שער

**<Secure IDE>**

מסמך עיצוב

<אדם ליבנה>

<גרסה 1.0>

<22/03/2020>

**היסטוריית גרסאות המסמך**

|  |  |  |
| --- | --- | --- |
| **תאריך** | גרסה | **תקציר השינויים** |
| 22/03/2020 | 1.0 | טיוטה ראשונה |
|  |  |  |
|  |  |  |
|  |  |  |



תוכן עניינים

[שער 1](#_Toc36664337)

[תוכן עניינים 2](#_Toc36664338)

[1 הקדמה 3](#_Toc36664339)

[1.1 מטרה 3](#_Toc36664340)

[1.2 המוצר 3](#_Toc36664341)

[1.3 קישור למסמכים קודמים 4](#_Toc36664342)

[1.4 הגדרות 4](#_Toc36664343)

[2 ארכיטקטורת המערכת 7](#_Toc36664344)

[2.1 מבט על 7](#_Toc36664345)

[2.2 פירוט רכיבי המערכת 7](#_Toc36664346)

[2.3 דיון בנושא העיצוב הנבחר 8](#_Toc36664347)

[3 עיצוב נתונים ופרוטוקולים 10](#_Toc36664348)

[4 ממשק משתמש 11](#_Toc36664349)

[File 12](#_Toc36664350)

[Window 12](#_Toc36664351)

[Edit 12](#_Toc36664352)

[Tools 12](#_Toc36664353)

[Run 12](#_Toc36664354)

[5 נספחים 16](#_Toc36664355)

1 הקדמה

החלק הזה ישמש להצגה כללית של מסמך העיצוב. הסבר כללי, וקישור למסמכים קודמים כמו מסמך האפיון עליו אנו מתבססים

מסמך זה מתאר את המבנה של הפרויקט המסכם של מגמת סייבר בתיכון הראל במסגרת תוכנית "גבהים" של משרד החינוך. במסמך מופיע תיאור של המוצר, קהל היעד, רכיבים מרכזיים ועוד.

1.1 מטרה

*מה המטרה של המסמך ומי קהל היעד שלו*

מטרת הפרויקט היא ליצור סביבת פיתוח תוכנה המהווה ערכת למידה ועבודה בתחום בדיקת בעיות אבטחה של מערכות מחשוב, כאשר קהל היעד של הפרויקט הוא מתכנתים אשר רוצים ללמוד ולעסוק בתחום אבטחת מחשבים.

1.2 המוצר

*חלק זה צריך להכיל את הדברים הבאים (ניתן לתמצת מתוך מסמך האפיון):*

* *שם המוצר אותו מפתחים*
* *הסבר כללי של מה המוצר עושה*
* שם המוצר: secureIDE
* המוצר מספק סביבה תכנותית – לימודית בתחום אבטחת מחשבים. כמו כן גם ספרייה עם חומרי לימוד ומידע מועיל (עם אפשרות להוסיף עוד), ומכונה ווירטואלית (Sandbox) למטרת בדיקה.
* מטרת המוצר לספק כמות מקסימלית של כלים ללמידה ופיתוח תוכנות בתחום אבטחת המחשבים. יתרונות המוצר בכך שהוא מרכז את הכל תחת תוכנה אחת, ושהוא מתרכז בעיקר בתחום אחד.
* בנוסף, יהיה גם אתר שממנו אפשר להוריד את התוכנה ~~(כולל גוגל פליי)~~ וגם סטאפ וויזארד
* להוסיף גם הסבר על המבנה של התוכנה עצמה למי שמעוניין (מה נכתב ותרשימי זרימה של המערכת)
* להוסיף רשימה של כל הדברים שניתן לעשות עם התוכנה (כל הפיצ'רים)
* להוסיף חלק שמבהיר מה שצריך מבחינה משפטית

1.3 קישור למסמכים קודמים

*קישור למסמכים / קבצים חיצוניים הרלוונטיים למסמך. לדוגמא: מסמך אפיון, עמוד web עם עיצוב הממשק הכללי שאתם מתכננים וכו'.*

**תיק פרוייקט**

**מסמך אפיון**

**האתר ממנו מותקן הפרויקט:** <http://adam.harelwebs.com/>

1.4 הגדרות

*בחלק זה יש לציין את כל ההגדרות, המושגים וראשי התיבות בהם תשתמשו בהמשך המסמך.*

* IDE (Integrated Development Environment) - סביבת פיתוח משולבת, [תוכנת מחשב](https://he.wikipedia.org/wiki/%D7%AA%D7%95%D7%9B%D7%A0%D7%94) המסייעת ל[מתכנתים](https://he.wikipedia.org/wiki/%D7%9E%D7%AA%D7%9B%D7%A0%D7%AA) לפתח תוכנה.
* אבטחת מחשבים – תחום רחב הכולל בתוכו כל נושא הקשור בגרימת/מניעת נזק לתפקוד של מערכת ממוחשבת.
* ספריות - אוסף של תת-תוכניות המשמשות לפיתוח [תוכנה](https://he.wikipedia.org/wiki/%D7%AA%D7%95%D7%9B%D7%A0%D7%94). ספריות מכילות קוד או מידע, שמספקים שירות לתוכניות עצמאיות.
* GUI (graphical user interface) – ממשק משתמש גרפי, צורה אינטואיטיבית ופשוטה לאינטראקציה של המשתמש עם המחשב בעזרת גרפיקה ממוחשבת.
* UI (user interface) – ממשק משתמש, הדרך בה המשתמש מתקשר עם התוכנה הרצה על המחשב.
* API (Application Programming Interface) - ממשק תכנות יישומים, ערכה של ספריות קוד, [פקודות](https://he.wikipedia.org/wiki/%D7%A4%D7%A7%D7%95%D7%93%D7%94_(%D7%9E%D7%97%D7%A9%D7%91)), [פונקציות](https://he.wikipedia.org/wiki/%D7%A4%D7%95%D7%A0%D7%A7%D7%A6%D7%99%D7%94) ו[פרוצדורות](https://he.wikipedia.org/wiki/%D7%A9%D7%92%D7%A8%D7%94) מן המוכן, בהן יכולים ה[מתכנתים](https://he.wikipedia.org/wiki/%D7%9E%D7%AA%D7%9B%D7%A0%D7%AA) לעשות שימוש פשוט, בלי להידרש לכתוב אותן בעצמם כדי שיוכלו להשתמש במידע של היישום שממנו הם רוצים להשתמש לטובת היישום שלהם.
* VM (virtual machine) - מכונה וירטואלית, [תוכנה](https://he.wikipedia.org/wiki/%D7%AA%D7%95%D7%9B%D7%A0%D7%94) היוצרת סביבה הנחוצה להפעלתה של תוכנה אחרת, מבלי שיהיה צורך במימוש פיזי של סביבה זו. מכונה וירטואלית משחררת את התוכנה המתבססת עליה מתלות ב[חומרה](https://he.wikipedia.org/wiki/%D7%97%D7%95%D7%9E%D7%A8%D7%94) מסוימת, משום שכל חומרה שעליה תמומש המכונה הווירטואלית תאפשר את הפעלת התוכנה. בנוסף, המכונה הווירטואלית מדמה מערכת הפעלה, מה שמאפשר לערוך בדיקות העלולות להיות הרסניות מבלי לדאוג לשלמות המחשב המריץ את הבדיקה.
* Sandbox – ארגז חול, סביבה שבה מאפשרים למשתמשים לעשות ככל העולה על רוחם, ללא חשש מנזק ל[מערכת ההפעלה](https://he.wikipedia.org/wiki/%D7%9E%D7%A2%D7%A8%D7%9B%D7%AA_%D7%94%D7%A4%D7%A2%D7%9C%D7%94). זוהי תוכנה אשר עוטפת תוכנות אחרות ומבדילה את משאביהן ממשאביה של מערכת ההפעלה.
* Sandboxie – תוכנה מסוג Sandbox אשר בה נעשה שימוש בפרויקט זה
* פקודת מערכת (System call) – בקשה שמבצעת תוכנת מחשב מ[ליבת מערכת ההפעלה](https://he.wikipedia.org/wiki/%D7%9C%D7%99%D7%91%D7%94_(%D7%9E%D7%A2%D7%A8%D7%9B%D7%95%D7%AA_%D7%94%D7%A4%D7%A2%D7%9C%D7%94)) כדי לבצע פעולה שהיא אינה יכולה לבצע בעצמה. קריאות מערכת הן האחראיות על החיבור שבין המשתמש לליבת המערכת, ובכך מאבזרת את המשתמש ונותנת לו שימוש מרבי בפונקציונליות שהיא מציעה. הדבר כולל בין היתר יכולת קבלת גישה למרבית רכיבי החומרה של המחשב.
* טרמינל (שמות נוספים: ממשק שורת הפקודה, CLI/CMD - Command Line Interface) הוא [מעטפת](https://he.wikipedia.org/wiki/%D7%9E%D7%A2%D7%98%D7%A4%D7%AA_(%D7%9E%D7%97%D7%A9%D7%91%D7%99%D7%9D)) שמספקת [מערכת הפעלה](https://he.wikipedia.org/wiki/%D7%9E%D7%A2%D7%A8%D7%9B%D7%AA_%D7%94%D7%A4%D7%A2%D7%9C%D7%94) או [תוכנה](https://he.wikipedia.org/wiki/%D7%AA%D7%95%D7%9B%D7%A0%D7%94) אחרת למשתמש על מנת שזה יוכל להקיש פקודות טקסטואליות שיפעילו אותה. ממשק שורת הפקודה הוא טקסטואלי, בניגוד לממשקים גרפיים אחרים שהתוכנה או מערכת ההפעלה יכולות לספק.
* Cmd – הטרמינל שבא כברירת בחדל במערכת ההפעלה windows, ובהדרגה מתחיל להיות מוחלף ע"י Power Shell
* Eclipse IDE – [כלי פיתוח](https://he.wikipedia.org/wiki/%D7%A1%D7%91%D7%99%D7%91%D7%AA_%D7%A4%D7%99%D7%AA%D7%95%D7%97_%D7%9E%D7%A9%D7%95%D7%9C%D7%91%D7%AA) ליישומים בשפת [Java](https://he.wikipedia.org/wiki/Java)‏. Eclipse מספקת סביבת פיתוח מתקדמת לשפה זו, והיא בעצמה משמשת לפיתוח פלטפורמת Eclipse, הכתובה גם היא ב־Java. נוסף על כך, ל־Eclipse יש מספר רב של פרויקטים משניים, המספקים סביבות פיתוח לשפות אחרות כגון [C++](https://he.wikipedia.org/wiki/C%2B%2B)‎, [פיתון](https://he.wikipedia.org/wiki/%D7%A4%D7%99%D7%99%D7%AA%D7%95%D7%9F_(%D7%A9%D7%A4%D7%AA_%D7%AA%D7%9B%D7%A0%D7%95%D7%AA)) ורבות אחרות. ההשערה הרווחת היא שהשם "Eclipse" נבחר בגלל התחרות עם חברת [סאן מיקרוסיסטמס](https://he.wikipedia.org/wiki/%D7%A1%D7%90%D7%9F_%D7%9E%D7%99%D7%A7%D7%A8%D7%95%D7%A1%D7%99%D7%A1%D7%98%D7%9E%D7%A1), מפתחת שפת java. סאן פירושו [שמש](https://he.wikipedia.org/wiki/%D7%94%D7%A9%D7%9E%D7%A9) ו־"Eclipse" פירושו [ליקוי חמה](https://he.wikipedia.org/wiki/%D7%9C%D7%99%D7%A7%D7%95%D7%99_%D7%97%D7%9E%D7%94).
* NetBeans IDE –  [סביבת פיתוח משולבת](https://he.wikipedia.org/wiki/%D7%A1%D7%91%D7%99%D7%91%D7%AA_%D7%A4%D7%99%D7%AA%D7%95%D7%97_%D7%9E%D7%A9%D7%95%D7%9C%D7%91%D7%AA) חינמית מבית אורקל אשר נכתבה בשפת ג'אווה והשייכת לקטגוריית ה[קוד הפתוח](https://he.wikipedia.org/wiki/%D7%A7%D7%95%D7%93_%D7%A4%D7%AA%D7%95%D7%97). אפשר לפתח בה [אפליקציות](https://he.wikipedia.org/w/index.php?title=%D7%AA%D7%95%D7%9B%D7%A0%D7%AA&action=edit&redlink=1) שולחניות, אפליקציות [אינטרנט](https://he.wikipedia.org/wiki/%D7%90%D7%99%D7%A0%D7%98%D7%A8%D7%A0%D7%98) ואפליקציות ל[סלולרי](https://he.wikipedia.org/wiki/%D7%98%D7%9C%D7%A4%D7%95%D7%9F_%D7%A1%D7%9C%D7%95%D7%9C%D7%A8%D7%99) באמצעות [שפות תכנות](https://he.wikipedia.org/wiki/%D7%A9%D7%A4%D7%AA_%D7%AA%D7%9B%D7%A0%D7%95%D7%AA) רבות.
* Launch4j – כלי חוצה פלטפורמות לעטיפת יישומי Java המופצים כקובצי Jar במוצרי הפעלה מקוריים של Windows. ניתן להגדיר את ההפעלה לחיפוש אחר גרסת JRE מסוימת או להשתמש בגירסה מקובצת וניתן להגדיר אפשרויות זמן ריצה. העטיפה מספקת גם חוויית משתמש טובה יותר באמצעות אייקון יישום, מסך התזה לפני JRE ודף הורדה של Java למקרה שלא ניתן למצוא את JRE המתאים.
* Inno Setup Compiler – תוכנת התקנה חינמית לקובצי הרצה מסוג exe על פלטפורמת windows. Inno Setup, שהוצג לראשונה בשנת 1997, מתחרה ואף מתעלה על מתקינים מסחריים רבים בסט תכונות ויציבות.
* JRE – סביבת ההרצה של ג'אווה
* JVM – מכונה וירטואלית המפרשת ומריצה Java Bytecode. לרוב, קוד זה מופק על ידי מהדר לשפת Java, אם כי ה-JVM יכולה לשמש גם כפלטפורמת היעד של מהדרים לשפות אחרות. ה-JVM היא תוכנת מחשב ככל תוכנה אחרת, והיא משמשת כמעין שכבת-מיפוי בין מערכת ההפעלה לבין תוכניות הרצות על גביה.
* JDK – מוצר של סאן המיועד עבור מפתחי ג'אווה. מאז הופעתה של ג'אווה, ה-JDK הייתה ערכת פיתוח התוכנה (Software Development Kit, SDK) הנפוצה ביותר עבור ג'אווה. ה-JDK כוללת מהדר לשפת Java, עותק מלא של סביבת ההרצה של ג'אווה (JRE), והרבה כלי פיתוח חשובים נוספים.
* Java – שפת תכנות מונחית עצמים אשר פותחה בחברת סאן מיקרוסיסטמס (כיום חברת־בת של אורקל) על ידי צוות בראשות ג'יימס גוסלינג בשנת 1991, והיא אחת משפות התכנות הנפוצות ביותר הנמצאות בשימוש כיום. השפה הוצגה לראשונה בשנת 1995, והיא מהווה את אחד מרכיבי הליבה של פלטפורמת התוכנה ג'אווה. התחביר של השפה מבוסס במידה רבה על התחביר של ++C, אך כולל הרחבות רבות. לרוב עוברות תוכניות ג'אווה הידור ל־Java bytecode, שפת ביניים דמוית שפת מכונה, שאותה מריצה מכונה וירטואלית (Java Virtual Machine; JVM). הודות לכך התוכנית יכולה לרוץ על כל מחשב ועל כל מערכת הפעלה המריצים JVM, החל מטלפונים סלולריים ועד למחשבי-על.
* קובץ הרצה (.exe) – קובץ אשר מכיל נתונים המתורגמים לתוכנית על ידי המחשב. על פי רוב, הקובץ כולל ייצוג בינארי של פקודות מכונה בעבור מעבד מסוים. באופן תאורטי כל קובץ, אשר קיימת תוכנית מחשב אחרת, למשל מערכת הפעלה, המסוגלת להריץ אותו, הוא קובץ הרצה. כך למשל, אם מותקנת במחשבך סביבת הריצה של ג'אווה תוכל להריץ קובצי JAR. לעיתים כולל קובץ ההרצה מידע אשר אינו חלק מהתוכנית עצמה, כגון מידע על ניפוי שגיאות. קובצי הרצה עשויים לכלול קריאות לשירותים שמספקת מערכת ההפעלה. מכאן אפשר להבין מדוע מרבית קובצי הרצה הם ספציפיים לסביבה בעבורה הם נכתבו. קובץ הרצה המיועד עבור מערכת ההפעלה חלונות (שהיא גם סוג של תוכנית מחשב), לא יוכל לרוץ מעל UNIX.
* קובץ התקנה – תוכנה המצורפת לתוכנה אחרת שאינה ברת הרצה במצבה הראשוני, ואחראית על התקנתה. תוכנות התקנה מתאימות לתוכנה מסוימת אליה הן נכתבו.
* Adobe xd -
* Draw.io -

2 ארכיטקטורת המערכת

חלק זה כולל את תיאור מבנה המערכת ופירוט המודולים השונים בה

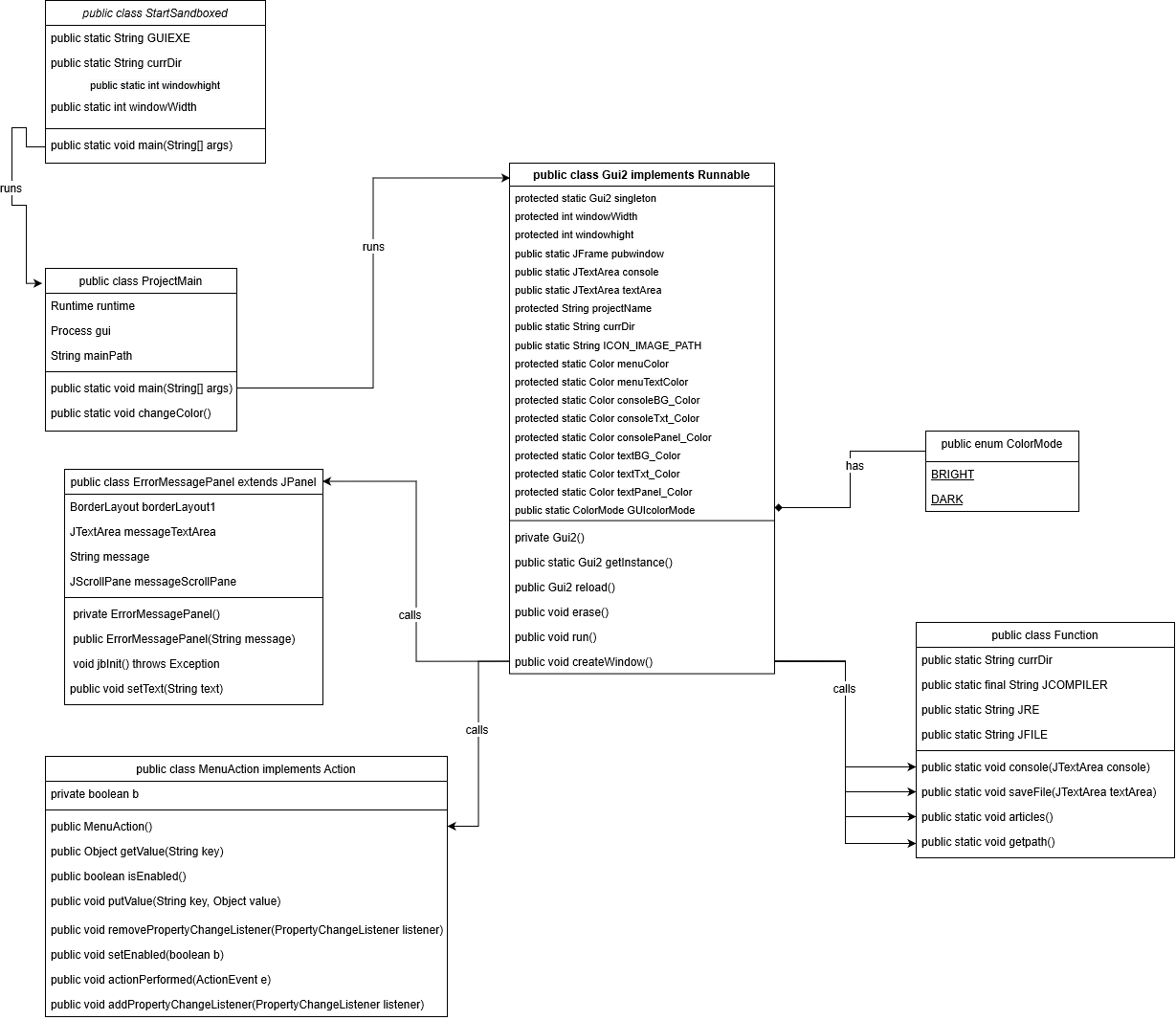
2.1 מבט על

בסעיף זה יש להסביר את החלוקה העיקרית למודולים השונים במערכת, והיחסים/קשרים ביניהם. יש להגדיר באופן ברור את תחום האחריות של כל מודול או רכיב, ולאיזו פונקציונליות (כפי שהוגדרה במסמך האפיון שנכתב לפני כן) הוא מתקשר (בין אם מודול אחד בלבד או מספר מודולים יחד).  
חשוב להסביר לא רק מה/איך נעשה אלא גם מדוע דווקא כך.

2.2 פירוט רכיבי המערכת

**כאן יופיע פירוט לכל רכיב (ניתן לעשות באיטרציות – כל פעם עבור הרכיבים הרלוונטיים, לפני תהליך הפיתוח שלהם)**  
בסעיף זה יש לפרט את המבנה הפנימי של כל מודול/רכיב - כלומר:

* מבנה המחלקות השונות המרכיבות אותו, המאפיינים שלהם (public/private/protected + אילו טיפוסי משתנים כל פונקציה מקבלת ו/או מחזירה) והקשרים ביניהם (מי קורא למי, משתמש במי ולאיזה צורך) – מומלץ לצייר גרף UML (לא חייבים לפרט משתנים/פונקציות פרטיות אבל שיהיה ברור מה קשור למה ובאיזה אופן או לאיזה צורך)
* יש לתאר את ה- data flow / use case של שימושים שונים בכל תת-רכיב או פונקציה (מה קורה אם מתקבלים סוגים שונים של ערכים, איך/מדוע/באילו מקרים הרכיבים מתנהגים בצורה מסוימת או אחרת וכן הלאה)



**המלצות:**

* כדאי לתכנן רכיבים תשתיתיים עבור צרכים גלובליים במערכת - כמו למשל יצירת תקשורת רשתית, כתיבה וקריאה לבסיס נתונים (או קבצים), כתיבת תיעוד (log) הפעולות של המערכת (מקובל להגדיר "רמה" לכל שורה שנכתבת - debug, info, warning, error כדי שיהיה קל לסנן בזמן ריצת התוכנית. בנושא זה – מוזמנים להסתכל על מודול logging בפייתון).
* מומלץ לתכנן את הרכיבים באופן שיאפשר אבסטרקציה ומודולריות – בכדי לאפשר הרחבה עתידית ותחזוקה קלה. אם המימוש הפנימי של רכיב או מודול מסוים משתנה אז שאר הרכיבים/מודולים אינם מושפעים או נפגעים מכך ואין צורך לשנות גם אותם, או כאשר רוצים להוסיף פונקציונליות/רכיב חדש אז אין צורך לשכתב את כל שאר המערכת במיוחד בשבילו.

2.3 דיון בנושא העיצוב הנבחר

כאן יש לפרט ובעיקר להסביר מדוע בחרתם דווקא בחלוקת הרכיבים/תפקידים הזו ולא אחרת. מה היתרונות שלכם מבחינתה, וגם מה החסרונות שאתם מודעים אליהם. במידה וחשבתם על חלופות אחרות, יש לציין אותן ולהסביר בקצרה כיצד הן שונות מהפתרון שנבחר - ומדוע החלטתם בסופו של דבר שלא לבחור בחלופות האלו.  
כאן יש להתייחס גם לגבי שפת התכנות שבה בחרתם לכתוב כל רכיב, ומדוע דווקא בשפה זו ולא אחרת.

3 עיצוב נתונים ופרוטוקולים

כאן יופיע תיעוד של מבני נתונים שונים / פרוטוקולים המשמשים אותנו במערכת

בסעיף זה יש לפרט את כל סוגי המידע אשר מועבר או נשמר במערכת - בין אם בין רכיבים, בין מודולים או בכל צורה אחרת (כלומר גם אם זה משמש רק לתקשורת פנימית ולא ע"ג הרשת, או למשל אם זה משהו ששומרים לקובץ/בסיס נתונים).

* לכל סוג מידע כזה יש לפרט על השדות אותן הוא מכיל ומאיזה סוג כל שדה, מה טווח הערכים הרלוונטי לגביו וכל הגבלה או מידע נוסף אחר שאתם מוצאים לנכון (למשל: האם מותר שהשדה יהיה ריק ומה המשמעות של זה, אורך או ערך מינימלי/מקסימלי למספר או למחרוזת)
* יש לציין לאיזו מטרה משמש כל מבנה נתונים.
* עבור פרוטוקול תקשורת-נתונים בין רכיבים שונים:
* מה המצבים השונים לכל סוג של תקשורת (למשל אותנטיקציה, הרשמה, שליחת קובץ וכו')
* באילו ייצוגי מידע נעשה שימוש + תרשים זרימה של המצבים השונים שיכולים להיות וסדר השלבים שלהם (למשל כיצד נעביר מידע על מקרים של סיסמה שגויה, משתמש חסום, התחברות מוצלחת וכו')
* מומלץ להגדיר מראש קודים של בקשות/פעולות, תוצאות/תגובות ושגיאות אשר ישותפו בין כך הרכיבים הרלוונטיים
* יש להתייחס לאופי המידע הנשמר (האם הוא טקסטואלי או בינארי) ולייצגו בצורה נוחה והולמת בהתאם
* יש לפרט אודות מבנה בסיס הנתונים (חלוקה לטבלאות, השם והטיפוס של כל עמודה/שדה, קשרים בין טבלאות ואילוצים כלשהם על עמודות)

4 ממשק משתמש

כאן יופיע פירוט ממשקי המשתמשים וכן תופיע הסקיצה עבורם

בסעיף זה יש לפרט את הפונקציונליות של המערכת כפי שהיא מתבטאת עבור משתמש חיצוני. במידה ויש סוגים שונים של משתמשים, יש להתייחס לכולם בהתאם - אילו רכיבים או נתונים רלוונטיים עבורם, וכיצד הם מתקשרים איתם.  
  
בנוסף, יש לצרף המחשות ויזואליות של המסכים השונים, ולהסביר את התוכן שלהם (למשל מה התפקיד של כל שדה או כפתור, מה קשור/תלוי במה - למשל כפתור שמכובה בהתאם לתנאים מסוימים במערכת וכן הלאה) והקשרים ביניהם (איזה מסך מוביל לאיזה מסך ובאילו מקרים)  
  
דוגמה (לא מלאה):  


ממשק המשתמש של הפרויקט שלי נועד להיות יחסית מינימליסטי, פונקציונלי ופשוט להבנה. במבט קל על המסך הראשי ניתן לזהות שלושה אלמנטים עיקריים. העליון זהו התפריט, דרכו מתבצעות כל הפעולות אותן מאפשרת התוכנית. במרכז ובתחתית נמצאות שתי תיבות טקסט ריקות. התיבה העליונה משמשת כאזור בו נכתב הקוד של המשתמש, ואילו התחתונה משמשת כאזור של הקונסולה, בו מודפסים הפלט והשגיאות של התוכנית ושדרכו מוכנס הקלט. באחת הפינות של כל אחת מתיבות אלה ישנו משולש קטן, אשר גרירה שלו באמצעות העכבר תשנה את גודלה של תיבת הטקסט בו הוא נמצא. במקרה של שגיאה כלשהיא בפעולתה של התוכנית, השגיאה תוצג בחלון קופץ (popup) למשתמש ובנוסף תודפס הודעת שגיאה ותכולת המחסנית (Stack trace) לקונסולה אשר דרכה הופעלה התוכנית. בתפריט ישנם מספר תפריטי-משנה, אשר תוכנם מופיע בהמשך:

File

* Save As… מפעיל חלון קופץ אשר מבקש להכניס מיקום קובץ (path). התוכנית תשמור את הטקסט שנכתב באזור הקוד כקובץ במיקום שהוכנס, והמיקום יישמר בזיכרון התוכנית
* Save As File… מבצע פעולה זהה לקודמו, אך הקלט של הקובץ מתבצע בעזרת UI נוח יותר לשימוש

Window

* Show
* Change Color יוצר חלון נוסף של התוכנית אשר פלטת הצבעים שלו משתנה. אם המשתמש עבד במצב בהיר החלון יווצר במצב כהה, ולהפך
* Articles מציג חלון אשר כל כפתור בו הינו לינק לאתר או מאמר המעניק למשתמש מידע על לימוד בתחום אבטחת המידע והמחשבים
* Legal Information מציג חלון אשר כל כפתור בו הינו לינק לאתר או מאמר המעניק למשתמש מידע על חוקים ומעמדן המשפטי של פעולות פריצה (האקינג)

Edit

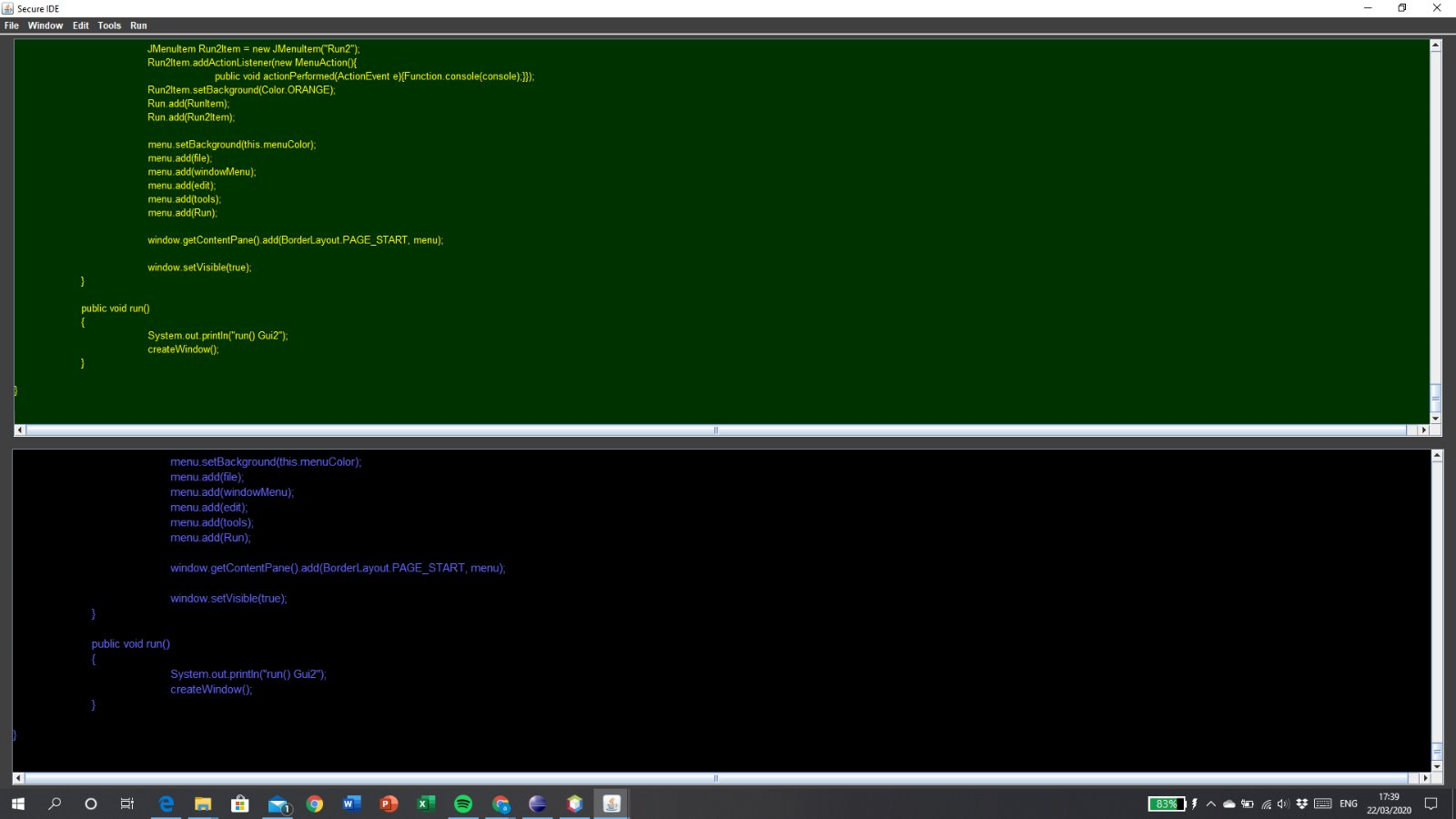
* Change Path מציג למשתמש חלון קופץ אשר מבקש לבחור את המיקום של הקובץ אותו מבקש המשתמש לקמפל ולהריץ (כרגע התמיכה היא רק בקבצי .java) ושומר מיקום זה בזיכרון התוכנית

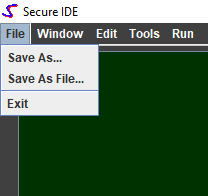
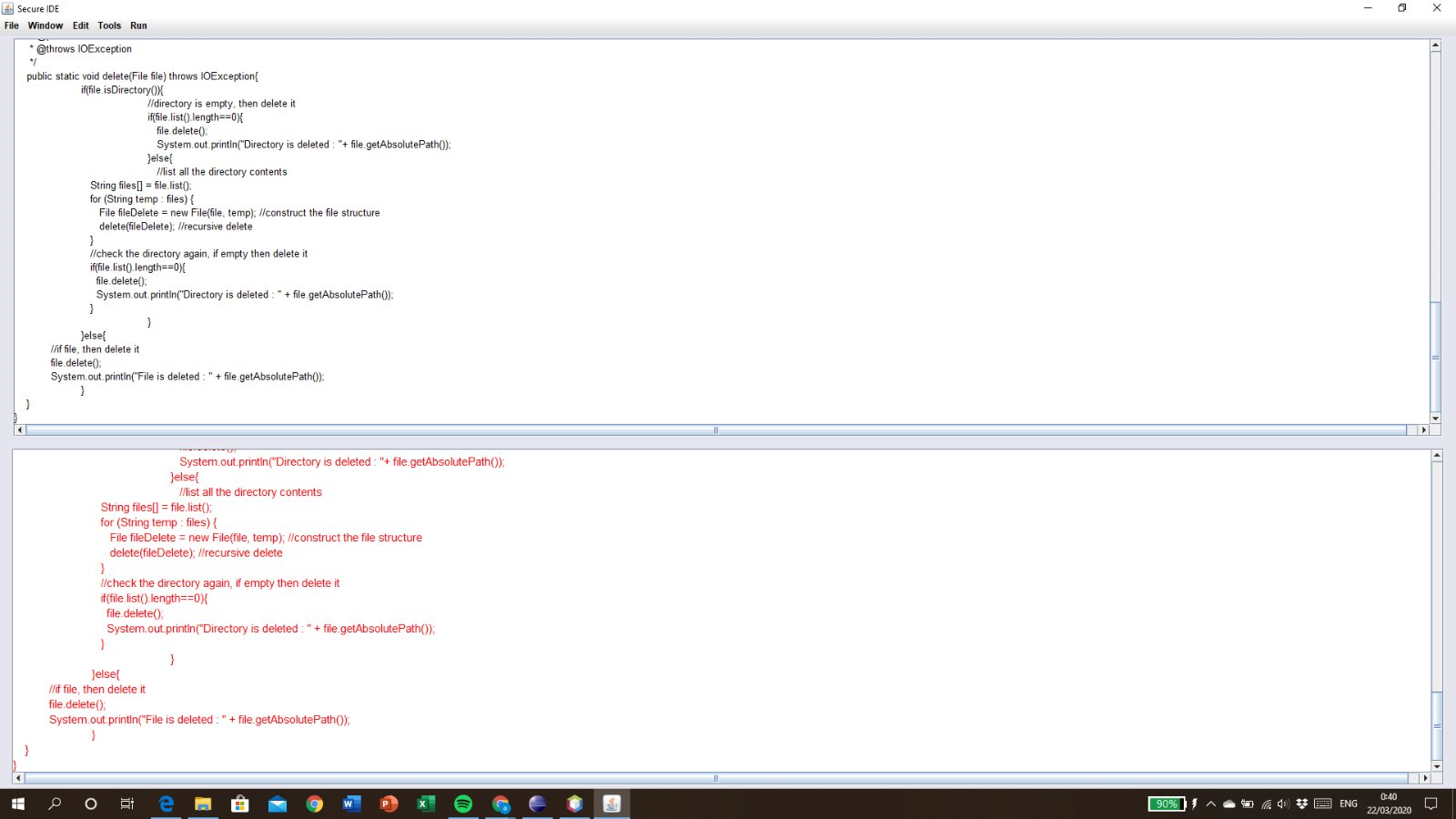
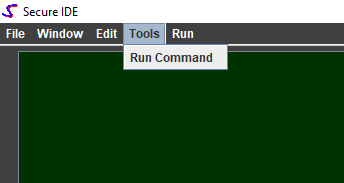
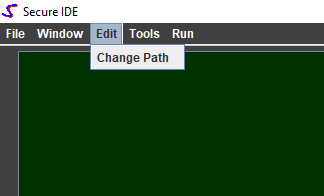
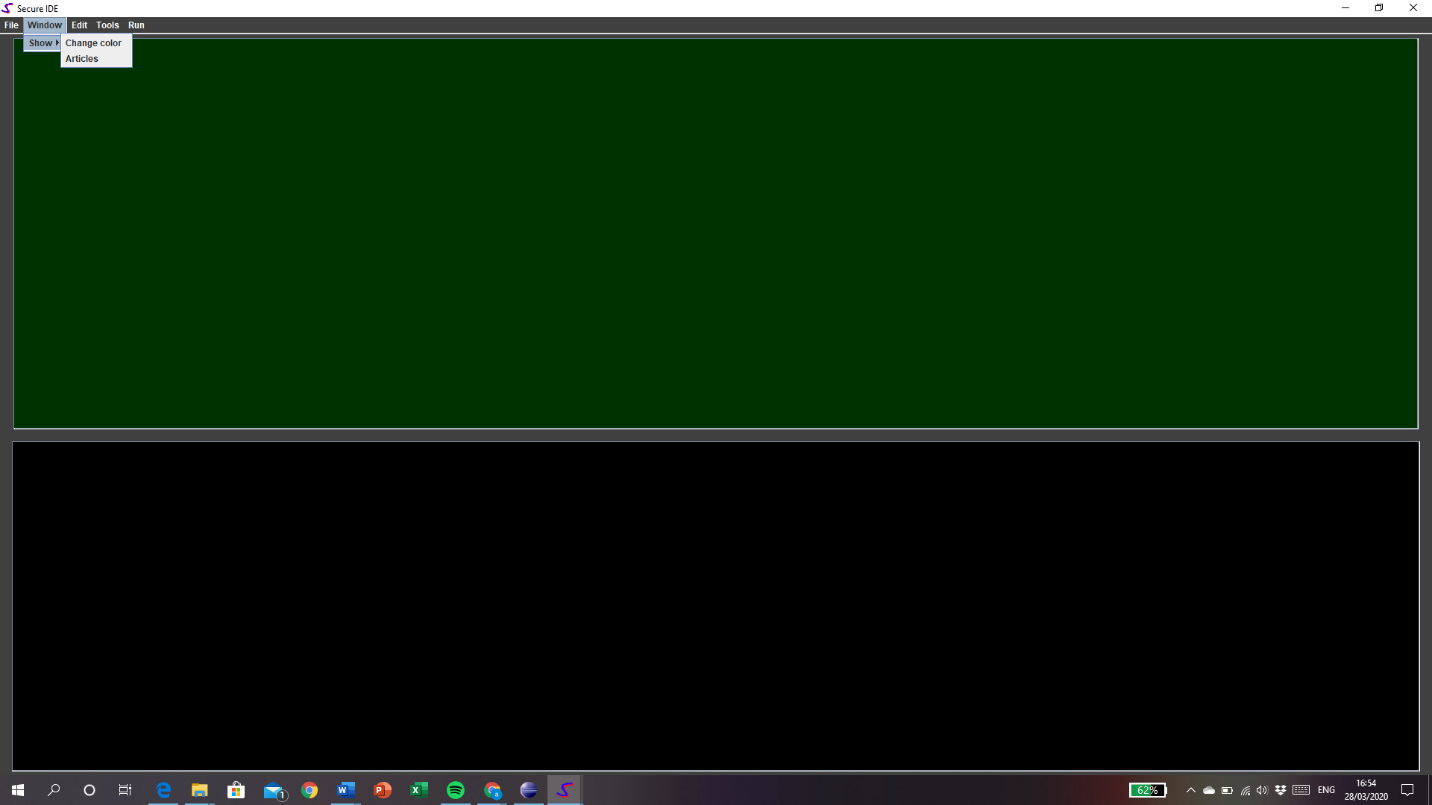
Tools

* Run Command מפעיל חלון קופץ (popup) אשר מבקש להכניס פקודה. התוכנית תריץ את הפקודה שהוכנסה דרך הטרמינל של המערכת ותציג את הפלט של הפקודה בחלון קופץ נוסף

Run

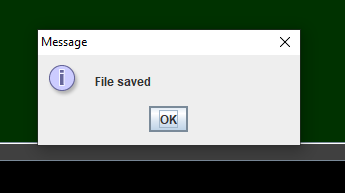
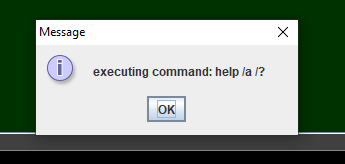
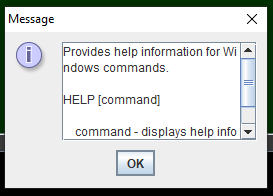
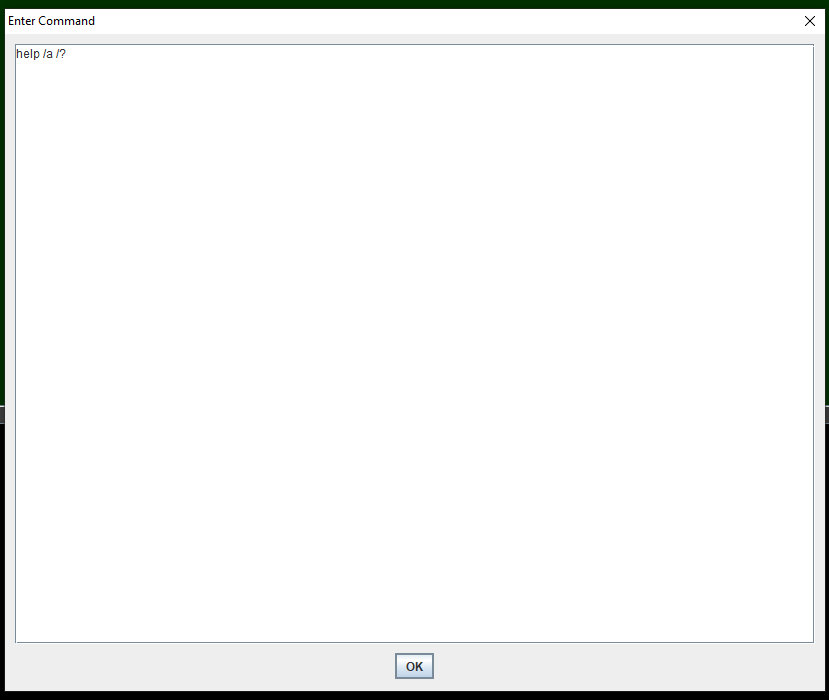
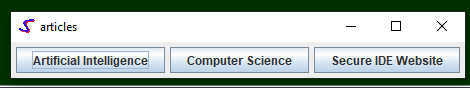
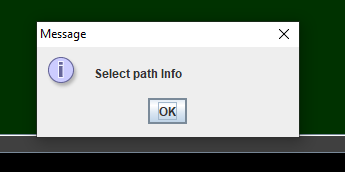
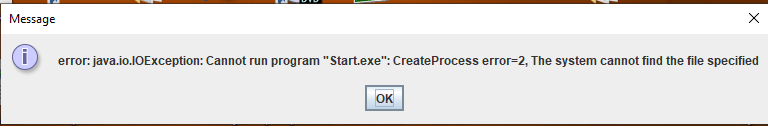
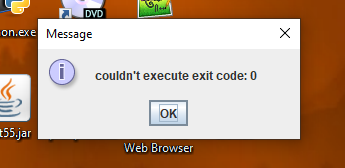
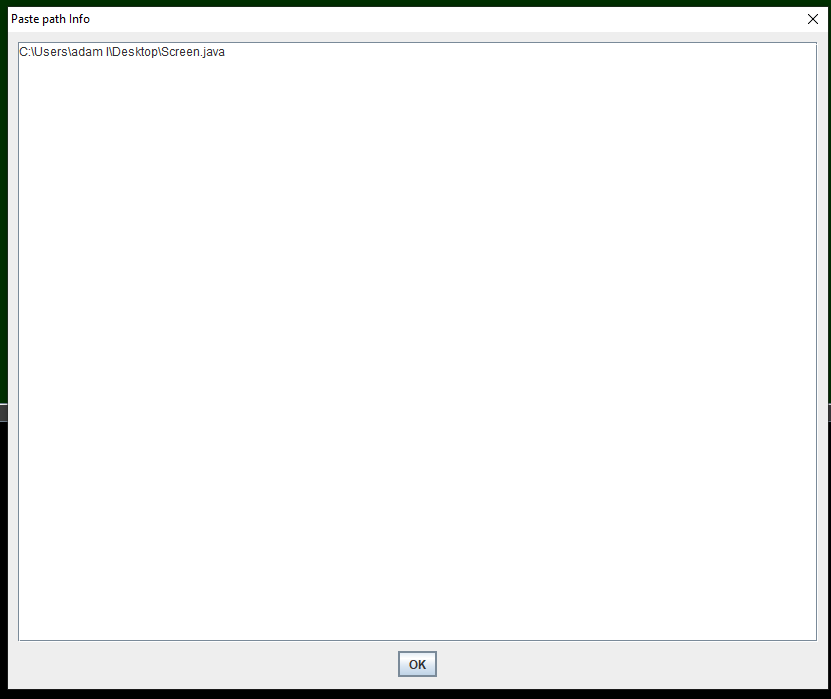
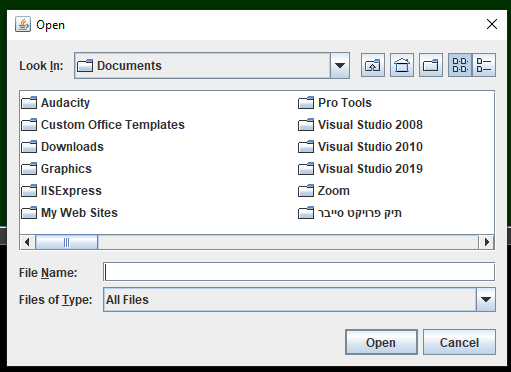
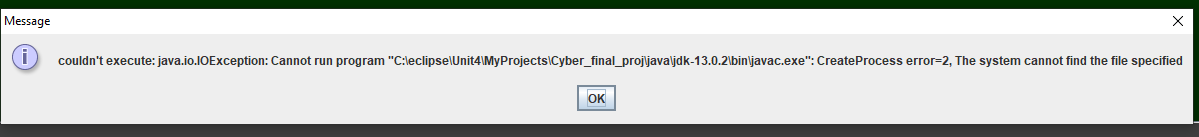
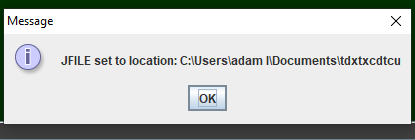
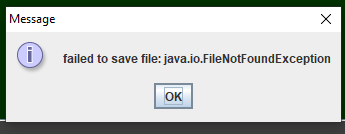
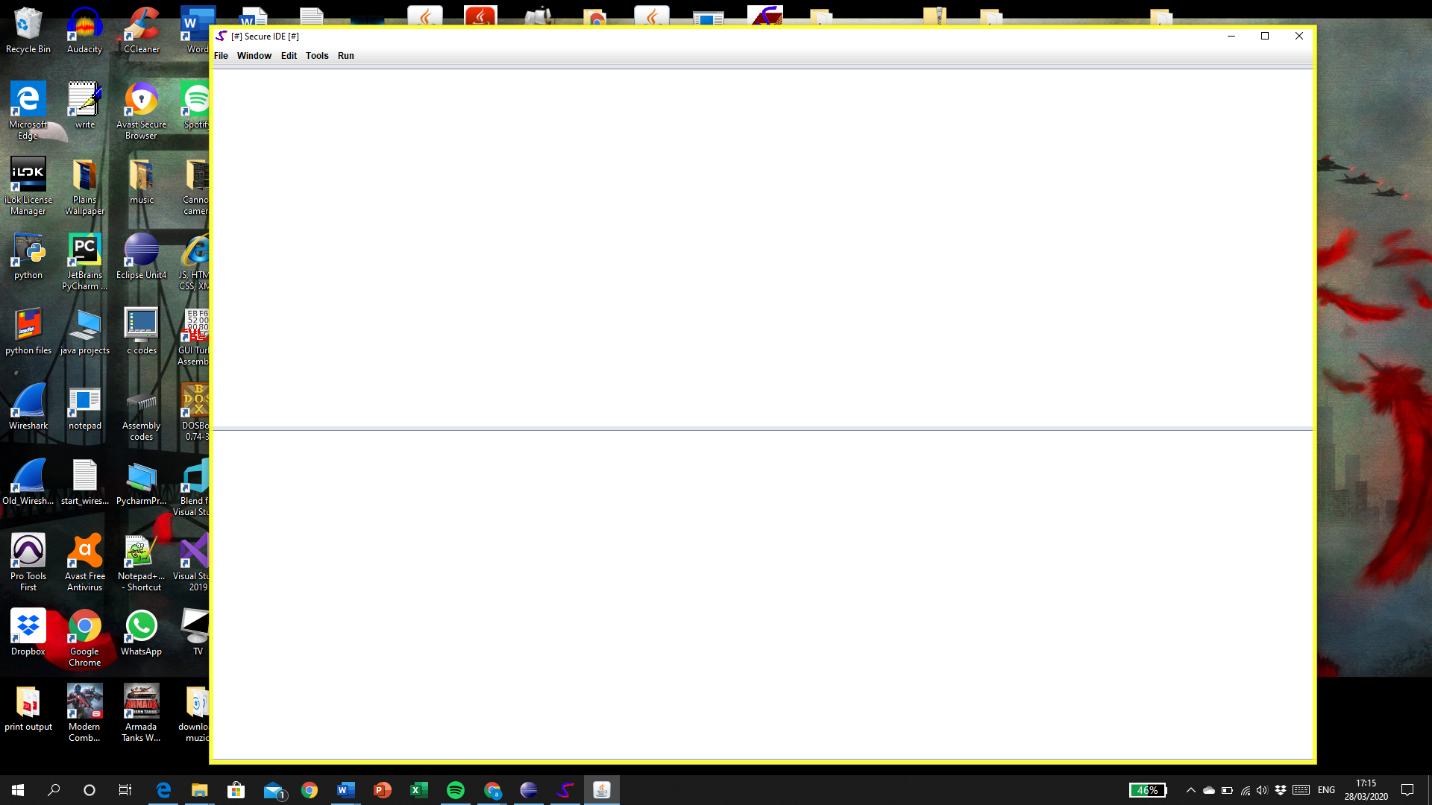
* Run מפעיל את ההדרה וההרצה של הקובץ שמיקומו נשמר בזיכרון התוכנית



5 נספחים

**כל דבר שהייתם רוצים להוסיף בנוגע לעיצוב המערכת**

****