|  |  |  |
| --- | --- | --- |
| **תאריך הגשה:** | | **קבוצה 9** |
| **אימייל** | **ת.ז** | **שם** |
| [haimazu4@gmail.com](mailto:haimazu4@gmail.com) | 203374855 | חיים אזולאי |
| [hodayamekonen@gmail.com](mailto:hodayamekonen@gmail.com) | 204632988 | הודיה מקונן |
| [nastyak6@gmail.com](mailto:nastyak6@gmail.com) | 321241192 | אנסטסיה קוקין |
| [barkatz0610@gmail.com](mailto:barkatz0610@gmail.com) | 315818567 | בר כץ |
| [rinat20968@gmail.com](mailto:rinat20968@gmail.com) | 316237577 | רינת סטודנץ |
| [roi.amar@e.braude.ac.il](mailto:roi.amar@e.braude.ac.il) | 203118666 | רועי עמר |

**G9\_Answers.Ass3**

****

**סעיף א:**

דילמה מס.1

בדיקת כמות מבקרים בפועל אל מול מס' המבקרים שמצויין בהזמנה- אם הגיעו בפועל פחות אנשים מההזמנה. או אם הגיעו בפועל כמות גדולה יותר.

* אופציה מס.1 – לגבי כמות אנשים קטנה מהמוזמן . אם בהזמנה מצויין מס' מבקרים ובפועל הגיעו פחות אנשים אז נבצע גבייה ע"פ ההזמנה ועדכון של כמות המבקרים בפועל.
* אופציה מס. 2- אם בהזמנה מצויין מס' מבקרים ובפועל הגיעו יותר אנשים אז נבצע התייחסות ובדיקה של יתר האנשים ע"פ כניסה של מבקר מזדמן. כלומר , נציג השירות יבצע עבור המבקרים שאינם רשומים מראש הזמנה חדשה (והם לא יזכו להנחות של הזמנה מראש)
* אופציה מס.3 – ההזמנה עבור ביקור מזדמן נוצרת באופן אוטומטי שקוף למשתמש.

החלטה: ההחלטה שנבחרה היא שילוב של אופציות 1 ו3. אופציה מס.2 מהווה פתרון מסורבל ולא יעיל . במקום שנציג בפארק יבצע הזמנה רק עבור מספר האנשים העודף , ניתן לבצע זאת בלחיצת כפתור מבלי לבצע תהליך ארוך ומסורבל.

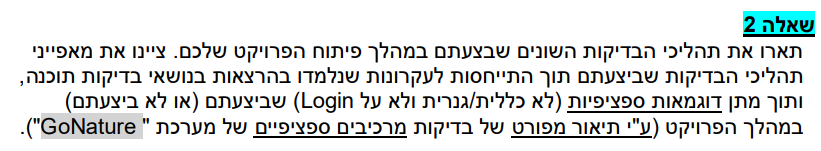
דילמה מס.2

תהליך בקרת הכניסה לפארק של מבקר שהוא מנוי. מבקר שהינו מנוי – יכולות להיות ברשותו מספר הזמנות בעבר ובעתיד- כיצד נדע לזהות מהי הזמנתו כרגע בעת הכניסה לפארק?

* אופציה מס.1 - לבדוק לפי מספר הזמנה ולראות האם הזמנה היא לעכשיו
* אופציה מס.2 – לבדוק לפי מספר מנוי ולראות האם קיימת הזמנה לעכשיו

החלטה : ההחלה שנבחרה היא לייצר מנגנון שבו בודקים או לפי מסר הזמנה או לפי מספר מנוי או לפי מס. ת.ז של הלקוח אם קיימת לו לDB הזמנה לעכשיו. ההחלטה נבעה מכך שיתכן שלקוח לא יזכור את אחד מהפרטים המזהים שלו ועל מנת שתהיה לנו דרך לזהות אותו בכל אחת מהאופציות השונות.

**סעיף ב: רועי !!!!!!!!!**



נתאר את בדיקות המערכת השונות שביצענו במרכיבים השונים של הפרויקט :

ביצענו עבור כל רכיב GUI בדיקות פונקציאונליות ובדיקות שפיות שכללו בדיקות חיוביות ושליליות. ביצענו בדיקות שילובים(אינטגרציה) בין רכיבי הפרויקט השונים וביצענו בדיקות UNIT Test

נתאר מספר דוגמאות ספציפיות כעת :

בדיקות חיוביות-

עבור מסך של מנהל פארק-

|  |  |  |
| --- | --- | --- |
| שם הבדיקה | תאור | תוצאה צפויה |
| 1. הזנה של הנחת מנהל פארק | לחץ על כפתור הזנת הנחה והזן מספר בטווח של בין 0 ל100 (אחוזים) | קבלת חיווי שההודעה נקלטה בהצלחה ונשלחה למנהל המחלקה |
| 1. הזנה של מכסה למבקרים מזמינים מראש | לחץ על כפתור קביעת כמות מזמינים מראש והזן מספר כלשהו | קבלת חיווי שההודעה נקלטה בהצלחה ונשלחה למנהל המחלקה |
| 1. הזנה של מכסה מבקרים בפארק | לחץ על כפתור קביעת כמות מבקרים והזן מספר כלשהו | קבלת חיווי שההודעה נקלטה בהצלחה ונשלחה למנהל המחלקה |
| 1. הצגת דוח ביקורים לפי סוגים | בחר תאריכים רצויים להצגת דוח הביקורים מפולח לפי סוגי מבקרים | הגרף מוצג היטב . המידע המופיע , מופיע בצורה ויזואלית טובה , שמות הצירים נכונים , המספרים תואמים למידע שיש בDB ומוצגים היטב בגרף |
| 1. הצגת דוח שימוש בפארק – כמות מבקרים מפולח לפי קפסולות של השעות | בחר תאריכים רצויים להצגת דוח השימוש מפולח לפי שעות | הגרף מוצג היטב . המידע המופיע , מופיע בצורה ויזואלית טובה , שמות הצירים נכונים , המספרים תואמים למידע שיש בDB ומוצגים היטב בגרף |
| 1. הצגת דוח הכנסות | בחר חודש להצגת דוח הכנסות המפולח לפי הכנסה יומית | הגרף מוצג היטב . המידע המופיע , מופיע בצורה ויזואלית טובה , שמות הצירים נכונים , המספרים תואמים למידע שיש בDB ומוצגים היטב בגרף |

בדיקות שליליות :

עבור מסך של יצירת הזמנה חדשה-

|  |  |  |
| --- | --- | --- |
| שם הבדיקה | תאור | תוצאה צפויה |
| 1. בדיקת שדות ריקים | השאר שדות ריקים ובצע הזמנה | תופיע הודעה : אחד או יותר מהשדות ריקים, יש למלא אותם |
| 1. הכנסת ערכים לא תקינים | הכנס תווים שאינם מספרים לשדה כמות מבקרים | תופיע הודעה : ערך המבקרים שהוזן , שגוי |
| 1. הכנס כמות ספרות לא מספקת | הכנסת מספר טלפון ומזהה לא חוקי - הכנס פחות מ9 ספרות בשדה id ופחות מ10 ספרות במספר טלפון | תופיע הודעה על כמות ספרות לא תקינה |
| 1. הכנס כתובת מייל בפורמט לא תקין | הכנס מייל בפורמט שלא כולל '@' ו – '.' | תופיע הודעה שכתובת המייל הוכנסה בפורמט לא תקין |
| 1. הזמנה לתאריך שאין בו מקום פנוי בפארק | בחר פארק וודא את כמות המבקרים בו . בצע הזמנות עד כדי כמעט תפוסה מלאה של הפארק ולאחר מכן בצע הזמנה נוספת לכמות אנשים כך שתחרוג את כמות התפוסה המלאה | נקבל הודעה שאין מקום בפארק ויוצע לנו להכנס לרשימת המתנה או לבחור תאריך אחר |
| 1. שדות בכרטיס אשראי | -הזן אותיות ותווים בשדה מספר כרטיס  -הזן פחות מ16 ספרות במספר כרטיס אשראי  -השאר שדות ריקים  -הזן יותר מ3 ספרות בCVV  -הזן תווים בCVV | נקבל הודעת שגיאה בהתאם |

בדיקות אינטגרציה:

* ביצענו בדיקות אינטגרציה רבות בפרויקט מכיוון שכל אחד עבד על מודול אחר והיינו משלבים אותם באמצעות הGIT. היינו מבצעים לרוב בדיקות שפיות קטנות כאשר חלקים משמעותיים בקוד היו מתווספים כדי לראות שהפונקציאונליות לא נפגעה.

דוגמה לבדיקה שביצענו לאינטגרציה בין 2 מודולים :

במסך עובד פארק מתעדכנת כמות המבקרים בפארק בכל רגע נתון עקב כניסה ויציאה של מבקרים שמתבצעת דרכו.

לכן ביצענו את הבדיקה הבאה :

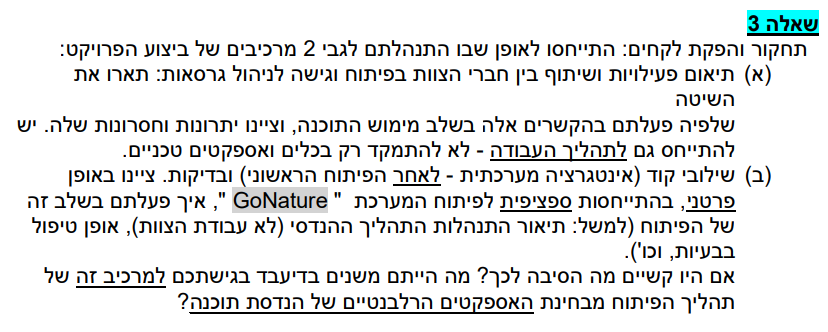
|  |  |  |
| --- | --- | --- |
| שם הבדיקה | תיאור | תוצאה צפויה |
| עדכון כניסת/יציאת מבקר במסך מנהל פארק | היכנס למסך של עובד בפארק והזן כמות של מבקרים שרוצים להכנס/לצאת ובצע כניסת/יציאת מבקרים . פתח client נוסף והכנס למסך מנהל פארק. | במסך עובד הפארק מספר המבקרים הנוכחי התעדכן. גם במסך מנהל הפארק כמות הבמקרים התעדכנה ללא צורך בפתיחה מחודשת של החלון |
| עדכון כניסת מבקרים במספר פארקים בו זמנית | פתח מספר clients של עובד פארק עבור פארקים שונים.היכנס למסך של עובד בפארק והזן כמות של מבקרים שרוצים להכנס/לצאת ובצע כניסת/יציאת מבקרים . פתח client נוסף והכנס למסך מנהל פארק. | במסך של מנהל המחלקה ניתן לראות את מספר המבקרים הנוכחי בפארק מתעדכן בכל פארק. |

בדיקות רגרסיה:

למעשה בכל הוספה של מודול חדש על פונקציונאליות שכבר עובדת דאגנו לבצע בדיקות לכך שלא פגענו בפונקציאונליות הקיימת. לדוגמה – מסך יצירת הזמנה כתהליך תקין (כאשר יש מקום בפארק) מומש לפני שמומש מנגנון רשימת המתנה להזמנה כאשר אין מקום בפארק. לכן , כאשר הוספנו את המודול לקוד ,טרם בידקתו ,בדקנו כי לא פגענו בפונקציאוליות של הזמנה חדשה כתהליך תקין כאשר יש מקום בפארק.

דוגמה נוספת – מימוש של דוח ביקורים . תחילה מימשנו את האפשרות להציג דוח מבחינה ויזואלית, כלומר שהפונקציאונליות עובדת מקצה לקצה (הבקשה נשלחת לשרת והנתנוים מתקבלים בחזרה ומוצגים היטב). לאחר מכן , הבנו שנרצה לטפל בבעיה של מחסור בנתונים בתארייכם מסוימים -אז הוספנו מנגנון שבודק זאת ומחזיר תשובה בהתאם והתקבל מהשרת שאין מידע בתאריכים הנתונים. כאשר התחלנו לבדוק את המנגנון החדש ,בדקנו תחילה שעדיין הגרף מציג כראוי את הנתונים וגילינו שלא - כנראה הוכנס באג בעת ההטמעה. לכן בדיקת הרגרסיה שלנו הייתה טובה ואפשרה לנו בשלב מוקדם להבין שהקוד שהוספנו לא תקין ונאלצנו לתקן אותו.

בדיקות יחידה ???



**סעיף א:**

במימוש המערכת על מנת לבצע תיאום פעילויות ושיתוף בין חברי הצוות יצרנו פרויקט שיתופי לחלוקת משימות באתר JIRA כך התאפשר לנו סנכרון וידיעה על איזו משימה עובד כל אחד.כמו כן עבדנו בגיט על מנת לשתף את קוד הפרויקט. בנוסף לכך, גם את הקבצים השונים ששיתפנו בינינו כגון פרוטוקולי עבודה ומסמכים עיוניים שמרנו בתיקייה שיתופית בגיט.ביצענו חלוקה של חברי הצוות לצוות שרת וצוות לקוח. אחד מחברי הצוות הציע פרוטוקול להעברת המידע בין השרת ללקוח והטמענו אותו על מנת לשמור על אחידות העבודה בין הרכיבים השונים בפרויקט ועל מנת שיהיה קו מנחה אחיד לפיו כולם יפעלו. הקפדנו על שיתופיות המידע בינינו ודאגנו לשמור על סנכרון – אם קרו תקלות חמורות או הייתה הודעה חשובה היא הייתה מועלת מיד בצ'אט הקבוצתי.

יתרונות:

* לכל חברי הצוות יש גישה לכל הקוד
* חברי הקבוצה יכלו לראות את חלקי הקוד של יתר חברי הקבוצה ולבצע שימוש בקוד שלהם בעזרת reuse.
* יצירת עבודה במקביל על מספר רב של מודולים.
* ניתן לצפות בשינויים שביצעו ומי חבר הצוות שביצע אותם.
* ניתן לבצע חזרה משינוי שביצע חבר הצוות- לדוגמא אם חבר צוות אחד יצר תקלה בקוד או מחק חלק מהקוד ניתן לבצע revert ובעזרת כך לבטל את הפעולה האחרונה.
* יצירת גיבוי - לכל אחד מחברי הצוות קיים גיבוי במחשב של הפרוייקט בגרסה האחרונה שעדכן מהגיט.
* ניהול מיזוג גירסאות קוד סותרות- במצב כזה שישנם גרסאות סותרות הגיט מציג הודעה בפני מעדכן הפעולה.
* עבודה מקוונת בזמן אמת לכל חברי הצוות- עדכון שינויים בזמן אמת.

חסרונות:

* גישה בו זמנית לכל חברי הצוות לכל הפרוייקט עלולה לגרום ל"דריסות" בקוד-אם 2 חברי קבוצה עובדים על אותם דברים ולא מיידעים אחד את השני יכול היה להיווצר מצב בו חבר צוות אחד ביצע שינויים בקוד שגרמו לשינוי אצל חבר צוות אחרבזמן שהשני עבד על הקוד ולגרום למחיקה של דברים .
* שימוש לא נכון בגיט – חוסר הכרות בשימוש בפלטפורמה יכול לגרום לטעויות קריטיות כמו למשל לבצע דחיפה לפני משיכה שיכולה לגרום לדריסה של הקוד ולקונפליקטים בין שני חברי צוות שכתבו לאותה מחלקה.
* חוסר הכרות עם כלי הJIRA עלול למנוע שימוש בו מה שגורם לכך שאין בקרה אמיתית על המשימות ומהות השימוש בכלי מתפספסת.

**סעיף ב:**