**מטלה 3 – מענה לשאלות פתוחות**

**קבוצה G16:**

Erez Ben Ezra - 025438284  [erez.be10@gmail.com](mailto:erez.be10@gmail.com)

Tal Gilad - 037054095  [tgilad@gmail.com](mailto:tgilad@gmail.com)

Muhammad Zbeedat - 033933631  [zbeedatm@gmail.com](mailto:zbeedatm@gmail.com)

Maayan Salem - 307870311  [perach13@gmail.com](mailto:perach13@gmail.com)

Perach Ben David - 316218080  [salem.maayan@gmail.com](mailto:salem.maayan@gmail.com)

1. **תיאור תהליך התכן**
2. **דילמות הנדסיות בהן עסקנו בתכן של ביצוע הבחינות ממוחשב וידני**

היינו צריכים לחשוב איך נגדיר מבנה הטבלאות הקשורות בביצוע הבחינות ואופן ניהול הלוגיקה לשליפת הנתונים הקשורים לבחינה מסוימת שמבצע סטודנט כך שמבנה הנתונים יהיה פשוט יעיל לפעולות קריאה וכתיבה קלט ופלט וקל למימוש: לפי סיפור המעשה המורה יוצרת מבחן משאלות מתוך מאגר שאלות שזהו בעצם טופס הבחינה, ומגדירה לה משך זמן לפתרון, מספר נקודות לכל שאלה ותכונות נוספות ובנוסף קיימות תכונות רבות נוספות למבחן המתבצע בפועל (ממחושב/ידני, זמן בחינה בפועל, תאריך, מספר סטודנטים שהשלימו בחינה וכו'..), כדי להבחין בעצם בין 2 ישויות אלה החלטנו להפריד ל 2 טבלאות בבסיס הנתונים: tests היא הטבלה הכוללת את מבנה הנתונים הנדרש לשמירת הנתונים עבור טפסי הבחינה ו exams היא טבלה שבאמצעותה אנחנו שומרים נתונים על הבחינה בפועל, ההחלטה על מבנה זה התקבלה מכיוון שמורה יכולה ליצור טופס בחינה מבלי שיהיה מבחן פעיל באותו המקצוע. בטבלה StudentInExam החלטנו להשתמש לאחסון רשומות המידע על סטודנטים המבצעים את בחינות, כאשר המפתח לטבלה זו הוא מפתח מורכב הבנוי ממפתח ראשי של טבלה exams : examCode ומפתח ראשי של טבלת Users : UserID, אחד השדות בטבלה StudentInExam מכיל את הערך המאפיין את סוג הבחינה selectedExamType שיכול להיות ממוחשב ((computerized או ידני (manual), כך אנחנו מנהלים בטבלה אחת את 2 סוגי בחינות

המבוצעות ע"י סטודנטים ומפשטים את הלוגיקה לאחסון ושליפת הנתונים משותפים ושאינם משותפים ל 2 סוגי המבחנים, למשל: כאשר נדרש לשלוף נתונים על מבחן באותו מקצוע שנעשה ע"י סטודנטים גם באופן ממוחשב וגם באופן ידני (שדות שאינם בשימוש עבור אחד מהמבחנים יוגדרו בטבלה אבל יאוחסן בהם ערך null).

לכל אחת מהטבלאות יש קשרים (מפתח ראשי ומפתח משני) לטבלאות נוספות מהן נדרש לקבל מידע,

לדוגמא, טבלה StudentInExam מקושרת לטבלה Users כדי שניתן יהיה לשלוף את שם הסטודנט לפי שדה UserID המוגדר בטבלת Users כמפתח ראשי, ובטבלת StudentInExam כמפתח משני.

דילמה נוספת שדנו בה היא האם נדרשת טבלה אחת, או יותר כדי לנהל את מאגר השאלות והתשובה הנכונה לכל שאלה. לאחר ששקלנו את העניין, החלטנו לפשט כמה שניתן את סכמת בסיס הנתונים והגדרנו טבלה אחת בשם questions שבה בכל שורה מכילה גם את נתוני השאלה וגם 4 שדות של תשובה שכל אחד מהם מוגדר כ Boolean, התשובה הנכונה הוגדרה עם ערך 1, ושאר התשובות השגויות עם ערך 0, בטבלה זו הגדנו כמפתח משני את השדה coursID וכך בעצם הגדרנו סט של שאלות עם התשובות הנכונות אליהן בקורס מסוים.

1. **נושאים מתוך ההרצאות שבאו לידי ביטוי בתכן שלנו**

אחד מהנושאים שלמדנו בהרצאות הוא מודל 3 השכבות, זהו מודל שמגדיר את הפרדת האפליקציה לשכבת הנתונים, שכבת הלוגיקה ושכבת ממשק המשתמש.

בעבודה מימשנו מערכת שרת-לקוח, המסתמכת על מודל 3 השכבות:

**שכבת הנתונים** – את השכבה זו מימשנו באמצעות יצירת בסיס נתונים טבלאי מנורמל מבוסס SQL.

**שכבת הבקרה** – זו השכבה העוסקת בלוגיקת המערכת ועיבוד המידע, שמימשנו באמצעות

תכנון מבנה בסיס הנתונים שיוכל לתת מענה לדרישות ולאילוצים של המערכת, שריכזנו מתוך סיפור המעשה - בכל טבלה הגדרנו שדות (אורך וסוג) כך שיתאימו לאכלוס הנתונים של כל ישות שהגדרנו למערכת. יצרנו קשרים בין הטבלאות ע"י הגדרת מפתחות ראשיים ומפתחות משניים שבאמצעותן ניתן לבצע חיפוש ושליפה של הנתונים הרצויים מהטבלאות ולבצע פעולות קריאה וכתיבה (עדכון).

**שכבת התצוגה** – יצרנו ממשק משתמש שבאמצעותו משתמש יכול לבצע Login ולהיכנס למערכת, לקבל הנחיות ועל סמך הרשאותיו יכול לבחור באפשרויות העומדות לרשותו מתוך התפריטים.

ממשק המשתמש רץ על המחשב האישי שלנו, ועושה שימוש בממשק משתמש גרפי סטנדרטי. העיבוד של היישום מתבצע על גבי מודול אחד או יותר שרצים על מחשב נוסף שהגדרנו כשרת ומערכת בסיס הנתונים, במבנה טבלאי נמצאת על עמדת השרת ומכילה את הנתונים ואת הלוגיקה לאחסונם.

המשתמש יוצר אינטראקציה עם השרת שבו יושב בסיס הנתונים ומבקש ממנו שירותים.

נושא נוסף שבא לידי ביטוי בעבודה מתוך הנושאים שנלמדו הוא בדיקות שילוב/אינטגרציה

(integration testing): זהו תהליך שבו בודקים את תקינות הפעולה המשולבת של כל רכיבי המערכת.

בתכולת הבדיקות עולים הנושאים הבאים: פונקציונאליות, ממשקים בין הרכיבים, בדיקות תקשורת לנתונים בין עמדת שרת לעמדת לקוח, בדיקות אבטחת מידע, שילוב תרחישי כשל ועוד.

בפרויקט שלנו זה בא לידי ביטוי לאחר השלב שבו חיברנו בין ממשק המשתמש והקוד לבסיס הנתונים, יצרנו Connection String שזוהי בעצם שורת קוד שמאתחלת את הקשר לבסיס הנתונים הטבלאי שיצרנו למערכת, הרצנו את הקוד בעמדת מחשב שהגדרנו כצד השרת והחלטנו על סדרה של בדיקות שבהן הגדרנו את סוג הבדיקה, תרחיש הפעולה המבוצעת והתוצאה הרצויה שאנו מעוניינים והתוצאה קיבלנו בפועל ולאחר בדקנו האם המערכת הגיבה באופן שבו ציפינו. פירוט לשלב הבדיקות שביצענו בתשובה לשאלה מספר 2.

1. **תהליכי הבדיקות שביצענו במהלך פיתוח הפרויקט**

החלטנו על מספר תהליכי בדיקה, כשלכל אחד מהם הגדרנו את תרחיש הבדיקה, התוצאה שאנחנו מצפים לקבל והשוואתה לתוצאה בפועל. בכל תרחיש בדקנו הזנה נכונה ושגויה של נתוני קלט ובדקנו את תגובת המערכת בהתאם.

**בדיקות ממשק משתמש** - וידאנו שהממשק עומד בדרישות שהגדרנו לו להצגת המידע באופן ברור ומלא,

נוח וידידותי למשתמש למשל: ראות ברורה של השאלות והתשובות ובחירה נוחה של התשובה הנכונה, ההנחיה שרשמנו לסטודנט לביצוע הבחינה, השאלות המוצגות לו ובדיקה טכנית שהממשק עובד באופן

תקין למשל: שהכפתורים מגיבים ומבצעים את המעבר הנכון בין שלב אחד לשלב הבא, מעבר בין מסכים.

**בדיקות פונקציונאליות** - בדיקות אבטחה ובדיקות שילוב, למשל: בדיקת ההתחברות הראשונית של משתמש באמצעות קוד וסיסמא כאשר האימות מתבצע אל מול טבלת Users שאליה הזנו את נתוני ה Login של המשתמש, בהמשך זיהוי המשתמש כסטודנט והכנסתו למסך לביצוע בחינה. את הבדיקה לביצוע Login ביצענו עם נתוני התחברות נכונים ושגויים.

בדיקת נכונות הצגת המידע**:** "בדיקות קופסה שחורה" (נכונות הפלט והתגובה, ערכים חוקיים/לא חוקיים) -בדקנו את פעולות קלט/פלט, בדיקה שהקוד פונה לשדות הנכונים ב DB ומוצג המידע הנכון למשתמש, בדקנו שמבוצעת קריאה נכונה של הנתונים שהזנו לשדות בטבלה, למשל : לאחר זיהוי במשתמש במערכת כמשתמש וכסטודנט, הכנסתו של הסטודנט למסך שבו הוא בוחר את סוג הבחינה ידני/ממוחשב ומשם להוא אכן עובר למסך שבו הוא יכול להתחיל בביצוע הבחינה שבה בחר ויכול לענות לכל שאלה, עד לסיום הבחינה. "בדיקות קופסה לבנה" (נכונות חישוב, החלטות לוגיות): חישוב הציון מבוצע באופן נכון, בסיום הבחינה באמצע או בסיום המענה לשאלות מתקבל הציון הנכון בהתאם לתשובות ולניקוד שהגדרנו לכל שאלה בטבלה questions.

**בדיקת ביצועים** – בעיקר בשלב שבו בדקנו תקשורת שרת-לקוח על תקשורת ה WIFI הביתית והסלולרית,

כדי לוודא שזמן התגובה הוא סביר עד 3 שניות לעליית המערכת ו 1 שניה עד 2 שניות לביצוע Login, למעבר בין מסכים, להצגת ציון הבחינה למשתמש.

1. **תחקור והפקת לקחים**
2. **שיתוף ותיאום פעילויות בין חברי הצוות בפיתוח וגישה לניהול גרסאות**

השיטה שבה פעלנו: לאחר שהתפרסמה מטלה במודול, אנחנו, חברי הקבוצה ניסינו לקבוע מועד מועדף על כולם להיפגש. בגלל הקושי למצוא זמן מתאים לכולנו ביחד, החלטנו לחלק את הקבוצה ל 2-3 קבוצות בכל פעם לפי שלב ההתקדמות במטלות. חלוקת המטלות נעשתה דרך קבוצת WhatsApp שפתחנו שבה נמצאים כל חברי הקבוצה, ובאמצעותה ניהלנו את השיח וקבענו את חלוקת המטלות בינינו, מי אחראי על איזה חלק בהתאם לכישוריו וניסיונו ובנוסף קבענו את מועד הסיום עבור כל מטלה.

סיכמנו שכאשר אחת הקבוצות משלימה את משימתה, לפני המפגש המתוכנן הבא של הקבוצה, היא מעבירה את החומר (תיעוד, קבצים) להתייחסות ולבדיקת שאר החברים.

כאשר נפגשנו, ווידאנו את רמת מוכנות המטלה וההתקדמות בעבודה ובמידת הצורך תיקנו אחד את השני עבור כל המטלות שהגדרנו לכל אחד בעבודה, כך עד אשר המטלה מוכנה להגשה. בסיום ארזנו לקבצים בהתאם לדרישות המטלה והעלינו למודול.

יתרונות:

חלוקת עבודה ברורה בין חברי הצוות, הגדרת תאריך סיום לכל חלק ומתן זמן לסנכרן ולהתקדם לשלב הבא.

עזרה אחד לשני דרך המייל/טלפון/WhatsApp, נפגשים רק באמת כשיש לכך תועלת.

חסרונות:

מכיוון שפרופיל חברי הקבוצה הוא של אנשים עובדים במשרה מלאה, חלקנו הורים לילדים ומשלימים לתואר שני, ו 2 מחברי הקבוצה אף לומדים לקורסים לתואר השני במקביל, כל אחד טרוד בענייניו והיה קושי גדול לתאם את המפגשים מכיוון שכך, רוב העבודה נעשתה בנפרד ולא הייתה פרונטלית, פנים מול פנים, דבר שהקשה על ביצוע המשימות.

1. **שילובי קוד (אינטגרציה) ובדיקות**

תחילה, לאחר הגש מטלה 2 בשלמותה, נפגשנו והתקנו את האבטיפוס (תוצר המטלה) על כל המחשבים האישיים של כולם ובדקנו ריצה. כך ידענו שכולנו מתחילים מאותה נקודת פתיחה.

לאחר מכן, ביצענו את החלוקה בינינו 2 מחברי הקבוצה עבדו על תכנון ויצירת הDB המפורט של המערכת ואכלוסו בנתונים, שנבנה על אחד מהמחשבים ונשלח לשאר חברי הקבוצה כקובץ SQL במייל עם הסכמה טבלאית המקלה על הבנת מבנה הנתונים, לסבב התייחסות עדכונים והתאמות כך שיהיה מקובל על כולם וניתן יהיה להתקדם.

במקביל, אחד מחברי הקבוצה התחיל בכתיבת הקוד על סמךDB בסיסי, הקוד ברובו נכתב על מחשב אחד מרכזי של אחד מחברי הקבוצה. לאחר קבלת ה DB המפורט והתקנתו בעמדת הפיתוח, הקוד הותאם אליו.

כאשר היה צורך בשילוב קוד ממחשב אחר השתמשנו בפלטפורמת GitHub. המשכנו לעדכן ולהתעדכן בינינו דרך קבוצת הWhatsApp.

בדיעבד, כדי להקל על התהליך היינו פועלים אחרת בשלב הקמת מבנה הנתונים, חושבים על הסכמה הטבלאית וממנה יוצרים את הקשרים ומבצעים Reverse Engineering וליצירת הטבלאות ונעזרים בה כדי להקל תהליך בדיקת הקשרים בין הטבלאות (מפתחות ראשיים ומשניים, הגדרת מפתחות מורכבים), כיוון שכאשר מוצג בפנינו מבנה ויזואלי הרבה יותר נוח וקל לתקשר לנתח, לעלות על כשלים, ולשפר את מבנה הנתונים. הבנו את זה בשלב מאוחר יותר ויישמנו זאת, מה שהקל מאוד לאחר מכן לבצע את השילוב והאינטגרציה בין בסיס הנתונים, לקוד המערכת וממשק המשתמש.