מסמך תיאור עבודת הפרויקט

.**1 עבודת הצוות :**

**תהליך העבודה:** בהתחלה התארגנו לקבוצה , התחלנו עם הבנת דרישות הפרויקט בנוסף להיכרות עם המשחק (snake and ladders) מבחינת חוקים, אילוצים , ישויות , דרישות וכו'...

אחר כך עברנו לחלק של אפיון הפרויקט (SRS),שהוא מפרט דרישות שתיארנו בו במדויק את הדרישות הפורמליות והפונקציונליות של מערכת התוכנה שאנחנו אמורים לפתח בהמשך לפיתוח שהמטרה העיקרית ממנו היא להגדיר במדויק את מהלך הפרויקט ולהבטיח שיש הבנה משותפת בין הלקוחות, הפיתוח והבדיקה.

ולצורך האפיון היעזרנו בתקן IEEE.

אחרי הבנה מעמיקה לתיאורית המשחק עברנו לשלב של SDP (תוכנית פיתוח תוכנה) , זהו מסמך שתיארנו בו את תוכנית הפיתוח שלנו (המשחק) מתחילתו ועד לסיומו .

SDP הוא ראשי תיבות של "Software Development Plan" או "תוכנית פיתוח תוכנה". זהו מסמך המתאר את תוכנית הפיתוח של פרויקט תוכנה מתחילתו ועד לסיומו.

במסמך ה-SDP ציינו שאנחנו נשתמש ב- Agileמתדולוגיה שהיא גישת פיתוח תוכנה איטרטיבית , ועל פי זה תארנו בטבלה את כלל האיטרציות שלנו ואת סיפורי המשתמש שהם כוללים את הפעולות שאנחנו רוצים לבצע כך שסדרנו אותם לפי החשיבות ועם תאריך הסיום של כל איטרציה.

אחרי זה בנינו את אב טיפוס משלנו שבו רצינו להבהיר ולחדד את הפעולות העיקריות שהמערכת אמורה לבצע ( דרישות פונקציונליות) שעזר לנו לזהות את התקלות שנוכל להתקל בהם ולהחליט איך לעצב את המערכת שלנו כך שתהיה גמישה וניתנת לתחזק .

בשלב הבא התחלנו עבודה עם ה- Github , שהמטרה ממנו היא לשתף את הקוד בין חברי הקבוצה שלנו ולהתחיל בשלושת האיטרציות של מימוש המשחק.

בשלושת האיטרציות היינו עוברים על סיפורי המשתמש שציינו אותם מקודם לפי החשיבות ומחלקים משימות .

הגשנו כל איטרציה יחד עם קובץ Scrum שהוא קובץ טקסט כולל את הגדרת תפקידי הצוות, וזרימת את כלל התהליך.

באיטרציות גם הכללנו בדיקות קופסה שחורות ולבנות בנוסף לתבניות עיצוב.

אחר כך עשינו סגירת פרויקט ודאגנו להוסיף בונוסים ופיצ'רים נוספים שהרחבנו עליהם בקובץ נפרד של הבונוסים.

**תקשורת בין חברי הקבוצה :**  מבחינתנו התקשורת בין חברי הקבוצה היא מרכיב עקרוני להצלחת הפרויקט וליעילות העבודה , לצורך כך היינו משתפים פעולה תמיד ומתקשרים דרך קבוצת וואטצפ שבה היינו מתכננים לפגישות , לשלוח עדכונים במהלך העבודה ולחשוב ביחד על בעיות שנתקלנו בהם .

בנוסף לזה היינו נפגשים באופן פרונטלי גם כדלי לעבוד יחד על משמיות משותפות , בנוסף לזה גם השתמשנו במפגשים מכוונים דרך google meet לצורך עבודה על הפרויקט.

* **אתגרים איתם התמודדתם :**
* ניהול זמן : יש מקרים שבהם היינו מקרים זמן פחות למשימה שציפינו שלא תיקח הרבה זמן לכן לא תמיד היינו מצליחים לנחש את הזמן שצריך להקצות אותו למשימה מסוימת.
* ניהול משאבים : משאבים כמו eclipse, swing.. לא השתמשנו בהם מזמן לכן לקח לנו הרבה זמן כדי לסדר אותם .
* בעיות טכניות: בעיות בקוד כמו שגיאות בקוד ובגלל שהקוד ענק ומוסך זה היה אתגר שתמיד קשה להתמודד איתו.
* ניהול שינויים: בהרבה מקרים היינו נאלצים לעשות שינויים , כמו למשל הגדלת לוח המשחק שעקב זה היינו גם עושים שינויים על כל האובייקטים בלוח וזה היה מסובך ולוקח הרבה זמן ומאמץ.
* בעיות עם ה- github : קיבלנו הרבה קונפלקטים , דריסת קוד , עשינו כמה repositories מחדש.

**מה למדתם מהעבודה בקורס ובצוות** :

קודם כל הקורס סיפק לנו ניסיון משמעותי בתחום ההנדסת תכונה גם מבחינה תיאורתית ומבחינה מעשית , בהקשר של העבודה בצוות למדנו כיצד לתקשר ושתף פעולה וידע ואיך זה משפיע על זרימת העבודה שלנו .

בנוסף לכך למדנו איך לנהל פרויקט ענק בצורה יעילה להגיע למטרה שלנו כמו למשל ניהול זמן , סדר של משמיות.

גם קיבלנו את ההזדמנות להשתמש בכלים משמעותיים בתחום כמו github, למדנו גם איך להתמודד עם באגים ושגיאות לא צפויות ולפתור אותם באופן יעיל .

כל זה ספק לנו יכולת למידה והתפתחות פרקטית בתחום ההנדסה והפיתוח של תוכנה, ומספק יסודות לקריירה בתחום.

1. **אלמנטים מיוחדים שהשתמשנו בהם:**

**טכנולוגיות מיוחדות :**

השתמשנו ב chatGbt שעזר לנו לבצע debugging לקוד .

עבור sound effects הורדנו אותם מ - pixabay.com בנוסף לזאת עשינו המרה לקבצי הקטל מ-MP3 ל WAV דרךcloudconvert.com.

צפינו בסרטוני יוטיוב שהסבירו איך להפעיל קבצי קול בגאווה .

והשמשנו ב JUNIT כדי לבצע בדיקות לקוד . בנוסף ל SWING ו Windows Builder לבניית ממשק המשתמש.

השתמשנו הקבצי jar עבור ספריות ה. json השתמשנו בשיטת העיצוב המודרנית FlatLaf-Flat look and feel.

קצת הסבר עליה:

**FlatLaf** is a modern **open-source** cross-platform Look and Feel for Java Swing desktop applications.It looks almost flat (no shadows or gradients), clean, simple and elegant. FlatLaf comes with **Light**, **Dark**, **IntelliJ** and **Darcula** themes, scales on **HiDPI** displays and runs on Java 8 or newer.

**תבניות עיצוב :**

**MVC** (1

לפי בקשת צוות הקורס חלקנו את הקוד שלנו ל-3 packages

שהם Model View Control

**Observer pattern** (2

השתמשנו בתבנית זו במחלקה של SysData, במחלקה זו יש לנו פונקציות של הוספה\עריכה\ עדכון של שאלה בקובץ JSONבכל פעם שהמשתמש מבצע פעולה במסך של ניהול השאלות הפונקציות האלה מקבלות התראה ומעדכנות את מסד הנתונים בהתאם וזה גם בא ע"י הביטוי בממשק המשתמש כך אם המשתמש מוחק שאלה מסוימת היא נמחקת מיד מהטבלה בנוסף לזאת היא תמחק מיד ממסד הנתונים שהוא קובץ הJSON וזה נכון גם עבור הוספה ועריכה. בחרנו בתבנית זו כי הוא עוזרת לנו לנהל את השאלות בצורה נכונה ולהציג למשתמש מידע עדכני בקשר לשאלות כך שקובץ ה JSON תמיד מעודכן בהתאם לפעולות המשתמש.

Template Pattern (3

יש לנו מחלקה אבסטרקטית בשם MovePlayerTemplate שמכילה את החתימה של הפונקציה האבסטרקטית MovePlayer(int playerX, int playerY) שמקבלת קןררדינאטות השחקן פונקציה זו מספקת שלד לפעולה של הזזת השחקן.

**Factory** (4

יש לנו את המחלקה DiceFactory שמשמשת כ factory ליצירת קוביית המשחק לפי הרמה שהיא מקבלת ויש את המחלקה dice שהיא מחלקת האב לכל סוגי הקובייה השונים שהם תתי מחלקות שיורשות ממנה והם EasyDice ,MidDice,HardDice

כל אחת מהם שייכת לסוג שונה של קובייה שתואם לרמת המשחק כלומר EasyDice מתאימה למשחק קל כך שהסטטיסטיקות וההסתברויות שלה מתאימות לכללי משחק מסוג זה וזה נכון עבור שאר הסוגים.פונקציית createDice במחלקה DiceFactory אחראית על יצירת סוג הקוביות המתאים בהתבסס על הרמה המסופקת.

**ממשקי המשתמש:**

אב הטיפוס של ממשק במשתמש עוצב באמצעות. Figma

השתמשנו בספריית Java Swing בנןסף השתמשנו בשיטת העיצוה המודרנית FlatLaf.

**אלגוריתמים:**

אלגוריתמי מיון למיון תור השחקנים.