

פרויקט רשתות מחשבים ואינטרנט - Kahoot

מגישה: גל גרין

תז: 214883415

מערכת הפעלה:

מהדורה Windows 11 Pro

גירסה H222

הותקן ב 06/11/2023

גירסת Build של מערכת ההפעלה 22621.3155

חוויה Windows Feature Experience Pack 1000.22684.1000.0

שפת תכנות: פייתון

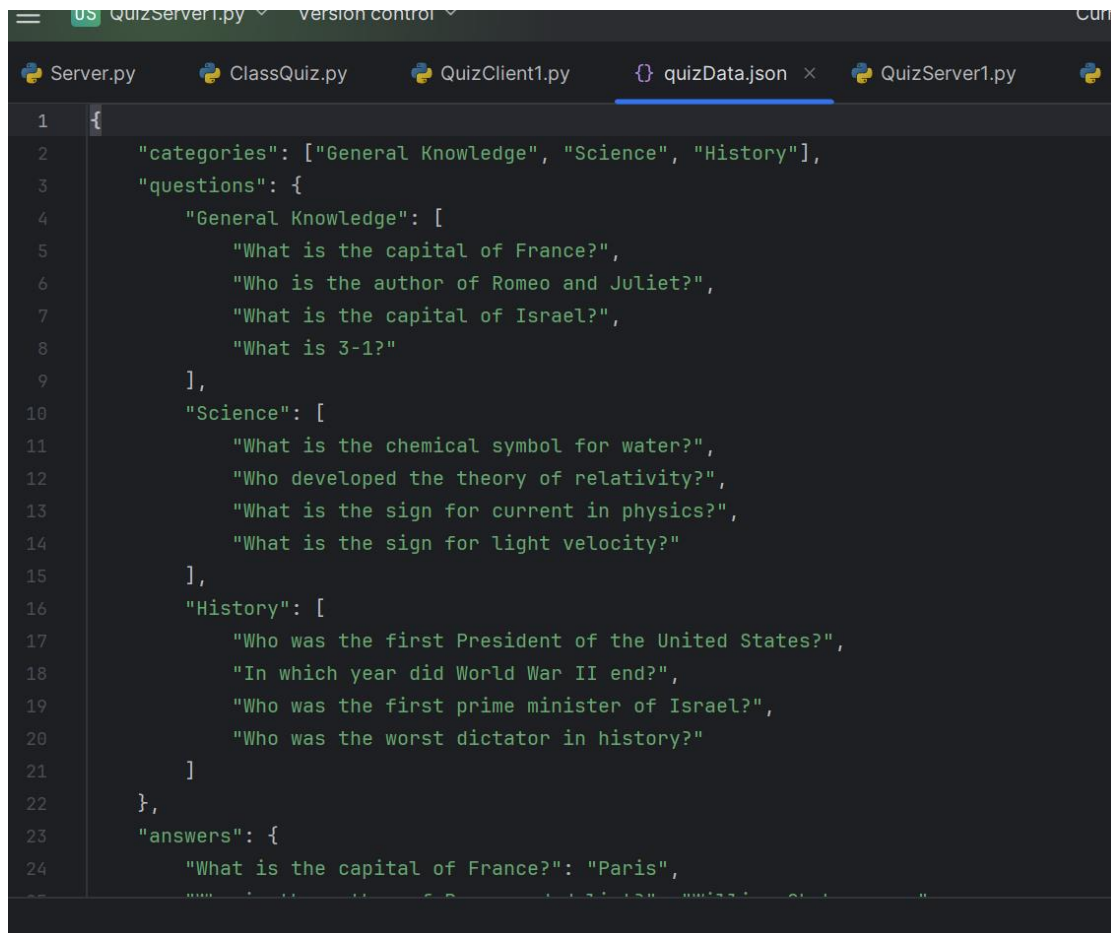
גרסא: תכנתתי ב: PyCharm Community Edition 2023.2.4

הסבר כללי של הקוד שכתבתי:

למעשה הוא מחולק לארבעה קבצים:

קובץ ראשון: quizData.json

קובץ זה הוא הדטא בייס של השאלון- כלומר הוא מכיל את כל הקטגוריות, השאלות, התשובות הנכונות ואופציות לתשובות.



```
1 {
2   "categories": ["General Knowledge", "Science", "History"],
3   "questions": {
4     "General Knowledge": [
5       "What is the capital of France?",
6       "Who is the author of Romeo and Juliet?",
7       "What is the capital of Israel?",
8       "What is 3-1?"
9     ],
10    "Science": [
11      "What is the chemical symbol for water?",
12      "Who developed the theory of relativity?",
13      "What is the sign for current in physics?",
14      "What is the sign for light velocity?"
15    ],
16    "History": [
17      "Who was the first President of the United States?",
18      "In which year did World War II end?",
19      "Who was the first prime minister of Israel?",
20      "Who was the worst dictator in history?"
21    ]
22  },
23  "answers": {
24    "What is the capital of France?": "Paris",
25    "Who is the author of Romeo and Juliet?": "William Shakespeare",
26    "What is the capital of Israel?": "Jerusalem",
27    "What is 3-1?": "2",
28    "What is the chemical symbol for water?": "H2O",
29    "Who developed the theory of relativity?": "Albert Einstein",
30    "What is the sign for current in physics?": "A",
31    "What is the sign for light velocity?": "c",
32    "Who was the first President of the United States?": "George Washington",
33    "In which year did World War II end?": "1945",
34    "Who was the first prime minister of Israel?": "David Ben-Gurion",
35    "Who was the worst dictator in history?": "Adolf Hitler"
```

הקובץ אכן מכיל רק מידע על קטגוריות שאלות ותשובות.

כאשר נרצה לגשת לשאלה או לקטגוריה וכו' בשאלון בעצם נטען את המידע מקובץ זה לאחר שנפתח אותו.

```

4
5
6 # read the question data and save him to variables
7 with open('quizData.json', 'r') as file:
8     quiz_data = json.load(file)
9
10 categories = quiz_data['categories']
11 questions = quiz_data['questions']
12 answers = quiz_data['answers']
13 options = quiz_data['options']
14
15

```

טעינת המידע.

קובץ שני: ClassQuiz.py

למעשה קובץ זה הוא מימוש של שאלון כללי.

ראשית, בשאלון זה אני קוראת מקובץ הדטא בייס של השאלות- אני טוענת לתוך משתנים מתאימים את הקטגוריות, השאלות, התשובות הנכונות והתשובות האפשריות.

תפקיד הקובץ העיקרי הוא:

פתחתי Class עם תכונות של שאלון: מי המשתתפים בשאלון, מה קטגוריית השאלון ומהם הציונים של המשתתפים במשחק, מהו המספר המזהה של המשחק, האם המשחק התחיל או שלא ועוד תכונות שנדרשות להפעלת המשחק כמו מהי השאלה הנוכחית ומהי התשובה שמתאימה לה וכו'.

למעשה בכל פעם שאפתח שאלון אצור אובייקט חדש מסוג Class quiz .

בנוסף בניתי פונקציות עזר כמו : לצרף שחקן חדש לשאלון, לקבל שאלה שדואגת לספק שאלות באופן רנדומלי בלי חזרה על שאלות ומספקת ביחד עם זה את התשובה הנכונה ואופציות התשובה, וגם פונקציה לנהל את ציוני המשתתפים ועוד..

קובץ שלישי: QuizServer1.py

למעשה זה קובץ השרת- הוא דואג לכל התנהלות השרת: מי מתחבר לשרת (מאפשר להתחבר רק אם משתמש מתחבר עם יוזרניימ חדש שלא תפוס), שהשרת יקשיב כראוי ופשוט דואג להתנהלות השרת מול הלקוח (handle client func) ודואג לטפל בשגיאות התחברות לשרת.

נשים לב שלכל לקוח שמתחבר לשרת נפתח thread חדש כלומר מתאפשר טיפול באופן מקביל ללקוחות שמתחברים בו זמנית.

נשים לב שגם בעצם הלקוחות מתקשרים אחד עם השני דרך הסרבר: אם אדמין רוצה להתחיל משחק שעוד משתתפים מחוברים אליו הוא ישלח הודעה לסרבר על כך והסרבר ידאג בעזרת ברודקסט לשלוח את השאלות לכל המשתתפים.

פירוט קטן על ברודקסט: למעשה פונקציה הברודקסט שבניתי שולחת את ההודעה המתאימה (שהיא מקבלת כפרמטר) לכל המשתתפים בשאלון המתאים (שהיא מקבלת כפרמטר).

למעשה ברודקסט לא שולחת הודעה לכל קליינט שמחובר לשרת אלא רק לקליינטים שמחוברים למשחק הספציפי שבו מעוניינים.

הסבר נוסף ומעמיק יותר על התנהלות הסרבר:

הסרבר כל הזמן מאזין לחיבורים ומקבל כל חיבור. יכול לטפל בכמה חיבורים במקביל- מוגבל על ידי ערך הדיפולט של לקבל חיבור לסרבר (של הפונקציה).

לכל משתמש חדש הסרבר פותח טרד חדש שמפנה לפונק: `handle client`

פונק זו מקבלת שם משתמש מהלקוח- אם הוא תפוס על ידי משתמש שמחובר לסרבר היא שולחת הודעה על כך ללקוח ואם השם לא תפוס שולחת ללקוח הודעה שהשם בסדר ומוסיפה את הלקוח לרשימת הלקוחות שמחוברים לסרבר (רשימה גלובלית).

לאחר מכן הסרבר מקבל מה הלקוח בחר- להתחיל משחק חדש או קיים.

אם הלקוח בחר להתחיל משחק חדש הסרבר יאזין לאיזה קטגוריה הלקוח בחר ויפתח לו שאלון חדש על ידי פונק: `start new quiz` ולפי התז שהמשחק קיבל שומר את המשחק ברשימת המשחקים הפעילים בסרבר (גם רשימה גלובלית).

פונק: `start new quiz` : בונה אובייקט חדש מסוג `class quiz` ושולחת את המספר המזהה שלו ללקוח ואז מחכה להודעה מהקליינט להתחיל את השאלות. כאשר מקבלת הודעה זו עושה ברודקסט לכל המשתתפים בשאלון שהמשחק מתחיל.

לאחר מכן עושה ברודקסט של השאלות כל עוד נשארו שאלות בשאלון שלא הוצגו ויחד איתן את אופציות התשובה והתשובה הנכונה.

כאשר נגמרו השאלות נשלחת הודעה על כך לקליינט וכתגובה מקבלים ממנו את הציון של האדמין (את שאר הציונים מקבלים לא בפונקציה) ומוסיפים אותו למילון הציונים של המשתתפים. מחכים זמן שהיה על מנת שבוודאות כל הציונים התקבלו מהלקוחות ואז שולחים את מילון הציונים לאדמין.

אם הלקוח בחר להתחבר למשחק קיים: הסרבר מקבל מהלקוח את המספר המזהה של המשחק אליו הוא רוצה להתחבר. אם אכן יש משחק כזה: אם המשחק כבר התחיל הוא שולח הודעה שהמשחק כבר התחיל ללקוח ואם לא אז הוא מצרף את הלקוח לשחקני המשחק ושולח הודעת אישור ללקוח, אם המשחק לא קיים אז הוא שולח הודעה על כך ללקוח שסוגר את החיבור ואת המשחק ללקוח.

כעת הפונק מחכה שהמשחק ייגמר וכאשר הוא נגמר היא מקבלת מכל הקליינטים (שהם לא אדמינים) את הציון שלהם ומוסיפה אותו למילון הציונים ולאחר זמן השהיה על מנת שבוודאות כל הציונים התקבלו מהלקוחות ואז שולחים את מילון הציונים ללקוח.

בסוף המשחק מוחקים לגמרי את המשחק: מוחקים את משתתפיו מהמשתתפים שמחוברים לסרבר ומוחקים את המשחק ממילון המשחקים הנוכחיים.

קובץ רביעי: QuizClient1.py

למעשה קובץ זה הוא מימוש של לקוח כללי שמתייחס לכל האפשרויות של לקוח יכול להיות לבחור.

בקובץ זה פתחתי Class של לקוח:

כל משתתף חדש שירצה להשתתף במשחק יוצר אובייקט חדש מסוג class client .

לאובייקט יש תכונות של לקוח: מהו השם משתמש של הלקוח, האם הוא אדמין של משחק או שלא (עם הגדרה דיפולטיבית שכן ובמקרה הצורך שמתחברים למשחק קיים משתנה ללא) וכו'.

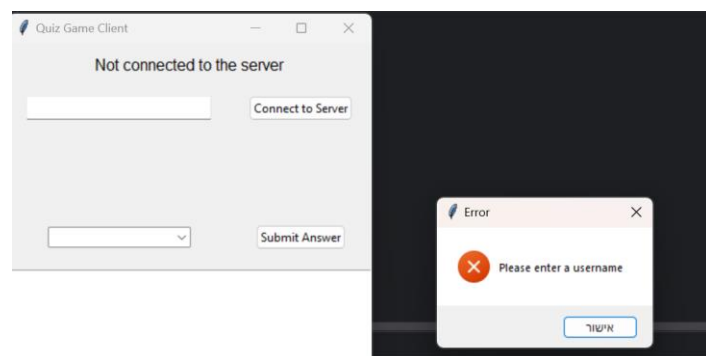
בנוסף בניתי פונקציות עזר כמו : לצרף שחקן חדש לשאלון, לקבל שאלה שדואגת לספק שאלות באופן רנדומלי בלי חזרה על שאלות ומספקת ביחד עם זה את התשובה הנכונה ואופציות התשובה, וגם פונקציה לנהל את ציוני המשתתפים ועוד..

הסבר נוסף ומעמיק יותר על התנהלות לקוח:

למעשה הלקוח אחראי על ה GUI של המשחק:

כאשר מריצים את הלקוח ראשית מציגים כפתורים ולייבלים:

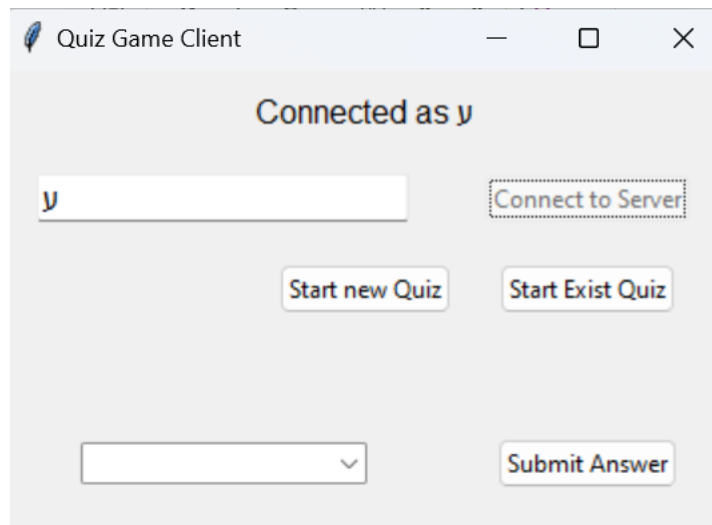
לייבל של לא מחובר לשרת וכפתור שדורש להקליד שם משתמש על מנת להתחבר לשרת(אם ינסו ללחוץ על כפתור התחברות כאשר לא הקלידו שם משתמש תקפוץ הודעת נא להקליד שם משתמש). כפתור ההתחברות למעשה מנתב לפונק: connect to server



הערה קטנה: במסך זה יש כפתור איקס שכאשר הוא נלחץ : החיבור ייסגר והמסך ייסגר. נשים לב שגם אם המשחק אליו הקליינט מחובר בפעולה המסך ייסגר בלי להפריע לשאר המשתמשים לא משנה אם הוא אדמין או לא.

לאחר שכפתור ההתחברות נלחץ בהצלחה: הפונק בודקת מול הסרבר האם שם המשתמש תקין כלומר האם הוא לא תפוס על ידי משתמש אחר שכבר התחבר לסרבר. אם הוא לא תקין קופצת הודעה של שם משתמש תפוס ומסך המשחק נסגר לאחר כמה שניות.

אם השם משתמש תקין אז יוצג במסך טקסט של מחובר כ {שם משתמש} ויוצגו שני כפתורים: לבחור במשחק חדש או קיים ומנטרלים את האופציה לבחור שוב בכפתור התחברות לסרבר כי אין צורך לגעת בו שוב.



הערה קטנה: אם יש שגיאה בשרת או בהתחברות תקפוץ הודעת שגיאה : לא מסוגל להתחבר לשרת והמשחק ייסגר.

בנוסף כעת נפתח טרד חדש לכל קליינט שמטפל בקבלת הודעות מהסרבר- הטרד מנתב לפונק receive messages שתפקידה הוא להאזין כל הזמן להודעות ולפעול בהתאם למה שכתוב בהן.

הערה קטנה: אם במהלך המשחק יש בעיה בתקשורת אז יקפוץ טקסט שהחיבור נסגר והחיבור ייסגר והמסך ייסגר.

פונק : receive messages

ראשית הפונק מחכה שהלקוח יבחר משחק חדש או קיים (כאשר לוחצים על אחד מהכפתורים אז הפונק שאליהם הם מנתבים מגדירות בתכונה של הקליינט שבחרנו).

Quiz Game Client

Connected as y

y

Select a Category

Science

ולאחר השליחה יופיע המספר המזהה של המשחק:

Quiz Game Client

Connected as y

y id of the game:1145

Select a Category

Science

אם הלקוח לא אדמין כלומר הוא בחר להצטרף למשחק קיים אזי הלקוח ישלח את המספר המזהה של המשחק אליו הוא רוצה להצטרף לסרבר שיבדוק אם קיים משחק כזה או לא.

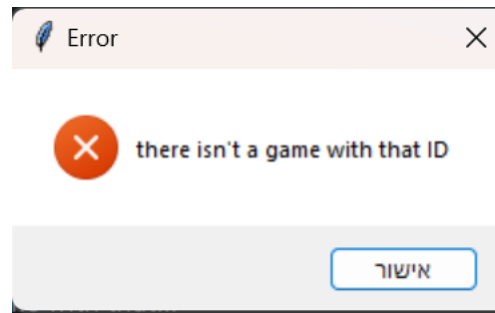
Quiz Game Client

Connected as '

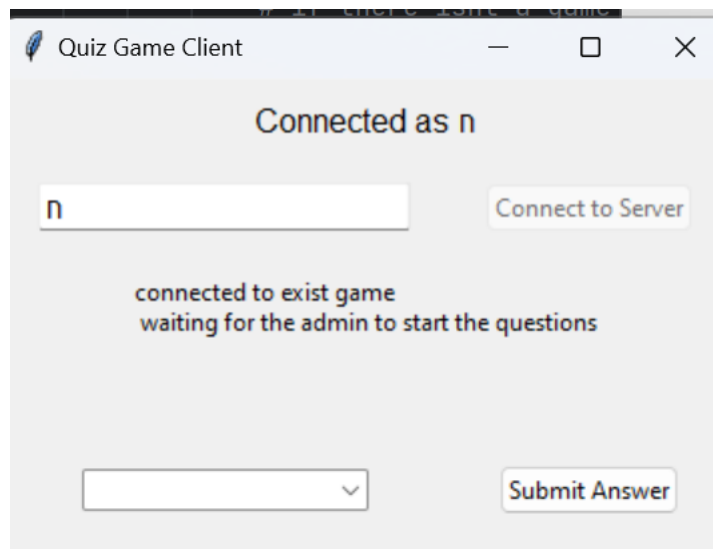
'

entry game ID

המשתמש מקליד את המספר המזהה של המשחק ושולח אותו ומחכה לתגובה.
אם הלקוח יקבל הודעה שלא קיים משחק כזה תקפוץ הודעת שגיאה שלא קיים משחק כזה
והחיבור ייסגר והמסך ייסגר.



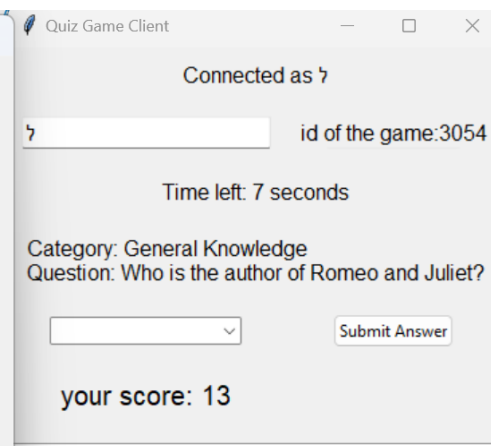
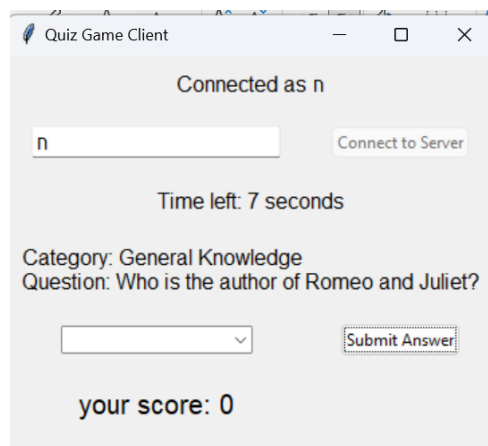
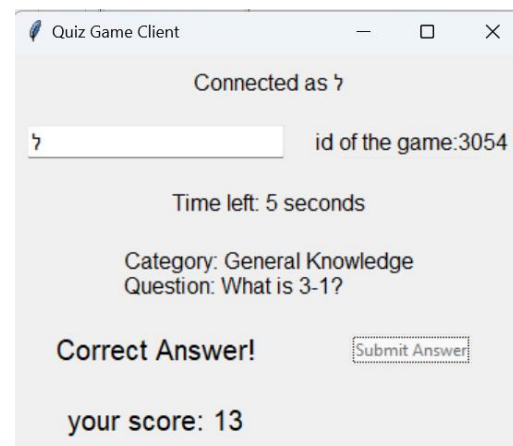
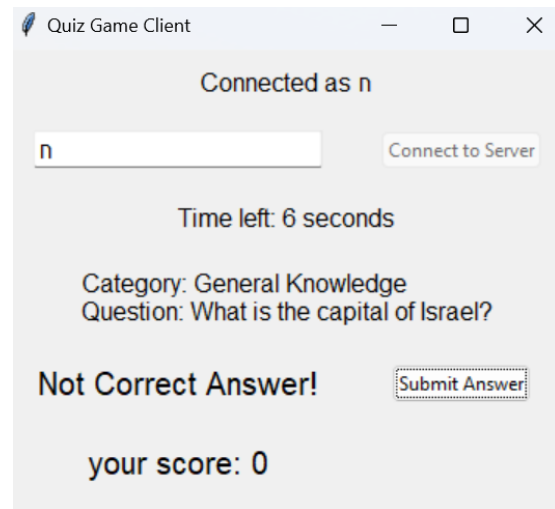
אם כן קיים משחק כזה אזי הסרבר גם בדק אם המשחק כבר התחיל או שלא: אם המשחק
התחיל תקפוץ לקליינט הודעה שהמשחק כבר התחיל והחיבור ייסגר והמסך ייסגר אחרת
יציג הודעה שההתחברות למשחק בוצעה וכעת מחכים שהאדמין יתחיל את שאלות המשחק.

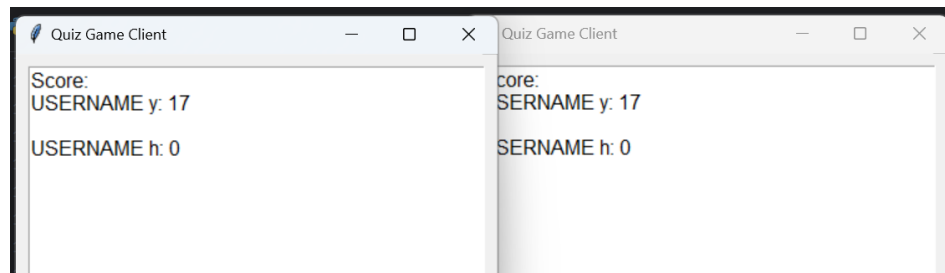


כעת הלקוח ממשיך להאזין עד לקבלת הודעה מהסרבר שהמשחק מתחיל ובהתאם משנה
את ה GUI ומקבל מהסרבר את השאלות האופציות והתשובה הנכונה עד שנגמרות
השאלות. אם הלקוח בחר תשובה נכונה יוצג לו טקסט של תשובה נכונה ובהתאם לכמה
מהר ענה יתעדכן לו הציון ואם הלקוח בחר תשובה לא נכונה יוצג לו טקסט של תשובה לא
נכונה ולא יתעדכן לו הציון(יישאר כפי שהוא).

כך עד שיקבל הודעה מהסרבר שהמשחק נגמר ואז יציג טקסט על המסך שהמשחק נגמר
ואז לאחר זמן השהייה יקבל מילון ציונים מהסרבר. כאשר קיבל את המילון הוא יראה על
המסך רק את מילון הציונים לאחר שמילן אותו ולאחר כמה שניות יסגור את טבלת הציונים.
כמו בקהוט אמיתי טבלת הציונים המלאה מופיעה אצל האדמין.

מצורפות תמונות ממשחק של שני משתתפים: מצב של תשובה נכונה, תשובה לא נכונה ובסוף טבלת הציונים הממויינת.





הצילום הזה הוא ממשחק שונה

שאלות תיאורטיות:

אבטחה:

1. בעיות אבטחה שעלולות להיות:

התקפות זדוניות שניתן לפתור על ידי אימות קלטים קפדני של המשתמשים שרוצים להתחבר .

עוד בעיה שעלולה להיות היא קיבוע: משתמשים התחברו למשחק אך לא התחילו אותו וסתם מבזבזים משאבים לאורך זמן רב. ניתן לפתור זאת על ידי הקצאת זמני time-out: להקצות זמן שאם המשתמש לא עשה כלום הוא אוטומטית ינותק מהמשחק.

בעיה נוספת היא גניבת מידע על המשתמשים אז מה שניתן לעשות הוא לאחסן את המידע עליהם באופן מאובטח.

בנוסף בתקיפות זדוניות יכולים לפגוע בנתוני השאלונים אזי פתרון אפשרי הוא פתרון שמימשי בקוד: לממש דטא בייס לשאלות שכל המידע נמצא בו ובתוכנית לא ניגשים אליו כלל אלא רק טוענים מידע ממנו שנמצא בקובץ נפרד וכך גם לא נפגע במידע השאלון בטעות.

2.

הסיכונים שמתלווים להעברת הודעות כפשוט טקסט הן בעיות אבטחה! עלולים להאזין להודעות שנשלחות ואם ההודעות נשלחות פשוט כטקסט יכולים לקרוא אותן וגם לחשוף מידע רגיש וגם לפגוע בהתנהלות התוכנית בהתאם.

בנוסף אם ההודעות הן פשוט טקסט אנשים שמאזינים יכולים גם לשנות את הטקסט ולפגוע בתוכנית.

על מנת לפתור בעיות אלה ניתן להשתמש בהצפנה: כך רק מי שיש לו את מפתח ההצפנה יוכל לקרוא את ההודעות וכך גם המידע יישאר חסוי ובנוסף בהצפנה יש סימנים דיגיטליים שמוודאים אמינות של ההודעה וכך ניתן לבדוק אם מישהו התעסק עם ההודעה.

Scalability:

1. על מנת להתמודד עם חיבור במקביל של מספר רב של משתמשים יכול לגרום ל over load על הסרבר ולתפוקה גרועה כלומר המשחק יהיה איטי ותקוע. בנוסף יש בכלל בעיה של הגבלה של לכמה משתמשים הסרבר יכול להאזין בו זמנית. פיתרון אפשרי הוא למעשה לחלק את העומס לכמה סרברים שיתקשרו ביניהם.

גישה לדטא בייס יכול להוות צוואר בקבוק כאשר משתמשים רבים מאוד רוצים לגשת בו זמנית אליו אז נבחר בדטא בייס שיודע להתנהל טוב מול מספר רב של משתמשים.

2. על מנת לחלק את העומס על משאבי השרת נוכל להשתמש בפונקציות שאנו מכירים ממערכות הפעלה, כגון:

Round robin, least connections, least response time and more...

אמינות ורמת סבילות לתקלות:

1. על מנת להבטיח את מהימנות היישום שלי בכל פונקציה שכתבתי השתמשתי ב try ולאחר מכן ב except כך שכל חריגה מתאימה תיתפס בפונק מתאימה כך שגם אם אפילו השרת נופל והחיבור מתנתק מה שיקרה זה : תודפס הודעת שגיאה והמשחק ייסגר ללקוח.

2. על מנת ליישם התמדה בהודעות אפילו במידה והשרת עושה ריסטרט מה שאוכל לעשות הוא : לאחסן את ההודעות במערכת תור שמאחסנת את ההודעות באופן קבוע כך שגם אם השרת עושה ריסטרט ההודעות נשמרות.

אימות משתמש:

1. ישנה חשיבות רבה לאימות משתמשים:
ראשית למנוע גישות לא מורשות כגון גישה של משתמש לא מורשה למידע חסוי ובנוסף מאפשר לתכננה לזהות כל משתמש בקלות ולהבדיל בין משתמשים.
על מנת לאמת משתמשים באופן מאובטח ניתן : לבקש סיסמא כך שלכל שם משתמש יש סיסמא ייחודית תואמת משלו.

2. על מנת לנהל רישום סיסמאות ומשתמשים כדי שלמערכת תהיה אבטחה טובה יותר מה שהייתי עושה הוא:

ראשית הגבלת הקלטים: לשים הגבלות ברורות על איזה מידע ניתן להקליד.
בנוסף הייתי מוודאה שלכל משתמש שמתחבר מוקצה מזהה ייחודי לו.
ואם היו שדות כגון טלפון ומייל הייתי מאשרת את קיומם ושיוכם ללקוח על ידי שליחת הודעה שהלקוח צריך לפתוח מהם ולהקליד במערכת את הקוד שהתקבל בהודעה.

התמודדות עם לקוחות במקביל וסנכרון:

1. האתגרים עם התחברות כמה לקוחות במקביל לשרת הם:
לוודא שתהליכים של לקוחות שונים לא יתנגשו אחד עם השני, חלוקת משאבים של השרת באופן אפקטיבי בין כל הלקוחות וכאשר המשאבים משותפים כי בסופו של דבר יש שרת אחד יכולים להיווצר מצבי מירוך שיפגעו ביציבות השרת

2. על מנת למנוע מצבי מירוך ופגיעה במידע מה שניתן לעשות הוא לדאוג לסנכרון threads ואבטחה בצורה הבאה:

ראשית להשתמש במנגנוני נעילה על מנת לנהל את הגישה למשאבים משותפים: לנעול את המידע לפני שניגשים למידע משותף ולשחרר אותו בסוף הגישה. ובנוסף לנסות לזהות תהליכים אטומיים ולהשתמש בהם. נוסף על כך אפשר לשים מגבלה על כמות הטרדים שיכולים לרוץ במקביל מה שיעזור מאוד לנהל את הגישה למשאבים המשותפים.

:Protocol Design

1. השתמשתי בפרוטוקול TCP וזאת מכיוון ש:
שאלון הוא לא תהליך שדורש תשובה מיידי, אם יש השהייה קטנה עד לקבלת תשובה והתנהלות השאלון אז זה בסדר ולכן בחרתי להשתמש בפרוטוקול שאמין יותר וטיפה איטי יותר כיוון שאם מהירות לא קריטית עדיף להשתמש בפרוטוקול אמין יותר.
2. TCP handshake :
תהליך שמתקיים על מנת ליצור חיבור בין לקוח לשרת:
הלקוח שולח לשרת הודעת בקשת התחברות, השרת מגיב בהודעה שהוא מקבל את הבקשה ומוכן ליצור חיבור ואז הלקוח מגיב באישור ונוצר חיבור אמין.

:Message Ordering and Delivery

1. הודעות עלולות להגיע ליעדן בסדר שונה מהסדר בו הן נשלחו, על מנת לפתור בעיה זו אתן לכל חבילה שיוצאת את מספר השליחה שלה וכך היעד יידע להרכיב את ההודעה השלמה לפי הסדר.
בנוסף חלקי הודעות עלולים להיאבד בדרך אז פתרון אפשרי הוא לחכות לקבלת אישור מהיעד שההודעה הגיעה בשלמותה (ניתן לדעת גם לפי מספרי החבילות וגם לפי סיגנל של הודעה אחרונה בשליחה זו) ואם לא התקבלה הודעה זו לשלוח שוב את כל ההודעה.
2. על מנת להבטיח שההודעות מועברות בצורה מהמינת ובזמן מה שניתן לעשות הוא: השולח מצפה להודעת אישור מהיעד שאכן החבילה התקבלה ותקינה ואם כעבור זמן מסוים לא התקבלה הודעה זו השולח ישלח שוב את ההודעה.

:Persistent Storage

1. החשיבות של אחסון מתמשך בשרת מוודאת שהמידע במערכת לא נעלם במקרה של תקלה כמו ריסטרט וכו'.
בתוכנה שלי על מנת לטפל בכך בניתי דטא בייס לשאלון שמכיל את כל המידע ששאלון צריך ונשמר גם במקרה של תקלה בקובץ נפרד לגמרי.
2. ישנם כמה סוגי אחסון בתוכנה שלי:
משתנים גלובליים שמשותפים לכולם (בעיקר רשימות) : נוחים כי הם קלים לגישה אך ישנו מידע רב שקליינטים רוב הזמן לא צריכים וסתם מאחסנים ללא צורך.

משתנים לוקליים: רק הקליינטים שזקוקים להם ניגשים אליהם כלומר אין שמירה
מיותרת שלהם אך במידה ורוצים לשנות את התוכנית צריך מאוד לשנות אותם
בהתאם.
קובץ דטא בייס: מאחסן את כל מידע השאלון ושומר עליו כיוון שרק קוראים ממנו ולא
משנים את תוכנו וכך לא יהרסו אותו בטעות.