G9_Answers.Ass3

קבוצה 9	תאריך הגשה: 10.01.2021	
שם	ת.ז	אימייל
חיים אזולאי	203374855	haimazu4@gmail.com
הודיה מקונן	204632988	hodayamekonen@gmail.com
אנסטסיה קוקין	321241192	nastyak6@gmail.com
בר כץ	315818567	barkatz0610@gmail.com
רינת סטודנץ	316237577	rinat20968@gmail.com
רועי עמר	203118666	roi.amar@e.braude.ac.il

שאלה 1

תארו את תהליך התכן (design) שביצעתם עבור המערכת " GoNature " באופן הבא:

(א) עבור יכולות **בקרת כניסה לפארק** בהקשר של ביקור מוזמן (מתוכנן) ולא מתוכנן (מזדמן).
תארו דילמות הנדסיות ספציפיות שעסקתם בהן בתכן של יכולות אלה. השתמשו בפורמט של תיאור Design issue כפי שמופיע בהרצאה על Design.
תארו את הדילמות, השיקולים וקבלת ההחלטות שלכם והסבירו את הפתרונות שבחרתם.

(ב) ציינו אילו מהעקרונות שנלמדו בהרצאות הקורס בנושאים: Architecture, Design, Reuse, ו- Design patterns באו לידי ביטוי בתהליך ה**תכן הכלל-מערכתי** שביצעתם והסבירו באיזה אופן – באמצעות דוגמאות ספציפיות (לא כלליות) מתוך המערכת שפיתחתם. לכל אחד מהעקרונות, הסבירו מה הם היתרונות המתקבלים משימוש בהם.

סעיף א:

דילמה מס.1

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

- אופציה מס.1 – לגבי כמות אנשים קטנה מהמוזמן . אם בהזמנה מצוין מס' מבקרים ובפועל הגיעו פחות אנשים אז נבצע גבייה ע"פ ההזמנה ועדכון של כמות המבקרים בפועל.
- אופציה מס. 2- אם בהזמנה מצוין 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 ושקוף למשתמש.

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

<u>דילמה מס.2</u>

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

<u>החלטה</u>: 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 בDB 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.

<u>סעיף ב:</u>

בחרנו להשתמש ב singleton : design pattern . נסביר כעת כיצד מימשנו זאת:

קראנו למחלקה שמממשת את הsingleton : design pattern – contextController וכך אנו מבצעים את השימוש בו: Controller נפתח פעם אחת בלבד, כדי שתהיה לנו גישה לכל ה controllers שפתוחים בעצם השתמשנו ב singleton משום שבאמצעות שיטה זו ישנו אובייקט אחד מכל מסך שפתוח בבת אחת וזה מאפשר לנו לגשת לכלל המסכים באופן פשוט, לדעת מי מהם פתוח בכל רגע נתון ובנוסף לקבל גישה לכלל הנתונים שנמצאים בקונטרולר. שימוש עיקרי במחלקה זו הוא עבור מימוש של sendToAllClients כאשר אנחנו רוצים לעדכן בכל מסך של עובדי ומנהלי הפארק את כמות המבקרים הנוכחית, אנחנו מבצעים בדיקה איזה מסך פתוח כעת ואליו אנו נשלח את העדכון.

<u>שאלה 2</u>

תארו את תהליכי הבדיקות השונים שבצעתם במהלך פיתוח הפרויקט שלכם. ציינו את מאפייני תהליכי הבדיקות שביצעתם תוך התייחסות לעקרונות שנלמדו בהרצאות בנושאי בדיקות תוכנה, ותוך מתן <u>דוגמאות ספציפיות</u> (לא כללית/גנרית ולא על Login) שביצעתם (או לא ביצעתם) במהלך הפרויקט (<u>ע"י תיאור מפורט</u> של בדיקות <u>מרכיבים ספציפיים</u> של מערכת " GoNature").

נתאר את בדיקות המערכת השונות שביצענו במרכיבים השונים של הפרויקט :

ביצענו בדיקות קופסא שחורה וקופסא לבנה.

בדיקות קופסא שחורה :

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

נתאר מספר דוגמאות ספציפיות כעת :

בדיקות חיוביות-

עבור המסך של מנהל הפארק-

שם הבדיקה	תיאור	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
2) הזנה של מכסה למבקרים מזמינים מראש	לחץ על כפתור קביעת כמות מזמינים מראש והזן מספר כלשהו	קבלת חיווי שההודעה נקלטה בהצלחה ונשלחה למנהל המחלקה
3) הזנה של מכסה מבקרים בפארק	לחץ על כפתור קביעת כמות מבקרים והזן מספר כלשהו	קבלת חיווי שההודעה נקלטה בהצלחה ונשלחה למנהל המחלקה
4) 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
5) 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
6) 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	תיאור	תוצאה צפויה
1) בדיקת שדות ריקים	השאר שדות ריקים ובצע הזמנה	תופיע הודעה : אחד או יותר מהשדות ריקים, יש למלא אותם
2) הכנסת ערכים לא תקינים	הכנס תווים שאינם מספרים לשדה כמות מבקרים	תופיע הודעה : ערך המבקרים שהוזן , שגוי
3) 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
4) הכנס כתובת מייל בפורמט לא תקין	הכנס מייל בפורמט שלא כולל '@' ו – '.'	תופיע הודעה שכתובת המייל הוכנסה בפורמט לא תקין
5) 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 להיכנס לרשימת המתנה או לבחור תאריך אחר
6) 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 שהפונקציונליות 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 הבדיקות הבאות :

שם הבדיקה	תיאור	תוצאה צפויה
עדכון כניסת/יציאת מבקר במסך מנהל פארק	היכנס למסך של עובד בפארק והזן כמות של מבקרים שרוצים להיכנס/לצאת ובצע כניסת/יציאת מבקרים . פתח client נוסף והכנס למסך מנהל פארק.	במסך עובד הפארק מספר המבקרים הנוכחי התעדכן. גם במסך מנהל הפארק כמות המבקרים 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 להיכנס/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 בפונקציונליות 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 בדיקתו, בדקנו כי לא פגענו בפונקציונליות של הזמנה חדשה כתהליך תקין כאשר יש מקום בפארק.

דוגמה נוספת – מימוש של דוח ביקורים . תחילה מימשנו את האפשרות להציג דוח מבחינה ויזואלית, כלומר שהפונקציונליות עובדת מקצה לקצה (הבקשה נשלחת לשרת והנתונים מתקבלים בחזרה ומוצגים היטב). לאחר מכן, הבנו שנרצה לטפל בבעיה של מחסור בנתונים בתאריכים 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 קופסא לבנה :

בדיקות יחידה-

ביצענו בדיקות של מסלולי החישוב השונים בקוד והנתיבים השונים שיש בקוד . לדוגמה כאשר אנחנו מבצעים בדיקה של תנאי בפונקציה שבודקת האם זמן של הזמנה חלף אנו בודקים מספר מסלולים אפשריים- האם גם התאריך קטן מהיום או שרק השעה כבר חלפה או האם התאריך טרם חלף וניתן לתת למזמין לגשת להזמנה.

שאלה 3

תחקור והפקת לקחים: התייחסו לאופן שבו התנהלתם לגבי 2 מרכיבים של ביצוע הפרויקט:

(א) תיאום פעילויות ושיתוף בין חברי הצוות בפיתוח וגישה לניהול גרסאות: תארו את השיטה
שלפיה פעלתם בהקשרים אלה בשלב מימוש התוכנה, וציינו יתרונות וחסרונות שלה. יש להתייחס גם לתהליך העבודה - לא להתמקד רק בכלים ואספקטים טכניים.

(ב) שילובי קוד (אינטגרציה מערכתית - לאחר הפיתוח הראשוני) ובדיקות. ציינו באופן פרטני, בהתייחסות ספציפית לפיתוח המערכת " GoNature ", איך פעלתם בשלב זה של הפיתוח (למשל: תיאור התנהלות התהליך ההנדסי (לא עבודת הצוות), אופן טיפול בבעיות, וכו').
אם היו קשיים מה הסיבה לכך? מה הייתם משנים בדיעבד בגישתכם למרכיב זה של תהליך הפיתוח מבחינת האספקטים הרלבנטיים של הנדסת תוכנה?

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 הפרויקט בגרסה האחרונה שעדכן מהגיט.
- ניהול מיזוג גרסאות 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 הפרויקט 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 אחר בזמן 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:

בשלב האינטגרציה לאחר הפיתוח הראשוני השתמשנו במנגנון הגיט לביצוע שילובים בקוד. בחרנו להשתמש באינטגרציות יום יומיות ואפילו מספר פעמים ביום כלומר – לעבוד כל הצוות כולו על הענף הראשי ולדחוף ולמשוך ממנו.

בעצם בשלב זה של הפיתוח ובאופן זה , כאשר כל אחד מחברי הצוות היה עובד על הרכיב שלו ומסיים חלק כלשהו בקוד , הוא מיד היה דוחף את הקוד על מנת שיופיע אצל כולם. עבודה בצורה כזאת אפשרה לנו התעדכנות on line בשינויים של הקוד מה שהקל עבורנו את עבודת הצוות השיתופית ופתירת התקלות במהירות ובכך מספר חברי צוות שיכלו לעבוד במקביל. לדוגמה באחת הפעמים מצאנו תקלה שגרמה להשבתה של כלל האפליקציה, אופן עבודה זה אפשר לנו לחזור אחורה בהיסטוריית השינויים ולעבור על כל שינוי בכדי לבדוק מה מהשינויים שהוכנסו ועל ידי מי נגרמה התקלה.

ההנחה המרכזית בתהליך הפיתוח בעבודתנו הייתה שכולם עובדים על הגרסה המעודכנת ביותר של הקוד כדי למנוע בעיות באינטגרציה ורגרסיה של הקוד ולמנוע מצב של עבודה עם קבצים ישנים מדי ולא מעודכנים.

הקשיים שעלו במהלך העבודה באופן זה :

- קרו מספר מקרים בהם הייתה דריסה של קוד על ידי חבר צוות. בגלל שלא נמשך קוד, והוא ומספר קבצים היו לא מעודכנים, נוצרו קונפליקטים בין הקבצים ונדרסו דברים בעת הדחיפה לגיט.

השינויים שהיינו מבצעים בדיעבד :

- היינו יוצרים branch נפרד לכל חבר צוות ומבצעים מיזוגים בזמן קבוע מראש. חלוקת העבודה לענפים נפרדים תהפוך את העבודה למקצועית ונכונה יותר ויכול להיות שהייתה פותרת את בעיית הדריסות בקוד שהייתה אחד מהקשיים שנתקלנו בהם.
- הקדשת זמן ללמידה והבנה עמוקה יותר של דיאגרמת ה package. אם במטלה 2 היינו מבינים יותר לעומק את הדיאגרמה , היינו כנראה עושים אותה בצורה אחרת. כאשר הגענו לשלב המימוש והסתכלנו על הדיאגרמה בדיעבד, ביצענו שינויים רבים עד לכך שכמעט אף דבר לא תאם לדיאגרמה. אם היינו מבינים את חשיבותה יותר לעומק ,ייתכן והיינו מבצעים אותה אחרת וחוסכים זמן יקר בחשיבה על אופן המימוש והחלוקה ל packages.