# ***עבודת גמר בתכנון ותכנות מערכות התמחות סייבר***

# 5 יח"ל – שאלון 883589

# מערכת Cloud Walkie-Talkie

# מגיש: אריאל אורן (ת.ז: 215358904)

# מורה מנחה: שרית שוורץ

# בית הספר: אמי"ת גוש-דן בר-אילן

# שנת הגשה: 2023, תשפ"ג

## מבוא

### ייזום

הפרוייקט שלי הוא תוכנה בשביל מדריכי טיולים, היא יכולה לתת למדריך ליצור חדר ואז כל מי שנמצע איתו בטיול יוכל להקשיב לו דרך התוכנה כשהוא נכנס לחדר, זה שימושי במקום להשתמש במקרופון או בווקי טוקי לכל אחד שעלול להישבר או ללכת לאיבוד. בחרתי בפרוייקט הזה בגלל שכשהייתי בסיור ביד ושם המדריכה נתנה ווקי טוקי לכל אחד עם אוזניות חד פעמיות וזה היה לט נוח לשימוש וכנראה שגם ישן, בנוסף אחד מהווקי טוקי כמעט אבד וזה גם בעיה. האתגרים שיש הם: לעשות שזה יעבוד בכל הפלטפורמות והטלפונים וגם בווב, לעשות שזה יהיה נגיש לכולם

האפליקציה תהיה מיועדת בעיקר למדריכי טיולים אך יהיה אפשר להשתמש בה גם בכיתות או במקומות עם הרבה אנשים שצריך להקשיב למישהו אחד

המטרה שלי זה שזה יהיה מספיק נוח כדי שכל אחד יוכל להשתמש בזה בלי בעיה ושזה יהיה בשימוש על ידי קשת רחבה של משתמשים

התועלת של זה שזה חסכוני בכסף ושלא צריך לקנות מכשיר בשביל שכולם יוכלו לשמוע

הטכנולוגיה קיימת אך לא לשימוש כזה, יש שיחות טלפון רגילות אך זה לא נוח לעשות שיחה משותפת וזה לא נוח לעשות שמישהו אחד מדבר, יש זום אבל זה כבד מדי לשימוש פשוט וזולל הרבה בטרייה והוא גם לא מספיק קל לשימוש וצריך לשלם בשביל חדר שלא נסגר, יש גם תחליפים לזום אבל יש להם את אותם בעיות של זום

הפרויקט קשור בעיקר לרשתות ותקשורת בין משתמשים וגם בין פלטפורמות ומערכות הפעלה שונות

#### הגדרת לקוח:

יש שתי סוגים של לקוחות: מדריכים (או מורים) שפותחים את החדר ומדברים, ומשתמשים רגילים שהולכים עם המדריכים (או מורים) ונכנסים לחדר שלהם ומקשיבים למה שהם אומרים

#### הגדרת יעדים/מטרות:

לעשות שזה יהיה כמה שיותר נוח לשימוש ונגיש לכמה שיותר אנשים אבל להשאיר שזה יהיה תוכנה קלה

#### בעיות תועלות וחסכונות:

התועלת של התוכנה שלי זה שהיא נוחה למדריכים (או מורים) שרוצים להדריך לקבוצה של אנשים שצריכים להקשיב להם והם נמצאים המרחק מה מהם וקשה להקשיב, עוד תועלת זה שזה יהיה מוכן למדריכים בעיקר עם פונקציות שהם צריכים כמו למשל שיתוף מיקום למקרה שמישהו הלך לאיבוד

#### סקירת פתרונות קיימים:

Zoom – זה טוב לשיחות אבל יש זמן מוגבל לחדר אם לא משלמים, האפליקציה לא כל כך אינטואיטיבית והיא זוללת סוללה ומאוד כבדה, בנוסף היא מיועדת בעיקר ללמידה ולא לטיולים

יש עוד תוכנות טיולים אבל זה בעיקר לתכנון מסלולים ולא למדריכים

#### סקירת טכנולוגית הפרויקט:

צריך מקרופון, רמקול וgps למשתמשים, אין טכנולוגיה מיוחדת

#### תיחום הפרויקט:

יש שירות צד לקוח בשביל להתחבר לחדרים וליצור משתמשים

יש שירות peer to peer בשביל השיחות

יש שירות במערכות הפעלה בשביל להשמיע קול, להשתמש בהקלטה וgps

### פירוט תיאור המערכת (אפיון)

במערכת יהיה אפשר ליצור חדר, זה יצור קוד וכל משתמש שיכניס את הקוד יכנס לחדר ורק מי שיצר את החדר יוכל לדבר ויהיה לו גישה של admin, יהיה אפשר לשוחח בצאט והadmin יוכל להביא לאנשים לדבר או להעיף משתמשים ולנעול את החדר, יהיה גם text to speech שיכתוב את מה שנאמר (טוב לכבדי שמיעה) וזה יתרגם גם לשפות אחרות בגוגל תרגום, יהיה אפשר גם לשתף מקום ולסמן מקומות למקרה שמישהו הולך לאיבוד או שעושים סיבוב בשוק ואז רוצים להיפגש שוב בנקודה מסויימת

יהיה משתמש admin, ומשתמשים רגילים שלפעמים יהיה להם פריווילגיות מסוימות

לסיים את הgui, לסיים את האפשרות לייצור שיחה, לעשות שרת שאחראי על שליחת הקודים וחיבור לחדרים ויצירת משתמשים, לחבר הכל ביחד, להוסיף את שאר הפיצרים

יש סיכונים שיהיה אפשר להתחבר לחדרים בלי שיודעים אז צריך לעשות שלא יהיה קל לפצח את הקוד, יש סיכון שיצותתו גם אז להצפין את השיחה, יש אפשרות שיגנבו מידע על משתמשים אז להצפין את הdb

## תיאור תחום הידע

### פירוט מעמיק של היכולות שהוצגו בשלב הקודם

צד שרת צריך ליצור חדרים וליצור משתמשים מסוג רגיל ומסוג guides

שם היכולת: ליצור חדרים

מהות: ליצור חדר שבו יהיה אפשר לדבר

אוסף יכולות: לקבל את המשתמש שפותח את החדר ולבדוק אם הוא מסוג guide, אם כן אז ליצור קוד שלא היה קיים לפני זה ולשמור בdatabase עם הip שלו כדי שיהיה אפשר להתחבר

שם היכולת: ליצור משתמשים

מהות: ליצור משתמשים משתי סוגים: רגיל ומדריך (לא חובה להתחבר בתור משתמש רגיל, יהיה אפשר גם להתחבר אנונימית)

אוסף יכולות: לקבל שם משתמש, סיסמה ומייל, לבדוק שלא השתמשו במייל הזה לפני, צריך גם לשמור בdatabase ולתת את הסוג המתאים

שם היכולת: לקבל ולשלוח ip וport של מדריך על פי הקוד

מהות: לקבל ולשלוח ip וport בשביל שיהיה אפשר להתחבר לשיחה

אוסף יכולות: לקבל קוד של חדר כלשהו, לעשות שאילתה לדאטה בייס ולשלוח ip וport של המדריך

שם היכולת: לפתוח חדר חדש

מהות: לקבל בקשה לפתיחת חדר אם הוא מדריך

אוסף יכולות: לבדוק אם הוא מדריך, להוסיף את הקוד לטבלה של החדרים הפועלים, לקבל את הip והport של המדריך ולהכניס גם אותם לטבלה

צד לקוח צריך לדעת לקבל את השיחה ולהשמיע, לשלוח את הקול לכולם אם הוא מדבר, לקבל ולשלוח הודעות ומיקום

שם היכולת: לקבל את השיחה ולהשמיע

מהות: לקבל את השיחה ולהשמיע

אוסף יכולות: לקבל את השיחה ולהשמיע

שם היכולת: לשלוח את הקול לכולם אם הוא מדבר

מהות: לשלוח את הקול לכולם אם הוא מדבר

אוסף יכולות: להקליט ולשלוח

שם היכולת: לקבל ולשלוח הודעות

מהות: לקבל ולשלוח הודעות

אוסף יכולות: לקבל הודעות ולשלוח הודעות ולהראות מי שלח מה ומתי

שם היכולת: לקבל ולשלוח מיקום

מהות: לקבל ולשלוח מיקום

אוסף יכולות: לקבל את המיקום של מי שמשתף את המיקום שלו ולהראות איפה הוא על המפה, לשלוח לכולם את המיקום כשהוא רוצה

אובייקטים נחוצים: ממשק משתמש, הצפנה, תקשורת, שרת, בסיס נתונים

## מבנה / ארכיטקטורה של הפרויקט

1. תיאור הארכיטקטורה של המערכת המוצעת:

למערכת המוצעת יש ארכיטקטורת שרת-לקוח וארכיטקטורתpeer to peer , יצירת המשתמשים נעשית בין המשתמש לשרת, החדרים גם נעשים בין המשתמש לשרת, השרת שומר את כל החדרים הפתוחים ומתאים בין הקוד שהמשתמש מכניס לבין המדריך שפתח את החדר ונותן לו את הip והport הפתוחים כדי שיהיה אפשר לתקשר ביניהם. אפליקציית המדריך פועלת כשרת ואפליקציות המבקרים פועלות כלקוחות בשיטת peer to peer. המדריך יוצר חדר ומקבל קוד ייחודי בו יוכלו המבקרים להשתמש כדי להצטרף לחדר. המבקרים מתחברים לחדר באמצעות הקוד שנשלח לשרת והם מקבלים את הip והport. המדריך יכול לדבר דרך התוכנה. התוכנה משתמשת בפרוטוקול UDP לתקשורת בין המדריך למבקרים כדי שיהיה תקשורת מהירה גם אם חלק מהמידע לא עבר טוב. המערכת משתמשת גם במסד נתונים מקומי של SQLite לאחסון המשתמשים והחדרים הפתוחים.

תיאור החומרה:

המערכת מיושמת באמצעות Python כשפת התכנות העיקרית. PySide6 משמש לבניית ממשק המשתמש. Sockets ופרוטוקול UDP משמשים לתקשורת בין המדריך למשתמש וללקוח. PyAudio משמש להקלטה והשמעת השיחה. SQLite משמש לאחסון משתמשים וחדרים.

(להוסיף תרשים זרימה)

2. תיאור הטכנולוגיה הרלוונטית:

שפת תכנות: פייתון

פריימוורק לגרפיקה: pyside שמשתמש בqt לייצירת גרפיקה קרוס פלטפורם וcss לעיצוב העמודים

תקשורת: socket בשימוש פרוטוקול udp כדי שיהיה סטרים מהיר של מידע בזמן אמת, השתמשתי בjson כדי להעביר את המידע

התוכנה מיועדת לשימוש במוזיאונים כי שם יש רשת סגורה עם קליטה טובה ואנשים בדרך כלל רוצים להסתובב קצת מחוץ לטווח השמיעה של המדריך אבל אפשר להשתמש גם בטיולים בחוץ ובכיתות למקרה שהתלמידים בעבודת כיתה רועשת

3. תיאור זרימת המידע במערכת:

ניתן לחלק את זרימת המידע במערכת לשלוש יכולות עיקריות:

- יצירת חדר: המדריך יוצר חדר ומקבל קוד ייחודי. הקוד נשלח למבקרים.

- הצטרפות מבקרים: המבקרים מתחברים לחדר באמצעות הקוד שמספק המדריך.

- צ'אט: המדריך יכול לדבר למשתמשים, והמשתמשים יכולים לשלוח הודעות טקסט למדריך.

(להוסיף תרשים זרימה)

4. תיאור האלגוריתמים המרכזיים בפרויקט:

שרת:

תהליך הכניסה דורש מהמשתמש לספק את המייל והסיסמה שלו, אשר נשלחים לאחר מכן לשרת לאימות. השרת בודק במסד נתונים את המשתמשים, ואם המשתמש בטבלה השרת מעניק למשתמש גישה למערכת. אם המייל והסיסמה לא תואמים או לא במסד נתונים, השרת ימנע גישה ויבקש מהמשתמש לנסות שוב.

תהליך ההרשמה מקבל את השם וכתובת האימייל של המשתמש וסיסמה ואת סוג המשתמש (מדריך או משתמש). לאחר מכן השרת בודק אם המייל בשימוש ואם לא אז הוא מאחסן את המידע במסד הנתונים ומעניק למשתמש גישה למערכת.

השרת מקשיב לפקודות, אם הפקודה זה "פתח חדר", השרת בודק תחילה אם המשתמש מורשה לפתוח חדר על ידי בדיקה אם המשתמש הוא מדריך, הקוד מקבל את ה-IP וה-port של המדריך, ואז מכניס לטבלה את החדר הפתוח עם ה-IP וה-port ומקשר למדריך ומחזיר את הקוד של החדר.

אם הפקודה זה "סגור חדר", השרת בודק תחילה אם המשתמש מורשה לפתוח חדר על ידי בדיקה אם המשתמש הוא מדריך ואם המשתמש מדריך, הקוד מוחק את נתוני החדר המתאימים ממסד הנתונים.

אם הפקודה היא "קבל מארח לפי קוד", הקוד מחלץ את קוד החדר מההודעה, בודק את נתוני ה-IP וה-port המתאימים של המדריך ממסד הנתונים, ושולח את הנתונים ללקוח.

מדריך:

המדריך מקשיב לכל הודעה ותמיד שולח לכל מי שמחובר את ההקלטה בזמן אמת, אם הוא מזהה שמישהו נכנס הוא שומר את הip והport והשם שלו ברשימה של האנשים שיש בחדר.

אם הוא מקבל שמישהו שולח הודעה הוא מקבל את תוכן ההודעה ומי ששלח ושולח לשאר המאזינים מי שלח את ההודעה ומה תוכן ההודעה.

לקוח:

הלקוח שולח בקשה לip וport שהוא קיבל מהשרת כששלח את הקוד.

כשהמדריך מאשר לו להיכנס הוא תמיד מקשיב למדריך. אם זה שמע הוא משמיע ואם זה json הוא בודק מה התוכן ועושה, בדרך כלל זה הודעה או קריאה לסגירת החדר.

5. תיאור סביבת הפיתוח:

סביבת הפיתוח לפרויקט זה כוללת כלים שונים, כגון Python, PySide6, PyAudio ו-SQLite. הבדיקה נעשית בעיקר בשרת שמשתמש במספר טכניקות אבטחה כדי להבטיח את בטיחות נתוני המשתמש.

ראשית, הקוד משתמש ב-Salted Password hashing כדי לאחסן את הסיסמה של המשתמש בצורה מאובטחת. כאשר משתמש יוצר חשבון, הסיסמה שלו עוברת גיבוב עם רעש (רצף אקראי של בייתים) באמצעות האלגוריתם SHA256. הרעש מתווסף לסיסמה כדי להפוך את ה-hash לבטוח יותר, מכיוון שאותה סיסמה תייצר גרסאות שונות עם רעשים שונים. לאחר מכן ה-hash המתקבל מאוחסן במסד הנתונים, במקום בסיסמה הפשוטה. כאשר בודקים את הסיסמה מצפינים מחדש את הסיסמה ומשווים את הhash ככה אם פורצים למסד נתונים אי אפשר להשתמש בסיסמה או לדעת מה היא הייתה.

שנית, הקוד מאמת את כתובת האימייל של המשתמש לפני יצירת חשבון. כתובת הדוא"ל נבדקת מול ביטוי רגולרי כדי לוודא שהיא בפורמט חוקי. זה עוזר למנוע מתוקפים להחדיר קוד או נתונים זדוניים למסד הנתונים.

שלישית, הקוד משתמש בהצהרות מוכנות כדי למנוע התקפות של הזרקת SQL. הצהרות מוכנות משמשות להפרדה בין קוד SQL לבין קלט המשתמש. זה מונע מתוקפים להחדיר קוד SQL משלהם למסד הנתונים על ידי שינוי קלט המשתמש. בנוסף הקוד משתמש בregex בשביל לבדוק אם יש sql injection.

רביעית, השרת מתקשר בjson ולא בpickle שיש לו בעיות אבטחה כאשר משתמשים בו. זה יכול להפעיל קוד במהלך תהליך הדה-סריאליזציה וניתן להשתמש בו עבור התקפות דה-סריאליזציה.

6. תיאור פרוטוקול התקשורת:

פרוטוקול התקשורת כולל שליחת וקבלת הודעות בין המדריך למבקרים באמצעות פרוטוקול UDP. מבנה ההודעות כולל שדה סוג הודעה, שדה שולח, שדה נמען ושדה תוכן הודעה. ההודעות נשלחות בפורמט בינארי שכשהופכים לסטרינג יש הודעת json שאפשר להפוך לdictionary בעזרת הספרייה json של פייתון.

7. תיאור מסכי המערכת:

המערכת כוללת מסכים שונים כמו מסך הראשי שבו אפשר גם לשים קוד ולהתחבר לחדר, מסך ההרשמה, מסך התחברות, מסך יצירת החדר ומסך השיחה.

8. תיאור מבני הנתונים:

Users:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| id | username | password | email | type | room\_code | host\_ip | host\_port |
| 1 | user1 | gO2v0JW1B | [us1@gmail.com](mailto:user1@example.com) | normal | NULL | NULL | NULL |
| 2 | user2 | hsg9yQVPjW | [us2@gmail.com](mailto:user2@example.com) | guide | A1B2C3 | 192.168.0.1 | 8080 |
| 3 | user3 | P7hEKxrn4J1 | [us3@gmail.com](mailto:user3@example.com) | normal | NULL | NULL | NULL |

Active\_rooms:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| id | room\_code | host\_ip | host\_port | host\_listener\_port |
| 1 | A1B2C3 | 192.168.0.1 | 8080 | NULL |

9. סקירת חולשות והאיומים:

איומים אפשריים למערכת כוללים התקפות Injection SQL, התקפות אתרים, התקפות MITM, והתקפות DDOS.

## מימוש הפרויקט

חלק א:

socket: מודול זה מספק דרך ליצור שקעים ולקיים אינטראקציה עם שקעים.

threading: מודול זה מספק אפשרות להריץ חלקים מהקוד במקביל.

sqlite3: מודול זה מספק דרך ליצור אינטראקציה עם מסד נתונים של SQLite.

hashlib: מודול זה מספק דרך ליצור ולנהל hashs.

hmac: מודול זה מספק דרך ליצור ולנהל HMACs.

json: מודול זה מספק דרך ליצור ולנתח נתוני JSON.

מודולים שפיתחתי:

utils.py

מחלקה: לממש כל מיני פונקציות ומחלקות שצריך במהלך הפרויקט.

תפקיד: מספק פונקציות ומחלקות עבור המערכת.

פעולות במחלקה:

create\_room\_id()

תפקיד: יוצר מזהה ייחודי לחדר.

פרמטרים: אין

הצהרת החזרה: קוד לחדר.

send\_response()

תפקיד: שולח תגובה ללקוח.

פרמטרים: נתוני התגובה וכתובת הלקוח.

הצהרת החזרה: אין.

compare\_passwords()

תפקיד: השוואה בין שתי סיסמאות.

פרמטרים: שתי הסיסמאות.

הצהרת החזרה: נכון אם הסיסמאות תואמות, אחרת לא נכון.

התחברות()

תפקיד: כניסה למשתמש.

פרמטרים: האימייל והסיסמה של המשתמש.

הצהרת החזרה: מזהה המשתמש אם הכניסה מוצלחת, אין אחרת.

get\_room\_code()

תפקיד: מקבל את קוד החדר עבור משתמש.

פרמטרים: מזהה המשתמש.

הצהרת החזרה: קוד החדר למשתמש, אין אם למשתמש אין חדר.

insert\_room()

תפקיד: הוספת חדר למסד הנתונים.

פרמטרים: נתוני החדר.

הצהרת החזרה: אין.

get\_user\_type()

תפקיד: מקבל את סוג המשתמש עבור משתמש.

פרמטרים: מזהה המשתמש.

הצהרת החזרה: סוג המשתמש עבור המשתמש, אין אם המשתמש אינו קיים.

יחסי גומלין:

מודול utils.py משמש את כל שאר התוכנה. מודול server.py הוא המודול הראשי של המערכת והוא מטפל בחיבורים נכנסים מלקוחות. מודול utils.py משמש גם את מודול server.py לשליחת תגובות ללקוחות ולהשוואת סיסמאות. client\_gui.py משתמש ב class\_broadcast.py וב class\_listener.py בשביל להשתמש בתקשורת של החדרים והם גם משתמשים בutils.py והם גם מתקשרים עם הserver.py

חלק ב:

def compare\_passwords(password:bytes, hashed\_password:bytes, salt:bytes):

"""This function compares two passwords.

Args:

password: The first password.

hashed\_password: The hashed version of the second password.

salt: The salt used to hash the second password.

Returns:

True if the passwords match, False otherwise.

"""

# Hash the first password with the same salt.

salted\_password = hashlib.sha256(password + salt).hexdigest().encode()

# Compare the hashed passwords.

return hmac.compare\_digest(salted\_password, hashed\_password)

הפונקציה compare\_passwords() פועלת על ידי הצפנת הסיסמה הראשונה עם אותו רעש ששימש להצפנת הסיסמה השנייה. לאחר מכן הפונקציה משווה את הסיסמאות המוצפנות. אם הסיסמאות תואמות, הפונקציה מחזירה True. אחרת, הפונקציה מחזירה False.

הפונקציה compare\_passwords() מסובכת מכיוון שהיא משתמשת בשני אלגוריתמי הצפנה שונים: hashlib.sha256() ו-hmac.compare\_digest(). האלגוריתם hashlib.sha256() משמש לגיבוב הסיסמאות, בעוד שהאלגוריתם hmac.compare\_digest() משווה את הסיסמאות המוצפנות באותו כמות זמן אם הסיסמה נכונה או לא נכונה.

def receive\_data():

"""This function receives data from the socket and starts a new thread to handle the connection.

Args:

sock: The socket.

Returns:

None.

"""

# Receive data from the socket.

data = b""

while True:

chunk, addr = sock.recvfrom(1024 \* 2)

data += chunk

if len(chunk) < 1024:

# End of message.

break

# Print the data.

print(data, "hi")

# Start a new thread to handle the connection.

threading.Thread(target=handle\_connection, args=(data, addr)).start()

הפונקציה receive\_data() על הזמן מקשיבה לחיבורים ולאחר מכן משרשרת להודעה ובודקת אם זה כל ההודעה שנשלחה, אם לא היא ממשיכה להקשיב עד לקבלת ההודעה המלאה. הפונקציה קוראת לhandle\_connection() שתעבד את הנתונים על thread אחר.

הפונקציה receive\_data() מסובכת מכיוון שהיא משתמשת בשתי טכניקות שונות לטיפול בנתונים: היא משתמשת בלולאה כדי לקבל את הנתונים והיא משתמשת בשרשור כדי לעבד את הנתונים. הלולאה משמשת לקבלת הנתונים מכיוון שהיא לפעמים קיבלה רק חלק מהנתונים ואז השרת לא ידע לעבוד איתם.

sql\_injection\_pattern = re.compile(r"(?:')|(?:--)|(\b(select|update|delete|insert|drop|alter)\b)")

הקוד בודק אם יש sql injection

def send\_udp\_large\_data(sock, data, buffer\_size=1024):

"""This function sends a large amount of data over UDP.

Args:

sock: The socket to send the data over.

data: The data to send.

buffer\_size: The size of each packet to send.

Returns:

None.

"""

# Calculate the number of packets required to send the entire data.

packet\_count = len(data) // buffer\_size

if len(data) % buffer\_size != 0:

packet\_count += 1

# Send the data in pieces.

for i in range(packet\_count):

# Get the current piece of data to send.

start = i \* buffer\_size

end = min(start + buffer\_size, len(data))

data\_chunk = data[start:end]

# Send the current piece of data to the server.

sock.sendto(data\_chunk, (SERVER\_IP, SERVER\_PORT))

הקוד פועל על ידי חישוב תחילה של מספר החבילות שידרשו לשליחת כל הנתונים. זה נעשה על ידי חלוקת אורך הנתונים בגודל המאגר. אם אורך הנתונים אינו כפולה של גודל המאגר, אזי תידרש מנה נוספת.

לאחר שחושב מספר החבילות, הקוד שולח את הנתונים בחתיכות. פיסת הנתונים הראשונה נשלחת על ידי קבלת פיסת הנתונים הנוכחית לשליחה, מה שנעשה על ידי קבלת האינדקס של הביט הראשון בנתונים והאינדקס של הבית האחרון בנתונים. פיסת הנתונים הנוכחית נשלחת לאחר מכן לשרת באמצעות שיטת sock.sendto() .

לאחר מכן הקוד ממשיך לשלוח את הנתונים בחתיכות עד שכל הנתונים נשלחים.

חלק ג:

sql\_injection\_pattern

מטרה: לבדוק התקפות הזרקת SQL.

מה הוא עושה: הביטוי הרגולרי מחפש התאמה לדפוס ספציפי של תווים שיכול לשמש להחדרת קוד זדוני לשאילתת SQL.

תוצאות: אם נמצא התאמה, הפונקציה תעלה חריגה.

email\_regex

מטרה: לבדוק אם כתובת דוא"ל תקפה.

מה הוא עושה: הביטוי הרגולרי מחפש התאמה לתבנית ספציפית של תווים שתקפים בכתובת דוא"ל.

תוצאות: אם נמצאה התאמה, הפונקציה מחזירה True. אם לא נמצא התאמה, הפונקציה מחזירה False.

login()

מטרה: לבדוק אם סיסמת המשתמש נכונה.

מה היא עושה: הפונקציה לוקחת שני ארגומנטים: סיסמת המשתמש והסיסמה הגובבת המאוחסנת במסד הנתונים. הפונקציה משווה את הסיסמה של המשתמש לסיסמת הגיבוב.

תוצאות: אם הסיסמאות תואמות, הפונקציה מחזירה את מזהה המשתמש. אם הסיסמאות אינן תואמות, הפונקציה מחזירה ללא.

compare\_passwords()

מטרה: להשוות סיסמה של משתמש לסיסמה הגובבת המאוחסנת במסד הנתונים.

מה היא עושה: הפונקציה לוקחת שני ארגומנטים: סיסמת המשתמש והסיסמה הגובבת המאוחסנת במסד הנתונים. הפונקציה משתמשת בפונקציית hash קריפטוגרפית כדי להמיר את סיסמת המשתמש לסיסמה hashed. לאחר מכן, הפונקציה משווה את הסיסמה הגובבת לסיסמת הגיבוב המאוחסנת במסד הנתונים.

תוצאות: אם הסיסמאות הגובבות תואמות, הפונקציה מחזירה True. אם הסיסמאות הגובבות אינן תואמות, הפונקציה מחזירה False.

get\_room\_code()

מטרה: לבדוק אם למשתמש יש קוד חדר.

מה היא עושה: הפונקציה לוקחת ארגומנט אחד: מזהה המשתמש. הפונקציה מבצעת שאילתות במסד הנתונים כדי לראות אם למשתמש יש קוד חדר.

תוצאות: אם למשתמש יש קוד חדר, הפונקציה מחזירה את קוד החדר. אם למשתמש אין קוד חדר, הפונקציה מחזירה ללא.