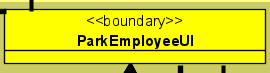
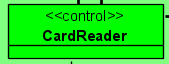
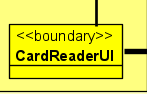
|  |  |  |
| --- | --- | --- |
| **תאריך הגשה:** | | **קבוצה 9** |
| **אימייל** | **ת.ז** | **שם** |
| [haimazu4@gmail.com](mailto:haimazu4@gmail.com) | 203374855 | חיים אזולאי |
| [hodayamekonen@gmail.com](mailto:hodayamekonen@gmail.com) | 204632988 | הודיה מקונן |
| [nastyak6@gmail.com](mailto:nastyak6@gmail.com) | 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 |

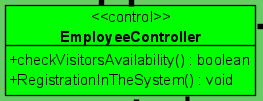
**G9\_Answers.Ass3**

****

**סעיף א:**







כניסה ע"י עובד בפארק

כניסה ע"י קורא הכרטיסים

ביקור בפארק

ביקור מזדמן

ביקור מתוכנן

Check id\memberNum

Check id\memberNum

Check availability

Check availability

בCLASS אנחנו התייחסנו אל בקרת הכניסה לפארק ב-2 דרכים דרך אחת דרך הקורא כרטיסים ודרך שנייה ע"י העובד בפארק.

הדילמות ההנדסיות שעמדנו מולם היו:

* בדיקת כמות מבקרים בפועל אל מול מס' המבקרים שמצויין בהזמנה- אם הגיעו בפועל פחות אנשים מההזמנה היה עלינו לבצע חישוב חדש לכמות המתאימה. אך אם הגיעו בפועל כמות גדולה יותר מההזמנה היה עלינו לבצע כניסה נוספת ולהתמודד אל מול ההתייחסות של אותם מבקרים נוספים.
* ביצוע סימולציה של כניסה מהירה בעזרת קורא הכרטיסים שבהתייחסות שלנו עד כה הייתה אל מערכת נפרדת מהמערכת שלנו.

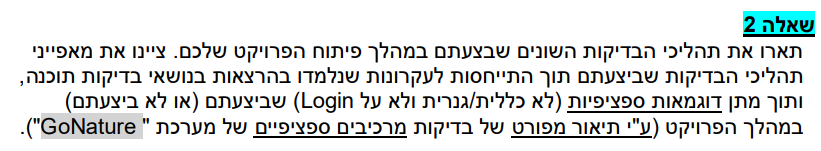
שיקולים והחלטות:

מכיוון והבנו שעלינו להתייחס אל כל המבקרים המגיעים לפארק החלטנו לבצע את בקרת הכניסה בדרך הזו. העובד בפארק ייקבל ממבקר מוזמן מס הזמנה וכמות אנשים וממבקר מזדמן כמות ומס מזהה(ת.ז או מנוי).

* אם זהו מבקר מוזמן בעל הזמנה ומס מבקרים מתאים אז הכניסה והתשלום מתבצע לפי ההזמנה ובתוספת של הנחה אם קיימת של מנהל הפארק.
* אם בהזמנה מצויין מס' מבקרים ובפועל הגיעו פחות אנשים אז נבצע גבייה ע"פ ההזמנה ועדכון של כמות המבקרים בפועל.
* אם בהזמנה מצויין מס' מבקרים ובפועל הגיעו יותר אנשים אז נבצע התייחסות ובדיקה של יתר האנשים ע"פ כניסה של מבקר מזדמן.
* אם זהו מבקר מזדמן- נבצע בדיקה של כמות האנשים בפארק במידה וישנו מקום פנוי נבצע כניסה ותמחור לפי סוג מבקר ובסוף התהליך נבצע עדכון של כמות האנשים בפארק.

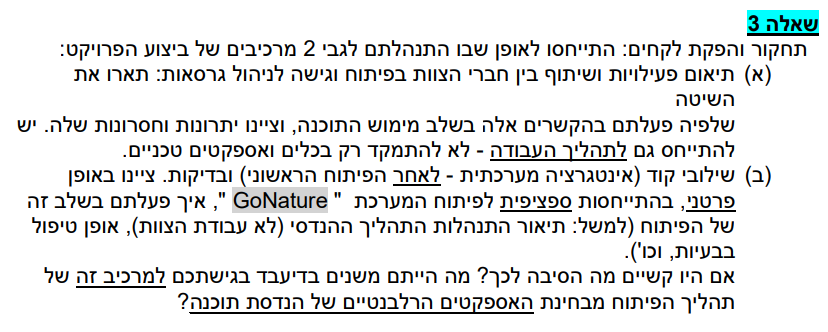
על מנת שנוכל לבצע הדמייה של קורא הכרטיסים הוספנו כפתור שע"י לחיצה בו יוצג לעובד בפארק מס הזמנה ולאחר מכן את שאר הפרטים הכוללים של ההזמנה.

**סעיף ב: רועי נסטיה אמרה שאתה יודע הכי טוב**



בדיקות מערכת:

* בדיקות קלט- בדיקות של מידע שמוזן בGUI בהתאם לדרישות המערכת. (לדוגמא במסך של נציג השירות כל השדות הם שדות חובה לכן יש בדיקה שאין שדות ריקים במסך. כמו כן שלא מזינים מספרים בשדה של שם פרטי ושם משפחה . שהאיימיל שמוזן הוא ע"פ תבנית מסויימת . וגם שדה ת.ז בעל בדיוק 9 ספרות ).
* בדיקות מידע שחוזר מהסרבר- בכל פניה אל השרת אנו מקבלים מידע אודות הפעולה שרצינו לבצע, לכן נבדוק את המידע בצד הלקוח.(לדוגמא כאשר מבצעים הוספה של מנוי חדש נבדוק האם קיבלנו true שזה הצלחה או false ככישלון.בנוסף הוספנו גם שליחת מס' המנוי אותו יוכל להעביר נציג השירות לבעל המנוי החדש שיצר ואותו אנחנו מציגים עם הודעת ההצלחה של הוספת המנוי למסד הנתונים שלנו).
* בדיקות הכנסה לDB – בכל פניה מצד הלקוח לשינוי מסד הנתונים שלנו קיימת בדיקה של הנתונים שהתקבלו. לדוגמא בהוספת מנוי חדש בצד השרת ישנה שאילתא שבודקת את קיום המנוי בDB ורק במידה ולא קיים יוצרת אותו בDB ע"י שאילתת הוספה. בכל אחד מהמצבים קיימת הודעה שמעדכנת על מצב הפעולה.
* בדיקת סוג משתמש- קיימת בדיקה שבודקת את סוג המשתמש. אם הוא משתמש רגיל/משפחתי/מדריך. בהוספת מנוי חדש לDB במסך של נציג השירות יש CHECKBOX לסימון במידה וההוספה היא של מנוי עבור מדריך ובהתאם לכך נבחר עבורו מס מנוי שמסמן את היותו מדריך.
* בדיקת זמינות- למשל בדיקה של תאריך ההזמנה אותו הזין המשתמש האם ישנו מקום פנוי בפארק או שלא ואז להפנות אותו לרשימת המתנה. או בדיקת מקום פנוי בפארק (האם הפארק לא בתפוסה מלאה) עבור מבקר מזדמן שמגיע לפארק. לדוגמא בבקרת הכניסה לפארק במצב שבו מגיעים מבקרים נוספים, מעבר למה שמצויין בהזמנה אנחנו יוצרים רישום שלהם בDB, על מנת שנוכל לאחר מכן להשתמש במידע הזה עבור הפקת הדוחת.
* בדיקות אימות מידע- בדיקה שאכן קיים המידע בDB עבור הנתון שהזין המשתמש בשדה . לדוגמא במסך של העובד בפארק צריך להזין מס' הזמנה וכמות משתמשים. ברגע שהעובד לוחץ בכפתור הcheck מתבצעת בדיקה האם בDB שמור מידע עבור מס ההזמנה. המידע נשלף בעזרת שאילתא ונשלח בחזרה אל הלקוח ומוצג במסך שלו בהתאם.
* בדיקת רישום לרשימת המתנה- האם המשתמש מעוניין להכנס לרשימת המתנה כמו שהתבקשנו כחלק מדרישות הלקוח.



**סעיף א:**

במימוש המערכת בכתיבת הקוד על מנת לבצע תיאום פעילויות ושיתוף בין חברי הצוות יצרנו גישה משותפת ע"י חיבור האקליפס אל הגיט. ביצענו חלוקה של חברי הצוות לצוות שרת וצוות לקוח . עבדנו יחדיו כל חברי הצוות על מימוש המערכת.  
יתרונות:

* הגישה לכתיבת הקוד הייתה משותפת.
* חברי הקבוצה יכלו לראות את חלקי הקוד של יתר חברי הקבוצה ולבצע שימוש בקוד שלהם בעזרת reuse.
* כל חברי הצוות יכלו להתקדם בעבודתם ללא צורך בהמתנה לחלון זמן מסויים.
* ניתן לצפות בשינויים שביצעו ומי חבר הצוות שביצע אותם- לדוגמא באיזה קונטרול ביצעו שינויים.
* 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-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 להשפעה על חבר הצוות השני.
* אם חבר צוות אחד לא סיים את המשימה שמוטלת עליו והיא משימת הקדם של חבר הצוות השני הוא תקוע בעבודתו ולא יכול להתקדם.
* הקצאת זמן לא מבוטל ללימוד השימוש בגיט. ישנם פקודות רבות שבהתחלה לא ניתן לדעת את משמעותם.
* לפעמים מעוניינים לבצע push על מנת לשמור על שינויים גדולים בפרקי זמן קצרים ובכל ביצוע של פעולתpush עלינו לבצע גם פעולת commit. פעולת commit מחייבת אותנו לציין כותרת עבור הPush אותו ביצענו מה שיוצר ריבוי הודעות בעלות אותה משמעות.

**סעיף ב:**