסמה ולאחר מכן ישנו צורך לטעון חוזה שירות ומשלב זה המערכת מבצעת את כול עבודתה בצורה אוטמאטית.

כול הרכיבים הנמדדים במערכת מוגדרים מראש במבנה הנתונים ,לכול רכיב מוגדרים סוגי הניטורים המתאימים ספציפית אליו כאשר חלק מסוגי הניטורים מתאימים לכלל הרכיבים במערכת.

מנחה הפרויקט : ד"ר אלי וינטראוב.

מפתח המערכת : יניב נחום.

1. **מטרות יעדים ומדדים**
2. **מטרת הפרויקט:**

פיתוח אפליקציה המבצעת ניטור בזמן אמת של השירותים הניתנים לארגון בהתאם להסכם שנקבע עם הספק.

האפליקציה תקבל זרם נתונים לגבי השירותים הניתנים מהספק, ותייצר דיווחים לגבי עמידה במדדי השירות שהוגדרו. האפליקציה תותקן במערכת המידע של הארגון מקבל השירותים.

1. **יעדי הפרויקט:**

* אינדיקציה וזיהוי מהיר של חריגה מזמני תגובה הכתובים בהסכם.
* צמצום טעויות אנוש בחישוב עמידה בזמנים.
* פיתוח ממשק משתמש נוח ופשוט לשימוש למנהלים.
* הפקת דוחות התואמים את כמות התקלות ומשך הטיפול בהם.
* קיצור משך טיפול בתקלות.
* עלות פיתוח ותחזוקה נמוכה.

1. **מדדים:**

* התראה על 95% מחריגות בהסכם שנחתם.
* קבלת חווי על תקלה כלשהיא במערכת בתווך זמן שלא יעלה על 5 שניות מרגע התקלה.
* יבוצע Benchmark בין הפרויקט המבוצע לתוכנה מקבילה בשוק, תוצאות ה-Benchmark יצביעו על שיפור בזמנים בין הפרויקט המוצע לתוכנה הקיימת בשוק.

1. **סקירה ספרותית**

ישנם מספר נושאים עיקריים אפשריים לספרות הרלונטית לפרויקט זה: תחילה, נעסוק בהבנת מהות רמות השירות בעסקי המחשוב. בהמשך אנו נעסוק בדרכים למימוש וניטור (ניתוח ומדידה) של תהליכי מתן השירות על ידי נותן השירות ללקוח.

* ספרו של Blokdijk G **[6]** מתאר בצורה מפורטת את מהותה של רמת השירות והערכתה על ידי מקבל השירות. תהליך ההערכה כולל בתוכו מספר תת-תהליכים:

- תיכון תהליך השירות

- הגדרת דרישות רמות השירות

- משא ומתן והסכמה על רמת השירות

ובשלב שני:

- ניטור

- דיווח

- הערכה

- יישום הלקחים, (ובשאיפה) שיפור תהליך מתן השירות

כאשר המיקוד הוא על זיהוי נכון של נקודות אפשריות בשיפור השירות.

על מנת לתאר נושא זה היטב נוכל לקחת כדוגמה את העלייה החדה בשימוש במחשוב בענן. בעבר, כל ארגון היה בונה לעצמו חדר מחשב שבו היו נמצאים כל כוח המחשוב בארגון. בכדי לתחזק חדר מחשב היה צורך באנשי IT בעלי ידע רב שעלותם הייתה גבוהה לארגון. על מנת להוזיל עלויות (או אף במקביל לתעל את כוח האדם לטובת משימות אחרות) ארגונים רבים כיום מנצלים את מחשוב הענן ושירותיו לטובתם.

ספר זה בא לאגד את הדרישות ואת ה- SLA (רמות שירות) שבהם נותן השירות מחוייב לעמוד בהן על מנת שיהיה ניתן להשתמש באופן תקין בשירות מסוג מחשוב הענן.

ה- SLA הוא הבסיס על פיו הלקוח יכול לסמוך על נותן השירות ולצפות לשירות שהוגדר בו, אשר מכיל בעיקרו את

הגדרת התנאים למדידת השירות (SLO), ומנגנון ביקורת.

עניין הביקורת ותנאי המדידה של נותן השירות וסיפוק השירות הם נושאים חשובים למקבל השירות שכן עליו לבחון באופן תמידי ובמקביל לקבלת השירות את ההסכמים הללו, ולוודא שהמוצר (בין אם הוא חומרי או שירות) אותו הוא מקבל עונה על ההגדרות שנכתבו ב – SLA.

בפרויקט זה נתמקד במתן פיתרון לתהליך המדידה ובחינת ההסכמים הללו על ידי כך שנספק אפליקציה אשר תנטר (לפי מספר פרמטרים קבועים מראש ולפי ה – SLA) בזמן אמת את רמת השירות ותתריע על עמידה או חריגה מההסכם.

הבעיה העיקרית העומדת בפני ארגונים רבים היא בעיית ההתייעלות בין אם היא כלכלית או בין אם היא במתן\קבלת השירות מ\לקוחות תוך עמידה זהירה בדרישות. ווקר מציג בסיפרו **[2]** את העקרונות לפתירת בעיות בארגון בצורה מהירה ומקצועית, ובנוסף מציג איזה מידע צריך להכיל ארגון שעובד עם SLA .

SLA – זהו חוזה בין מספק השירות למקבל השירות. חוזה זה בדר"כ מתאר את השירותים שיסופקו , את איכות או רמת השירות ואת האחריות של שני הצדדים.

SLA – עשוי או לא עשוי לכלול תמחור שירות, החוזה גם יגדיר דרך למדוד את היענות שני הצדדים לתנאים שנקבעו.

Xiong K & Perros מתארים במאמרם **[4]** את קביעת המדדים, הקריטריונים והגישה להקצאת משאבי מחשב בענן (או כל שירות אחר) בסביבה כזו שמצמצמת את העלות הכוללת של משאבי המחשב. בנוסף, הם מציגים מדדים לחישוב כגון זמן אחוזון תגובה, ניצול אשכול מחשוב, שיעור אובדן מנות וזמינות אשכול מחשוב.

לבסוף, הצורך לכלי שיאגד את רמות השירות התגלה עוד לפני 15 שנים כאשר Hiles Aתיאר בספרו **[5]** את הבעיה שנובעת מצרכים אלו וכמובן את הפתרון שעל פיו פרויקט זה יונחה.

בספרו "המטרה" **[8],** ד"ר אלי גולדרט (מפתח תורת האילוצים) מציע דרכים להתמודד עם בעיות הגורמות לצוואר בקבוק .

בכול ארגון נוצר צוואר בקבוק שמונע את השגת המטרה :רווח, רק איתור צוואר הבקבוק,פתיחתו ושיפורו יעלה את

התפוקה של המערכת כולה וימנע תקלות קריטיות למערכת , בכדי להצליח בהתרת צווארי הבקבוק נדרש ליישם את

חמשת המהלכים הבאים: לאתר ולמצוא את האילוץ, להחליט כיצד לנצלו, להכפיף את המשאבים לאילוץ, להגדיל את יכולות האילוץ, ולבסוף להתחיל הכול מחדש כיוון שאילוץ,צוואר בקבוק חדש נוצר.

בעזרת תורת האילוצים ניתן לזהות מבעוד מועד תקלות אפשריות באחד מרכיבי המערכת .

# תיאור מצב קיים:

כיום, מספר הכלים המספקים פתרון לניטור רמת השירות המתקבלת מספק השירות הוא דיי גדול ואף קיים במגוון תחומים. כלים אלו יודעים למנף את המידע מהסביבה המבצעית, מבינים כיצד מידע זה מנוצל במדידות ביצועים, ומשווים ביצוע כנגד התחייבות שנעשתה בין לקוח לספק. כתוצאה מכך, ארגון יכול לנהל באופן פרואקטיבי חוזים כגון הסכמי רמת שירות. בעזרת כלים אלו, יכול הארגון להפחית את כמות המאמץ הנדרש בדיווח על קשרים אלו ולשפר את שביעות רצון לקוחותיו.

דוגמאות לכלים מסוג זה הנמצאים כיום בשוק:

**Oblicore Guarantee** – חברת Oblicore **[7]** נקנתה על ידי חברת CA ונחשבת כבמובילה בתחום ניטור רמת השירות. הכלי שאותו מציעה מפעיל ומאיץ את הניהול, הניטור והדיווח של כול הסכמי רמת השירות ואספקת שירות עבור ארגונים וספקי שירותים. הפיתרון של חברה זו, משפר את יכולתו של הארגון (המקבל שירות מספק) בדברים הבאים:

* נותן אמצעי לאגד את כול הנתונים בתחום השירות ונותן מדדים למדידה.
* נותן אפשרות להקים התחייבות חוזית ולקבוע יעדי ביצועים.
* עוקב אחר הביצועים כנגד המטרות שהוגדרו בזמן אמת ונותן אפשרות לנקוט בפעולות על סמך ביצועים אלו.
* מספק דוח על ביצועים לשני הצדדים בהסכם, הן לספק השירות והן לצרכן השירות.

**Maximo SLA Manager** – חברת IBM **[8]** היא חברה ידועה שמספקת פתרונות של חומרה ותוכנה. מוצר זה שמציעה הינו תוכנה המסייעת בהגדרת הסכמי רמת שירות, עוזרת ליישם נהלי דיווח ולפקח על אספקת רמת השירות. הפתרון של חברה זו מספק את הדברים הבאים:

* נותן אפשרות להגדיר מספר גופים לשם דיווח במקרה הצורך במטרה למנוע אירועים לפני שהם קורים. ניתן גם להגדיר למי להודיע ואיך להודיע.
* מציג ממשק גרפי שנותן תמונה ברורה של המצב הנוכחי ומציג מדדי ביצועים.
* מספק תבניות תקשורת הנועדו לספק תקשורת מובנית, על מנת להבטיח שמידע קריטי ישותף מבעוד מועד, להקטין אי הבנות ואירועים המשפיעים על השירות.
* תומך במסגרות ITIL – (Information Technology Infrastructure Library) על מנת לקשור הסכמי רמת שירות לתהליכי ITIL המאפשרים ניהול הדוק של תצורות שינויים, מהדורות, בעיות ואירועים.

**IT SLA Management Software** – חברת Up time Software **[9]** היא חברה שמספקת תוכנות לניהול מערכות של שרתים וירטואליים או שרתים פיזיים וניטור אפליקציות. הפיתרון שלה מספק את הדברים הבאים:

* נותן לוח מחוונים שמציג נתונים על סטטוס המערכת בזמן אמת.
* מספק אפליקציה בשימוש במכשירים סלולריים וכך עוזר להיות תמיד בשליטה.
* נותן דיווח מפורט של ההשפעה של כל אחד ממרכיבי התשתית שמשפיעים של סיפוק השירות בזמן.

האפליקציה המוצעת בפרויקט זה בעלת יכולת התאמה מלאה לצורכי הלקוח ויכולת גמישות בהגדרת סעיפי הסכם השירות איתו. ביכולתה של האפליקציה להביא בזמן אמת עדכונים על מצב המערכת והתרעה על חריגות ביחס להסכם.

האפליקציה המוצעת תהיה שונה משאר המערכות ביכולות הבאות:

* חיזוי SLA על ציר הזמן של רכיב חומרה, למשל MTBF ,לא רק מדידה בזמן אמת אלא גם חיזוי מההיסטוריה של הרכיב בהנחת התפלגות מסוימת.
* מדידת SLA של רכיב (למשל דיסק קשיח) בהשוואה ליתר הרכיבים במערכת ,הצגת זמני תגובה מינימום/מקסימום אך גם סטיות תקן ביחס ליתר הרכיבים.

# ניתוח חלופות מערכתי:

**5.1 פיתוח המערכת כתוכנת PC (Local Access)**

זוהי גישה המאפשרת לבעלי התפקידים במשמרת להיכנס למערכת האוטומטית לבדוק את הסטטוס של ביצוע מדידת השירות. כלומר, בדיקת מצב המערכת מתבצעת פיזית על ידי המשתמש על המחשב בו מותקנת המערכת. התאמת מערכת זו לצרכי המשתמש היא גבוהה היות והיא קלה לשימוש.

**5.2 פיתוח המערכת כ Agent Base**

זוהי גישה הדוגמת את השירותים השונים או מושאי הדגימה (כגון: מערכת חומרה, שירותי ענן, סלולר ועוד) ותדווח על מצבם או תתריע כשאירוע מסוים קורה למקום מרכזי. חלופה זו אמנם מנצלת את היותה מסונפת בכך שההתרעות מתקבלות במקביל. אך מצד שני, מימוש חלופה שכזו הוא מסובך למפתח יחיד ומורכב לפרק זמן מוקצב לפרויקט זה.

**5.3 פיתוח המערכת כאפליקציית Web**

פיתוח המערכת כאפליקציית Web מצד אחד, תיהיה נוחה לשימוש עבור המשתמש ותענה על צרכיו. חלופה שכזו תדרוש תוספת מורכבות לפיתוח, ותקשה על מטרתנו להביא לידי ביטוי פיתוח מהיר ו"זול". כמו כן, פיתוח מערכת זו כאפליקציית Web דורשת הבנה במספר שפות פיתוח שונות דבר שיצריך מהמפתח זמן למידה יחסית ארוך על מנת להתמקצע בצורה סבירה בהן.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| קריטריון | משקל | חלופה 6.1 | חלופה 6.2 | חלופה 6.3. |
| עלות מערכת  (זול = ציון גבוה) | 15% | 4 | 2 | 2 |
| זיהוי מהיר של חריגה (מהיר = ציון גבוה) | 30% | 5 | 5 | 5 |
| צימצום טעויות אנוש | 15% | 3 | 3 | 3 |
| נוחות שימוש | 20% | 3 | 4 | 4 |
| משך טיפול בתקלה  (קצר = ציון גבוה) | 20% | 3 | 3 | 3 |
| שקלול התאמה | 100% | 3.85 | 2.6 | 2.6 |

הסבר המשקלים והציונים:

לכל קריטריון ניתן משקל. המשקל ניתן באחוזים מתוך 100%.

לכל קריטריון ניתן ציון בין 0-5. הציון המסכם חושב כסכום המכפלות של ציון הקריטריון כפול המשקל.

**מסקנה:** הפתרון שנבחר הוא תוכנת PC, המשלב בתוכו מערכת המנטרת בזמן אמת את השירותים הניתנים לארגון בהתאם להסכם שנקבע מול הספק. המערכת תתריע ותדווח על עמידה/חריגה מההסכם בזמן אמת, הדיווח יוצג על גבי לוח מחוונים , גרפים וטבלה המאגדת את כול החריגות שקרו עד לזמן הנוכחי.

בצורה כזו המערכת שומרת על מורכבות מימוש פשוטה תוך שמירה על יכולת פונקציונלית גבוהה (שימוש בספריות וכלים קיימים).

שמירה על נוחות שימוש גבוהה- שכן היא לא תדרוש מהמשתמש הכנות רבות מראש ועקומת למידה יורדת של המערכת.

סביבת הפיתוח שנבחרה היא Visual Studio 2012ושפת הפיתוח היא #C, היכרות מפתח המערכת עם סביבת הפיתוח וניסיון קודם בפיתוח בה כמו כן תיעוד ותמיכה המוצעים עליה.

# דרישות מערכת:

דרישות המערכת לא השתנו מהדרישות המופיעות במסמך הגדרת פרויקט אך חודדו לצורך דיוק מירבי, ניתן לראות את הדרישות במסמך הרחבה חמישי, **מסמך הגדרת פרויקט**

**6.1 דרישות פונקציונאליות**

1. תחום העבודה - אבטחת איכות שירות ללקוחות וניהול פרויקטים.
2. תחום המוצר – הפרויקט מיועד לחברות הקונות שירותים מספק מסוים ורוצות לוודא קבלת שירות הולם העומד ביעדים אשר הוגדרו לספק ואשר הספק התחייב לספקם.
3. דרישות מידע ופונקציונליות:

* הצגת תמונת מצב עדכנית של כול הרכיבים הנבדקים בארגון .המערכת תחזיק תמונת מצב עדכנית אחרונה , המערכת תדגום את כול אחד מהרכיבים המוגדרים.התוצאות יוצגו על גבי מסך תמונת המצב האינטגרטיבית של המערכת. אם במהלך ריצת המערכת תזוהה חריגה כולשהי באחד הרכיבים או יותר, המערכת תציג את הזיהוי לגבי כול רכיב ורכיב בטבלה .
* הצגת מצב רכיב לעומק - ניתן להתמקד על רכיב ספציפי על ידי לחיצה על הרכיב המסוים לאחר מכן יהיה ניתן לקבל עליו מידע רב יותר,כגון :סוגי הבדיקות המבוצעות על ידי המערכת.
* הצגת חיזוי תקלות על פי נתוני עבר – חיזוי תקלות שיופיעו בעתיד על סמך נתוני העבר, חיזוי זה יוצג על גבי גרף ויוכל להציג על כול רכיב ורכיב את החיזוי לגביו.
* הצגת המדידה של SLA של רכיב בהשוואה ליתר הרכיבים הרלוונטים במערכת.
* המערכת תנהל משתמשים והרשאות , ישנים שני סוגי משתמשים , Viewer או Admin, לאחר הפעלת המערכת רק למשתמש מנהל תהיה האפשרות להוסיף משתמשי Viewer או Admin נוספים למערכת. בנוסף רק למשתמש מנהל תינתן האפשרות לבצע שינויים מערכתיים כוללים: כגון טעינת קובץ חוזה שירות חדש.

**6.2 דרישות לא פונקציונליות**

1. **דרישות עיצוב**

* המערכת צריכה להציג באופן ברור את מדדי המערכות.
* .המערכת צריכה להציג באופן גרפי סטטוס מערכות עדכני.

1. **דרישות שימוש וממשק משתמש**

* המערכת תודיע על כל מצב בו היא נמצאת – ריצה(ניטור הרכיבים), עצירה(סגירת המערכת), זיהוי חריגות.
* ממשק המשתמש הינו גרפי וישנם מספר שלבים החייבים להתבצע לפני שהמערכת מתחילה לעבוד(יצירת משתמשים, טעינת חוזה למערכת).

1. **דרישות ביצועים**

* המערכת תזהה חריגות ב-95% בהסכם שנחתם.
* המערכת תספק מידע אמין ומדויק בלמעלה מ-96% מהשימוש בה.
* המערכת תהיה זמינה ב-99.8% מהזמן.

1. **דרישות הפעלה**

* אין למערכת דרישות הפעלה מיוחדות. ממשק המשתמש יהיה ידידותי לכל משתמש בארגון, בנוסף על כך ממשק הניהול יהיה ידידותי ופשוט לשימוש, המשתמש צריך להיות בעל ידע בסיסי בתחום המחשבים והמערכות המופעלות בארגון .

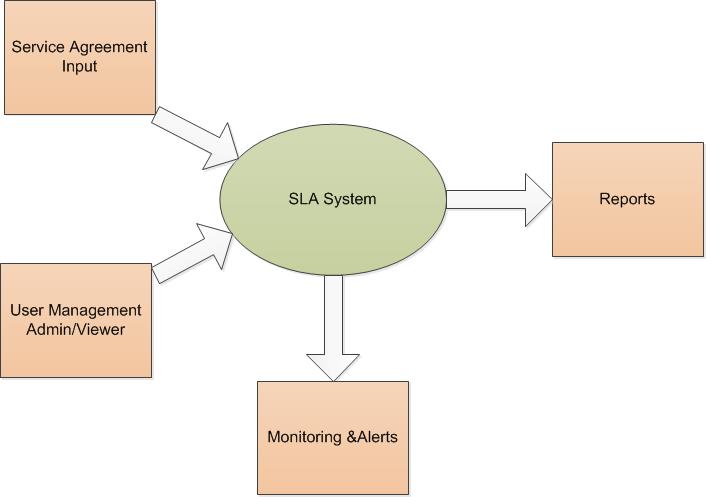
1. **דרישות תחזוקה ותמיכה -**

* המערכת נבנתה בצורה מודולרית כאשר ישנה הפרדה בין המודולים השונים על מנת לאפשר שינויים ותוספות פונקציונאליות עתידיות למערכת בצורה ברורה שימנע פגיעה בפונקציונאליות הקיימת.
* לאחר מסירת המערכת לא דרושה עבודת תחזוקה נוספת.

# אפיון המערכת

ניתן לראות את אפיון המערכת במסמך הרחבה ראשון , **מסמך SRD , סעיף 3.**

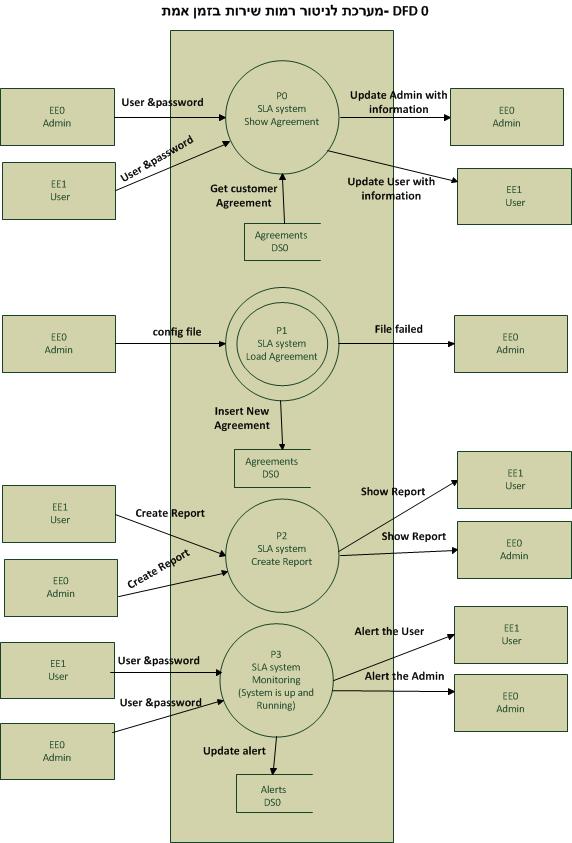
# מודל המערכת



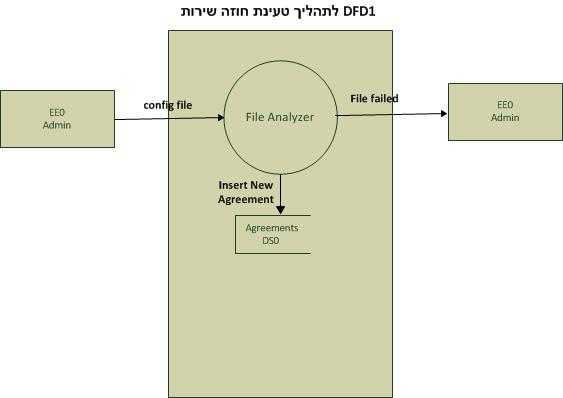
**איור 1: תרשים בלוקים של מבנה המערכת**

# תרשימי זרימת נתונים (DFD )

# 



**איור 2 : תרשים פונקציונאלי DFD -0**



**איור 3 : תרשים פונקציונאלי DFD -1**

# UML

Main UseCase(Admin)

****

**איור 4: תרשים זרימה מנהל מערכת**

Main UseCase(Viewer)

****

**איור 5: תרשים זרימה משתמש רגיל במערכת**

System Found Exception and Alert

****

**איור 6: תרשים זרימה מערכת מזהה חריגה ומתריעה**

Overall System Status



**איור 7: תרשים זרימה מנהל מערכת בודק את מצב המערכת**

# איפיון פונקציונאלי

המודולים האחראים על הביצועים יחד עם תרחישי המערכת המוזכרים בסעיף **4.1 במסמך ההרחבה הראשון (מסמך SRD )** הם המודולים הראשיים.

* גדגשד

# ביצועים עיקריים

הביצועים העיקריים המוזכרים בסעיף 4.2 במסמך ההרחבה הראשון (מסמך SRD) נותרו ללא שינוי.

* המערכת תזהה חריגות ב-95% בהסכם שנחתם.
* המערכת תספק מידע אמין ומדויק בלמעלה מ-96% מהשימוש בה.
* המערכת תהיה זמינה ב-99.8% מהזמן.

# אילוצי סביבה

מעבר לאילוצי סביבה המוזכרים בסעיף 2.3 ההרחבה הראשון (מסמך ה- SRD ) אין אילוצים נוספים.

# ניתוח חלופות טכנולוגיות

**7.5.1 C# -**

סביבה נוחה, ספריות מובנות לתאימות למערכת הפעלה המיועדת וכן לצרכי DB וכמו כן משלבת תמיכה וקהילה רחבה.

**7.5.2 Eclipse Java**

סביבה נוחה, ספריות מובנות במודל "קוד פתוח" ותאימות למערכות הפעלה שונות. אין ספריות מובנות ל DB דבר שיצריך זמן ארוך ללמידה ופיתוח.

|  |  |  |  |
| --- | --- | --- | --- |
| **קריטריון** | **משקל** | **חלופה 7.1** | **חלופה 7.2** |
| קלות פיתוח (ספריות קיימות כתובות באותה שפה) | 25% | 4 | 2 |
| עלות (זול = ציון גבוה) | 20% | 1 | 4 |
| הכרות המפתח עם הסביבה | 30% | 4 | 2 |
| קהילה, תיעוד ותמיכה | 15% | 4 | 4 |
| תמיכה בפלטפורמות מרובות | 10% | 4 | 4 |
| שקלול התאמה | 100% | 3.4 | 2.9 |

הסבר המשקלים והציונים:

לכל קריטריון ניתן משקל. המשקל ניתן באחוזים מתוך 100%.

לכל קריטריון ניתן ציון בין 0-5. הציון המסכם חושב כסכום המכפלות של ציון הקריטריון כפול המשקל.

הקריטריונים החשובים ביותר הם הכרות המתכנת עם הסביבה וקלות פיתוח. כיוון שסביבת העבודה הוא הקריטריון החשוב ביותר מבחינת אפשרויות ונוחות פיתוח – #C אכן מספק את הקריטריונים הללו משום ששאר הספריות (שנכלול בפרויקט ואחרות) כתובות בשפה זו (זוהי שפה מאוד פופולרית). בנוסף היכרות המתכנת עם הסביבה והשפה יאפשרו למתכנת לבצע את הנדרש ללא עלות למידה של השפה והתאמות אחרות.

ניתן לכתוב את המערכת בשפת JAVA אך בגלל שאין את תמיכה מובנית בבסיס נתונים דבר זה יצרוך זמן למידה ומחקר של המפתח את השפה ויצרוך כתיבה מחדש של מודול זה.

לאחר מכן עלות (20%), קהילה תיעוד ותמיכה (15%) ותמיכה בפלטפורמות רבות (10%).

קריטריון “קהילה תיעוד ותמיכה” קיבל ציון 4 מתוך 5 היות וכמתכנת אני סבור כי ישנה חשיבות רבה ללמידה של נושאים וכשיש תיעוד רחב שניתן להיעזר בו קל יותר לממש פתרונות לאתגרים בפרויקט.

קריטריונים אלו מהווים חלק כמעט עיקרי בשיקוליי בבחירת החלופה הטכנולוגית לכן משקלם הכולל מהווים כמעט מחצית מהציון הכולל.

**מסקנה :** סביבת הפיתוח שנבחרה היא Visual Studio ושפת הפיתוח היא C# היכרות המפתח עם הסביבה וניסיון קודם בפיתוח בה כמו כן התיעוד והתמיכה המובנית בבסיס הנתונים.

שיקול נוסף וקריטי הינו ספריות הקוד הפתוח הקיימות ברחבי האינטרנט המוצעות לשימוש חופשי המקלים על התאמתם לצרכי הפרויקט.

# תיכון המערכת (Software Design )

ניתן לראות את תיכון המערכת ,פירוט תתי המערכות והממשקים  **במסמך הרחבה שני (מסמך SDD)**.

# ארכיטקטורת המערכת

הארכיטקטורה שנבחרה הינה הארכיטקטורה עפ"י מודל השכבות , מימוש בשיטת השכבות מקנה יכולת החלפה של כול מודול תכנותי אחד באחר מבלי לפגוע בפונקציונאליות הכוללת.

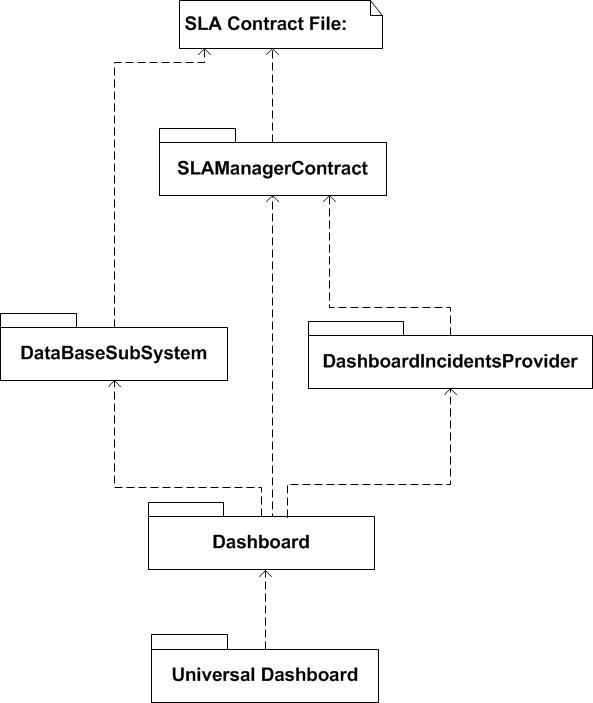


**איור 8: הארכיטקטורה במבנה של שכבות**

הסבר על איור 8 :

* **GUI** - שכבת התצוגה: שכבת זו כוללת את הממשק למשתמש , אותו רואה משתמש המערכת, בשכבה זו נמצאים האלמנטים הגרפים כגון : כפתורים , לוח מכוונים , גרפים, טבלאות
* **Business Logic -** שכבת הלוגיקה: שכבה זו כוללת את הלוגיקה והפונקציונאליות של הממשק הגרפי אותו רואה המשתמש בשכבת התצוגה, בשכבה זו מתבצעת העבודה עם החוזה שנטען למערכת , הנתונים שזורמים למערכת מהלקוח , ניטור הרכיבים וזיהוי החריגות ברמות ה-SLA.
* **Data –** שכבת המידע: כול זרימת הנתונים שמתקבלים מהלקוח ישמרו במבנה הנתונים, כמו כן החוזה הנטען למערכת שעל פיו המערכת מנטרת את רמות ה- SLA.

# חלוקה לתתי מערכות

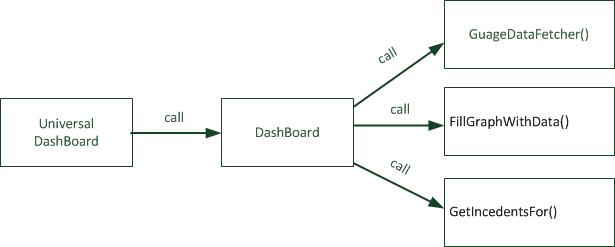


# תיכון מפורט

תיאור מפורט של הרכיבים ניתן למצוא במסמך הרחבה שני , מסמך SDD סעיף 3.

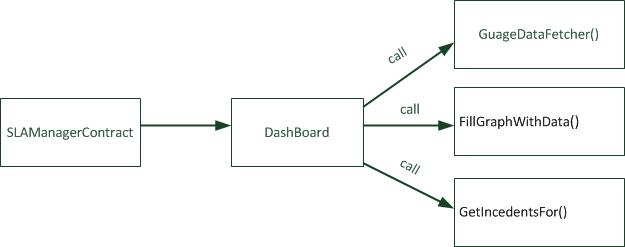
# Universal Dashboard

תכלית : מאפשר לראות את מצב המערכת במבט על ומאפשר לקבל החלטה מושכלת כאשר נרצה לבחור להתמקד ברכיב ספציפי אשר מאותת לנו .



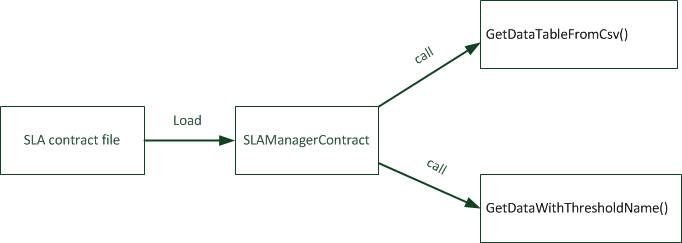
# Dashboard

תכלית : מציג את הסטטוס והניטורים הרלוונטים לרכיב הנבחר ואת החריגות הקיימות זמן אמת במערכת.



# - SLAManagerContract

תכלית :– ניהול חוזה השירות המסופק מהלקוח.



# DashboardIncidentsProvider

תכלית :– מספק למסך ה-Dashboard את המידע ,האם ישנה חריגה לגבי כול רכיב ורכיב.

# C:\yaniv\Dropbox\Public\שיעורים\פרויקט גמר\ספר פרויקט\כול מיני\DashboardIncidentProivder.jpg

# אלטרנטיבות לתיכון מערכת

תיכון מערכת תוכנה הינו תהליך של פתרון בעיות ותכנון פיתרון התוכנה שמוצע.

הפתרון מתוכנן ע"י מפתח התוכנה והוא יכלול בתוכו את הארכיטקטורה, אלגוריתמים בשימוש ורכיבים במערכת.

**קישור ל- Data Base**

האלטרנטיבות שנבחנו לקישור המערכת ל-Data Base הן:

1. Stored Procedures
2. שאילתות דינאמיות (שימוש במנגנון LINQ המובנה בתוך Visual Studio ובSQL command)

להלן סקירה המוצגת בטבלה של הקריטריונים שנבחנו בכל אלטרנטיבה:

|  |  |  |
| --- | --- | --- |
|  | **Stored Procedures** | **שאילתות דינאמיות** |
| **אבטחה** | SQL injection  מאפשרת הצגת נתונים גם למשתמשים שאין להם הרשאות לצפייה בטבלאות מסוימות. | SQL injection  נמנע ברוב המקרים ע"י כך שקלט משתמש הופך לערכי פרמטרים וזה מונע מפקודות זדוניות להשתמש במשוב המגיע מהלקוחות |
| **ביצועים** | שאילתה מאוחסנת וקבועה, ניתן לשנותה והשינוי ישפיע על כל המערכת בהתאם, ללא צורך בשינוי שאילתות המקושרות אליה, מאפשרת הגדרת תנאים, הגדרת משתנים ועוד. - יתרון בשאילתות גדולות ומורכבות. | שיטה פשוטה מאוד שמנהלת בצורה טובה עבודה מול בסיס הנתונים בשאילתות פשוטות. שיטה פחות טובה בשאילתות גדולות ומורכבות. |
| **מיקום קוד** | נמצא בשכבת בסיס הנתונים | נמצא בשכבת הלוגיקה ובשכבת DAL (Data access layer) |
| **תחזוקה** | שינוי לא דורש recompilation של חלקים באפליקציה | שינוי דורש recompilation של חלקים באפליקציה |
| **שינוי בסיס נתונים** | יש צורך לשכתב את כל הפרוצדורות השמורות בשכבת בסיס הנתונים כיוון שכל שאילתות ה-SQL נמצאות בשכבה זו. | משנים מספר מתודות בשכבת ה-DAL שאחראית על הקישור לבסיס הנתונים, שאילתות ה-SQL נשארות בשכבת הלוגיקה. |

טבלה 1: סקירת האלטרנטיבות לקישור ל-Data Base

להלן היתרונות והחסרונות של כל אלטרנטיבה:

|  |  |  |
| --- | --- | --- |
| **חסרונות** | **יתרונות** |  |
| 1.בעת שינוי בסיס הנתונים יש צורך לשכתב מחדש את כל הפרוצדורות השמורות בשכבת בסיס הנתונים כיוון שכל השאילתות נמצאות בשכבה זו. | **1.** שאילתה מאוחסנת וקבועה שכל שינוי בה משפיע על כל המערכת בהתאם ללא צורך בשינוי שאילתות קשורות. | **Stored Procedures** |
|  | **2.**מאפשרת הגדרת תנאים והגדרת משתנים ועוד. |
|  | **3.**מאפשרת שיתוף בין שאילתות |
|  | 4.אופטימיזציה מערכתית על השאילתות – נטענות ל- cache ומעודכנות בכל פעם שמבצעים insert לטבלה |
| 1.בשאילתות גדולות ומתרחבות שיטה פחות טובה | **1.**במקרה של שינוי בסיס הנתונים משנים רק מספר מתודות בשכבת ה-DAL כיוון שכל השאילתות נשארות בשכבת הלוגיקה | **שאילתות דינאמיות** |
| 2.שינוי בשאילתה דורש recompilation של חלקים אחרים באפליקציה. | **2.**שיטה פשוטה מאוד שמנהלת את העבודה מול בסיס הנתונים בצורה טובה בשאילתות פשוטות |
|  | 3. SQL injection  נמנע ברוב המקרים ע"י כך שקלט משתמש הופך לערכי פרמטרים וזה מונע מפקודות זדוניות להשתמש במשוב המגיע מהלקוחות |

טבלה 2: יתרונות וחסרונות של כל אלטרנטיבה

**הפתרון שנבחר:**

בחרתי לעבוד בשיטת השאילתות הדינאמיות תוך שימוש במנגנון ה LINQ של Visual Studio וב-SQL Command לאור העובדה כי מרבית השאילתות בפרויקט פשוטות ולא מצריכות הגדרות משתנים ותנאים רבים ומורכבים מאוד ,במידה ואדרש לשאילתא כבדה ומורכבת ישנה אפשרות לשלב שימוש ב-Stores procedures בעלת ביצועים טובים יותר שתפשט את אופן הקישור אך כרגע אני לא רואה צורך בכך.

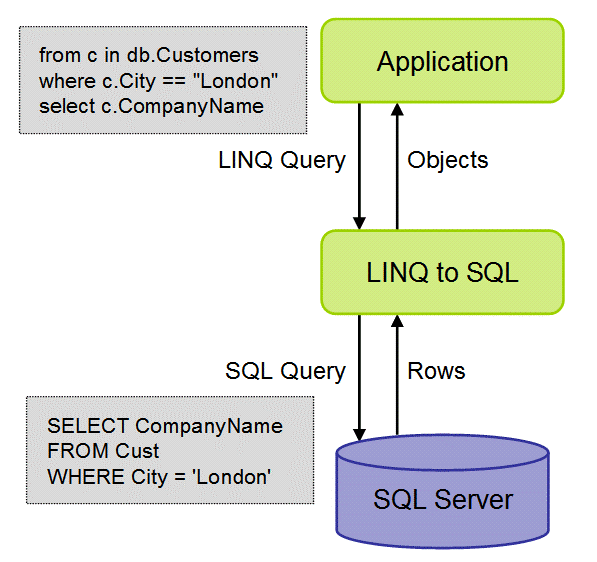
**הפתרון שנבחר לתיכון המערכת:**

טכנולוגית הפיתוח שנבחרה היא C# עבודה בתצורת תוכנת PC מקומית מהסיבות הבאות:

* + עניין רב של המפתח בשפה זאת ומיומנות הקיימת בטכנולוגיה.
  + קישור קל ל-DB מובנה ושימוש ביכולות חדשניות של Visual Studio כגון LINQ ו -SQL Command
  + נוחות פיתוח בסביבת הפיתוח Visual Studio

יתרונותיה של טכנולוגיה זו באים לידי ביטוי במודל פיתוח מובנה ומבוסס, פיתוח מונחה אובייקטים והיכולת להשתמש ברכיבים והמחלקות רבות אשר קיימות כחלק מתשתית ה-.Net

* מסד הנתונים שנבחר הוא: SQL Express Server 2008 מהסיבות הבאות:
* מיומנות קיימת של המפתח במסד הנתונים.
* רמת תאימות גבוהה ל- Framework ולמיקרוסופט.
* התוכנה חינמית ולכן לא נדרשת השקעה כספית.



תרשים 1: הצגת הקישור בין שכבת האפליקציה למבנה הנתונים בעזרת ה-LINQ

# תכנון הפרויקט

# ניהול סיכונים:

* את ניהול הסיכונים המורחב ניתן לראות במסמך הרחבה 4 SPMP סעיף 2 .
* בעת ניהול הפרויקט היה קשר רציף בין המנחה לסטודנט , הסיכון של אי עמידה בלוחות הזמנים עקב חוסר זמן שהוגדרו בדוח תכנון ראשוני התממש והמנחה קיבל עידכון בזמן על העיקוב והיה מודע לכך.
* היו מספר קשים במהלך הפרויקט שהעיקרי בינהם היה לאגד את כול חלקי הפרויקט למסך עיקרי אחד (UniversalDahsBoard) שידע להתריע בזמן אמת ולהפנות אותנו לרכיב או לרכיבים העיקרים בעלי החריגות בזמן נתון.

# תוכנית עבודה

* תוכנית העבודה שונתה קלות עקב עיקוב בהגשת דוח תכנון ראשוני אשר השפיע על הכנת דוח תכנון מפורט. מנקודה זו הסטודנט הקפיד לעמוד ביעדי הזמן שהוקצו בתוכנית ניהול הפרויקט אשר מצורף במסמך הרחבה רביעי – SPMP סעיף 2.

# בדיקות והערכה

# תוכנית בדיקות תוכנה

חלק זה של הפרויקט מראה כי כול מטרות הפרויקט מומשו ומבטיח כי הפרויקט נכתב בהתאם לדרישות שהוגדרו.

התוכנית מורכבת באופן כללי מבדיקות של כול מודול תוכנה כי שמתואר במסמך ה- STR המצורף כנספח הרחבה שלישי.

**הרכיבים שנבדקו:**

* + אישרור משתמשים בהתאם להרשאות שנקבעו להם בעת הרישום.
  + בדיקת העלאת קובץ חוזה שירות
  + וידוא תקינות הנתונים בחוזה השירות.
  + זיהוי וניטור פערים בין המוסכם בקובץ ה- input למצב הקיים במערכת.
  + התרעה בזמן אמת על כול פער או חריגה מהמוסכם בקובץ ה-Input.

**דווח בדיקות תוכנה**

במהלך כתיבת הפרויקט בדקתי מספר רב של פעמים של כול במערכת ובנוסף נעשו בדיקות מסודרות ופרטניות. בנוסף על כך תוקנו מספר רב של באגים שנמצאו במהלך כתיבת הפרויקט, ניתן לראות את דווח הבדיקות המלא במסמך הרחבה שלישי , מסמך STR.

**להלן תוצאות בדיקות התוכנה:**

* בדיקת הרשאות המשתמשים – הבדיקות עברו בהצלחה.
* בדיקת העלאת קובץ חוזה - הבדיקות עברו בהצלחה.
* בדיקת תקינות נתוני חוזה השירות - הבדיקות עברו בהצלחה.
* זיהוי וניטור פערים בין המוסכם בקובץ ה- input למצב הקיים במערכת - הבדיקות עברו בהצלחה.
* בדיקת התרעות בזמן אמת - הבדיקות עברו בהצלחה.

# דוגמאות הפעלה מפורטות מקצה לקצה

להלן דוגמאות הפעלה מקצה לקצה של מספר תהליכים עיקריים במערכת בליווי תצלומי מסמך והסברים בהתאם.

**התהליכים העיקריים:**

* כניסה למערכת ע"י משתמש בעל הרשאות Admin, העלאת חוזה שירות למערכת והצגת פעולת ניטור המערכת על הרכיבים בהתאם לחוזה.
* כניסה למערכת ע"י משתמש בעל הרשאות Admin או Viewer והצגת היכולות השונות.
* כניסה לדוחות השונים במערכת

# כניסה למערכת ע"י משתמש בעל הרשאות Admin ,העלאת חוזה שירות למערכת והצגת פעולת ניטור המערכת על הרכיבים בהתאם לחוזה

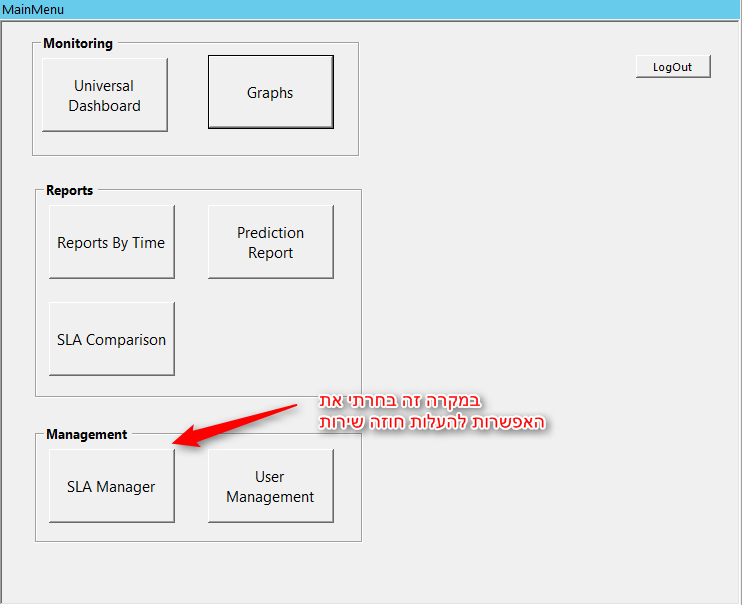
# כניסה למערכת על ידי משתמש בעל הרשאות Admin

זהו מסך הכניסה למערכת ,על המשתמש לבחור את שם המשתמש שלו ולספק סיסמה אישית.

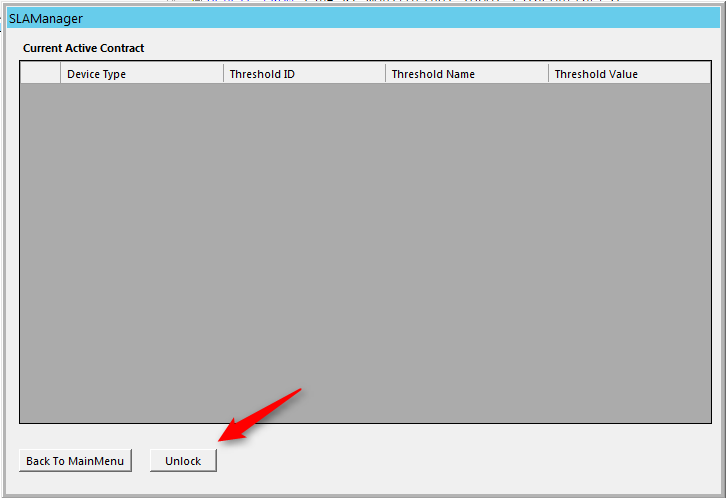


# מסך ראשי – MainMenu

מסך זה מתקבל לאחר שהמשתמש סיפק סיסמה נכונה למערכת (מסך זה יספק את האפשרויות הרלוונטיות בהתאם למשתמש שביצע כניסה (Admin או Viewer ) במקרה זה בחרתי לעלות חוזה שירות

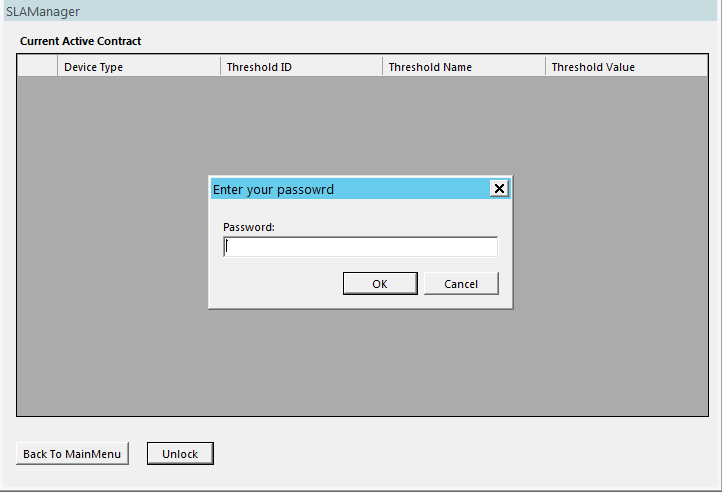


# העלאת חוזה שירות למערכת



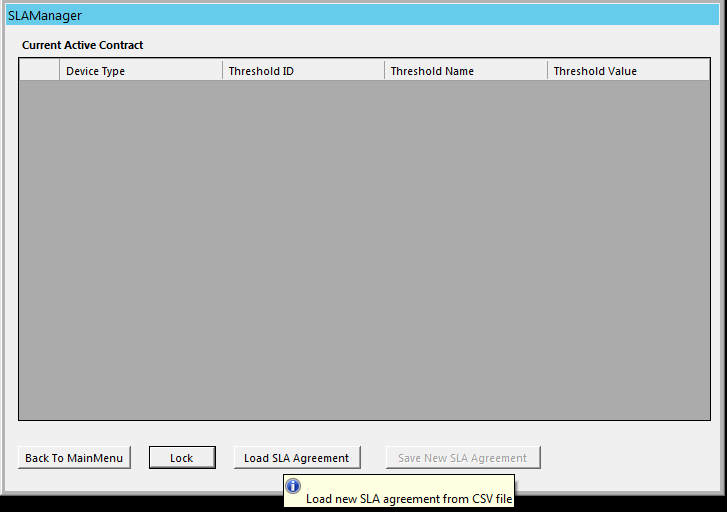
# בקשת סיסמה פעם נוספת

עקב חשיבותו של חוזה השירות לפעולתה התקינה של המערכת המשתמש מתבקש לספק את סיסמתו פעם נוספת.

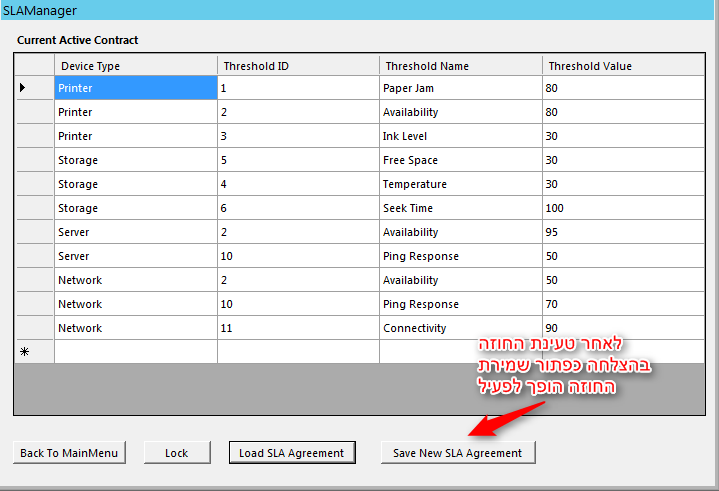


# הכנסת הסיסמה בהצלחה

לאחר הכנסת הסיסמא בהצלחה כפתור העלאת חוזה שירות מאופשר ופעולה זו אפשרית לביצוע

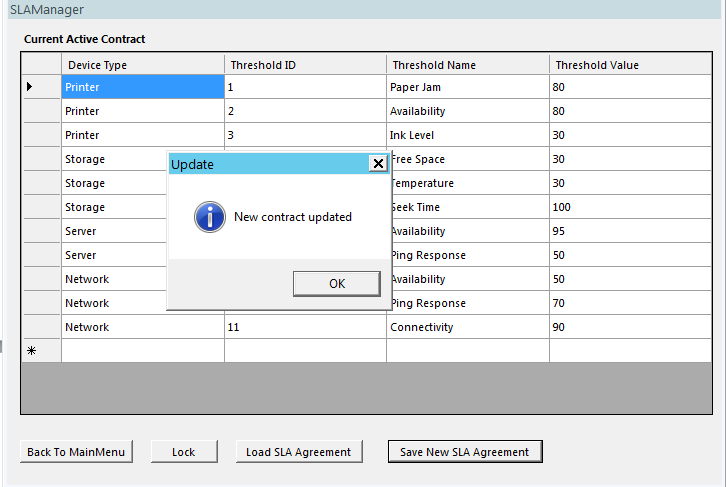


# חוזה השירות לאחר טעינתו מקובץ ה-CSV .



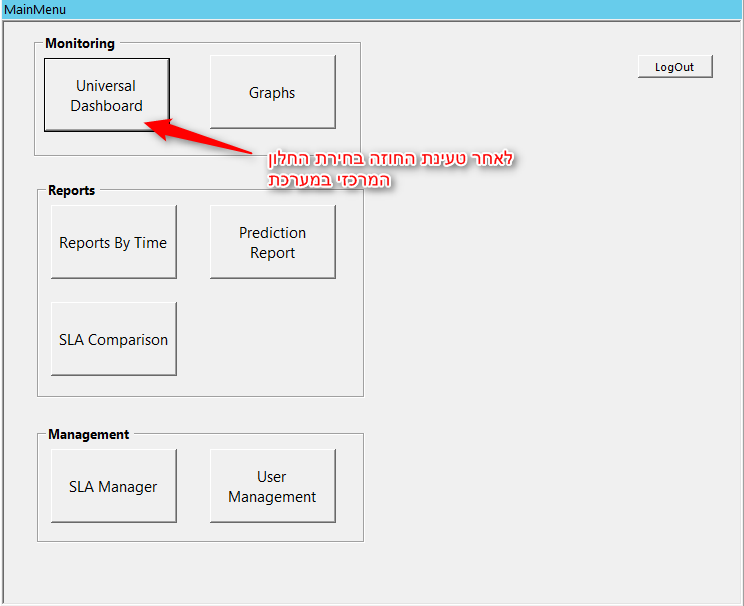
# שמירת חוזה השירות

החוזה מוצג לאחר שמירתו במבנה הנתונים עם הודעה מתאימה



# חזרה לתפריט הראשי

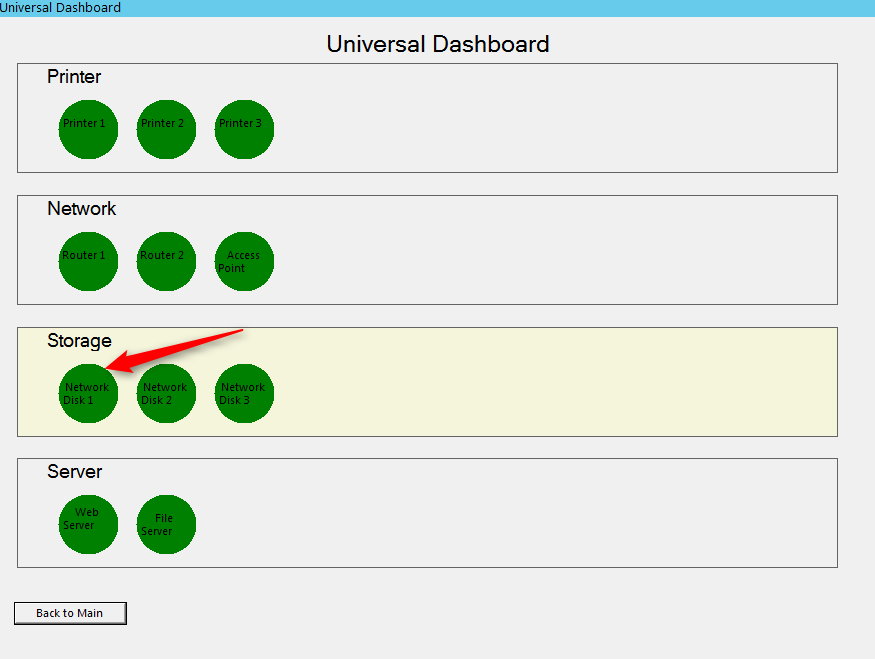
לאחר טעינת החוזה נחזור אל התפריט הראשי לבחירת המודול הרצוי.



# החלון המרכזי של המערכת

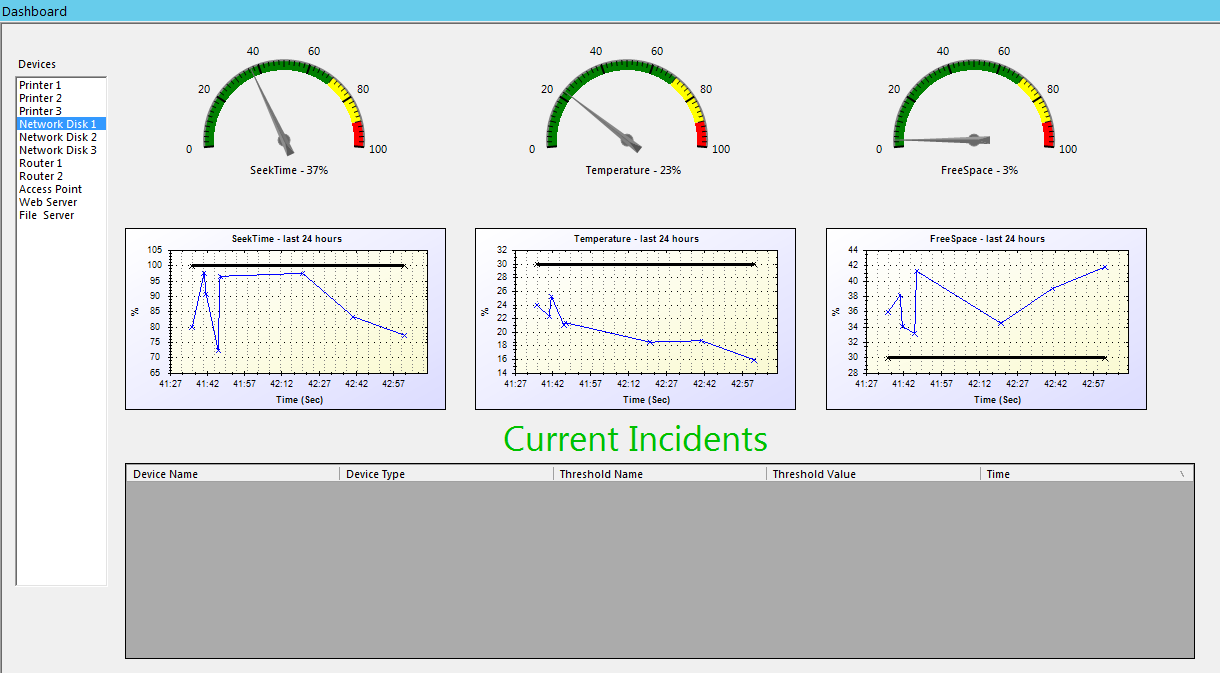
החלון המרכזי במערכת מוצג לאחר טעינת חוזה השירות , בשלב זה כול הרכיבים מדווחים עמידה ביעדי ה-SLA

ולכן כול הרכיבים בצבע ירוק ( כאשר חריגה או חריגה אפשרית מתקרבת הצבע ישתנה לאדום או צהוב בהתאמה)



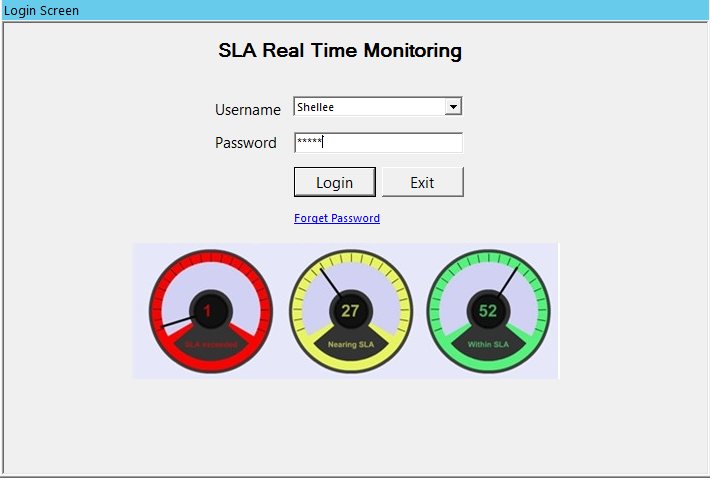
# תמונה מפורטת של רכיב

לאחר בחירת אחד הרכיבים מקבלים תמונה מפורטת אשר מספקת מידע רב יותר וממוקד על הרכיב הנבחר.



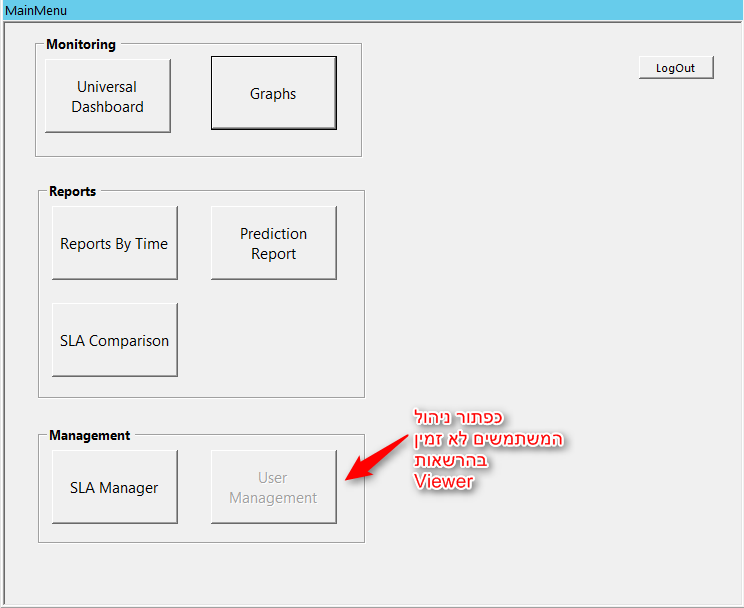
# כניסה למערכת ע"י משתמש בעל הרשאות Admin או Viewer והצגת היכולות השונות

# כניסה למערכת ע"י משתמש בעל הרשאות Viewer



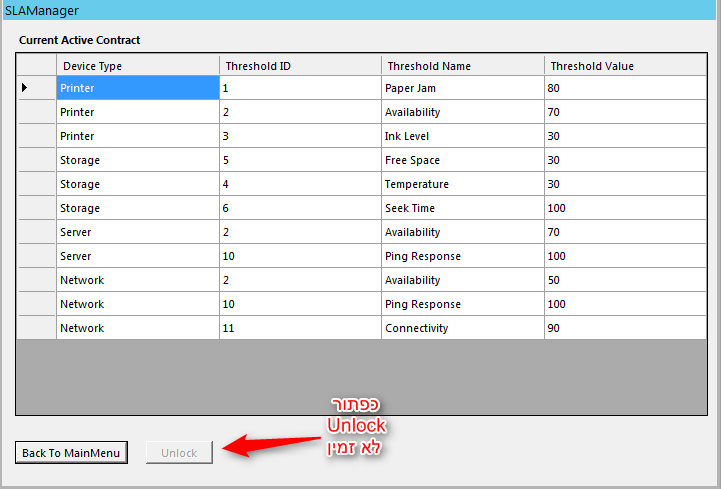
# מסך ראשי – MainMenu

מסך זה מתקבל לאחר שהמשתמש סיפק סיסמה נכונה למערכת (מסך זה יספק את האפשרויות הרלוונטיות בהתאם למשתמש שביצע כניסה (Admin או Viewer ) במקרה זה המשתמשת היא בעלת הרשאות Viewer



# בחירת מסך SLA Manager

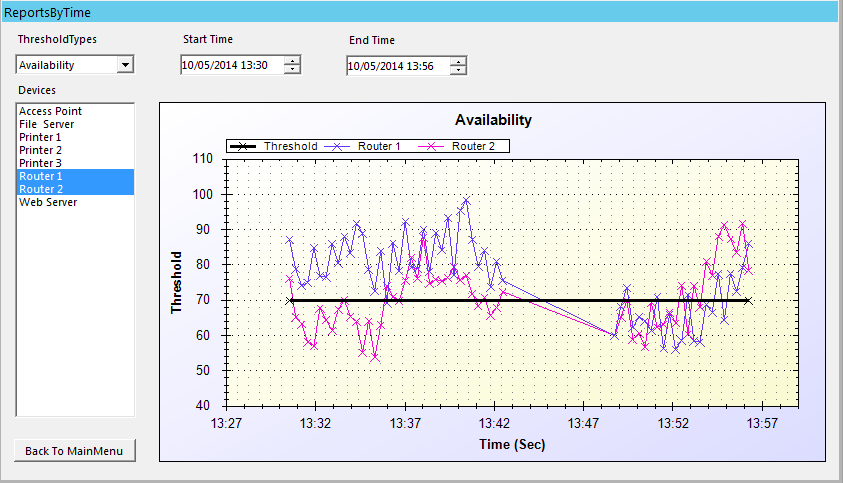
ניתן לראות כי לא ניתנת האופציה לשנות את חוזה השירות והאפשרות היחידה היא רק לצפות בחוזה הקיים .



# כניסה לדוחות השונים

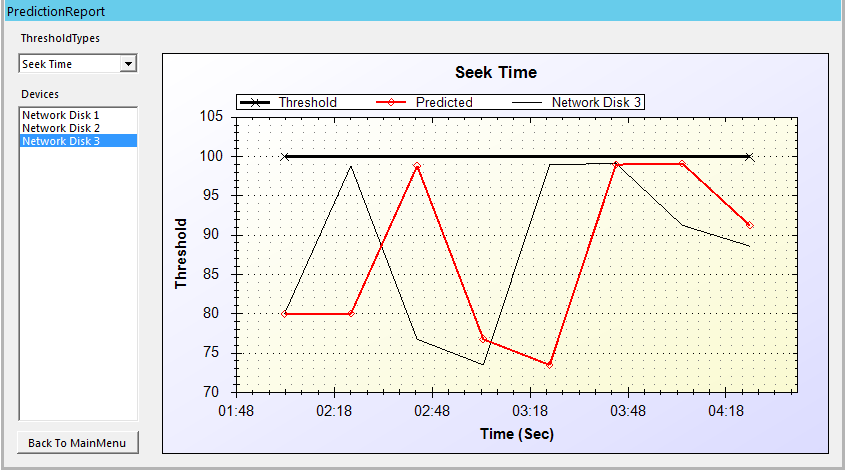
# דוח ReportsByTime

בדוח זה ניתן לראות את התנהגות הרכיב או הרכיבים לאורך זמן ולבחור טווח יום וזמן רצויים , כמו כן ניתן להשוות בין מספר רכיבים.



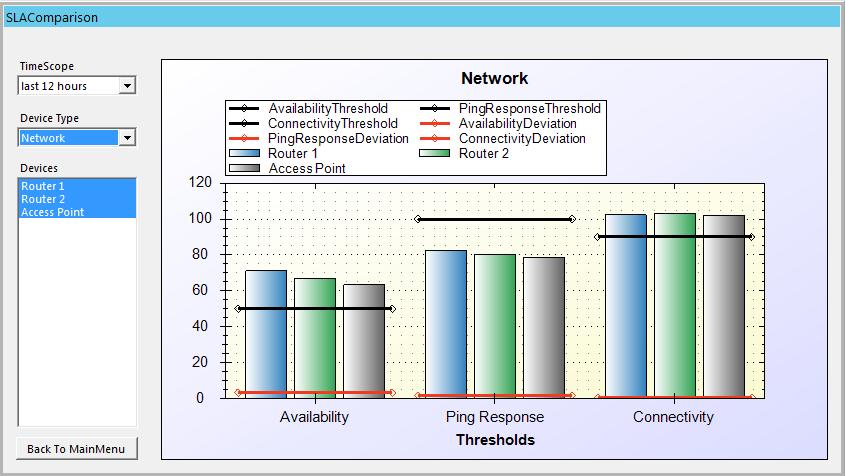
# דוח PredictionReport

בדוח זה ניתן לראות את חיזוי התנהגות הרכיב בעתיד ובעזרתו ניתן להעריך ולמנוע בעיות עתידיות צפויות



# דוח SLAComparison

בדוח זה ניתן לראות מדידת SLA ביחס לשאר הרכיבים במערכת ,הצגת זמני תגובה מינימום/מקסימום אך גם סטיית תקן ביחס ליתר הרכיבים.



# ניתוח יעילות

# ביצועי מערכת

המערכת עובדת בצורה מהירה (זמן אמת) ומדוייקת , המערכת מומשה עם מספר Threads שבעזרתם המערכת נותנת תמונה מלאה ומהירה של מצב כול רכיב וכול אלמנט שנבדק על הרכיב . שלב טעינת וניתוח חוזה השירות נעשה פעם אחת ובתחילת תהליך הפעלת המערכת , שלב זה הוא הכי איטי במערכת ואורך כ-9 שניות בממוצע לחוזה של 11 רכיבים , שלב זה הוא ארוך יחסית כי מספר האלמנטים שנבדקים הוא רב וזהו שלב קריטי לצורך פעולתה התקינה של המערכת כולה .

# עלות מערכת

המערכת נבדקה ותוצג על מחשב בין 5 שנים הנחשב מיושן מאוד לימינו, כיום המערכת צורכת משאבים מועטים ,שימוש המעבד הוא בין 8 ל-20 אחוז , והשימוש בזיכרון ה-RAM הוא כ -37 מגה בייט.

נעשו מאמצים רבים לבנות את המערכת בצורה יעילה ופשוטה למרות שנושא היעילות לא נדון בצורה עמוקה בפרויקט זה ,והדגש הוא על איכות זיהוי החריגות וההתרעה בזמן הקצר ביותר .

**הוא טעינת חוזה שירות נעשה פעם יחידה ובהפעלה הראשונית של המערכת בלבד לאחר מכן כול פעולת המערכת מתבצעת בצורה אוטמאטית .**

# אמינות המערכת

איכות ואמינות המערכת נמדדים בביצוע מטרות הפרויקט וביחס למערכות דומות הקיימות בשוק ,

מטרת הפרויקט מומשה במלואה .

יעדי הפרויקט בוצעו בצורה טובה והמערכת עומדת בהצלחה במדדים שהוגדרו.

כמו כן הבדיקות שבוצעו על המערכת עברו בהצלחה ,במהלך הבדיקות הראשוניות נמצאו מספר באגים שתוקנו וכעת המערכת נקייה לחלוטין מבאגים קריטיים.

# שלמות המערכת

מטרת המערכת הייתה גדולה מאוד והמערכת מבצעת את מטרה זו בצורה מיטבית .

# ייחודיות ומקוריות

המערכת מנתחת מסמך חוזה שירות הנגזר מקובץ CSV , ובעזרת מסמך זה המערכת גוזרת מהם אלמנטי בדיקה לרכיב ובהמשך מבצעת את כול פעולתה ללא התערבות המשתמש.

רעיון הפרויקט ניתן ע"י המנחה וכמו כן ניתנה עזרה רבה בהגדרת ייחודיות המערכת שנבנתה.

בניית ממשק המשתמש נבנה לאחר מחשבה עמוקה על יצירת ממשק פשוט ונעים לשימוש , ישנם מוצרים דומים שחלקם נראים מיושנים ולא ברורים למשתמש הפשוט .

יחודיות מערכת זו היא בשני דברים עיקריים :

* חיזוי SLA על ציר הזמן של רכיב חומרה, למשל MTBF ,לא רק מדידה בזמן אמת אלא גם חיזוי מההיסטוריה של הרכיב בהנחת התפלגות מסוימת.
* מדידת SLA של רכיב (למשל דיסק קשיח) בהשוואה ליתר הרכיבים במערכת ,הצגת זמני תגובה מינימום/מקסימום אך גם סטיות תקן ביחס ליתר הרכיבים.

# אבטחת מידע

למערכת יש דרישות אבטחה שבאות לידי ביטוי בכניסה למערכת ע"י בחירת שם משתמש והכנסת סיסמה מתאימה, החלק הראשון נעשה על מנת לתת את האפשרות לייצר שני סוגי הרשאות למשתמשים , משתמש מנהל(Admin) ומשתמש רגיל (Viewer) , כאשר רק משתמש מנהל יהיה בעל יכולת לשנות את חוזה השירות ובנוסף למחוק או להכניס משתמשים חדשים למערכת.

# התוצר

בפרק זה יפורטו חבילות התוכנה שפותחו במהלך כול תקופת הכנת הפרויקט.

* **לב המערכת:**
  + כול רכיב מנוטר לפי סוגי החריגות הרוולנטיות לגביו .
  + חלון מרכזי המאגד את כול הרכיבים והתנהגותם בזמן אמת וב-24 השעות האחרונות , כמו כן את כול החריגות הקיימות במערכת לגבי כלל הרכיבים.
* **זיכרון המערכת:**
  + בסיס נתונים Microsoft SQL express server 2008
  + מתודה לשמירת חוזה השירות .
  + מתודה לשמירת כול הנתונים הזורמים למערכת
* **תצלומי מסך של המערכת:**
  + דוגמאות הפעלה מקצה לקצה ניתן למצוא בפרק 10 בדיקות והערכה תחת סעיף ג' דוגמאות הפעלה מפורטות מקצה לקצה.
* **תשתיות נדרשות להפעלת המערכת:**
  + מחשב נייח או נייד בעל מעבד פנטיום 4 לפחות.
  + מערכת הפעלה Windows 7 .
  + מסד נתונים Microsoft SQL Express server 2008
* **סוג התוצר הסופי:**
  + אפליקציה המבצעת ניטור בזמן אמת של השירותים הניתנים לארגון בהתאם להסכם שנקבע עם הספק.

# סיום

# סיכום ומסקנות

במהלך הפרויקט למדתי שפת תיכנות חדשה C# ,בזמן העבודה על הפרויקט עשיתי את קורס C# שניתן במכללה והיווה קשר ישיר להתקדמות וליכולות השונות שמומשו בקוד , בנוסף על כך למדתי לנהל את הזמן שלי בצורה טובה יותר,היו קשיים רבים במהלך הפרויקט אך תמיד הסתכלתי קדימה אל היעד החשוב ביותר סיום פרויקט הגמר בהצלחה.

חשוב לציון שהחלק המשמעותי ביותר במהלך עבודה על פרויקט בסדר גודל כזה הוא מסמך הגדרת הפרויקט , בזמן כתיבת מסמך זה הידע שלי על הפרויקט היה ממש בחיתוליו ובהמשך הרגשתי כמה היה חשוב לבצע את מסמך זה כהלכה וכמו כן חשיבות תמיכת והנחיית המנחה בשלב כול כך ראשוני.

תחום של ניטור רמות שירות מרגיש כתחום שטרם מומש לגמרי ונראה כי חברות מתחילות להבין את חשיבות מוצר זה , כיום אני עובד עם חברה שמספקת שירותי CDN לחברות שונות והפרויקט מתקשר בצורה מושלמת לעבודתי כיום , המוצר שלנו מחוייב לעמות בזמני SLA שונים וכאשר זה לא קורה , הלקוח ברוב המקרים דורש דוח ליקויים על סיבת התקלה ופיצוי כספי עקב אי עמידת המוצר בזמני ה- SLA שהובטחו בחוזה שנחתם בין שני הצדדים.

תהליך העבודה על הפרויקט כלל תהליכים עיקריים כגון : הגדרת דרישות והגדרת מטרות המערכת ,סקירה סיפרותית , סקר מצב קיים, תיכון המערכת , חלופות מערכתיות, כתיבת מערכת שלמה הכוללת גם את הצד התיכנותי וגם את הצד המנהלי.

הפרויקט נבנה בשלבים , כאשר כול שלב עזר מאוד לראות את השלב הבא בצורה ברורה , למנחה ולי הייתה חשובה מימוש האב טיפוס בצורה אופקית צרה אך אשר מממשת את כלל יכולות המערכת ודרך זו איפשרה מימוש מהיר ונוח של כלל הפרויקט.

למדתי הרבה על חשיבותו של תהליך תכנון ראשוני נכון של מערכת והערכת מורכבותה , כמו כן את חשיבותו של תיכון מפורט אשר מהווה אבן דרך לכתיבת קוד בצורה טובה ויעילה, ראוי לציין שתכנון לקויי במקרים מסוימים יכול לגרום לעיקובים רבים ולכן מומלץ להשקיע את כול הזמן שניתן בשלב זה.

הפרויקט בשלב זה מוכן לגמרי , כול מתודות הלוגיקה ממושו בשלמותם , המערכת מגובה ומוכנה לפעולה.

# פיתוחים עתידיים והמשך עבודה

* + 1. **צפי לפלט:** עבור משתמשים רשומים המערכת תכנס מיד ושמם יופיע מצד ימין למעלה, קיימת גם אפשרות לשמירת נתוני הכניסה.  
       עבור משתמשים שאינם רשומים – המערכת תבקש מהמנהל לרשום אותם למערכת כדי שיוכלו להיכנס .
    2. **שלבי הבדיקה:**
       - כניסה למערכת
       - הכנסת פרטי משתמש
       - קבלת הפלט הצפוי (כניסה למערכת או בקשה לרישום)
  1. **בדיקת שאילתות**
     1. **תכלית:** בדיקת כל סוגי השאילתות שיהיו אפשריות במערכת ותוצאותיהן
     2. **קלטים:** כל שאילתא אפשרית
     3. **צפי לפלט:** עבור כל שאילתא אפשרית במערכת הצפי לפלט הוא פלט תקין, כל שאילתא שמתאפשרת למשתמש דרך מסכי המערכת היא תקינה ועבור כל שאילתא כזו תתקבל תוצאה רצויה.
     4. **שלבי הבדיקה:**
* . בנית שאילתא במערכת
* הרצת השאילתא.
* קבלת פלט סטטיסטי מתאים.
* בדיקה מול מבנה הנתונים כי אכן תוצאות השאילתא נכונות.

* 1. **בדיקת עקביות**
     1. **תכלית:** בדיקת עקביות של המערכת – ווידוא שהמידע מהימןועקבי
     2. **קלטים:** שאילתות במערכת
     3. **צפי לפלט:** עבור אותה שאילתא הצפי הוא לפלט זהה לכל הרצה
     4. **שלבי הבדיקה:**
* הרצת השאילתא במערכת.
* שמירת התוצאות לצורך השוואה.
* הרצת אותה שאילתא פעם נוספת.
* השוואה לתוצאות הקודמות שהתקבלו.  
  1. **בדיקת ביצועים**
     1. **תכלית:** בדיקת עמידה בדרישות הביצועים של המערכת
     2. **קלטים: שאילתות במערכת**
     3. **צפי לפלט:** מענה על כל שאילתא לא עולה על מספר שניות על פי הגדרת דרישות
     4. **שלבי הבדיקה:**
* הרצת השאילתא על מבנה הנתונים ומדידת זמן עד לקבלת תוצאה חזרה.

# 

הדו"ח מכיל סעיפים אשר נשארו ללא שינוי מהגדרת הפרויקט המורחבת ודו"ח מספר 1 ועל כן יפנה להגדרה המצורפת לדו"ח זה במקומות הרלוונטים.

בדו"ח זה אציג את תוצרי החשיבה המקדימה שביצעתי לנושא עיצוב המערכת ע"פ הדרישות שהוגדרו בשלבים והמסמכים הקודמים.

בדו"ח זה אציג את תיכון המערכת ואסקור את הנושאים הבאים :

1. סעיפים המתייחסים לתיכון המערכת:

* ניתוח חלופות לתיכון- תיאור השוואתי של מתודולוגיית התיכון שנבחרה אל מול אלטרנטיבות אפשריות.
* הפתרון שנבחר- סעיף המציג את תמצית הפתרון שנבחר לתיכון המערכת, עם הפניה למסמך SDD מפורט.
* סקר ספרות- תיאור תמציתי של נושאים בהם ישנו דגש מבחינת השלמת פערי ידע, או מקורות התייחסות ספרותיים הנדרשים לצורך השלמת הפרויקט.
* תוכנית בדיקות- תמצית תוכנית הבדיקות אשר תהווה מדד איכותי לעמידת המערכת בדרישות, עם הפניה למסמך STD מפורט.

1. סעיפים מערכתיים:

* לוחות זמנים- תיאור לוחות הזמנים המעודכן של הפרויקט.
* ניהול סיכונים מעודכן- הצגת תוצרי החשיבה של תהליך ניתוח הסיכונים של המערכת, ודרכי ההתמודדות האפשריים עם סיכונים אלו.

1. נספחים:

* מסמך SRD- Software Requirements Document

מסמך המתאר את הדרישות הפונקציונאליות והמערכתיות אשר בהן מערכת התוכנה צריכה לעמוד.

* מסמך SDD- Software Design Description

מסמך המתאר את תיכון המערכת וכולל את ארכיטקטורת התוכנה, רכיבי התוכנה השונים ופירוט ממשקיהם, תרשימי UML ו-Use Cases .

* מסמך STD- Software Test Documentation

תיאור תיכון תוכנית הבדיקות של המערכת, אשר תשמש כמדד איכותי לעמידה בדרישות המערכת.

* מסמך SPMP- Software Project Management Plan

מסמך זה מתאר מרכיבים של ניהול סיכונים ופרמטרים נוספים הרלוונטיים לאופן ניהול הפרויקט.

הפרויקט נמצא כעת בעת כתיבת מסמך זה בשלבי סיום של שלב התיכון ובאמצע מימוש אב הטיפוס. המשך פיתוח הפרויקט יחל מיד לאחר הגשת דו"ח זה.

# מטרות העבודה

* 1. **מטרת הפרויקט:**

פיתוח אפליקציה המבצעת ניטור בזמן אמת של השירותים הניתנים לארגון בהתאם להסכם שנקבע עם הספק.

האפליקציה תקבל זרם נתונים לגבי השירותים הניתנים מהספק, ותייצר דיווחים לגבי עמידה במדדי השירות שהוגדרו. האפליקציה תותקן במערכת המידע של הארגון מקבל השירותים.

* 1. **יעדי הפרויקט:**

1. אינדיקציה וזיהוי מהיר של חריגה מזמני תגובה הכתובים בהסכם.
2. צמצום טעויות אנוש בחישוב עמידה בזמנים.
3. פיתוח ממשק משתמש נוח ופשוט לשימוש למנהלים.
4. הפקת דוחות התואמים את כמות התקלות ומשך הטיפול בהם.
5. קיצור משך טיפול בתקלות.
6. עלות פיתוח ותחזוקה נמוכה.
   1. **מדדים**
7. התראה על 95% מחריגות בהסכם שנחתם.
8. קבלת חווי על תקלה כלשהיא במערכת בתווך זמן שלא יעלה על 5 שניות מרגע התקלה.
9. יבוצע Benchmark בין הפרויקט המבוצע לתוכנה מקבילה בשוק, תוצאות ה-Benchmark יצביעו על שיפור בזמנים בין הפרויקט המוצע לתוכנה הקיימת בשוק.

מטרות המערכת מוצגות ללא שינוי אך אני מוצא לנכון לפרט את תצורת הפיתוח לאור שאלות שעלו בסקר התכנון.

# סקירה ספרותית

הסקירה הספרותית שבוצעה בדו"חות הקודמים רלוונטית מאוד לפיתוח הפרויקט המתבצע בימים אלו,כמו כן התווספו מקורות ספרותיים נוספים.

* **המטרה (ד"ר אלי גולדרט - מפתח תורת האילוצים,וגף קוקס)**

בכול ארגון נוצר צוואר בקבוק שמונע את השגת המטרה :רווח, רק איתור צוואר הבקבוק,פתיחתו ושיפורו יעלה את התפוקה של המערכת כולה וימנע תקלות קריטיות למערכת , בכדי להצליח בהתרת צווארי הבקבוק נדרש ליישם את חמשת המהלכים הבאים: לאתר ולמצוא את האילוץ, להחליט כיצד לנצלו, להכפיף את המשאבים לאילוץ, להגדיל את יכולות האילוץ, ולבסוף להתחיל הכול מחדש כיוון שאילוץ,צוואר בקבוק חדש נוצר.

בעזרת תורת האילוצים ניתן לזהות מבעוד מועד תקלות אפשריות באחד מרכיבי המערכת .

# אלטרנטיבות לתיכון המערכת:

תיכון מערכת תוכנה הינו תהליך של פתרון בעיות ותכנון פיתרון התוכנה שמוצע.

הפתרון מתוכנן ע"י מפתח התוכנה והוא יכלול בתוכו את הארכיטקטורה, אלגוריתמים בשימוש ורכיבים במערכת.

* 1. **קישור ל- Data Base**

האלטרנטיבות שנבחנו לקישור המערכת ל-Data Base הן:

1. Stored Procedures
2. שאילתות דינאמיות (שימוש במנגנון LINQ המובנה בתוך Visual Studio )

להלן סקירה המוצגת בטבלה של הקריטריונים שנבחנו בכל אלטרנטיבה:

|  |  |  |
| --- | --- | --- |
|  | **Stored Procedures** | **שאילתות דינאמיות** |
| **אבטחה** | SQL injection  מאפשרת הצגת נתונים גם למשתמשים שאין להם הרשאות לצפייה בטבלאות מסוימות. | SQL injection  נמנע ברוב המקרים ע"י כך שקלט משתמש הופך לערכי פרמטרים וזה מונע מפקודות זדוניות להשתמש במשוב המגיע מהלקוחות |
| **ביצועים** | שאילתה מאוחסנת וקבועה, ניתן לשנותה והשינוי ישפיע על כל המערכת בהתאם, ללא צורך בשינוי שאילתות המקושרות אליה, מאפשרת הגדרת תנאים, הגדרת משתנים ועוד. - יתרון בשאילתות גדולות ומורכבות. | שיטה פשוטה מאוד שמנהלת בצורה טובה עבודה מול בסיס הנתונים בשאילתות פשוטות. שיטה פחות טובה בשאילתות גדולות ומורכבות. |
| **מיקום קוד** | נמצא בשכבת בסיס הנתונים | נמצא בשכבת הלוגיקה ובשכבת DAL (Data access layer) |
| **תחזוקה** | שינוי לא דורש recompilation של חלקים באפליקציה | שינוי דורש recompilation של חלקים באפליקציה |
| **שינוי בסיס נתונים** | יש צורך לשכתב את כל הפרוצדורות השמורות בשכבת בסיס הנתונים כיוון שכל שאילתות ה-SQL נמצאות בשכבה זו. | משנים מספר מתודות בשכבת ה-DAL שאחראית על הקישור לבסיס הנתונים, שאילתות ה-SQL נשארות בשכבת הלוגיקה. |

טבלה 1: סקירת האלטרנטיבות לקישור ל-Data Base

להלן היתרונות והחסרונות של כל אלטרנטיבה:

|  |  |  |
| --- | --- | --- |
| **חסרונות** | **יתרונות** |  |
| 1.בעת שינוי בסיס הנתונים יש צורך לשכתב מחדש את כל הפרוצדורות השמורות בשכבת בסיס הנתונים כיוון שכל השאילתות נמצאות בשכבה זו. | **1.** שאילתה מאוחסנת וקבועה שכל שינוי בה משפיע על כל המערכת בהתאם ללא צורך בשינוי שאילתות קשורות. | **Stored Procedures** |
|  | **2.**מאפשרת הגדרת תנאים והגדרת משתנים ועוד. |
|  | **3.**מאפשרת שיתוף בין שאילתות |
|  | 4.אופטימיזציה מערכתית על השאילתות – נטענות ל- cache ומעודכנות בכל פעם שמבצעים insert לטבלה |
| 1.בשאילתות גדולות ומתרחבות שיטה פחות טובה | **1.**במקרה של שינוי בסיס הנתונים משנים רק מספר מתודות בשכבת ה-DAL כיוון שכל השאילתות נשארות בשכבת הלוגיקה | **שאילתות דינאמיות** |
| 2.שינוי בשאילתה דורש recompilation של חלקים אחרים באפליקציה. | **2.**שיטה פשוטה מאוד שמנהלת את העבודה מול בסיס הנתונים בצורה טובה בשאילתות פשוטות |
|  | 3. SQL injection  נמנע ברוב המקרים ע"י כך שקלט משתמש הופך לערכי פרמטרים וזה מונע מפקודות זדוניות להשתמש במשוב המגיע מהלקוחות |

טבלה 2: יתרונות וחסרונות של כל אלטרנטיבה

**הפתרון שנבחר:**

בחרתי לעבוד בשיטת השאילתות הדינאמיות תוך שימוש במנגנון ה LINQ של Visual Studio לאור העובדה כי מרבית השאילתות בפרויקט פשוטות ולא מצריכות הגדרות משתנים ותנאים רבים ומורכבים מאוד ,במידה ואדרש לשאילתא כבדה ומורכבת ישנה אפשרות לשלב שימוש ב-Stores procedures בעלת ביצועים טובים יותר שתפשט את אופן הקישור אך כרגע אני לא רואה צורך בכך.

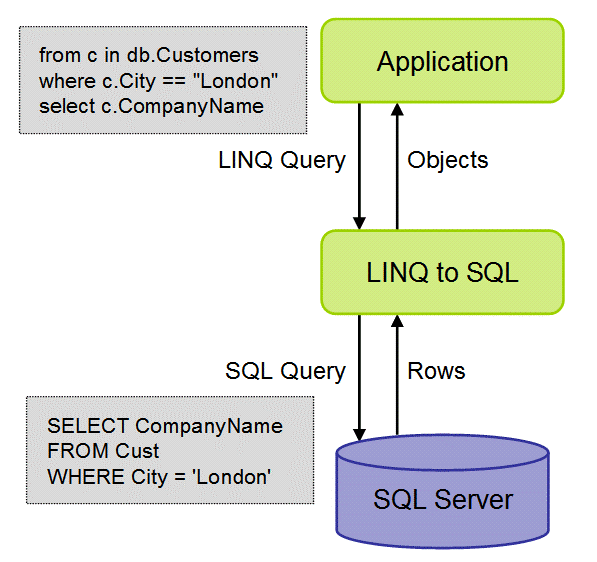
# הפתרון שנבחר לתיכון המערכת:

טכנולוגית הפיתוח שנבחרה היא C# עבודה בתצורת תוכנת PC מקומית מהסיבות הבאות:

* + עניין רב של המפתח בשפה זאת ומיומנות הקיימת בטכנולוגיה.
  + קישור קל ל-DB מובנה ושימוש ביכולות חדשניות של Visual Studio כגון LINQ.
  + נוחות פיתוח בסביבת הפיתוח Visual Studio

יתרונותיה של טכנולוגיה זו באים לידי ביטוי במודל פיתוח מובנה ומבוסס, פיתוח מונחה אובייקטים והיכולת להשתמש ברכיבים והמחלקות רבות אשר קיימות כחלק מתשתית ה-.Net

* מסד הנתונים שנבחר הוא: SQL Express Server 2008 מהסיבות הבאות:
* מיומנות קיימת של המפתח במסד הנתונים.
* רמת תאימות גבוהה ל- Framework ולמיקרוסופט.
* התוכנה חינמית ולכן לא נדרשת השקעה כספית.



תרשים 1: הצגת הקישור בין שכבת האפליקציה למבנה הנתונים בעזרת ה-LINQ

# תוכנית בדיקות תוכנה:

תוכנית הבדיקות נותנת את היכולת לבדוק את נכונות, עקביות,שרידות,יציבות המערכת.

תוכנית בדיקות התוכנה כוללת פירוט מלא של רכיבי המערכת לבדיקה בכל שלבי הפרויקט, תכונות המערכת שיבדקו בשלב הבדיקות, סביבה נדרשת לביצוע הבדיקות ומקרי בדיקה שונים למודולים המרכזיים של המערכת.

בתוכנית הבדיקות יש הגדרות של האובייקטים הנבדקים בשלב בדיקת התוכנה, תוך התייחסות לפי איזה קריטריונים לבדוק אותם.

פירוט מלא של תוכנית הבדיקות ניתן לראות בנספח ג' מסמך ה-STD.

# לוחות זמנים מעודכנים להמשך העבודה והערכת ביצוע המשימות:

לאחר מספר חודשי עבודה על הפרויקט , ישנם שינוים קלים בלוחות הזמנים עקב אילוצי עבודה ומשפחה

במסמך ההרחבה הרביעי - ה SPMP ישנה התייחסות לשינוי זה. הדבר לא יגרום לעיכוב בלוחות הזמנים המוגדרים והפרויקט יוגש בזמן.

1. **ניהול סיכונים מעודכן**

ישנם שינויים קלים בניהול הסיכונים וזאת באופן ישיר להתקדמות בפרויקט, כמובן גם סיכונים שהיו רלוונטיים לתקופת התיכון הראשונית שלפני הגדרת האב טיפוס או לימוד של חומר רלוונטי לפיתוח כבר פחות רלוונטים ולכן רמתם נמוכה יותר, כמו כן לאור לבטים בפיתוח המערכת , סיכון זה עלה בחשיבות

1. **אב טיפוס**

הצגת האב טיפוס הנבחר תציג באופן אנכי את אפשרויות המערכת תוך מתן דגש על

מודול ספציפי שיפותח באופן מלא.

האב טיפוס שתוכנן ובוצע במסגרת ההכנות לדו"ח זה, מציג את אופן הפעלת המערכת באופן   
 כללי .

האב טיפוס יציג את ממשק המסך הראשי של המערכת ואת תפריטי וכפתורי הכניסה לשאר   
 המודולים במערכת (ללא מימוש)

# ריכוז שינויים מדו"ח ביניים 1:

מלבד לוחות הזמנים שהתעדכנו, לא בוצעו שינויים בתכנון המערכת ובמימוש שהתבצע עד כה ומתכונן להמשיך להתבצע. עדכון לוחות הזמנים יוצג במסמך SPMP.

# רשימת מקורות :

כלל המקורות מדו"ח 1 רלוונטים וכן נוסף מקור המוצג מעלה בסעיף 4

1. [**מסמך הרחבה 1– Software Requirements Document - SRD**](#_Toc229729270)

**הקדמה**

* 1. **מטרת המסמך:**

מטרת המסמך היא לתאר את המערכת מבחינת מטרות ופונקציונאליות תוך כדי הגדרת המודלים השונים של המערכת ופירוט דרישות המערכת למרכיביה השונים.

* 1. **הגדרות** 
     1. DFD – Data Flow Diagram – תרשים זרימת הנתונים , זהו כלי בהנדסת תוכנה המציג בצורה גרפית את תהליכי עיבוד הנתונים המתרחשים במערכת המידע
     2. GUI – Graphic User Interface – ממשק משתמש – הוא החלק העיקרי והחשוף מבחינת המשתמש בתוכנה.
     3. C# - זוהי שפת תוכנה מונחית עצמים שתוכננה ופותחה על ידי מיקרוסופט בשנת 2000.
     4. SLA- Service Level Agreement - הסכם רמת שירות ,הוא חלק מחוזה שירות שבו שירות מוגדר באופן רישמי ,בדרך כלל משתמשים בשם זה כדי להתייחס לזמני תגובה המוגדרים בחוזה.
     5. MTBF – Mean Time Before Failure - זהו כלי מדידה שמראה עד כמה חומרה או תוכנה הם אמינים ,בדרך כלל לרכיבי חומרה זמן המדידה הוא במאות אלפי שעות, לדוגמא לדיסק קשיח יחושב זמן של 300,000 שעות.
  2. **מקורות:**

קיים במסמך מעלה , סעיף 12

* 1. **תיאור כללי:**
     1. **יחסים לפרויקטים נוכחים:**

לא קיים שום פרויקט בצורה ישירה או עקיפה הקשור לנושא זה.

* + 1. **יחסים בין פרויקטים קודמים :**

אין

* 1. **פונקציונאליות:**

למערכת תהיה את הפונקציונאליות הבאה:

* פיתוח אלגוריתם חכם לחישוב זמני SLA של רכיבי מחשוב בארגון המספק את השירות ,לדוגמא : דיסק קשיח, מדפסות , מעבדים וכדומה.  
  בעזרת האלגוריתם תתאפשר הצגת תמונת מצב עדכנית של כול הרכיבים הנבדקים בארגון המספק את השירות .

המערכת תחזיק תמונת מצב עדכנית אחרונה , האלגוריתם ידגום את כול אחד מהרכיבים המוגדרים.

התוצאות יוצגו על גבי מסך תמונת המצב האינטגרטיבית של המערכת.

אם במהלך ריצת המערכת יחול שינוי כולשהוא באחד הרכיבים או יותר, המערכת תציג את השינוי לגבי כול רכיב ורכיב .

* הצגת מצב רכיב לעומק - ניתן להתמקד על רכיב ספציפי על ידי לחיצה על הרכיב המסוים לאחר מכן יהיה ניתן לקבל עליו מידע רב יותר, כגון דגם הרכיב , החלקים השונים שמהם הוא מורכב , והבדיקות המבוצעות על ידי המערכת.
* הצגת חיזוי תקלות על פי נתוני עבר – פיתוח אלגוריתם שידע לחזות תקלות שיופיעו בעתיד על סמך נתוני העבר, חיזוי זה יוצג על גבי גרף ויוכל להציג על כול רכיב ורכיב את החיזוי לגביו.
* הצגת המדידה של SLA של רכיב בהשוואה ליתר הרכיבים במערכת.
* הצגת זמני תגובה של רכיב, רלוונטי רק לרכיבים מסוימיים כגון דיסק קשיח , גרפים אלו יציגו האם ישנה רגרסיה לינארית ברכיב המסויים .
* המערכת תנהל משתמשים והרשאות , ישנים שני סוגי משתמשים , רגיל או מנהל, משתמשים אלו יקבעו מראש.

לאחר הפעלת המערכת רק למשתמש מנהל תהיה האפשרות להוסיף משתמשים רגילים למערכת.

בנוסף רק למשתמש מנהל תינתן האפשרות לבצע שינויים מערכתיים כוללים: כגון טעינת קובץ חוזה שירות חדש, מחיקת משתמשים רגילים וכדומה.

* 1. **דרישות סביבתיות:**
     1. הפרויקט יעבוד על בסיס הרכיבים השונים הקיימים בארגון .

המערכת תקבל קובץ חוזה שירות ועל סמך רמות סף ודרישות הארגון תדע להתריע אם ישנה איזשהיא חריגה מההסכם שנחתם בין הארגון מקבל השירות לספק המספק את השירות.

משתמשי המערכת יהיו הגוף המנטר את המערכות השונות בארגון המקבל את השירות(בדר"כ מחלקת המחשוב בארגון). משתמשים אלו יוכלו להתריע מיידית לארגון המספק את השירות ובכך לקצר את זמן התקלה.

ההנהלה תסיק מסקנות באמצעות הניתוח שתבצע המערכת על השירותים השונים שהארגון מקבל מהספק.

המערכת אינה דורשת משאבים מיוחדים או ביצועים מיוחדים מעמדת הקצה עלייה היא מותקנת ועל כן תוכל לעבוד על כול מחשב נייח או נייד מה-5 שנים האחרונות.

המערכת תעבוד על סביבת עבודה:

חומרה:

* מעבד פנטיום 4 ומעלה .
* מינמום 2 גיגה זיכרון (בעיקר בשביל מערכת ההפעלה )
* דיסק קשיח של לפחותGB 80 (80 ג'יגה בית)

תוכנה:

* מערכת הפעלה Windows 7
* מסד נתונים SQL Express Server 2008
  1. **קשר למערכות אחרות:**

לא קיים קשר למערכות אחרות , כול הנתונים הנדרשים למערכת יוגדרו מראש בקובץ החוזה .

* 1. **אילוצים כללים:**
     1. **פיתוח:**

המערכת תבנה על בסיס כלים הקיימים כיום וניתנים לסטודנטים בחינם, הבסיס הוא C# תוך שימוש בטכנולוגיות חדישות ובסיס נתונים מוכר.

מלבד יתרון המפתח בהכרת כלים אלו וניתוח החלופות שנעשה, מנחה הפרויקט הגדיר לי את יכולות האפליקציה הדרושות והשאיר לי יד חופשית להחליט על דרך והכלים למימוש.

* + 1. **אילוצי זמן:**

הזמן הינו מרכיב קריטי מאוד בפיתוח הפרויקט , קיים לחץ גדול בפרויקט עקב אילוצי שונים כגון עבודה ומשפחה .

המטרה היא לממש את כול המודולים המתוכננים במהירות האפשרית על מנת להספיק לסיים את האפליקציה בזמן שהוגדר, בנוסף חשוב שהיה זמן נוסף על מנת לתקן באגים ושינויים שידרשו במהלך כתיבת הקוד.

* + 1. **חומרה :**

המערכת תרוץ על מחשבים סטנדרטים אשר קיימים כיום בשימוש בארגון , זמני התגובה של המחשבים הנ"ל יהיו מהירים יחסית, כאשר טעינת המערכת לא תערוך יותר מ-2-5 שניות , וכן הפעלת מודול תטען מיידית.

* + 1. **ניהול המערכת:**

המערכת תאפשר לנהל את המשתמשים, את הרכיבים הנבדקים ואת רמות הסף שנקבעו בקובץ החוזה.

המערכת תעמוד בדרישות אבטחת מידע והזדהות לכול משתמש במערכת , כאשר למשתמש מנהל ינתנו אפשריות נוספות של ניהול כגון : הוספה, מחיקה של משתמשים רגילים.

* + 1. **תפעול :**

ממשק המשתמש יהיה פשוט ונוח לשימוש, למרות שלמשתמשים ממחלקת ה- IT ישנו ניסיון מסויים בשימוש בתוכנות מחשב

* + 1. **עזרה למשתמש:**

יש לבנות מדריך בסיסי ביותר לשימוש המובנה בתוך המערכת , כאשר ישנה התייחסות לסוג המשתמש המחובר, כאשר למנהל ולמשתמש רגיל יהיה מדריך שונה .

* + 1. **ממשק המשתמש –GUI:**

המערכת נדרשת להיות אסתטית וידידותית למשתמש , נוחה לשימוש , ומגיבה במהירות רבה, כול משתמש יוכל להגיע לכול חלק מהמערכת על ידי מספר לחיצות קטן ככול שניתן.

* 1. **תיאור המודולים:**
     1. **ארכיטקטורה כללית:**

תרשים 1: ארכיטקטורה כללית של המערכת – כול הרכיבים המנוטרים מתחברים אל המערכת

* + 1. **ארכיטקטורת פיתוח:**



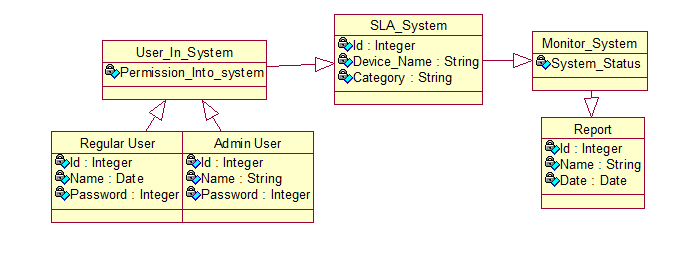
תרשים 2: ארכיטקטורת הפיתוח של המערכת.

UI- User Interface- השכבה המכילה את ממשק המשתמש, הצגת המערכת , מסכי המשתמש.

BL-business Logic - השכבה הלוגית הינה השכבה המקשרת ע"י פונקציות בין שכבת ה DAL לבין ה UI.

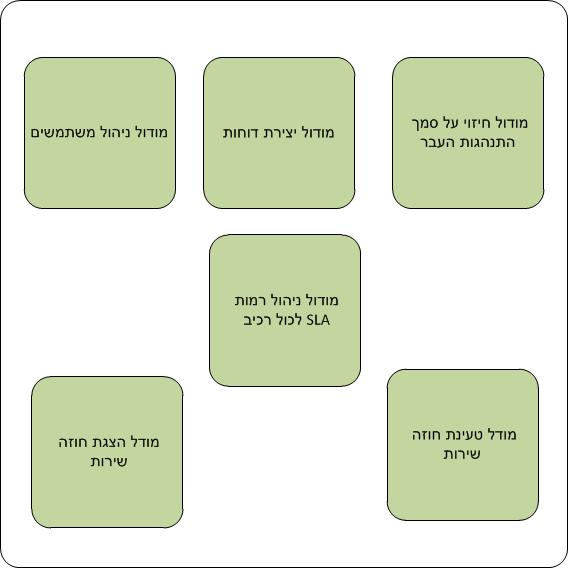
DAL-Data Access Layer – השכבה המכילה את הקשר לבסיס הנתונים ואל ישויות המערכת.

* + 1. **תרשים מחלקות –UML**



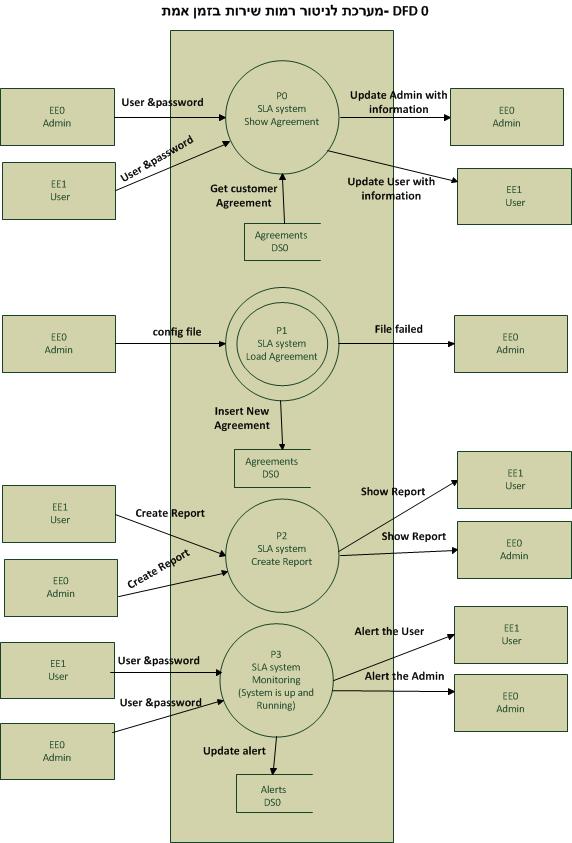
תרשים 3: מחלקות מרכזיות בפרויקט.

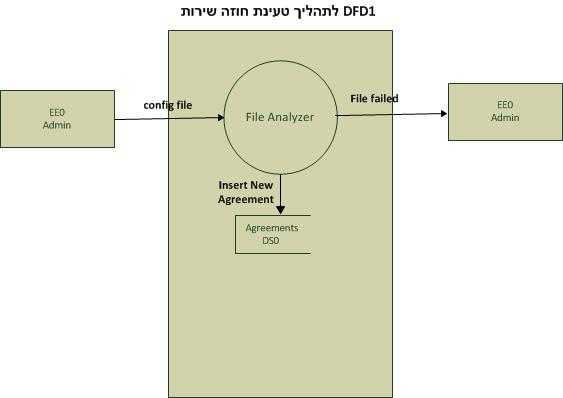
**מודלים מרכזיים במערכת לניטור רמות שירות בזמן אמת**



תרשים 4: מודולים מרכזים במערכת .

* + 1. **תרשימי DFD**





1. **דרישות ספציפיות:**
   1. **דרישות פונקציונאליות:**

**כללי:**

המודולים השונים במערכת נקבעים בעת טעינת חוזה השירות אך ישנם מודולים הדורשים תפעול ושינויים שונים שאותם יבצע מנהל המערכת(Admin )

* + 1. **אלגוריתם לקביעת רמת השירות לרכיבי המערכת המנוטרים**

במסגרת הפרויקט יפותח אלגוריתם אשר ידע לנטר בזמן אמת את הרכיבים השונים שהוטענו למערכת בשלב טעינת חוזה השירות שנקבע עם הספק.

* + - 1. בתחילה מנהל המערכת יתחבר בעזרת סיסמה אל המערכת ויטען קובץ חוזה שירות
      2. קובץ חוזה השירות יתווסף אל מבנה הנתונים לאחר בדיקת הקובץ ואימות על מבנה תקין.
      3. בשלב הבא מנהל המערכת יפעיל את המערכת על ידי לחיצה על כפתור ההפעלה
      4. המערכת תרוץ ועל סמך חוזה השירות שהוטען וזרם הנתונים שיוכנס למערכת המערכת תתריע אם ישנה חריגה כולשהיא במערכת.
    1. **ניהול משתמשים:**

מודול זה מאפשר למנהל המערכת להגדיר כמות משתמשים למערכת, לדוגמא אם המערכת מיועדת לחברה שהמשתמשים שלה עובדים במשמרות .

המודול יאפשר לערוך ולעדכן את נתוני המשתמשים במערכת , למחוק משתמש מהמערכת.

מודל זה מיועד בעיקר למנהל המערכת שמנהל את כול ההרשאות והמשתמשים במערכת.

* + - 1. תהליך קליטת משתמש : קיימת אפשרות לקלוט משתמש חדש, לעדכן משתמש ולמחוק משתמש.
      2. יש צורך להזין מספר עובד, תעודת זהות, שם פרטי, שם משפחה ,כתובת, תאריך לידה ,תאריך העסקה
      3. ניתן (לא חובה) לעדכן מספר שדות רשות : לדוגמא טלפון , תחביבים, וכדומה.
    1. **אלגוריתם חיזוי תקלות על סמך התנהגות העבר**
       1. במסגרת הפרויקט יפותח אלגוריתם אשר ידע לחזות תקלות שיקרו בעתיד על סמך התנהגות העבר
       2. בתחילת ריצת המערכת אין נתוני עבר והמערכת תתחיל לצבור נתונים מריצת המערכת .
       3. בשלב הבא לאחר שהמערכת צברה נתונים שונים המערכת תבצע חישובים ותבנה גרף שבו יהיה ניתן לראות כול כמה זמן המערכת "חושבת" שתקלה חדשה תצוץ במערכת
       4. יהיה ניתן לקבוע זמן מסויים (שעה) ולקבל חיווי האם תתקיים באותה שעה תקלה כולשהיא במערכת
    2. **ניהול רמות SLA לכול רכיב**

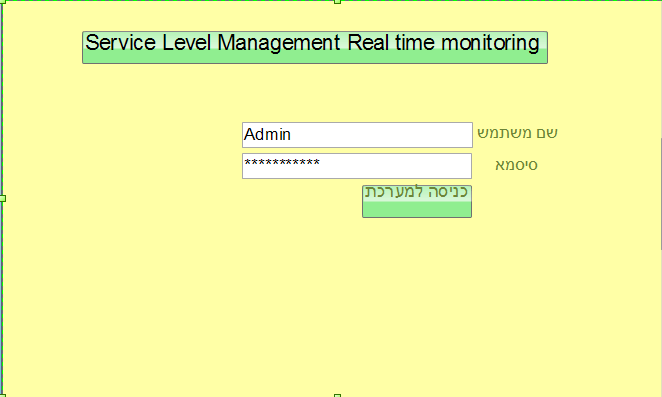
מודול זה מאפשר לקבוע לכול רכיב המנוטר במערכת רמת SLA שונה ,המשמעות היא שאם למשל הניטור מתבצע על מדפסת , אז ישנה אפשרות לנטר כמות דיו במדפסת באחוזים וגם האם קיימים דפים או לא .

לעומת זאת אם הניטור מתבצע על דיסק קשיח אז ניתן לנטר מהירות תגובה לפי זמן .

* + 1. **יצירת והצגת דוחות**

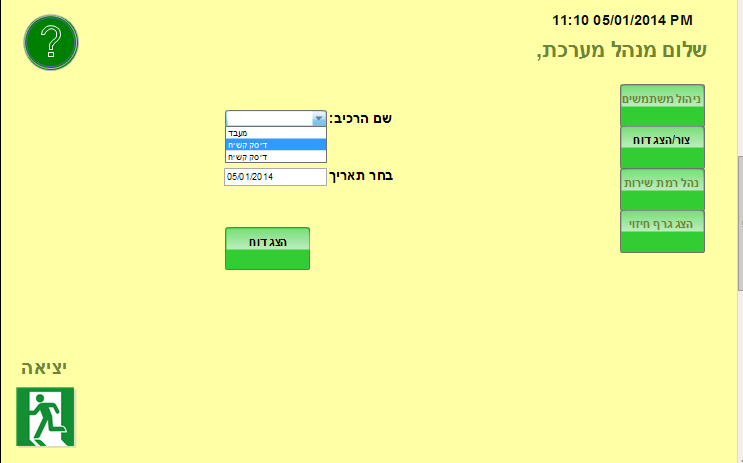
המערכת תצבור את ניטור הרכיבים במבנה הנתונים בזכות צבירת הנתונים יהיה ניתן להצג דוח לגבי כול רכיב ורכיב, וכך לקבל תמונה על הרכיב האם ישנה בעייה מסוימת איתו והוא מתקלקל לעיתים קרובות , או האם רמות הסף שנקבעו גבוהות או נמוכות מידי, בנוסף יהיה ניתן לזהות בזכות זאת צוארי בקבוק במערכת .

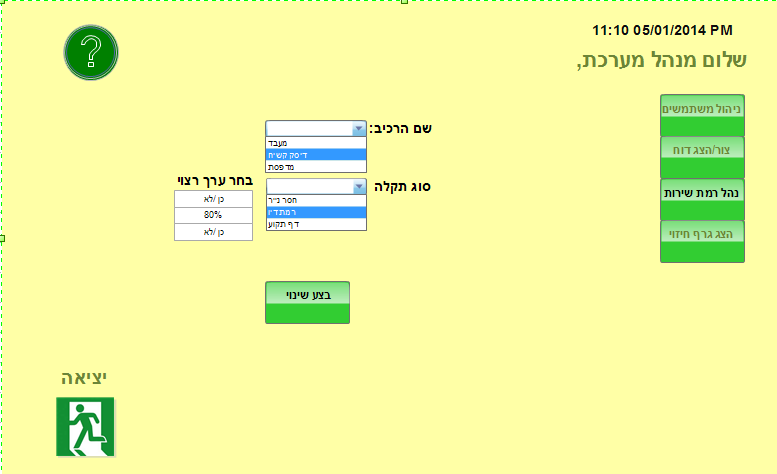
תרשים 5: פירוט מסכים במערכת . מצורף פירוט מסכים נכון לשלב זה של המערכת.

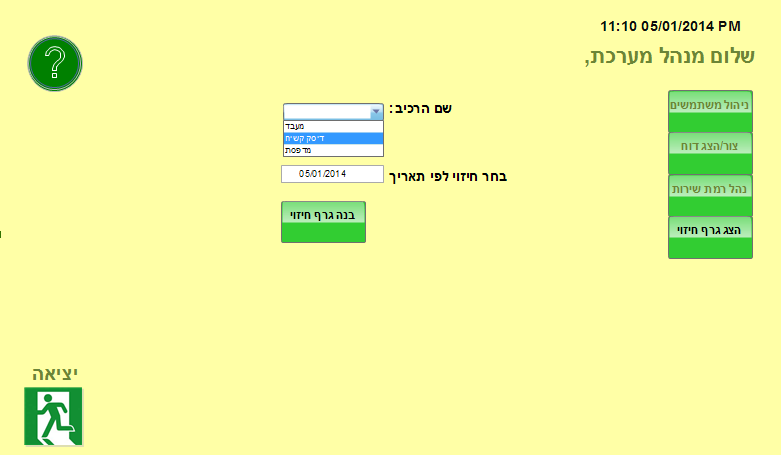












* 1. **דרישות ביצועים:**
     1. **ממשק גרפי וביצועי מערכת:**

הממשק הגרפי יהיה מהיר ככול הניתן וטעינת התוכנה לא תעלה על מספר שניות בודדות , מעברים בין המודולים השונים יבוצע במהירות על מנת לא לגרום למשתמש לחכות .

שליפת נתונים מבסיס הנתונים לא יעלה על מספר שניות בודדות.

* + 1. **אבטחה:**

כניסה למערכת תוכל להתבצע לאחר שהמערכת הופעלה ומסך של כניסה למערכת יוצג, מודול זה כי המשתמש קיים במערכת ובעל הרשאות גישה למערכת ויתנהג בהתאם להרשאות הספציפיות של המשתמש.

הסיסמאות של המשתמשים יוצפנו במבנה הנתונים על מנת למנוע מגורם לא רצוי לגשת אל המידע.

* + 1. **גיבוי:**

המערכת תבצע גיבוי לבסיס הנתונים אחת לזמן שיוגדר לקובץ מקומי על המחשב המריץ את המערכת באופן אוטומטי, בנוסף המערכת תתן את האפשרות למנהל המערכת לבצע גיבוי באופן יזום.

* 1. **דרישות משאבים:**
     1. **חומרה**
        1. מעבד פנטיום 4 ומעלה .
        2. מינימום 2 גיגה זיכרון (בעיקר בשביל מערכת ההפעלה ).
        3. דיסק קשיח של לפחותGB 80 (80 ג'יגה בית).
     2. **תוכנה:**
        1. מערכת הפעלה Windows 7.
        2. סביבת פיתוח Visual Studio 2010.
        3. בסיס נתונים SQL Express Server 2008.
        4. Microsoft Excel 2010 - המערכת תקבל קובץ חוזה בפורמט זה.
  2. **דרישות אבטחה**

נושא אבטחת המידע הוא נושא חשוב מאוד המעסיק את רובנו , אהיה מעוניין שרק למי שנדרש יהיו הרשאות גישה למערכת , ורק למספר מצומצם של משתמשים אפשרות לשנות ערכים קריטים במערכת שישפיעו על אופן עבודת המערכת והנתונים הקיימים בה.

אתייחס לשלושה היבטיים מרכזיים שעליהם נדרש להגן:

* + 1. **חיסיון המידע:**

המידע יהיה נגיש רק לגורם מורשה בלבד, ולכן כניסה למערכת תתבצע אך ורק באמצעות שם משתמש ייחודי וסיסמא.

* + 1. **זמינות המידע :**

מערכת המידע והמידע השמור בה יהיו בהתאם לרמת הזמינות, כלומר תתאפשר כתיבה ושליפת מידע אך ורק למשתמשים שהוגדרו לכך במבנה הנתונים,זאת אומרת שתהיה רמת גישה שונה לנתונים למשתמשים השונים.

* + 1. **שלמות ואמינות המערכת:**

המידע במערכת וכן המערכת באופן כללי הינה במצב מוגן בתוך הארגון בו פועלת המערכת.

לכול ארגון ישנה מבנה אבטחה משלו למידע הקיים בתוכו .כלומר פיירול וניטור הרשת על מנת לזהות ניסיונות פריצה , נקודה זו מבטיחה את אמינות ושלמות המידע.

[**מסמך הרחבה 2- Software Design Description**](#_Toc229729270) **SDD-**

1. הקדמה

1.1. מטרת המסמך

מטרת המסמך היא לתאר את המערכת מבחינת ארכיטקטורה ופונקציונאליות תוך כדי הגדרת המודלים השונים של המערכת ופירוט דרישות המערכת למרכיביה השונים. מטרת המסמך העיקרית הינה לתאר את תיכון, תכנון ועיצוב התוכנה.

1.2. הגדרות

אין תוספות למסמך SRD .

1.3. מקורות

מסמך זה מתייחס למקורות הקיימים במסמכים הנלווים שהם : SRD , STD ו-

תיכון ארכיטקטורת המערכת:

הארכיטקטורה שנבחרה הינה הארכיטקטורה עפ"י מודל השכבות שמתואר להלן:

* + - GUI Layer – השכבה שאחראית על התצוגה במסכי המערכת במחשב המשתמש . שכבה זאת תעביר את הנתונים לשכבה הלוגית לכל טיפול שלא נחשב בצד הוויזואלי של הממשק.
    - Business Logic/Business intelligent Layer – השכבה שאחראית על הצד הלוגי של המערכת, למעשה זוהי השכבה הבסיסית, בשכבה זו יתבצעו עיבודי הנתונים המגיעים מתצוגת המשתמש ,בנוסף היא אחראית לניהול המידע ומגשרת בין התצוגה ל –מבנה הנתונים.
    - DAL – Data Access Layer השכבה שאחראית לעבודה מול בסיס הנתונים שנבחר, בפרויקט יעשה שימוש ב Microsoft SQL Server EXPRESS 2008, פירוט בסיס הנתונים של הפרויקט נמצאת SRD.   
      שכבה זו תקבל נתונים או בקשות שונות משכבת ה Business Logic Layer ומול שכבה זו יתבצעו כל הפעולות שדרושות לעבודה עם מבנה הנתונים.



1. **ממשקי התוכנה:**

המערכת לניטור רמות שירות בארגון הינה מערכת המותקנת מקומית על המחשב בארגון שמעוניין לנטר את רמות השירות שהוא מקבל מהספק(נותן השירות).

הגישה אל המערכת תתבצע מהמחשב הספציפי שבוא הופעלה המערכת ,כניסה אל המערכת תתבצע על ידי הכנסת שם משתמש וסיסמה.

המערכת תכלול את המודולים הבאים אליהם תהיה גישה לאחר הכנסת שם המשתמש והסיסמה:

* מודול חיזוי על סמך התנהגות העבר.
* מודןל יצירת דוחות..
* מודול ניהול משתמשים.
* מודול ניהול SLA לכול רכיב.
* מודול טעינת חוזה שירות.
* מודל הצגת חוזה שירות.

לכול מודול יהיה ניתן להגיע מהמסך הראשי של המערכת, כאשר העכבר יהיה מעל אייקון המודול המערכת תציג מספר מילים על המודול הנוכחי, לחיצה על מודול כלשהוא מהמסך הראשי תעביר את המשתמש למסך הרלוונטי בהתאם לבחירה שהתבצעה.

**מסמך הרחבה 3 – STD – Software Test Documentation**

1. **הקדמה**

מסמך תכנון ועיצוב הבדיקות מוגדר לפי המודלים הקיימים במערכת.

לכול מודול תהיה סדרת בדיקות הרלוונטית לגביו ובסופו של דבר תתבצע סדרת בדיקות אינטגרציה למערכת כולה.

מטרתן של הבדיקות הללו הינו להבטיח את איכות התוכנה כולה בהתאם לסטנדרטים הקיימים בשוק.

הבדיקות יבדקו כי אכן כול הרכיבים עובדים כול אחד לחוד וכולם יחדיו כמערכת אחת. כמו כן הבדיקות יבדקו מקרי קצה המתאים לבדיקה הספציפית ואשר אמורים להיות מטופלים במסגרת התוכנה.

1. **פריטים שיבדקו:**
   1. **בדיקות ממשק משתמש**
      1. הרצת כול המסכים הקיימים והזנת נתונים ובדיקת תגובת המערכת בהתאם.
      2. בדיקת קישוריות לוגית בין כלל המסכים הנדרשים.
      3. בדיקות אינטואיטיביות הממשק ע"י ניסיון של מספר משתמשים.
      4. בדיקות פונקציונאליות על פי הגדרות.
   2. **בדיקת בסיס הנתונים:**
      1. ביצוע הכנסת חוזה גדול למבנה הנתונים ובדיקת עמידותו בעומס (ניתן גם להכניס חוזה קטן)
      2. ביצוע פעולות על מבנה הנתונים כגון עידכון ובדיקת תאימות הנתונים
      3. ביצוע פעולות מחיקה ממבנה הנתונים ובדיקת השפעת הפעולה על כול מבנה הנתונים.
2. **תכונות לבדיקה:**
   1. **ביצועים:**

בדיקת זמני עליית המערכת ע"י מספר יחידות קצה שונות, בדיקת תגובת המערכת לפעולות שונות ומדידת פרקי הזמן שלוקח למערכת להציג את הנתונים .

* 1. **נכונות:**

המידע המוצג על המסך שנשלף מבסיס הנתונים נדרש להיות נכון ולהציג את המידע שהוטען למערכת בצורה שלמה ומדוייקת.

* 1. **פונקציונאליות:**

בדיקה כי המערכת מבצעת את מה שהיא מיועדת לבצע בכול שלב הנדרש ממנה.

* 1. **אמינות:**

בדיקה שהמערכת עולה בכול פעם מחדש בצורה זהה , אין נפילה של המערכת, ועובדת בצורה סיסטמתית ללא סטיות בכול הפעלה מחודשת.

* 1. **הרשאות גישה:**

בדיקה כי משתמשים שונים יכולים לבצע פעולות שונות בהתאם להרשאות הגישה שניתנו להם.

* 1. **התממשקות:**

התממשקות כול רכיבי המערכת זה עם זה.

1. **תכונות שאינן לבדיקה**

* חומרה עליה רצה המערכת.
* אופטימיזציה של בסיס הנתונים.

1. **דרישות סביבתיות**

הבדיקות של המערכת יבוצעו באופן ידני מול מפרט בדיקות שמוגדר מעלה ויתאר את התוצאה הצפויה עבור כל אחד מהמודולים.

הבדיקות יתבצעו על מחשב בעל מערכת הפעלה WINDOWS 7.

הדרישות אינן דרישות מיוחדות היות ונדרשות לדמות עבודה אמיתית עם המערכת בהתאם לחומרה הקיימת כיום בקרב עמדת הקצה בארגון.

1. **אירועי בדיקה**
   1. **בדיקת נכונות המידע**
      1. **תכלית:** בדיקת מהימנות התקשורת עם בסיס הנתונים והצגת מידע באופן תקין על המסך.
      2. **קלטים:** 
         * משתמש רשום כבר במערכת.
         * משתמש לא קיים אשר נדרש תהליך רישום עבורו בעזרת מנהל המערכת.
      3. **צפי לפלט:** עבור משתמשים רשומים המערכת תכנס מיד ושמם יופיע מצד ימין למעלה, קיימת גם אפשרות לשמירת נתוני הכניסה.  
         עבור משתמשים שאינם רשומים – המערכת תבקש מהמנהל לרשום אותם למערכת כדי שיוכלו להיכנס .
      4. **שלבי הבדיקה:**
         * כניסה למערכת
         * הכנסת פרטי משתמש
         * קבלת הפלט הצפוי (כניסה למערכת או בקשה לרישום)
   2. **בדיקת שאילתות**
      1. **תכלית:** בדיקת כל סוגי השאילתות שיהיו אפשריות במערכת ותוצאותיהן
      2. **קלטים:** כל שאילתא אפשרית
      3. **צפי לפלט:** עבור כל שאילתא אפשרית במערכת הצפי לפלט הוא פלט תקין, כל שאילתא שמתאפשרת למשתמש דרך מסכי המערכת היא תקינה ועבור כל שאילתא כזו תתקבל תוצאה רצויה.
      4. **שלבי הבדיקה:**

* . בנית שאילתא במערכת
* הרצת השאילתא.
* קבלת פלט סטטיסטי מתאים.
* בדיקה מול מבנה הנתונים כי אכן תוצאות השאילתא נכונות.

* 1. **בדיקת עקביות**
     1. **תכלית:** בדיקת עקביות של המערכת – ווידוא שהמידע מהימןועקבי
     2. **קלטים:** שאילתות במערכת
     3. **צפי לפלט:** עבור אותה שאילתא הצפי הוא לפלט זהה לכל הרצה
     4. **שלבי הבדיקה:**
* הרצת השאילתא במערכת.
* שמירת התוצאות לצורך השוואה.
* הרצת אותה שאילתא פעם נוספת.
* השוואה לתוצאות הקודמות שהתקבלו.  
  1. **בדיקת ביצועים**
     1. **תכלית:** בדיקת עמידה בדרישות הביצועים של המערכת
     2. **קלטים: שאילתות במערכת**
     3. **צפי לפלט:** מענה על כל שאילתא לא עולה על מספר שניות על פי הגדרת דרישות
     4. **שלבי הבדיקה:**
* הרצת השאילתא על מבנה הנתונים ומדידת זמן עד לקבלת תוצאה חזרה.

1. **לוח זמנים**

שלב הבדיקות שכולל בדיקות מערכת ותיקוני באגים שהינה בדיקה מקיפה של כל רכיב נקבע בלוח הזמנים.   
חשוב לציין כי הבדיקות נעשות במשך כל תהליך פיתוח המערכת של הפרויקט שבו יבדק כל רכיב.   
בסיום בניית כל יחידה תתבצע בדיקה שתוודא תקינות ביצוע הפונקציונאליות שלה בהתאם לתוכנית.  
לאחר סיום הבדיקה תתבצע בדיקת אינטגרציה הכוללת את כל היחידות שבתוך אותו מודול בבדיקת המערכת יבדקו כל הסעיפים שנרשמו מעלה.

**מסמך הרחבה 4 – SPMP –Software Project Management Plan**

1. **מבוא:**

המסמך המוצג למטה מציג את הצד הניהולי של שלב התכנון והעיצוב של המערכת.

המסמך יכיל את נושא ניהול הסיכונים בטבלה המפורטת, בנוסף המסמך יגדיר תהליכים טכנייים וניהוליים הדרושים על מנת לספק מוצר אשר יעמוד בדרישות שהוגדרו.

1. **ניהול סיכונים:**

להלן מוצגת ניהול הסיכונים מדו"ח 1 עם שינויים קלים.

הסיכון – אי עמידה בלוחות הזמנים שהוגדרו עקב חוסר זמן התממש

מה שגרם לעיקוב בהכנת דוח 2 ובמימוש מלא של אבטיפוס

הפעילות להפחתה השפעתו של הסיכון השתנתה – עידכון המנחה על בסיס שבועי על התקדמות הפרויקט,

והגדלת כמות הפגישות על מנת לא לחרוג מהזמנים שוב.

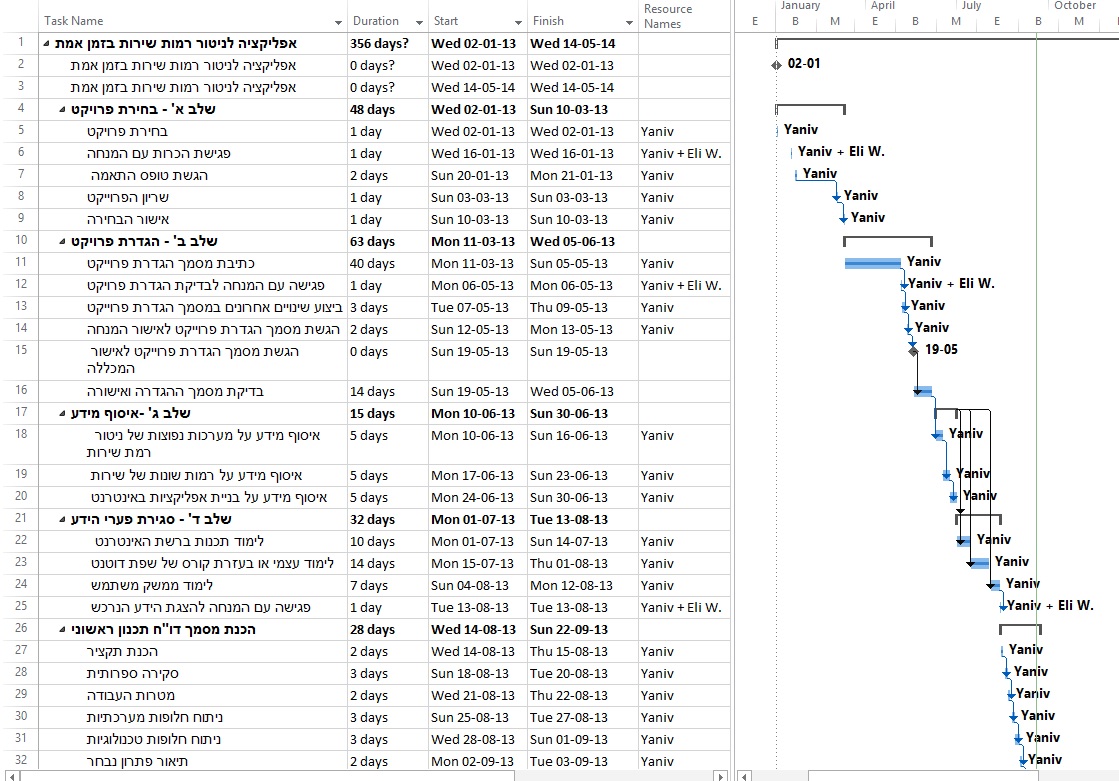
|  |  |  |  |
| --- | --- | --- | --- |
| **הסיכון** | **הסבירות להופעת הסיכון** | **חומרת הסיכון** | **פעילויות להפחתת השפעתו של הסיכון** |
| אי עמידה בלוחות זמנים שהוגדרו ,חוסר זמן עקב עבודה וילדים | 80% | 80% | בדיקה שבועית מול לוח הזמנים שהוגדר והזמנים שהוגדרו בנוהל הפרויקטים , עידכון המנחה על בסיס שבועי על התקדמות הפרויקט והגדלת כמות הפגישות |
| ממשק משתמש שאבנה לא יעמוד בדרישות הלקוח | 30% | 50% | אאכוף זאת על ידי סקרים שאעשה , אתן לאנשים מהמכללה להשתמש בתוכנה ולאחר מכן אותם אנשים ימלאו משובים שאליהם אתייחס על מנת לשפר את ממשק המשתמש |
| בחירת שפת תיכנות  לא נכונה כגון NET במקום Java | 40% | 30% | לימוד מעמיק של סביבת הפיתוח ויכולותיה. לחקור ממשק משתמש(פקדים , חלונות וכדומה) , מסד נתונים ודרכי התממשקות |
| אי הצלחת הבנת המערכות הקיימות כגון Oblicore | 50% | 80% | אחקור חלק מהמערכות הקיימות על ידי התקנת גרסאות ניסיון, והתייעצות עם המנחה, |
| אי הצלחה בכימות הטפסים לנתונים הניתנים לבדיקה | 60% | 70% | חשיבה עם המנחה על דרכים לקבלת מידע אמיתי מהשוק וחשיבה כיצד לכמת את הטפסים האמיתיים |
| אי הבנת האופן בו המערכות מנטרות בזמן אמת את התקלות | 70% | 70% | ביצוע לימוד מעמיק בנוגע לאופן פעולת הניטור של התקלות במערכות הקיימות |

1. **הערכת לוחות הזמנים:**

* הפרויקט בשלב זה עומד כמעט במלואו בלוחות הזמנים., חלו שינויים בהגשת דוח תכנון ראשוני בעקבות בעיות אישיות והתקבל שבוע דחייה ממחלקת הפרוייקטים.
* חלו שינויים במועד הגשת ספר הפרויקט ותערוכת פרויקטים בעקבות עידכון של מחלקת פרוייקטים.
  + חלק א' של הפרויקט - עד הגשת הגדרת הפרויקט. **בוצע**
    - פגישה עם המנחה.
    - כתיבת הגדרת פרויקט.
    - השלמת סעיפים חסרים בהגדרת הפרויקט (תיאום עם המנחה).
    - קבלת אישור המנחה להגשת הגדרת הפרויקט.
    - הגשת הגדרת הפרויקט.
  + חלק ב' של הפרויקט(עד הגשת דוח ביניים 1). **בוצע**
    - פגישה עם המנחה.
    - הרחבת אפיון וניתוח המערכת.
      * ניתוח מעמיק של החלופות המערכתיות.
      * הגדרת מודולים, תהליכים וקשרים במערכת.
    - כתיבת דוח ביניים ראשון.
      * כתיבת הפרקים המתבססים על אפיון המערכת.
    - הערכת ביצוע משימות ותיקון לוח הזמנים.
    - קבלת אישור המנחה להגשת דוח ביניים 1.
    - הגשת דוח ביניים 1.
  + חלק ג' של הפרויקט (עד הגשת דוח ביניים 2 ואב טיפוס). **בוצע כמעט במלואו**
    - כתיבת דוח ביניים 2.
    - כתיבת דוח ביניים 2 (בתיאום עם המנחה).
    - היערכות ביצוע משימות ועדכון לוח זמנים.
    - בניית אב טיפוס(נשאר עוד עבודה של שבועיים להשלמה מלאה).

* 1. מנקודה זו ועד סיום הפרויקט נותרו 2 החלקים הבאים:
  + חלק ד' של הפרויקט + ספר פרויקט.
    - סיום בנית הפרויקט.
      * סיום כתיבת קוד הפרויקט.
      * בדיקות מערכת.
      * בדיקות תוכנה.
      * קבלת אישור המנחה על סיום הפרויקט.
    - כתיבת ספר פרויקט.
      * השלמת פרקים חסרים בספר.
      * החתמת המנחה על הספר.
  + הגשת פרויקט + ספר.

1. **לוחות זמנים:**







****

**המחלקה להנדסת תוכנה**

שם הפרויקט:

אפליקציה לניטור רמות שירות בזמן אמת

Project Name:

Service Level Management Real time monitoring

הגדרת פרויקט Statement Of Work -

|  |  |
| --- | --- |
| שם הסטודנט: | יניב נחום |
| מספר ת.ז. | 021469408 |
| שם המנחה: | ד"ר אלי וינטראוב |
| תאריך ההגשה: | 19.05.2013 |

# תוכן עניינים

[1. תוכן עניינים 2](#_Toc229729266)

[2. מבוא 3](#_Toc229729267)

[3. מטרות יעדים מדדים 4](#_Toc229729268)

3.1 מטרת הפרויקט.................................................................................4

3.2 יעדי הפרויקט....................................................................................4

3.3 מדדים..............................................................................................4

[4. סקירה ספרותית 5](#_Toc229729270)

[5. סקירת מצב קיים 6](#_Toc229729270)

[6. מסמך דרישות ראשוני 8](#_Toc229729270)

6.1 דרישות פונקציונאליות........................................................................8

6.1.1 תחום העבודה...............................................................8

6.1.2 תחום המוצר.................................................................8

6.1.3 דרישות מידע ופונקציונאליות............................................8

6.2 דרישות לא פונקציונאליות....................................................................8

6.2.1 דרישות עיצוב.................................................................8

6.2.2 דרישות שימוש וממשק משתמש........................................8

6.2.3 דרישות ביצועים..............................................................8

6.2.4 דרישות הפעלה..............................................................9

6.2.5 דרישות תחזוקה ותמיכה..................................................9

6.2.6 דרישות אבטחה..............................................................9

6.2.7 בטיחות..........................................................................9

[7. ניתוח פונקציונליות ראשוני](#_Toc229729272) 10

7.1 תרשים מודולים כללי של המערכת........................................................10

7.2 UML.............................................................................................11

7.2.1 Main UseCase(Admin).................................................11

7.2.2 Main UseCase(User)....................................................11

7.2.3 System Found Exception and Alert..............................11

7.2.4 Overall System Status..................................................11

[8. הצגה ראשונית של חלופות מערכתיות](#_Toc229729273) 12

[9. תיאור דרך הביצוע המתוכנן](#_Toc229729274) 12

[10. האמצעים הכלים הדרושים](#_Toc229729275) 14

[11. פערי ידע שעל הסטודנט להשלים](#_Toc229729276) 15

[12. תוצרי הפרויקט](#_Toc229729277) 15

12.1 דו"ח ביניים 1................................................................................15

12.2 דו"ח ביניים 2 ...............................................................................16

12.3 ספר הפרויקט...............................................................................16

12.4 תוצרי סופי....................................................................................16

[13. בסיס לתוכנית בדיקות](#_Toc229729280) 16

[14. ניהול סיכונים](#_Toc229729297) 17

[15. חובות יזמיות](#_Toc229729298) 17

[16. תוכנית ניהול הפרויקט](#_Toc229729298) 18

16.1 תוכנית עבודה של הפרויקט................................................................18

[17. רשימת מקורות](#_Toc229729299) 20

# מבוא

**2.1 הקדמה**

פרוייקט זה יציג אפליקצית ניטור בזמן אמת של שירותים הניתנים לאירגון בהתאם להסכם הקיים עם הספק.

**2.2 מבוא**

כאשר ארגון קונה שירותים מספק, הספק בדרך כלל מתחייב לרמות שירות שונות: הן ברמה הטכנולוגית (לדוגמא: האם קווי האינטרנט, אחסון האתר וקווי הטלפון זמינים); הן בזמני תגובה לטיפול ראשוני; והן בהמשך הטיפול עד סגירת התקלה (SLA).

רוב החברות שמספקות שירות משתמשות בשיטה ידנית לניהל ולדווח על התחייבות רמת השירות. שיטה זו יקרה ובעלת נטייה לשגיאות רבות, כמו כן, היא מובילה לתרבות של סכסוך בין ספק השירות ללקוחותיו.

כיום ישנן חברות כדוגמאת oblicore, שנרכשה על ידי חברת CA, המציעות מערכת תוכנה שבודקת את רמות השירות המתקבלות מהספק. התוכנה בעיקר מסייעת ללקוחות לוודא שהשירות שהם מקבלים מחברות הטלקום וספקיות האינטרנט הוא אכן כפי שהובטח. החברה מספקת כלי תוכנה לשני הצדדים המוודאים שאכן השירות הוא מה שהוסכם ונדרש.

בפרויקט זה נייצר כלים אשר יאפשרו להציג נתונים על רמת השירות בזמן אמת ובכך תגדל שביעות רצון הלקוחות אשר קונים מוצר ושירות מספק. הלקוחות יראו בזמן אמת שהכסף שהם משלמים אכן שווה את ההשקעה. בנוסף, לקוחות שהינם Resellers, יוכלו להתחייב ללקוחות שלהם על רמת שירות מסוימת.

# מטרות יעדים ומדדים

**3.1 מטרת הפרויקט:**

מטרת פרוייקט זה היא בניית אפליקציה המבצעת ניטור בזמן אמת של השירותים הניתנים לארגון בהתאם להסכם שקבעו עם הספק. האפליקציה תתריע ותדווח על עמידה/חריגות מההסכם בזמן אמת.

**3.2 יעדי הפרויקט:**

* אינדיקציה וזיהוי מהיר של חריגה מזמני תגובה הכתובים בהסכם.
* קיצור משך טיפול בתקלות.
* צמצום טעויות אנוש. (זיהוי אוטומטי אמין יותר מאדם)
* פיתוח ממשק משתמש נוח ופשוט לשימוש למנהלים.
* הפקת דוחות התואמים את כמות התקלות ומשך הטיפול בהם.

**3.3 מדדים להצלחת הפרויקט:**

* התראה על 95% מחריגות בהסכם שנחתם.
* קבלת חווי על תקלה כלשהיא במערכת בתווך זמן שלא יעלה על 5 שניות מרגע התקלה.
* יבוצע Benchmark בין הפרויקט המבוצע לתוכנה מקבילה בשוק, תוצאות ה-Benchmark יצביעו על שיפור בזמנים בין הפרויקט המוצע לתוכנה הקיימת בשוק.

1. **סקירה ספרותית**

* **Discussion Group, Cloud Computing Use Cases,Cloud Computing use Case, [1]**

**2010**

בשנים האחרונות חלה עלייה משמעותית בשימוש במחשוב בענן. בעבר כול אירגון היה בונה לעצמו חדר מחשב שבו היו נמצאים כול כוח המחשוב של הארגון. בכדי לתחזק חדר מחשב היה צורך באנשי IT בעלי ידע רב שעולים המון כסף לארגון. בימינו ארגונים רבים הבינו את הכוח העצום בשימוש במחשוב ענן ואת החסכון הכספי ששירות זה מספק. המאמר *Cloud Computing Use Cases* בא לאגד את הדרישות ואת ה- SLAשבהם נותן השירות צריך לעמוד בהם כדי שיהיה ניתן להשתמש באופן תקין בשירות מסוג מחשוב ענן.

* **[2] Walker G. , IT Problem Management, Prentice Hall PTR,2001**

ארגונים רבים מחפשים דרכים רבות להתייעל ולספק שירותים באופן רציף ומיטבי ללקוחותיהם. הספר *IT Problem Management* מציג את העקרונות לפתירת בעיות בארגון בצורה מהירה ומקצועית, ובנוסף מציג איזה מידע צריך להכיל ארגון שעובד עם SLA .

* **[3] Baudoin C,Flynn J,McDonald J,Meegan J,Salsburg M & Woodward S, Public Cloud Service Agreements What to Expect and What to Negotiate,** **Cloud Standards Customer Council,2013.**

מאמר זה מספק לצרכני שירותי ענן גישה פרגמטית כדי להבין ולהעריך את הסכמי שירותי ענן ציבוריים.

* **[4]** **Xiong K & Perros H, SLA – Based Allocation Resource in Cluster Computing,** [**Parallel and Distributed Processing**](http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=4519061)**,2008**

מאמר זה מספק גישה להקצאת משאבי מחשב בסביבה כזו שמצמצת את העלות הכוללת של משאבי מחשב. כל זאת, באיכות שירות שהוגדרה בהסכם רמת השירות מול הלקוח. בנוסף, מוצגים מדדים לחישוב כגון זמן אחוזון תגובה, ניצול אשכול מחשוב, שיעור אובדן מנות וזמינות אשכול מחשוב

* **[5] Hiles A, The Complete Guide To IT Service Level Agreements ,Third Edition , Rothstein Associates,2002**

לפני יותר מ-15 שנה לא היה כלי אסטרטגי שעזר לאגד את רמת השירות והתמיכה בלקוחות. ספר זה מציג את נקודת מבטו של הכותב שכבר אז זיהה את הצורך בכלי.

* **[6] Blokdijk G,The Service Level Agreement SLA Guide,Emereo,2008**

ספר זה מציג מפת דרכים טכנית ומקצועית לניהול רמת שירות, ההסכמים, תכנון השירות, שיפור מתמשך של השירות, והצעת תבנית של שירות להטמעה באירגון.

1. **סקירת מצב קיים**

כיום, מספר הכלים המספקים פתרון לניטור רמת השירות המתקבלת מספק השירות הוא דיי גדול ואף קיים במגוון תחומים. כלים אלו יודעים למנף את המידע מהסביבה המבצעית, מבינים כיצד מידע זה מנוצל במדידות ביצועים, ומשווים ביצוע כנגד התחייבות שמעשתה בין לקוח לספק. כתוצאה מכך, ארגון יכול לנהל באופן פרואקטיבי חוזים כגון הסכמי רמת שירות. בעזרת כלים אלו, יכול הארגון להפחית את כמות המאמץ הנדרש בדיווח על קשרים אלו ולשפר את שביעות רצון לקוחותיו.

דוגמאות לכלים מסוג זה הנמצאים כיום בשוק:

**Oblicore Guarantee** – חברת Oblicore נקנתה על ידי חברת CA ונחשבת כבמובילה בתחום ניטור רמת השירות. הכלי שאותו מציעה מפעיל ומאיץ את הניהול, הניטור והדיווח של כול הסכמי רמת השירות ואספקת שירות עבור ארגונים וספקי שירותים. הפיתרון של חברה זו, משפר את יכולתו של הארגון (המקבל שירות מספק) בדברים הבאים:

* נותן אמצעי לאגד את כול הנתונים בתחום השירות ונותן מדדים למדידה.
* נותן אפשרות להקים התחייבות חוזית ולקבוע יעדי ביצועים.
* עוקב אחר הביצועים כנגד המטרות שהוגדרו בזמן אמת ונותן אפשרות לנקוט בפעולות על סמך ביצועים אלו.
* מספק דוח על ביצועים לשני הצדדים בהסכם, הן לספק השירות והן לצרכן השירות.

<http://www.ca.com/~/media/Files/ProductBriefs/1599-oblicore-guarantee-pb-042710.pdf>

**Maximo SLA Manager** – חברת IBM היא חברה ידועה שמספקת פתרונות של חומרה ותוכנה. מוצר זה שמציעה הינו תוכנה המסייעת בהגדרת הסכמי רמת שירות, עוזרת ליישם נהלי דיווח ולפקח על אספקת רמת השירות. הפתרון של חברה זו מספק את הדברים הבאים:

* נותן אפשרות להגדיר מספר גופים לשם דיווח במקרה הצורך במטרה למנוע אירועים לפני שהם קורים. ניתן גם להגדיר למי להודיע ואיך להודיע.
* מציג ממשק גרפי שנותן תמונה ברורה של המצב הנוכחי ומציג מדדי ביצועים.
* מספק תבניות תקשורת הנועדו לספק תקשורת מובנית, על מנת להבטיח שמידע קריטי ישותף מבעוד מועד, להקטין אי הבנות ואירועים המשפיעים על השירות.
* תומך במסגרות ITIL – (Information Technology Infrastructure Library) על מנת לקשור הסכמי רמת שירות לתהליכי ITIL המאפשרים ניהול הדוק של תצורות שינויים, מהדורות, בעיות ואירועים.

<http://www-01.ibm.com/software/tivoli/products/maximo-sla-mng/>

**IT SLA Management Software** – חברת Up time Software היא חברה שמספקת תוכנות לניהול מערכות של שרתים וירטואליים או שרתים פיזיים וניטור אפליקציות. הפיתרון שלה מספק את הדברים הבאים:

* נותן לוח מחוונים שמציג נתונים על סטטוס המערכת בזמן אמת.
* מספק אפליקציה בשימוש במכשירים סלולריים וכך עוזר להיות תמיד בשליטה.
* נותן דיווח מפורט של ההשפעה של כל אחד ממרכיבי התשתית שמשפיעים של סיפוק השירות בזמן.

<http://www.uptimesoftware.com/sla-management.php>

האפליקציה המוצעת בפרויקט זה בעלת יכולת התאמה מלאה לצורכי הלקוח ויכולת גמישות בהגדרת סעיפי הסכם השירות איתו. ביכולתה של האפליקציה להביא בזמן אמת עדכונים על מצב המערכת והתרעה על חריגות ביחס להסכם.

**6.** **מסמך דרישות ראשוני**

**6.1 דרישות פונקציונאליות**

**6.1.1 תחום העבודה**

הבטחת איכות שירות ללקוחות וניהול פרויקטים

**6.1.2תחום המוצר**

הפרויקט מיועד לחברות הקונות שירותים מספק מסוים ורוצות להבטיח קבלת שירות הולם העומד ביעדים אשר הוגדרו לספק ואשר הספק התחייב לספקם.

**6.1.3 דרישות מידע ופונקציונליות**

* בהפעלת התוכנה המערכת תבקש להזין שם חברה לשם בדיקת רמת השירות
* האפליקציה תקבל קובץ עם פרטי החוזה שמכומת לערכים הניתנים לבדיקה
* האפליקציה תנתח את הנתונים ותמיר את הקובץ שהוכנס לתצורת הרצה
* בעת לחיצת "הפעל" המערכת תתחיל לדגום את הנתונים ותתריע כאשר ימצאו חריגות
* באותו הזמן המערכת תרשום נתונים לקובץ לוג שבו תיתן זרימת נתונים שהתקבלו
* האפליקציה נותנת אפשרות לעצירה או לביטול הבדיקה
* האפליקציה תציג תמונה דינאמית בזמן אמת אשר תשתנה בהתאם לנתונים הזורמים לתוכה
* עם הפסקת האפליקציה יהיה ניתן לקבל תיעוד מסודר של הבדיקה

**6.2 דרישות לא פונקציונליות**

**6.2.1 דרישות עיצוב**

* המערכת צריכה להציג באופן ברור את מדדי המערכות.
* .המערכת צריכה להציג באופן גרפי סטטוס מערכות עדכני.

**6.2.2 דרישות שימוש וממשק משתמש**

* המערכת תודיע על כל מצב בו היא נמצאת – ריצה, עצירה, זיהוי חריגות.

**6.2.3 דרישות ביצועים**

* המערכת תזהה חריגות ב-95% בהסכם שנחתם.
* המערכת תספק מידע אמין ומדויק בלמעלה מ-96% מהשימוש בה.
* המערכת תהיה זמינה ב-99.8% מהזמן.

**6.2.4 דרישות הפעלה**

* אין למערכת דרישות הפעלה מיוחדות. ממשק המשתמש יהיה ידידותי לכל משתמש בארגון, ואילו ממשק הניהול יהיה ידידותי למשתמש בעל ידע בסיסי בתחום המחשבים והמערכות המופעלות בארגון .

**6.2.5 דרישות תחזוקה ותמיכה**

* לאחר מסירת המערכת לא דרושה עבודת תחזוקה נוספת.

**6.2.6 דרישות אבטחה**

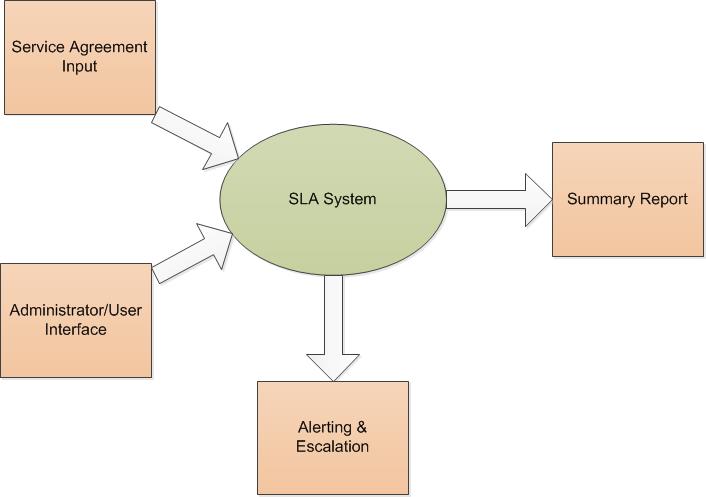
* למערכת יהיו דרישות אבטחת מידע שיבואו לידי ביטוי בכניסה למערכת בעזרת שם משתמש וסיסמה. בנוסף, ינוהלו הרשאות למשתמשים שונים וכך רמת מנהל תהיה שונה מרמת משתמש רגיל.

**6.2.7 בטיחות**

* לא רלוונטי לפרויקט שלי משום שבמערכת לא מוגדרים גורמי סיכון בטיחותיים.

# ניתוח פונקציונאלי ראשוני

**7.1 תרשים מודלים כללי של המערכת**



**7.2 UML**

7.2.1

Main UseCase(Admin)

****

7.2.2

Main UseCase(User)

****

7.2.3

System Found Exception and Alert

****

7.2.4

Overall System Status



# הצגה ראשונית של חלופות מערכתיות

|  |  |
| --- | --- |
| חלופה | תיאור החלופה |
| Centralized Remote Access | הגישה לשירות זה מתבצעת במקום מרכזי אחד. כניסה מתבצעת למערכת מרחוק ומשקפת את המצב הנוכחי של המערכת. |
| Agent Base | שירות זה מתבצע על ידי התקנה של Agent בכול השירותים שאותם אנחנו רוצים לנטר. כל Agent ידווח למיקום מרכזי. |
| Local Access | הגישה לשירות היא מקומית . בדיקת מצב המערכת מתבצעת פיזית על ידי המשתמש. |

1. **תיאור דרך הביצוע המתוכננת**
   1. **שלב ראשון:** **ביצוע מחקר עבור הנושאים הבאים:**

* סקירה רחבה של נושא הסכמי רמות שירות הקיימות בספרות המדעית ובארגונים שונים, כגון oblicore.
* סקירת התוכנות הקיימות בתחום ומופעלות בארגונים שונים.
  1. **שלב שני: תיכון המערכת**
* מיפוי קשיים ואתגרים בפיתוח הכלי.
* עיצוב ממשק משתמש – הגדרת ממשק משתמש ועיצוב כללי.
* זיהוי וחלוקת המערכת לתתי מערכות:
* מערכת המרת קובץ הסכם שירות הנחתם בין שני הצדדים לסביבת הרצה אוטומטית של הסכם שירות.
* בסיס נתונים לשמירת נתונים והפקת דוחות.
  1. **שלב שלישי: פיתוח כללי**
* פיתוח שכבת ה – GUI - ממשק האפליקציה.
* פיתוח המערכת – המרת קובץ הסכם של דרישות לסביבת הרצה אוטומטית של הסכם השירות.
* אינטגרציה עם בסיס הנתונים.
  1. **שלב רביעי: בדיקות**
* בדיקות עבור זיהוי נכון וקריאה נכונה של קובץ הסכם השירות.
* בדיקות נכונות ניתוח קובץ הסכם השירות – אימות הנתונים שמפיקה המערכת.
* בדיקת נאמנות הביצועים לדרישות הפרויקט.
* בדיקת הצגת התוצאות.
* בדיקת ה-GUI האפליקציה – בדיקת כול הפקדים והתפריטים הקיימים במערכת.

# האמצעים/הכלים הנדרשים, חומרה ותוכנה

* 1. **תוכנות\סביבת פיתוח**
     + Microsoft visual studio 2010/2012 או Eclipse IDE for Java Developers Juno
     + MS-Project 2010-2013
     + IBM® Rational Rose– לבניית תרשימי UML
     + Microsoft office – Visio 2010-2013 – לבניית תרשימי DFD
     + תוכנות Microsoft Office עם עדיפות לגרסאות 2010 /2013
     + [Subversion/ tortoise svn](http://tortoisesvn.net/) או GIT – ניהול גרסאות קוד
  2. **חומרה**
     + מחשב הפיתוח – AMD Dual Core 2.2 GHZ ,2 GB RAM
  3. **סביבת ריצה**
     + מחשב PC - Desktop או Laptop
     + מערכת הפעלה Windows 7/8

# פערי ידע של הסטודנט

* ניהול פרויקט – זהו פרויקט ראשון בסדר גודל שכזה שאותו אצטרך לנהל ולהוביל ועליי להרחיב את אופקי בנושא.
* תכנות ברשת NET - קורס הנלמד בסמסטר קיץ 2013 במידה ואראה ש-Java לא מתאים.
* לימוד תכנות ברשת האינטרנט
* הבטחת איכות תוכנה - קורס הנלמד בתחילת פרויקט זה בסמסטר ב' 2013
* לימוד עבודה עם Database לשליפה ואחסון נתונים רלוונטים לפרויקט

# תוצרי הפרויקט

* 1. דו''ח ביניים 1

### מטרות הפרויקט.

* 1. **סקירה תיאורטית על אופן פעולת המערכת.**
  2. **בדיקת חלופות למימוש הפרויקט.**
  3. **תוכנית עבודה מעודכנת.**
  4. **ניתוח פונקציונאלי מעודכן.**
  5. **ניתוח חלופות טכנולוגיות.**

**נספחים**

* 1. **תיאור הפתרון הנבחן -**SRD
  2. **תיאור בדיקות תוכנה –**STP
  3. **לוחות זמנים** -SPMP
  4. דו''ח ביניים 2

1. **תחילת פיתוח המוצר**
2. **הפקת מסמכים**
3. **בדיקות תוכנה**

**נספחים**

1. **הפתרון הנבחר –** SDD
2. **לוחות זמנים והערכת ביצוע המשימות –**SPMP
3. **תוכנית בדיקות תוכנה -**STD
   1. ספר הפרויקט
4. **סיכום את דו"חות הביניים .**
5. **שילוב דרישות מערכת ואפיון מערכת.**
6. **ארכיטקטורת המערכת.**
7. **דווח תוצאות בדיקות תוכנה.**
8. **דוגמאות להפעלת המערכת.**
9. **תיאור ביצועים עלות ואמינות.**
10. **תיאור הצלחת המערכת או אי ההצלחה.**
11. **איגוד כלל המסמכים שנעשו במשך העבודה על הפרויקט.** 
    1. תוצר סופי

* **אפליקציה ברשת, שתבצע את כל המתוכנן לגבי ניטור שירות בזמן אמת.**

# בסיס לתוכנית בדיקות

13.1 **בדיקות תוכנה**

1. בדיקת פעילות התוכנה בהתאם לדרישות.
2. בדיקת תוכנה על כול מכלול בנפרד ורק לאחר מכן על המערכת בשלמותה.
3. ביצוע ניסויים ובדיקה שחישוב המדדים מתבצע באופן נכון ומדויק.

# ניהול סיכונים

|  |  |  |  |
| --- | --- | --- | --- |
| **הסיכון** | **הסבירות להופעת הסיכון** | **חומרת הסיכון** | **פעילויות להפחתת השפעתו של הסיכון** |
| אי עמידה בלוחות זמנים שהוגדרו | 15% | 70% | בדיקה דו שבועית מול לוח הזמנים שהוגדר ומול הזמנים שהוגדרו בנוהל הפרויקטים |
| בעיות בפיתוח האלגוריתם הנדרש למערכת | 25% | 70% | פיתוח כבר בשלבים הראשונים של הפרויקט ע"מ לוודא שאכן עובד |
| פערי ידע לביצוע הפרויקט | 20% | 35% | למידה עצמית במהלך סמסטר ב' וסמסטר קיץ את הנדרש לפרויקט |
| גורמים אנושים | 20% | 55% | ניתן להתחשב מראש ולתכנן סיום כל שלב לפני הזמן המחויב. |

השיקולים לבניית טבלת ניהול הסיכונים: ניהול סיכונים הוא כלי מרכזי וחיוני ביותר עבור מנהל הפרויקט ועבור כלל הגורמים המעורבים בפרויקט. כלי זה מסייע באיתור נקודות התורפה וכשלים פוטנציאלים העשויים לגרום לחריגות משמעותיות בלוחות הזמנים, בעלויות הכספיות שנקבעו ואף לגרום לכשלון הפרויקט.

בעבודה זו אני השתמשתי במתודולגיה של נוהל מפת"ח ואשתמש בגישת מפת"ח לניהול סיכונים שהיא להקדים תרופה למכה.

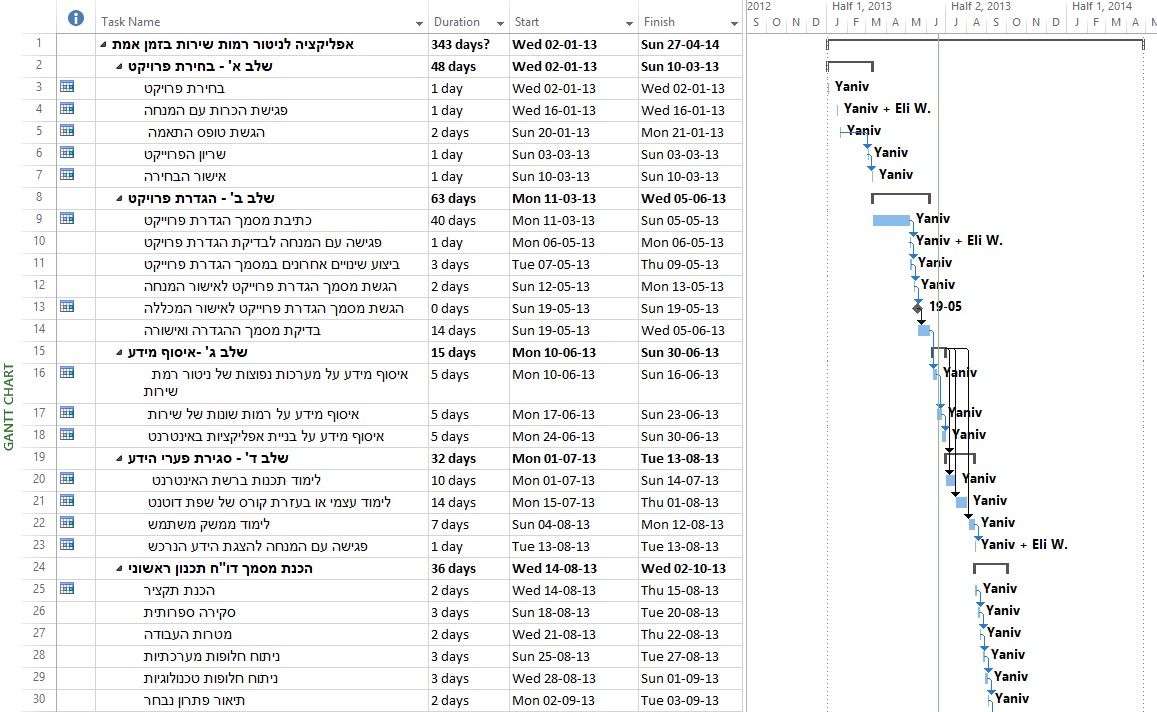
# חובות יזמיות

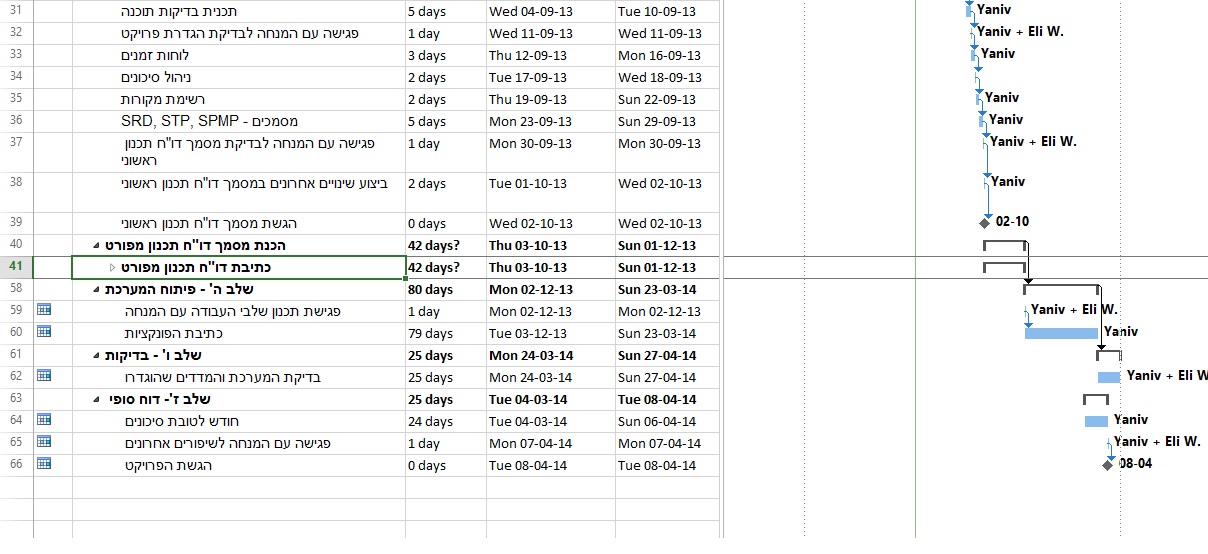
# לא רלוונטי לפרויקט זה.

# תוכנית ניהול הפרויקט



## 16.1 תוכנית עבודה של הפרויקט





# רשימת מקורות

* **Discussion Group, Cloud Computing Use Cases,Cloud Computing use Case, [1]**

**2010**

<http://opencloudmanifesto.org/Cloud_Computing_Use_Cases_Whitepaper-4_0.pdf>

* **[2] Walker G. , IT Problem Management, Prentice Hall PTR,2001**
* **[3] Baudoin C,Flynn J,McDonald J,Meegan J,Salsburg M & Woodward S, Public Cloud Service Agreements What to Expect and What to Negotiate,** **Cloud Standards Customer Council,2013.**

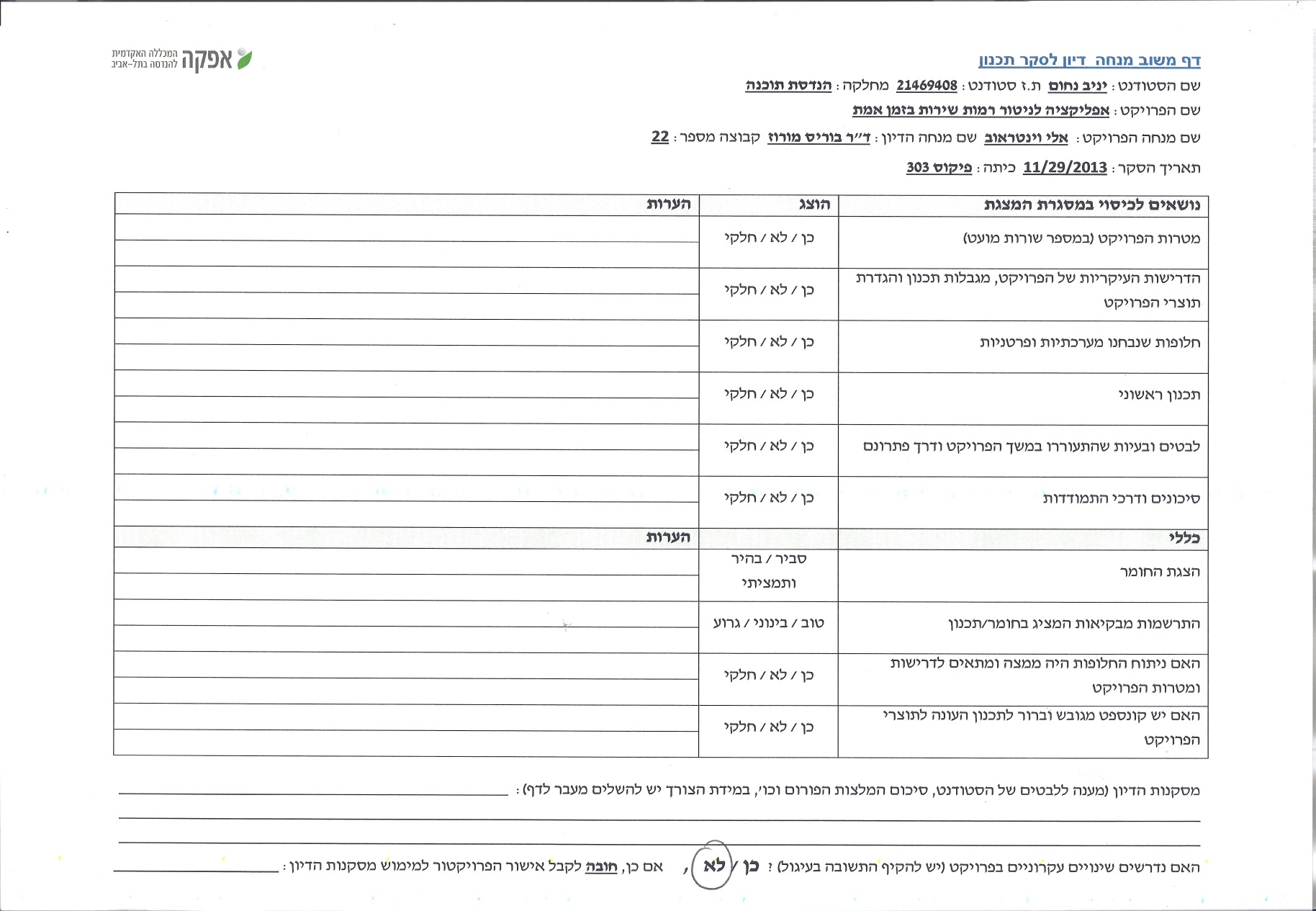
<http://cloud-council.org/publiccloudSLA.pdf>

* **[4]** **Xiong K & Perros H, SLA – Based Allocation Resource in Cluster Computing,** [**Parallel and Distributed Processing**](http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=4519061)**,2008**

<http://www4.ncsu.edu/~hp/Kaiqi6.pdf>

* **[5] Hiles A, The Complete Guide To IT Service Level Agreements ,Third Edition , Rothstein Associates,2002**
* **[6] Blokdijk G,The Service Level Agreement SLA Guide,Emereo,2008**

**דף משוב מנחה דיון לסקר תכנון**

****