**שאלות – משימה 3**

**שאלה 1:**

א.

דילמות הנדסית בנוגע לניהול מלאי הספרים ע"י הספרים:

הדילמה שלנו נוגעת לחיפוש ספר ע"י הספרן. האם בכל פעם שהספרן ירצה לחפש ספר, נצטרך לטעון את כל המידע ממסד הנתונים הכולל: את הספרים, נושאים, תחומים, מילות מפתח וכו', שמתוכו יוכל הספרן לבחור להציג ספר מסוים ואת פרטיו.

אפשרות שניה, על כל שאילתה לחיפוש ספר נטען מידע שרק רלוונטי לאותה שאילתה, ורק אם יבחר ספר מסוים נביא את כל המידע הרלוונטי אליו.

|  |  |  |
| --- | --- | --- |
| **יתרון** | **חסרון** |  |
| היתרון של השיטה זה שיש 2 פניות (קריאה/עדכון) למסד הנתונים. | החיסרון של השיטה הראשונה שקיימת העברה של הרבה מידע בתקשורת של client עם server  וזה מכביד על התקשורת בניהם. | **גישה ראשונה** |
| היתרון בשיטה השנייה שנקבל רק את המידע הרלוונטי וכך לא נכביד על התקשורת. | החיסרון של השיטה השנייה שמתבצעות הרבה פניות למסד הנתונים, ולכן הסיכוי שיהיו יותר רעשים וכתוצאה מכך נקבל נתונים לא נכונים. (reliability). | **גישה שניה** |

הפתרון שבחרנו ליישם הוא לפי הגישה השנייה, שלפיה לכל שאילתה מסוימת בניהול מלאי הספרים נטען רק את המידע הרלוונטי.

ב. בתהליך התכן בא לידי ביטוי נושא ה- . user interface

כאשר תכננו את המעבר בין החלונות, התלבטנו האם לכלול כפתור "אחורה" לחלון הקודם או לא.

לאחר ההבנה שפיתוח האפשרות הזו דורשת שמירה של כל החלונות הקודמים ודורשת הצגה של מידע מעודכן (גם אם שונה בחלונות הבאים), החלטנו לבצע מעבר בין חלונות שדורס את החלונות הקודמים ולא שומר את היסטורית המעברים.

2. כל אחד מחברי הקבוצה ביצע בדיקות על כל אחת מהפונקציות שהוא בנה על פי שיקול דעתו, תוך כדי תהליך פיתוח התוכנה.

בדיקות אלו כללו לדוגמא: בדיקות מול עדכון מסד הנתונים, בדיקת מעבר המסרים בין השרת ללקוח, בדיקות פלט, בדיקות עם פלטים שונים (מקרי קצה) ועוד

בתהליך הבדיקות כל חבר צוות השתמש בעיקרון ה Glass-Box כאשר הוא דיבג את הקוד שהוא כתב, משום שהקוד היה חשוף בפניו ותהליך הבדיקה כלל שינוי של הקוד עצמו. אך יחד עם זאת השתמש בעיקרון ה Black-Box לבדיקות שביצע על קוד שנכתב ע"י חבר צוות אחר, משום שהוא לכנס לתוכן הקוד אם כי היה יכול, אלא רק הזין קלט ובדק הוא הפלט הרצוי התקבל.

לדוגמא: ניסינו לבצע סידרה של פעולות עם לקוחות שאין להם מנוי או יש להם מנוי בתוקף או יש להם מנוי אך לא בתוקף. וביצענו את סידרת הפעולות הנ"ל כל פעם עם לקוח (קלט) אחר.

3. השיטות שלפיהם בחרנו לתאם את הפיתוח של התכנה

א.

השיטות לשיתוף ותיאום פיתוח תכנה:

* את שלב תכנון תכן הפרויקט ביצענו במשותף כל חברי הקבוצה.
* בשלב המימוש דאגנו שכל חבר צוות יממש מחלקה נפרדת כדי ליצור מינימום התנגשויות וסתירות בקוד.
* באשר למחלקות משותפות יצרנו פרוטוקול שלפיו כל עריכה של משאב משותף תעשה בסנכרון מלא בין כל חברי הצוות.

הכלים הטכניים בהם השתמשנו:

בחרנו להשתמש בתכנת ה- GitHub Desktop לשיתוף ותיאום הקבצים מתוך רצון שסנכרון הקוד יתבצע בצורה חלקה ויעילה.

היתרון ב-GitHub שהיא מבצעת סנכרון בין קבצי הקוד השונים באופן אוטומטי ובנוסף כל חבר צוות יכול להתעדכן בצורה מהירה בכל השינויים בקוד.

החיסרון ב- GitHub, כאשר חבר צוות מבצע שינוי בקובץ משותף הוא צריך להודיע לכל שאר חברי הצוות על השינוי כדי שיתעדכנו לגרסה העדכנית ביותר.

ב. כל חבר צוות שמבצע שינוי כלשהו בקוד, מבצע commit ומשתף את כל חברי הצוות בשינוי.

הקושי שנתקלנו בו הוא:

* תכנת ה GitHub לא הצליחה לסנכרן בין קבצי הקוד כאשר היו כמה גרסאות לאותו קובץ. במקרה כזה, חזרנו לגרסה הקודמת וביצענו את הסנכרון באופן ידני.

בדיעבד היינו בוחנים תוכנות שיתוף קבצים נוספות.