1. **היתרונות של מודל UML כעזר לתהליך התכנון:**

דיאגרמות UML מהוות סטנדרט שמשמש לתאר בצורה ויזואלית וחזותית תוכנה, בעיקר עבור תוכנה מבוססת פיתוח מונחה עצמים. הדיאגרמות עוזרות לארגן, לתכנן ולראות את התוכנה. בנוסף לכך, כיוון שדיאגרמות אלו מהוות סטנדרט פיתוח הרי שהן משומשות ומקובלות בתעשייה כדרך לתאר תוכנות ומשמשות בתור שפה משותפת בין מתכנתים. כמו כן יתרון נוסף הוא הנוחות שבקריאת הדיאגרמות והאפשרות לשימוש חוזר בחלקים נרחבים מהדיאגרמה בפרויקטים שונים.

* **תיאור חזותי של המידע:**

דיאגרמת Class היא בעצם תיאור חזותי של הקשרים בין המחלקות והישויות בתוך אפליקציות ותוכנות מחשב. מחלקה היא בעצם אובייקט תכנותי שמסדר משתנים ומתודות במקום אחד על פי הקשר ביניהם. בכדי להבין את הקוד והתוכנה, זה הכרחי להבין מה כל מחלקה עושה, המידע שהיא שומרת וכיצד זה בא לידי ביטוי בתמונה הגדולה ביחס לשאר המחלקות בתוכנה. על ידי הצגת המידע בדיאגרמה, ניתן לראות ולהבין בצורה פשוטה את הקשרים בין החלקים והמחלקות השונות בתוכנה.

* **נוחות קריאה:**

דיאגרמות UML הן מאוד קריאות, הן נועדו להיות מובנות על ידי מפתחים מתחומים שונים ועוזרות להסביר את הקשרים של התוכנה בצורה פשוטה ומובנת. באופן מסורתי, בכדי להבין תוכנה, על המפתח לקרוא את הקוד בצורה ישירה. כאשר מדובר בתוכנה גדולה ייתכן ומדובר במאות אלפים ואף מיליונים של שורות קוד. כאשר ניתן לחסוך את זמן ההבנה בהרבה על ידי התבוננות בדיאגרמות אשר ממחישות את הקשרים והפעולות המתבצעות.

* **שימוש חוזר:**

באמצעות דיאגרמות של Activity וSequence ניתן למדל ולהראות קטעי קוד שרצים ופועלים בתוכנה. מפתח התוכנה יכול לזהות ולראות קטעי קוד מיותרים או לחילופין קטעי קוד שחוזרים על עצמם, ובכך בעת תהליך המימוש יוכל לבצע שימוש חוזר (re-use) בקטעי קוד אלה במקום לכתוב את אותן פונקציות מחדש ובכך לחסוך זמן.

* **סטנדרט פיתוח:**

דיאגרמות UML מהוות בעצם סטנדרט או פרוטוקול פיתוח עבור פיתוח תוכנות בשפות מונחות עצמים. כאשר יוצרים מחלקות ואובייקטים עם קשרים ביניהם, דיאגרמת UML היא מה שמאפשרת לנו לתאר בצורה ויזואלית את קשרים אלו. כיוון שהדיאגרמות משמשות כסטנדרט הן ניתנות להבנה על ידי מפתחים רבים, ובפרט זה מאפשר למפתחים חדשים לקפוץ לתוך הפרויקט בקלות ולהיות מועילים ופרודוקטיביים מהיום הראשון בעבודה.

* **כלי תכנון:**

דיאגרמות UML עוזרות לתכנן את התוכנה לפני שמבצעים את שלב המימוש והתכנות. בכלי שאנו משתמשים ישנן דרכים לבצע יצירת קוד בצורה אוטומטית על בסיס הרכיבים שתוארו בדיאגרמה. עובדה זו יכולה לעזור לחסוך זמן בשלב המימוש של התוכנה. כמו כן ניתן לבצע שינויים בדיאגרמות בצורה פשוטה בעוד שלבצע שינויים ותיקונים בקטעי קוד יכולה להיות עבודה סיזיפית אשר גוזלת זמן רב. לכן מומלץ לבצע תכנון של הפרויקט באמצעות הדיאגרמות.

**יתרון ספציפי של הדיאגרמה:**

בפרויקט שלנו ישנו תהליך מרכזי וחשוב של הגשת בקשה וטיפול בבקשה. כאשר הטיפול בבקשה מורכב ממספר בעלי תפקידים שונים אשר כל אחד מהם מבצע שלב אחד בודד מתוך תהליך גדול. לאחר ניתוח הסיפור והדרישות שנובעות ממנו, הצלחנו בצוות לאבחן ולנתח שתהליך זה מהווה את לב ליבה של המערכת והפונקציונליות העיקרית שלה. כלומר ישנה חשיבות גבוהה ביותר להבנה של התהליך ומימושו בצורה שתענה על דרישות הלקוח וכמובן תאפשר ניצול אידאלי ומיטבי של משאבינו. נדרש מימוש שיאפשר לנו לבצע שימוש חוזר בקטעי קוד שונים ולפשט את הפונקציונליות לתתי פונקציות אשר כל אחת מהן עושה משהו בסיסי ופשוט ובכך לאפשר לנו קוד קריא יותר. כמו כן יקל על הנוחות בסביבת עבודה בצוות כאשר העבודה הינה שיתופית.

דרך דיאגרמת הActivity הצלחנו לתאר בצורה חזותית וברורה את זרימת המידע, השלבים והשחקנים המשתתפים בכל שלב עבור תהליך הגשה וטיפול בבקשה. הצלחנו לזקק את הפעולות המתבצעות ולהבין את כל ההסתעפויות השונות שיכולות לקרות במהלך התהליך.

לדוגמא בשלב "בחינה והחלטה" ייתכנו שלוש אופציות שונות: דחייה, אישור או בקשת מידע נוסף. לאחר המידול ברור לנו שכל אופציה בהסתעפות בעצם תעביר את הבקשה לשלב אחר, אישור הבקשה יקדם אותנו אל שלב "ביצוע", דחיית הבקשה תקפיץ אותנו היישר אל שלב ה"סגירה" ובקשת פרטים נוספים תחזיר אותנו אחרוה אל שלב "הערכת משמעות".

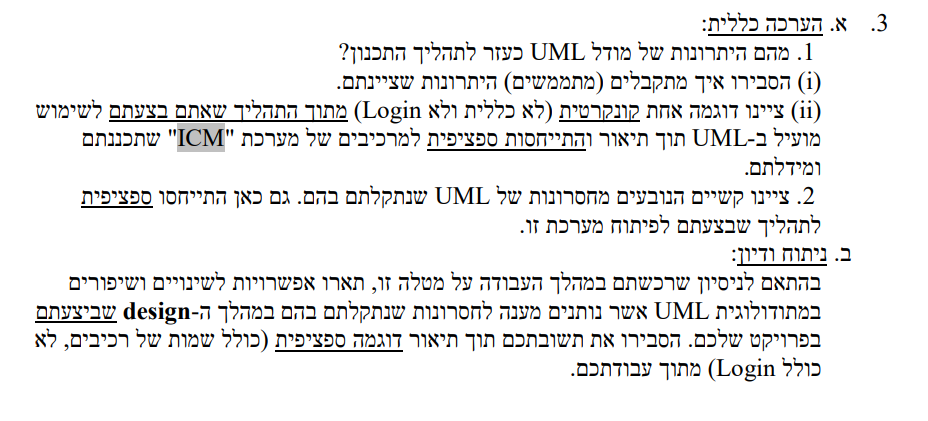
כמו כן זיהינו תוך כדי המידול שאומנם אנו מבינים את הזרימה של התהליך בגדול אך כל שלב בפני עצמו מהווה רצף שלם של אירועים שצריכים להתרחש ולכן בחרנו לתאר כל שלב בתור תת דיאגרמה שגם היא תהיה דיאגרמת Activity.

מידול זה מאפשר לנו רמות שונות של רזולוציה על התהליך והבנה עמוקה יותר של כל אחד מהשלבים שמתבצעים, כמו כן בעזרת ההסתכלות העמוקה ניתן לזהות מספר פעולות שחוזרות על עצמן בכל שלב כגון: קביעת זמן הנדרש לביצוע השלב (בשלבי "הערכת משמעות" ו"ביצוע") והמתנה לאישור המפקח.

במהלך תכנון הפרויקט ומימוש מייצג ההדגמה ואב טיפוס פעיל, השתמשנו בדיאגרמת Class בכדי לתאר את כל המחלקות והרכיבים השונים של הפרויקט. החל מכל הצגים השונים, אשר נקראים GUI – Graphical User Interface, דרך כל הבקרים שלהם (Controllers), וכמובן הישויות (Entities) והאובייקטים שיהיו לנו בפרויקט. מידול זה עזר לנו לתכנן מראש כיצד יראה הפרויקט וכמובן אילו מתודות עלינו לממש. זה מאפשר לנו לתת הערכת זמן והיקף עבודה די טובה לביצוע כלל העבודה.

תוך כדי תכנון הפרויקט בדיאגרמה צפו כל מיני שאלות על ארכיטקטורת התוכנה שנרצה להשתמש בה, בהתחלה רצינו להשתמש בבקר אחד כולל עבור כל התהליך של הגשה וטיפול בבקשה. אבל במבט על הדיאגרמה הבנו שבקר זה יהיה עמוס במתודות שונות, ויהיה עמוס מידי, דבר שיקשה לנו על מימושו וכמובן במבט לעתיד על תחזוקו ושמירת הקוד בצורה נקייה.

כמו כן זה עזר לנו להבין יותר טוב את זרימת המידע בפרויקט כאשר הוא יעבור מהפעולות של המשתמש בצג כלשהו (GUI) דרך הבקרים, אל מחלקת הלקוח שתדבר מול השרת שיפעיל את החיבור שלו למסד הנתונים וישלוף את המידע הרצוי ויחזירו באותו קו בכיוון הנגדי. ללא דיאגרמת Class היה לנו יותר קשה להבין את זרימת המידע הרצויה והדרך לממשה.



1. **קשיים הנובעים מחסרונות של UML**

* **יותר מידי דגש על עיצוב הפרויקט:**

דיאגרמות UML שמות דגש רב על עיצוב, דבר שיכול להיות בעייתי עבור צוותים קטנים כמו הצוות שלנו שמונה 5 מתכנתים. הרגשנו במהלך תכנון הדיאגרמות שאולי ניתחנו חלקים מסוימים מהעבודה והפרויקט יתר על המידה. כמו כן איבדנו ריכוז בכך שהשקענו זמן רב מאוד במחשבה על פיצ'רים של תוכנה שנרצה לכלול ולהוסיף.

בסופו של יום לא ניתן לפתור על בעיית תכנות באמצעות דיאגרמת UML, לבסוף צריך להפשיל את השרוולים ולהתחיל לקודד ולבצע בדיקות על הקוד. דיאגרמות UML אמורות לשמש למבט חזותי על התוכנה בכדי לעזור לנו לתכנן אותה תוך כדי פיתוח. אם נתקענו יותר מידי זמן בלאבחן ולפתור בעיות, זה יכול לגרום לעיכובים בעבודה שצריכה להתבצע בכדי לתקן את הבעיות.

* **דיאגרמות שנעשו גדולות ומסובכות:**

ככל שהתקדמנו יותר בדיאגרמות רצינו לכלול בתוכן כל תרחיש אפשרי, וכל סיטואציה שניתן להגיע אליה ולתת לה התייחסות בדיאגרמות. בסופו של יום זה גרם לנו לדיאגרמות די מסובכות ומבלבלות. לדוגמא עבור החלק של "מעקב ובקרה", יצרנו דיאגרמת Activity שלמה משלו. הרעיון היה שזוהי תהיה תת-דיאגרמה קטנה יחסית שתכלול התייחסות למקרים האלו עבור כל שלב בתהליך הגשת הבקשה, בכדי להקל על הנראות וההבנה של הדיאגרמות עבור כל שלב. בפועל מה שיצא זה דיאגרמה מאוד גדולה ומורכבת, שנעשתה אף יותר גדולה מהדיאגרמת Activty שמתארת במבט על את כל התהליך, דיאגרמה שעל פי התכנון הראשונה ציפינו שהיא תכלול את המסה העיקרית של המידע.

מחד גיסא הדיאגרמה מאפשרת לנו לתאר את התהליך בצורה מדויקת ומפורטת ומאידך גיסא יצרנו בלבול, הדיאגרמה גדולה וקשה להבנה. על אף כל הנוחות שמבט חזותי על הפעולות, יש צורך בבחינה מדוקדקת בכדי להבין את רצף הפעולות בדיאגרמה זו (Monitoring And Control).