קבוצה 4

שם מלא	ת"ז	כתובת דוא"ל
יואב כץ	207731118	yoav.katz@e.braude.ac.il
ליטל לשצ'ינסקי	208658948	Lital.Leschinsky@e.braude.ac.il
איריס קנטר	211566484	Iris.Kanter@e.braude.ac.il
רועי דרום	313264822	roi.darom@e.braude.ac.il
יעל קנטר	315823245	yael.kanter@e.braude.ac.il

תאריך ההגשה : 20.6.23

שאלה 1

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

(א) תיאור תהליך התכן עבור היכולת: יצירת מבחנים. -

תארו דילמות הנדסיות ספציפיות שעסקתם בהן בתכן של יכולת זו ואופן פתרונן. -

לצורך זה השתמשו בפורמט של תיאור Design issue כפי שמופיע בהרצאה על Design .

תארו את הדילמות, השיקולים וקבלת ההחלטות שלכם והסבירו את הפתרונות שבחרתם.

תהליך יצירת המבחן כולל מספר מסכים בו משתמש המרצה ליצירת מבחנים בתוך מערכת ה-CEMS.
זוהי בעצם סביבת עבודה שבה המרצים יכולים לבחור שאלות, למחוק אותן, להגדיר את מגבלת הזמן, להקצות נקודות לכל שאלה ולכלול הוראות או הערות נוספות.
הממשק נועד להיות קל לשימוש עבור המרצים על מנת שיוכלו לבנות מבחנים ביעילות.

המטרה הייתה ליצור ממשק ויזואלי ואינטואיטיבי שהמרצים יוכלו ליצור מבחן בקלות ולחסוך זמן.

דילמה 1: דילמת העיצוב- כיצד לעצב את תהליך יצירת המבחנים?

מטרה: יצירת ממשק ידידותי וברור למרצה.
חווית המשתמש חשובה לנו ועל כן שאפנו לספק חוויה יעילה למרצים בעת יצירת מבחנים.
ערכנו מחקר שבו שאלנו שאלות רלוונטיות על מנת להבין את הצרכים והעדפות שלהם לממשק נוח ויעיל ככל הניתן.

אפשרות 1:

לכל מבחן שאלות משלו והן נשמרות יחד איתו כרשומה אחת.

אפשרות 2:

שלוש טבלאות SQL. אחת למבחנים, אחת לשאלות ואחת המקשרת ביניהן.

אפשרות 3:

בנייה "אוטומטית" של מבחן ע"י הגרלת שאלות מתוך המקצוע ולא יהיה ניתן להשתמש בה אלא אם כן היא עוברת עריכה.

פתרון:

קיימות טבלת SQL של שאלות, טבלת מבחנים, וטבלה המקשרת בין כל שאלה למבחן.

פתרון זה מאפשר לשנות את השאלות במבחנים בצורה קלה ונוחה.

בעת יצירת מבחן, המרצה מוסיף את שאלות מהמאגר למבחן ולמעשה מוסיף לטבלה המקשרת את מספרי הזיהוי של המבחן והשאלות.

בחירת השאלות מהמאגר מתבצעת ע"י סינון על מסך נושא בקורס, דבר אשר סייע בייעול הזמן לבחירת השאלות הנדרשות למבחן.

לאחר סיום בניית המבחן ישנו מין דף סיכום שמציג באופן חזותי את כל השאלות ואת המאפיינים של השאלות על מנת שהמרצה יוכל לוודא תקינות המבחן, ווידוא שהשאלות לא חוזרות על עצמן או שרמת השאלות סבירה ואין שום טעויות ולבסוף מאפשר את יצירתו של המבחן.

אולם, בפתרון זה נדרש מהמרצה להוסיף את השאלות מראש בתהליך יצירת השאלות ואינו יכול "להשלים" שאלות בעת יצירת המבחן.

דילמה 2: הגדרות זמן מבחן וניקוד-
מטרה: היא לספק למרצים את האפשרות לקבוע את זמן הבחינה והניקוד בצורה קלה ונוחה.

אפשרות 1:

הגדרת ערכי זמן וניקוד באופן חופשי ע"י המרצה

אפשרות 2:

הגדרת ערכי זמן וניקוד מתוך רשימה מובנית אשר כוללת את האפשרויות ללא צורך בהקלדה.

אפשרות 3:

חלוקת זמן וניקוד באופן יחסי ושווה באופן אוטומטי לפי מספר השאלות.

פתרון:

בחירה מתוך רשימה מובנית את הניקוד ואת זמן המבחן מבלי ללא "טקסט חופשי" וכך למנוע שגיאות הקלדה.
באופן זה אנו מצמצמים את מרווח הטעויות ויחד עם זאת מספקים למרצים מגוון אפשרויות בחירה.

בנוסף, הגדרנו ששאלות שונות יכולות לקבל ניקוד שונה במבחנים שונים (בעזרת הטבלה המקשרת) וכך מאפשר למרצים להתאים את מבנה הבחינה בצורה אישית.

(ב) ציינו אילו מהעקרונות (*) שנלמדו בהרצאות הקורס בנושאים: Architecture , Design ,

Reuse, ו - Design patterns שימשו אתכם בתהליך התכן הכלל מערכתי שביצעתם -

והסביר ו באיזה אופן הבאתם אותם לידי ביטוי באמצעות דוגמאות קונקרטיות ספציפיות –

מתוך המערכת " CEMS " שפיתחתם (לא לכלול תיאורים כלליים גנריים כמו: - timer ,

packages JAVA, OCSF באופן כללי, וכו' '. יש לפרט בהקשר קונקרטי מפורט של -

המערכת).

(*) לכל אחד מהעקרונות ת האלה, הסבירו מה הם היתרונות המתקבלים משימוש בהם .

Architecture Principle: Divide and Conquer

במערכת חילקנו רכיבים ופונקציונלית למודולים נפרדים, מודולים אלה משפיעים אחד על השני אבל ניתן לשנות כל אחד מהם באופן עצמאי כך שהמודל האחר לא יפגע. כל מודול אחראי על פונקציונליות אחת.
לדוגמא: מודולים לאימות משתמשים (מסך סטודנט, מסך מרצה) ובעת הזיהוי נפתח המסך הרלוונטי של המשתמש, מסך ליצירת שאלות, מסך ליצירת מבחנים ומסך ניתוח תוצאות.
לתהליך העבודה במודולים ישנם יתרונות: תחזוקה קלה יותר של המערכת (כל חבר צוות אחראי על מודול אחר), ישנה השפעה מינימלית של שינויים או עדכונים במודול אחד עם מודולים אחרים דבר שמחזק את הגמישות והיכולת לעדכן או לשנות דברים מבלי לפגוע בשאר. בנוסף, בצורה זו נוכל לבצע בדיקות למערכת בצורה אמינה.

Reuse Principle:

דוגמאות לשימוש חוזר:

- שימוש בממשק Initializable לאתחול הבקרים להגדרת צורת המסך הראשונית.
- שימוש בממשק Serializable לכל המחלקות לצורך העברה שלהן בין הלקוח לשרת.
- שימוש בOCSF framework . הכוללת מתן שירותים התקשורת של השרת והלקוח.
- רכיבי ממשק-מסך בחירת שאלות, תצוגת תוצאות. ישנו שימוש חוזר גם בכפתורי חזרה או מעבר למסך הבא, שימור גם בהודעות קופצות באופן חוזר.

ע"י שימוש בחזרות ניתן להפחית כפילויות של קוד ולשפר את יעילות הפיתוח, כך בעצם מבטיחים עקביות. שימוש בחזרתיות, חסכון זמן וגמישות המערכת.

Design Pattern: Model-View-Controller (MVC) Pattern:

דרך מודל זה נבנה ממשק המשתמש, הלוגיקה והנתונים ב- CEMS.
המודל מייצג את יחסי הגומלין בין הנתונים מהDB (תלמידים, מרצים, שאלות, מבחנים, סטטיסטיקות) לחישובים התנאים השונים מחושבים לפי רצף פעולות שאותו הגדרנו ובין הצגת המידע למשתמש.

בפרויקט שלנו, קבצי הfxml מייצגים את התצוגה הנראית למשתמש. הקונטרולרים וקבצי הג'אווה תחת פרויקט "Server" מייצגים שכבת הלוגיקה בהם מתבצעים החישובים, שליחת המידע להצגה ובקשת/שליחת מידע מבסיס הנתונים והגדרת המחלקות בפרויקט common וטבלאות הSQL משמשים להגדרת הנתונים ולשמירתם.

דפוס עיצוב מסייע הן בארגון הפרויקט לצורך הבנתו ותחזוקו והן בהגדרת תפקיד הרכיבים השונים.

Design Pattern: Observer Pattern

מנגנון זה הוא דפוס עיצוב התנהגותי המאפשר יצירת תקשורת בין אובייקטים שונים, לדוגמא, כאשר מרצה משנה את הציון של סטודנט. הטבלה מתעדכנת באופן ישיר ללא צורך בהגדרת הטבלה מחדש.

Design Pattern: Singleton Pattern:

באמצעות דפוס זה אנו מבטיחים שישנו רק מופע אחד שניגש לDB, מחלקה יחידה זו מטפלת בחיבור למסד הנתונים ודרכו עובר המידע. דבר זה מבטיח זמינות של משאב משותף יחיד ומונע ריבוי מופעים. כלומר, לא נרצה שלסטודנטים תהיה גישה למסד נתונים, זוהי גם פגיעה בפרטיות וגם לא נרצה שתהיה להם אפשרות לעדכן ציונים לעצמם או לראות את המבחנים.

שאלה 2

תארו את תהליך המימוש (implementation) שביצעתם עבור המערכת " CEMS " בהקשרים :

(א) בדיקות. 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 דוגמאות ספציפיות (לא כללית/גנרית ולא על

Login) שביצעתם (או לא ביצעתם) במהלך הפרויקט ע"י תיאור מפורט של בדיקות

מרכיבים ספציפיים של מערכת " CEMS " ("ספציפי" כלומר: לא בהתייחסות כוללנית

גנרית כמו : "שדות ריקים", "התחברות לשרת", "כתיבה ל- DB ", "דוחות " וכו'. מה כן: יש

לפרט מה בדיוק נבדק בדגש על תהליכים ספציפיים של המערכת המפותחת).

בדיקת "יצירת מבחנים"

בדקנו את הפונקציונליות כ"קופסה שחורה".

תחילה, בדקנו את הפונקציה addQuestionSubmit :

1. הגדרנו מראש שאלה במקצוע מתמטיקה, את התשובות ואת סימון התשובה הנכונה בשדות המתאימים בחלון להוספת שאלות ואת המשתמש (מרצה). הנחנו כי הfxml תקין.
2. הגדרנו mock לClientMissionHandler כך שבעת קריאה לפונקציה ADDQUESTION נקבל את ערכי השדות שמילאנו
3. הגדרנו מקרה בו השדות תקינים ומקרה נוסף בו ישנו שדה null באחד מהשדות.
4. הרצנו את הפונקציה, כלומר דימינו לחיצה על כפתור הsubmit
5. וידאנו כי ערכי השדות שחזרו זהים לתנאים שהוגדרו בהתחלה. כך וידאנו כי הנתונים שישלחו לשרת זהים לקלט ובמידה שיהיה שינוי בקוד (למשל, הוספת פרמטרים), הבדיקה לא תעבור.

בדיקת "ביצוע מבחן ממחושב"

בדקנו את הפונקציונליות כ"קופסה לבנה".

תחילה, בדקנו את הפונקציה markQuestion אשר מחזירה את מספר התשובה שסומנה

1. הגדרנו OnlineTestController ורצינו לבדוק את סימון השאלה הנכונה ולכן יצרנו חלון עם ארבעה כפתורים מסוג "רדיו"
2. הרצנו את הפונקציה, כלומר, דימינו סימון שאלה
3. וידאנו כי הפונקציה מחזירה את אינדקס השאלה הנכון

בשלב 2, שמנו לב כי ניתן לסמן יותר מתשובה אחת נכונה, והמשתמש לא ידע איזו תשובה סימן. לכן, הוספנו תנאי כך שעבור כל תשובה אחרת שאינה התשובה שנבחרה ע"י המשתמש, סימונה יתבטל.

(ב) אינטגרציה. תארו את תהליכי האינטגרציה שביצעתם במהלך תהליך הפיתוח .

▪ תארו את מהלך האינטגרציה ציינו לוחות זמנים ופרוצדורה (תהליך) באמצעות – -

דוגמאות קונקרטיות ספציפיות (לא כלליות) מתוך המערכת " CEMS " שפיתחתם.

▪ ציינו באילו כלי תוכנה השתמשתם באינטגרציה והסבירו באיזה אופן .

▪ ציינו אילו מהעקרונות שנלמדו בהרצאה בהקשר של אינטגרציה.

החלטו לתזמן את האינטגרציות בהתאם לתהליך הלוגי:

איטרציה ראשונה: יצירה (מבחנים, שאלות)

איטרציה שנייה: ביצוע (מבחנים, שאלות)

איטרציה שלישית: ניתוח (דוחות וצפייה בנתונים)

בגישת Bottom-Up כלומר, בדקנו תחילה פונקציות בודדות ולאחר מכן בקרים שלמים ולבסוף התנהגות כוללת של שרת ולקוח – המערכות של התוכנה ובכל שלב שילבנו רכיבים ותתי מערכות שונות ליצירת מערכת בעלת פונקציונליות מיטבית.

השתמשנו במערכת ניהול מסדי נתונים: MySQL לצורך יצירת טבלאות והקשרים ביניהם. חיבור כל המערכת למסד נתונים תוך שאילת שאילתות רלוונטיות.

Eclipse- מכיל את הקוד של כל הפרויקט וגם קבצי הFXML של העיצוב של המערכת.

Scene Builder- השתמשנו על מנת לעצב את המסכים.

שאלה 3

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

(א) תיאום פעילויות ושיתוף בין חברי הצוות בפיתוח כולל גישה 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. –

תהליך העבודה:

תחילה נפגשנו כל הצוות יחד על מנת ליצור דף דרישות של הפרויקט, מה צריך להיות כלול בפרויקט תוך שימוש בהנחיות שנתנו.
לאחר מכן, התחלנו לעבוד יחד על ההתחברות בין השרת והלקוח והחיבור לDB
התפצלנו ל-3 צוותים כאשר כל צוות היה אחראי על מודל מסוים, כל צוות עשה את המסכים הרלוונטיים עבורו.
צוות 1: הפקת דוחות וסטטיסטיקות
צוות 2: פונקציונליות של סטודנט וראש מחלקה
צוות 3: פונקציונליות של המרצה

מידי שבוע נפגשנו על מנת לאחד את חלקי הפרויקט.
תחילה ניסינו להשתמש בGIT לאחר מכן לא הצלחנו לבצע merge מפאת חוסר הזמן אחדנו את החלקים בעצמינו.

יתרונות:
החלוקה הייתה שווה ביחס לעבודות, כל אחד לקח חלק במימוש הפרויקט, כשצוות אחד התקשה, כולם נרתמו לעזרתו וניסו לעזור.

חסרונות:

הייתה תלות בין המסכים- היו מקרים שבהם צוותים לא יכלו להתקדם במשימה עד אשר צוות אחר לא סיים.
למשל הצוות שהיה אחראי על ביצוע המבחן בזמן אמת מצד הסטודנט היה צריך להמתין עד שהצוות שאחראי על כתיבת שאלות ומבחן יסיים.
הקונפליקט היה האם לחכות לצוות שיסיים או לנסות בלעדיו.

לוז פגישות:

היו פגישות ספונטניות כשצוות מסוים היה צריך עזרה ובנוסף לכך היו 2 פגישות שבועיות של איחוד הפרויקט אחד בתחילת שבוע ושני בסוף שבוע.
שאר הפגישות שהיו בצוותים הקטנים התקיימו מידי יום.
מודל העבודה שבחרנו ליישם גרם לכל אחד מחברי הצוות להתמקצע בנושא מסוים ולא רק זה אלא גם לסייע לצוות אחר וכך גם להיות חשוף לנושאים אחרים שלא היו בתחומו.

(ב) ניהול הפרויקט בשלב הבניה (Construction) קידוד, שילובי קוד(אינטגרציה מערכתית) – ובדיקות.

ציינו באופן פרטני, בהתייחסות ספציפית לפיתוח המערכת " CEMS ", איך פעלתם בשלב זה של הפיתוח (כולל: תיאור התנהלות התהליך ההנדסי (לא דינמיקה ועבודת הצוות), בקרה: איך טיפלתם בבעיות טכניות הקשורות לתהליך הפיתוח, וכו'.

אם היו קשיים מה הסיבה לכך? מה הייתם משנים בדיעבד בגישתכם למרכיב זה של תהליך הפיתוח מבחינת האספקטים הרלבנטיים של הנדסת תוכנה?

הדבר שסייע לנו זה לכתוב לפני תחילת הכתיבה של הקוד את כל הדרישות ואת כל הפונקציונאליות הנדרשת

מבחינת טכנית היה לנו קושי בחיבור החלקים מכל צוות כיון שלא הייתה לנו היכרות מספיקה עם github.

נעזרנו במידע נוסף מהאינטרנט וכן פנינו לסגל עם הבעיות שנתקלנו בהן אך מאחר ועדיין לא הצלחנו לסנכרן את כל חברי הצוות, איחדנו את הקוד בצורה ידנית. בדיעבד, היינו צריכים לוודא כי אנו מבינים כיצד github עובד בטרם תחילת העבודה על מימוש הקוד.