

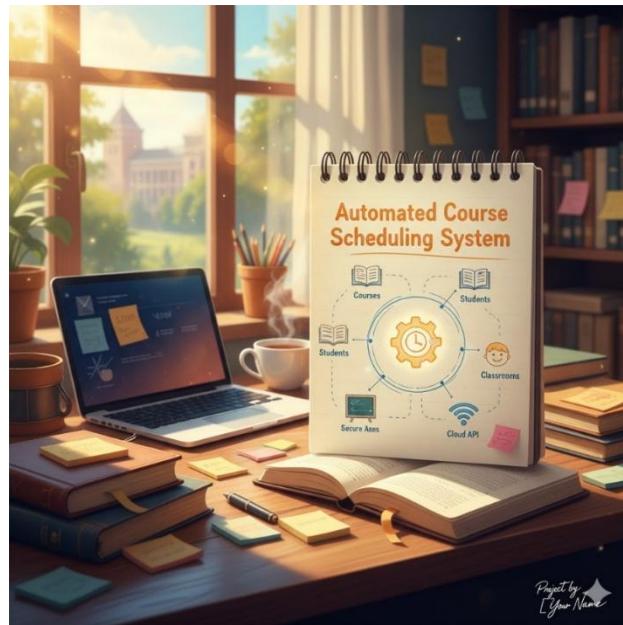


# הצעת פרויקט לעבודת גמר

י"ד הנדסאי תוכנה (שאלון 714918)

בנושא

**מערכת לשיזבוז קורסים אקדמיים ע"פ אילוצים**



שם הסטודנט: כפיר יוסף יעקבוב

ת"ז: 216232470

מנחה: אילן פרץ

תאריך : 13/1/2026

מכללה: כנפי רוח, קריית נוער ירושלים (סמל מוסד: 140129)

**תוכן עניינים:**

3.....	1.
4.....	2. רקע תיאורטי.....
7.....	3. תיאור הפרויקט.....
10.....	4. הגדרת הבעיה האלגוריתמית.....
15.....	5. הליכים עיקריים בפתרון בעיה בטכנולוגיות הנדסה מתקדמות.....
21.....	6. הליכים עיקריים בתחום למידת מכונה - לא רלוונטי.....
21.....	7. הליכים עיקריים במע' הפעלה/רשות/תקשורת/אבטחה - לא רלוונטי.....
21.....	8. תיאור פרוטוקולי תקשורת.....
22.....	9. פיתוחים עתידיים.....
22.....	10. תיאור טכנולוגיות הנדסה.....
24.....	11. מסד נתונים.....
26.....	12. פרטיים פורמליים.....

## 1. תיאור הנושא

תחום שיבוץ הממערכות (Scheduling) במוסדות אקדמיים מהוות אתגר לוגיסטי וחייב מרכיב, המשתריך למשפחת הביעות של "סיפוק אילוצים" (Constraint Satisfaction Problems - CSP). במוסדות השכלה גבוהה, יצירת לוח זמנים סמסטריאלי אינה מסתכמה רק בשיבוץ טכני של קורסים, אלא דורשת פתרון אופטימיזציה רב-מדדי שנועד לסנכרן בין משאבים מוגבלים לבין דרישות סותרות. האתגרים והבעיות הקיימות בתחום: התהילה הקיים של בניית מערכת שעות מתבצע לעיתים קרובות באופן ידני או חצי-אוטומטי, מה שmobiel במספר כשלים מרכזים:

- טעויות אנוש והתגשויות: קשי במקבב אחר מאות אילוצים בו-זמןית גורם לשיבוצים כפויים (התגשויות כתות או מרצים) או ליצירת מערכות שאינן בנות-ביצוע עבור הסטודנטים.
- חוסר יעילות וניצול משאבים: ללא אלגוריתם אופטימיזציה, קשה לנצל את חדרי הלימוד ואת שעות ההוראה בצורה מיטבית, מה שmobiel לבזבוז משאבים מסוימים.
- מרכיבות חישובית (Hard-NP): הבעיה מאופיינת בגידול מערבי (אקספוננציאלי) של מרחב הפתרונות ככל שמספר הקורסים והailוצים עולה. לכן, מציאת הפתרון "המושלם" בזמן סביר היא משימה כמעט בלתי אפשרית ללא שימוש בהיוריסטיות מתקדמות.
- ריבוי אילוצים סותרים: הצורך לאזן בין אילוצים "קשיים" (שאסור להפר, כגון מרצה שלא יכול ללמוד בשני מקומות במקביל) לבין אילוצים "רכים" (העדפות נוחות של סטודנטים ומרצים) יוצר קונפליקטים שקשה לפתור ללא כלים ממוחשבים חכמים.

המטרה בתחום זה היא לפתח מנגנונים שיאפשרו מעבר משיבוץ ידני ומסורבל לשיבוץ>Dינמי, הממקסם את שביעות הרצון של כלל הגורמים (סטודנטים, מרצים והנהלה) תוך עמידה קפเดנית בכל אילוצי המערכת.

השלכותיו של תכנון לקוי בתחום זה אין נעצרות ברמה הלוגיסטיבית בלבד, אלא משפיעות באופן ישיר על איכות ההוראה וההישגים האקדמיים. מערכת שעות שאינה אופטימלית עלולה ליצור חסמים משמעותיים בפני סטודנטים, כגון חוסר יכולת להירשם לקורסי חובה עקב חפיות בזמן, מצב המוביל לעיתים קרובות להארכת משך התואר שלא לצורך ולפגעה בחווית הלימוד הכללת. כמו כן, חלונות זמן ארוכים ובلتאי סבירים בין הרצאות גורמים לבזבוז זמן יקר עבור הסטודנטים והמרצים אחד, ומפחיתים את האטראקטיביות של המוסד.

מן היבט המוסדי, ההסתמכות על שיטות ידניות גבוהה מחיר יקר במשאבי אנוש. סגלי ההוראה והמציאות האקדמיות נדרשים להשיקו מאות שעות עבודה בכל תקופה רישום בניסיונות לפתור קונפליקטים נקודתיים, במקומם להתמקדש בפיתוח פדגוגי ומחקר. בעוד של טרנספורמציה דיגיטלית, המגמה הרווחת כיום בעולם היא מעבר למערכות חכמות ("Smart Scheduling") המבוססות על אלגוריתמים, אשר מסוגלות לעבד כמוניות אדרונות של נתונים בזמן אמיתי.

מערכות אלו הופכות את תהליך השיבוץ לכלי ניהול אסטרטגי, המאפשר חיסכון בעלותות תפעול (כגון צמצום ימי פעילות של קמפוסים שלמים) ושיפור דרמטי ברמת השירות הניתנת לסטודנטים ולסגל.

## 2. רקע תיאורטי

בעית השיבוץ האקדמי נחשבת לאחת הביעות המרכזיות בעולם האופטימיזציה הדיסקרטית ובתחומי הבינה המלאכותית, לאחר שהוא משלבת כמות גדולה של אילוצים ותלוויות. כמו שהוזכר קודם, בעיה זו שיכת למשפחת בעיות ה - (CSP) Constraint Satisfaction Problems, שבהן נדרש למצוא פתרון חוקי מתוך מרחב חיפוש רחב, כאשר כל פתרון חייב לעמוד באוסף גדול של אילוצים לוגיים, מבניים וזמןניים.

בבעיות CSP כל צעד משפיע על צעדים אחרים, וכך גם החלטה שנראית נכונה בשלב מוקדם יכולה להיות שגיאה בהמשך. למעשה זה יוצר צורך בגישה אלגוריתמית שמאזנת בין עמידה מלאה בכל האילוצים לבין יעילות חישובית, במיוחד כאשר מתמודדים עם כמותות גדולות של קורסים, סטודנטים וזמןנות מרצים.

בניגוד לבעיות פשוטות שבהן ניתן לבחור את הפתרון המידי הטוב ביותר, כאן מדובר בעיה רב-מדנית: כל קורס תלוי בקורסים אחרים, כל מרצה זמן רק בחלוקת המשבוע והמערכת יכולה צריכה להגיע ללוח זמנים אקדמי תקין, חוקי ויציב. לכן, לפני בחירת האלגוריתמים המתאימים לפרויקט, נבחנו מגוון גישות שונות - חלקן קלאליסטיות וחלקן מבוססות היוריסטיות - כדי להבין אילו מהן מתאימות בצורה הטובה ביותר ביותר להתמודד עם סוג זהה של בעיות.

**אלגוריתמים שנבחנו אך לא נבחרו:**

**אלגוריתמים חמدنيים - Greedy Algorithms:** האלגוריתמים החמדניים מבוססים על בחירה מקומית של האפשרות שנראית כרגע לטובה ביותר, ללא התייחסות למצב הכללי העתידי. למורתם ייעילותם והפשטות שלהם, הם מתאימים למערכות שבהן כל החלטה עומדת בפני עצמה, או כאלה שבהן התלות בין הצעדים קטנה. בעית השיבוץ האקדמי יש תלות הדידית בין>Userות ואף מאות אילוצים: קורסים שאסור شيئاً גמור, מרצים שמלמדים על פי אילוצים אישיים, סטודנטים שלומדים ומתקנים את המערכת על פי צרכיהם האישיים, תנאי קדם שנוגעים לסדר ההוראה, וכו'...

בחירה "מהירה" שאינה מתחשבת בכל האילוצים עלולה ליצור לוח זמנים חלקים שלא אפשריים או זהה שמספר עשרות תנאי מערכת. לכן, למרות ייעילותם, האלגוריתמים החמדניים נפסלו.

**אלגוריתמים גנטיים - Genetic Algorithms:** אלגוריתמים גנטיים פועלים לפי עקרונות אבולוציוניים: יצירה אוכלוסיות של פתרונות, ביצוע מוטציות ושהזרים, ובבחירה של הפתרונות המתאימים ביותר.

למרות שהם מתאימים היטב למרחבי חישוב גדולים, הם סובלים ממספר בעיות בהקשר של שיבוץ אקדמי: הם אינם מבטיחים שהפתרון עומד בכל האילוצים הנוקשים. נדרש זמן רב עד להתקנסות לפתרון תקין. יש צורך בפונקציית מטרה מורכבת וגדולה במיוחד, שיכולה להאט את המערכת בצורה משמעותית. גם אחרי זמן רב, האלגוריתם עלול לספק פתרון "כמעט תקין", שאינו עומד בכל הדרישות. בעית שיבוץ שבו כל אילוץ חייב להתקיים, אינה מתאימה לגישה אבולוציונית שמטבעה מקבלת לעיתים סטיות. لكن האלגוריתם הגנטי נפסל.

**חישול מדומה - Simulated Annealing :** Simulated Annealing הוא אלגוריתם שפועל בצורה הסתברותית ומנסה למצוא את הפתרון הכי טוב מתוך הרבה אפשרויות. הוא משתמש ברעיון של "חימום וקירור" - בהתחלה הוא מוכן לנסות גם פתרונות פחות טובים כדי לא להיתקע במקום לא מוצלח, וככל שהאלגוריתם מתקדם הוא מצמצם את הניסיונות האקראיים ומתקרב לפתרון ייצב אויצות. למרות יתרונו בפתרון בעיות אופטימיזציה, השיטה אינה מתאימה לבעה האקדמית מכמה סיבות: האלגוריתם אינו מבטיח פתרון חוקי, הוא עשוי להסתפק בפתרון טוב אך לא תקין מבחינה לוגית. כאשר יש הרבה אילוצים, האלגוריתם עלול להיתקע בפתרון חלק ולא להגיע לפתרון הטוב ביותר. תהליך ה"קירור" שלו עשוי להיות איטי, במיוחד כשהוילוט גודלה של קורסים ושיבוצים לבדוק. כדי שהוא יבודד היטב צריך לבנות פונקציית הערכה מדויקת שמכסה את כל האילוצים, וזה תהליך מסובך במיוחד. לכן, המערכת שיבוץ כל אילוץ חייב להתקיים לפחות פשיות. Simulated Annealing אינו מתאים, וכך נפסל.

לאחר בחינת מגוון גישות, נבחרו שני אלגוריתמים מרכזיים שימושיים יחד לפתרון מיטבי של בעית השיבוץ. **מיון טופולוגי לסדר על פי תנאי קדם, I - Pruning / Backtracking עם Branch- / Backtracking** **Bound-and-Bound** להצבה המדויקת בלוח הזמנים.

**מיון טופולוגי (Topological Sort) :** מיון טופולוגי הוא אלגוריתם המשמש ליצור סדר תקין של איברים במערכת שבה קיימות תלויות וכיוניות בין רכיבים. האלגוריתם מבטיח שככל איבר יופיע לאחר כל האיברים שהוא תלוי בהם, וכך הוא מתאים במיוחד לפתרון בעיות הcoliות יחסית קדם והיררכיה. תוצאה זו מאפשרת עבודה מסודרת ועקבית עם מבנים מורכבים תוך שמירה על כללי התלות המוגדרים.

**Pruning / Branch-and-Bound עם Backtracking** באlgorigthmi חישוב רקורייבים מסוג Backtracking לצורך בחינת שילובים אפשריים תחת מערכת אילוצים. במהלך החישוב מtbodyות בדיקות שוטפות לעמידה באילוצים שונים, וכל ענף שאינו עומד בתנאים נחתך מוקדם (Pruning). דבר המפחית משמעותית את מרחב החישוב וזמן החישוב. במקרים מסוימים משולבת גם שיטת Branch-and-Bound, מאפשרת להשוות בין פתרונות חלקיים, לקבוע חסמים, ולהעדיף מסלולי חישוב מבטיחים יותר על פני אחרים.

להלן טבלה שמסבירה על יתרונות והחסרונות של כל אחד מן האלגוריתמים שנבחנו:

שם האלגוריתם	עיקון הפעולה	יתרונות	חסרונות
<b>אלגוריתמים - חמדניים - Greedy Algorithms</b>	בחירה מקומית של האפשרות הטובה ביותר. ככל צעד נתון, ללא הסתכלות על ההשלכות העתידיות.	- פשטות מימוש. - יעיל ומהיר לריצה.	- בחירה "טובה" כרגע עלולה להוביל למבי סתום בהמשך. - אינם מתאימים לביעות עם תלות הדדית גבוהה, בין אילוצים.
<b>אלגוריתמים - גנטיים - Genetic Algorithms</b>	חיקוי תהליכי אבולוציוני: יצירת "אוכלוסיות" פתרונות, ביצוע מוטציות ושיפור הדרגתית של הדורות.	- מתאים מאוד למרחבי חיפוש עצומים. - יכול למצוא פתרונות, יצירתיים לביעות אופטימיזציה.	- אין הבטחה לפתרון חוקי: האלגוריתם עלול להחזיר פתרון "כמעט טוב" שמספר אילוץ קשיח. (למשל, התנגשות קורסי חובה) - זמן התכנסות ארוך.
<b>חישול מודמה - Simulated Annealing</b>	חיפוש הסתברות המסלב "חימום וקירור" כדי לצאת ממיןימום מקומי ולהתכנס לפתרון יציב.	טוב, במניעת היתקעות בפתרונות לא אופטימליים.	- בדומה לאלגוריתמים גנטיים, עשוי להסתפק בפתרון שאינו תקף לוגית ב-100%. - דורש חישוב מרכיב של פונקציית הערך ותהליך "הקירור" איטי.
<b>מיון טופולוגי ו- Pruning עם Backtracking Branch-and- / Bound</b>	חיפוש רקורסיבי בשיטתי: הצבת משאבים וחזרה לאחר. אם נתקלים במבי סתום, תור "גיזום" ענפים שגויים מראש.	- מבטיח פתרון חוקי: לא משאיר קצOOT פרומיים. - גמישות מלאה בטיפול באילוצים מורכבים. - הגזום חוסר זמן החישוב משמעותית.	במקרה הגרוע ביותר, זמן הריצה עלול להיות ארוך.

לסיכום, הבחירה לשלב בין מיון טופולוגי לבין Backtracking עם מנגןוני Pruning נובעת מה הצורך להתמודד עם בעיית שיבוץ מורכבת בעלת ריבוי אילוצים, יחסית תלות בין קורסים, מגבלות מערכת ומשתמשים שונים. מיון טופולוגי מאפשר ליצור סדר היררכי-לוגי של הקורסים והמשאבים, כך שהמערכת מתחשבת מראש בתלוויות מבניות ובכללים שאינם ניתנים להפרה.

לעומתו, מנגנון ה- Backtracking מספק גמישות מלאה להתקנת השיבוץ בפועל - תוך בחירת מסלולים אפשריים בלבד, בעוד ש- Pruning מונע בחינה של פתרונות שגויים עוד בשלבי החיפוש המוקדמים. שילוב זה מאפשר לאזן בין דיקט מקסימלי בעמידה בכל האילוצים לבין יעילות חישובית, ובבטיח שהמערכת לא רק מוצאת פתרון, אלא מוצאת את הפתרון הסביר והעקביו ביותר מרווח מרחב אפשריות עצום.

באופן זה, שילוב האלגוריתמים מאפשר בניית מערכת שיבוץ אקדמית אמינה וგמישה, הנוטנת מענה מלא לצרכים השונים של כלל המשתמשים, תוך שמירה על יעילות חישובית ועמידה בכל האילוצים המורכבים.

### 3. תיאור הפרויקט

מטרת המערכת ואופן פעולתה היא לבצע שיבוץ אוטומטי, חכם ואופטימלי של קורסים במוסד אקדמי, תוך התחשבות באילוצים מגוונים של מרצים, סטודנטים ומנהלי מערכת. המטרה היא לספק פתרון טכנולוגי מתקדם שיחליף את התהילה הידני והמסורבל של בניית מערכת שעوت, ויאפשר ניהול יעיל, גמיש ודינמי של מערכת הלימודים בכל מסטער. המערכת תאפשר למשתמשים להזין אילוצים, לנוהל נתונים ולהפעיל אלגוריתם שיבוץ מתקדם אשר יחשב מערכת שעותมาตรฐานת, ללא התנששות בין קורסים, מרצים או קבוצות לימוד. האלגוריתם ישאף להשיג אופטימיזציה מירבית וסיפוק הרצון של כלל המשתמשים.

חוויות המשתמש וה坦אמה אישית המערכת פותחה כיישום אינטרנט הנגיש מכל דפדפן, על מנת לאפשר זמינות מלאה ונוחות מרבית למשתמשים ללא צורך בהתקנה מקומית. השימוש במערכת מדומה "דיאלוג" בין הסטודנט/המרצה לבין המערכת: לדוגמה, סטודנט יכול להגיד אילוצים אישים כגון עבודה ביום מסוימים (למשל ימים א'-ג' משעה 16:00) או להגיד מגבלות שעות ליום יומיות (למשל לא יותר מ-8 שעות ברצף). בהתאם לקליטים אלו, המערכת תחשב עבורו מערכת שעות מותאמת אישית, תוך שמירה על מגבלות כלליות כמו זמינות מרצים, תפוזת כיתות ומועד קורסי חובה.

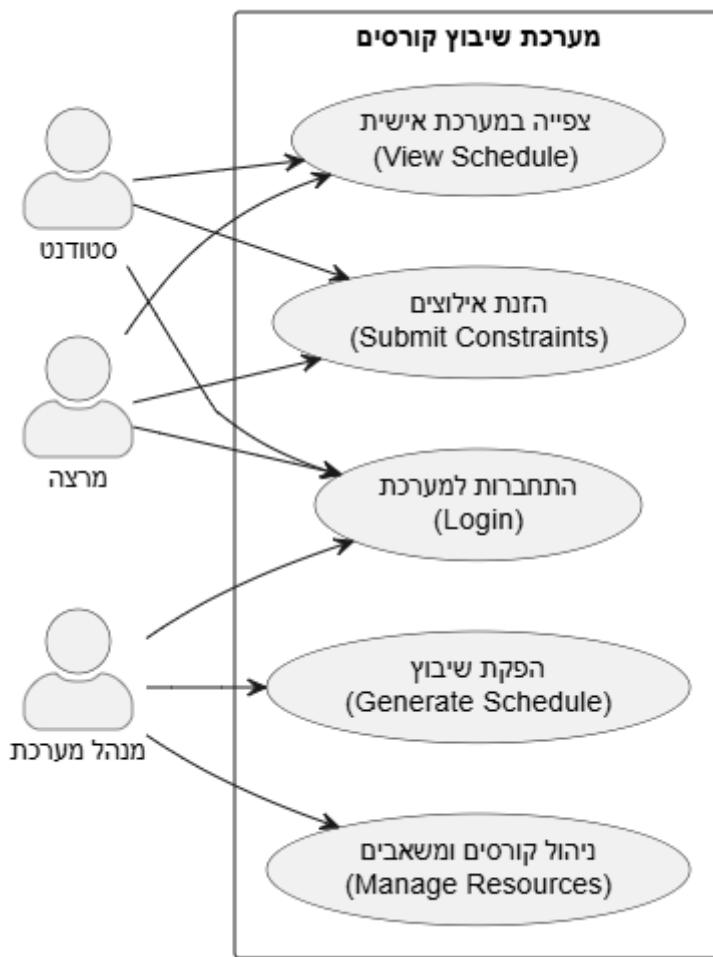
להלן תיאור השחקנים במערכת (Actors), המערכת משרתת שלושה סוגים משתמשים עיקריים, כאשר לכל אחד משק "יעודי המותאם לצרכיו":

**מנהל המערכת (Admin):** המשתמש בעל הרשות הגבוהות ביותר. המנהל אחראי על הזרת נתונים התשתיית (רשימת הקורסים, כיתות לימוד, שיוך מרצים לקורסים). תפקידו המרכזית הוא להגיד את פרמטרים של האלגוריתם, להפעיל את התהילה השיבוץ האוטומטי, ולבצע בקרת איכות על התוצאה הסופית לפני פרסום.

מרצה (Lecturer): משתמש המזמין את אילוצי ההוראה שלו. המרצה נכנס למערכת כדי להגדיר ימים ושעות בהם הוא אינו זמין להוראה, וכן להגדיר העדפות (כגון הימנעות משעות רצופות רבות מדי). לאחר השיבוץ, המרצה מקבל את מערכת השעות האישית שלו.

סטודנט (Student): משתמש הקצה העיקרי שננהה מהगמישות של המערכת. הסטודנט מתחבר כד' לבחור קורס או להזין אילוצים אישיים (עובדת, מגורים רחוקים וכו'). המערכת מציגה לו את האפשרויות הרלוונטיות ביותר עבורו ומאפשרת לו לצפות בלוח הזמנים הסופי ובציוןים.

להלן תרשים Use-Case המתאר את הליך הפרויקט:

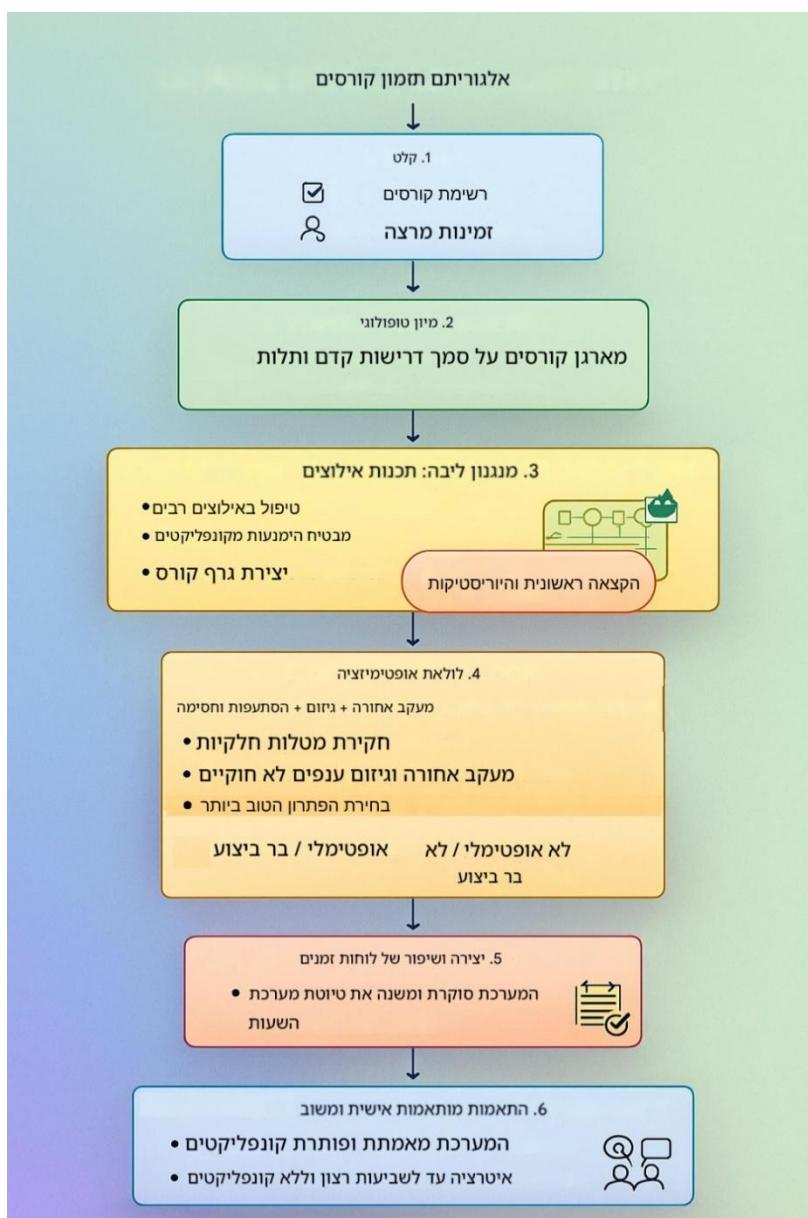


פירוט מקרי שימוש עיקריים (Use Cases) להלן תיאור הפעולות המרכזיות שהמשתמשים מבצעים במערכת:

- התחברות למערכת (Log):** כל משתמש (סטודנט, מרצה או מנהל) נדרש להזדהות באמצעות שם משתמש וסיסמה כניסה למערכת. המערכת מאמתת את הפרטים מול מסד הנתונים, ובמידה והם תקינים, היא טוענת את המשתמש המתאים לפי רמת הרשות של המשתמש (למשל, מנהל יראה אפשרות ניהול סטודנט לא יראה).

- הזנת אילוצים (Submit Constraints):** הסטודנט או המרצה ממלאים טופס מקוון ובו הם מציננים ימים ושעות חסומים. המערכת בודקת את תקינות הקלט ושומרת את האילוץ בבסיס הנתוניים.
- ניהול קורסים ומשאבים (Manage Resources):** מנהל המערכת מוסיף, עורך או מוחק קורסים מהמאגר, ומגדר לכל קורס את דרישות הקדם שלו ואת המרצה האחראי.
- פקת שיבוץ (Generate Schedule):** פועלה המבוצעת על ידי המנהל. בלחיצת כפתור, המערכת מפעילה את מנגנון השיבוץ (האלגוריתם), אשר מעבד את כל הנתוניים והאילוצים ומפיק טויטה של מערכת שעות.
- צפיה במערכת אישית (View Schedule):** כל משתמש (סטודנט/מרצה) יכול לצפות בלוח הזמנים השבועי שלו, המציג ויזואלית בטבלה ברורה, ולראות את המיקום והשעה של כל הרצאה.

להלן תרשيم המתאר את "זרימת העבודה של האלגוריתם":



תרשים זה מתאר את זרימת העבודה של אלגוריתם השיבוץ, החל משלב קליטת הנתונים הגלמים (קורסים וזמן מרצה) וארגון הלוגי באמצעות מין טופולוגי. בהמשך, המערכת מפעילה מנוע אילוצים המשלב לולאת אופטימיזציה וגיזום (Pruning) כדי לסנן פתרונות לא חוקיים ולמצוא את השיבוץ המקורי. התהיליך מסתיים ביצירת לוח זמנים סופי וביצוע התאמות אישיות למניעת קונפליקטים, עד לקבלת תוצר מוגמר ותקין.

## 4. הגדרת הבעיה האלגוריתמית

הבעיה המרכזית שהמערכת נועדה לפתור היא **בעיית שיבוץ עם אופטימיזציה רב-מדנית** של קורסים במועד אקדמי. את בעיית השיבוץ אני מתאר בצורה של גרפף מכון ללא מעגלים (גמ"ל) מדובר בבעיה קומבינטורית מורכבת שבה יש לשבץ קבוצה גדולה של משבבים (מורים, אילוצים וסטודנטים) על פני ציר זמן מוגדר, תחת מערכת ענפה של אילוצים. הקשיי המרכזי אינם מסתכם רק במצבת שיבוץ "חוקי" (כחז שמנוע התנגשויות פיזיות), אלא במצבת שיבוץ אופטימלי הממקסם את פונקציית המטריה - שביעות הרצון של כלל הגורמים במערכת.

הailוצים בבעיה זו מתחולקים לשניים:

**AILTZIM "KSHICHIM"** : אילוצים חשובות לקיום כדי שהמערכת תהיה תקפה, כגון מניעת מצב שבו מרצה משובץ לשני שיעורים בו-זמנית, מניעת חפיפה בין קורסי חובה באותו מסטר.

**AILTZIM "RECHIM"** : העדפות אישיות ואופטימיזציה, כגון ניסיון להיענות לבקשת סטודנטים לימי חופש מסויימים, צמצום "חלונות" במערכת, והתחשבות במגבילות שעות עבודה של מרצים.

מבחינה חשובה, מדובר בעיית NP-Hard. המשמעות היא שמרחב הפתרונות האפשריים גדול באופן מעריכי (אקספוננציאלי) ככל שמספר הקורסים והailוצים עולה, ובלתי אפשרי לסרוק את כל האפשרויות בזמן סביר כדי למצוא את הפתרון המושלם. לכן, האתגר האלגוריתמי דורש שימוש בפתרון חכם (כגון אלגוריתם יורייסטי) המציג את מרחב החיפוש וממקד את המערכת רק באפשרויות בעלות פוטנציאל גבוהה להצלחה. מטרת האלגוריתם היא למצאו "נקודות שיוי משקל" אופטימלית בין כל האילוצים המתנגדים, כך שיינתן פתרון יעיל ומאוזן שלא יפגע ביציבות המערכת הכללית.

בשביל לפתור את בעיה זו בחרתי בשני שלבים שיפtroו לי את הבעיה בצורה היילה ביותר:  
**מין טופולוגי (Topological Sort)**: השלב הראשון בפתרון הוא יצירת סדר תקין של קורסים בהתאם לתנאי קדם. באמצעות מין טופולוגי מתקבל סדר היררכי הגיוני של הקורסים - כך לדוגמה "מבוא לתכנות" מופיע לפני "אלגוריתמים". השלב הזה מבטיח שככל רץ' למידה עומד בכללי והדרישות האקדמיות

**Pruning / Branch-and-Bound עם Backtracking** : לאחר יצירת הסדר הבסיסי, המערכת מפעילה מנגן חיפוש רקורסיבי שמציב קורסים בלוח הזמן.

האלגוריתם מבצע בדיקות בזמן אמיתי:

זמן מ\_rfץ, עמידה בתנאי קדם, מגבלות עומס ועוד. בכל מצב שבו אפשרות לא עומדת באילוצים, היא נגזרת מיד (Pruning), וכך נחסך זמן חישוב משמעותי. במידה הצורך משולבת גם שיטת Branch-and-Bound, המאפשרת לטעוף פתרונות ולהתקדם במסלול החיפוש הטוב ביותר.

השלבים הבאים מתארים את תהליך העבודה המרכזי לפתור בעיית השיבוץ האקדמי במערכת, משלב איסוף האילוצים ועד ליצירת מערכת שעות מלאה ומותאמת אישית לכל משתמש. מהלך זה יוצר תשתיית חישובית מאורגנת, הנשענת על עקרונות הנדסה, תכנון מערכות ויישום אלגוריתמים מתקדמים, אשר יחד מאפשרים פתרון יעיל ומדויק לבעה מורכבת זו.

שלב 1: הגדרת הבעיה וניתוח הדרישות - בשלב זה מתבצעת הבנה عمוקה של מכלול האילוצים המערכתתיים: אילוצי מרצים (זמן, מגבלות הוראה), אילוצי סטודנטים, בניית קורסים וחסוי תלוות אקדמיים (דרישות קדם, מעבדות, תרגולים), וכן מגבלות מערכתיות כגון שעות לימוד וכו' ...

המטרה היא לתרגם מציאות אקדמית מורכבת לסת של אילוצים לוגיים ברורים, שמספקים בסיס לפתרון חישובי עקבי.

שלב 2: תרגום הבעיה למודל מבוסס גרף - לאחר איסוף הנתונים, הבעיה ממופна למודל גרפי המסדר את יחס התלות בין הקורסים. כל קורס מיוצג כקודקוד בגרף, וקשורת מייצגות יחס קדם. בנייה זו מאפשרת לזהות בקלות מבנים היררכיים, למנוע יצירת מעגלים אינסופיים, ולבסס את תהליך השיבוץ על עקרונות מסודרים.

שלב 3: הפחת סדר לוגי באמצעות מיזן טופולוגי - המערכת מפעילה מיזן טופולוגי על הגרף כדי לקבל סדר עקבי שבו ניתן לשבע את הקורסים. בשלב זה מודדא היעדר סתיות בין תנאי הקדם, ומספק מסגרת התקדמות ברורה לתהליכי השיבוץ. התוצאה היא סדר היררכי של קורסים שעליו מבוססים שלבי החיפוש הבאים - וכך האלגוריתם מתחילה למצוא תקופה ויציבה.

שלב 4: בניית מודל חיפוש וגיבוש גבולות הפתרון - בהתבסס על הסדר הטופולוגי, נבנה מודל חיפוש המגדיר את מרחב האפשרויות: לאיזה סטטוס ניתן לשבע כל קורס, כיצד מוצלב המידע עם זמינות מרצים, ומהם התנאים שמדוברים פתרון תקין. בשלב זה הבעיה מנוסגרת כך שניתן יהיה להפעיל עליה Backtracking בצורה יעילה ו邏輯ית.

שלב 5: חיפוש פתרון באמצעות Backtracking מושכל - זהו השלב שבו מתבצע החיפוש בפועל. האלגוריתם מנסה לשבע קורסים לפי המיזן הטופולוגי, ובכל צומת בוחן את האפשרויות התקפות.

אם נתקל באילוץ שאינו מאפשר התקדמות (כגון התנgesות בזמןניים, מרצה לא זמן, או חriegת מבנה אקדמי), האלגוריתם חוזר לאחר (Backtracking) ובודק חלופה אחרת. תהליך זה מאפשר חיפוש מודיק בטור מרחב פתרונות גדול, תוך שמירה על עמידה מלאה בחוקים שהוגדרו.

שלב 6: צמצום המרחב על ידי Pruning - במקביל לחיפוש, המערכת מפעילה מנגנון Pruning שמטרתם לעזר מסלולי חיפוש שלא יכולים להוביל לפתרון נכון. חיתוך זה מבוסס על זיהוי מוקדם של סתיות: אילוצי זמן שאינם ניתנים ליישוב, קורסים שאינן אפשריים במקומות מסוימים, או רצפים שמספרים את היררכיה הטופולוגית. כן נחסר זמן חישוב יקר ונמנעת בחינה של מסלולים לא ריאליים מראש.

שלב 7: הפקת שיבוץ כללי ובקרה איות - לאחר תהליך החיפוש, מופקلوح זמינים אקדמי מלא הכלול את כל הקורסים, הסטודנטים, המרצים, השעות והחדרים. בשלב זה מתבצעת בקרת תקינות פנימית: בדיקת תקיפות תנאי קדם, התאמת זמינות מרצים, מניעת התנגשויות, ושמירה על מבנה אקדמי זהה למידיות המודד. מנהל המערכת יכול לעיין בתוצאות, לבצע התאמות ידניות ולהגדיר גרסאות חלופיות במידת הצורך.

שלב 8: ייצרת מערכות אישיות לסטודנטים - המערכת מאפשרת ייצירת מערכת שעות מותאמת אישית לכל סטודנט, לפי אילוציו והעדפותיו. בשלב זה מופעל תהליכי התאמה חלקית המבוסס על תוצאות השיבוץ הכללי, תוך בדיקות עיקריות נוספת. כך מתבלטת מערכת שעות אישית נטולת התנגשויות, המשמרת את ההיגיון האקדמי ואת מגבלות הלימוד שהוגדרו.

חשוב לציין שהשילוב בין מיין טופולוגי לבין Backtracking עם Pruning נבחר כיוון שהוא מספק מענה מואזן בין בניית לוגית מוקדמת לבין חיפוש פתרון גמיש ומודיק.

המיון הטופולוגי מספק שלד היררכי קשיח המונע מראש סתיות ומאפשר התקדמות בטוחה.  
ה- Backtracking מאפשר גמישות מלאה בהתאם הפתרון לצרכים מורכבים ומשתנים.  
ה- Pruning מצמצם את מרחב החיפוש ומונע בזבוז זמן על מסלולים לא אפשריים.

יחד, תהליך זה יוצר מערכת שיבוץ אקדמית יציבה, עקבית, יעילה ומתאמת לכל המשתמשים, גם בסביבה עתירת אילוצים ונתונים.

## פואדי-קוד:

### מיון טופולוגי - Topological Sort

Pseudocode – Topological Sort (using DFS)

Algorithm TopologicalSort(Graph G):

Input:  $G = (V, E)$  where  $V$  = set of courses,  $E$  = prerequisite relations

Output: Ordered list of courses (topological order)

```

visited ← empty set
stack ← empty stack
For each node v in V:
    If v not in visited:
        DFS(v)
    Return stack in reverse order
Procedure DFS(node):
    Mark node as visited
    For each neighbor in node.adjacent:
        If neighbor not in visited:
            DFS(neighbor)
    Push node onto stack

```

הסבר:

1. עובר על כל הקורסים בגרף אחד-אחד.
2. בודק אם כבר ביקר בקורס. אם לא - מתחילה עליו חיפוש عمוק (DFS).
3. בתוך DFS הוא מסמן את הקורס כ"גובל" / ביקרתי בו.
4. עובר על כל הקורסים שתלויים בו (כל קורס שmagui אחרי בתור).
5. אם יש קורס תלוי שעדיין לא ביקרו בו הוא נכנס אליו רקורסיבית וממשיר להעמיק.
6. רק אחרי ששסיהם לעבור על כל הקורסים שתלויים בו, הוא מוסיף את הקורס הנוכחי למחסנית.
7. בסוף, אחרי של ה-DFS מסתיים, הופכים את המחסנית, וזה נותן את סדר הקורסים החוקי (קודם תנאי הקדם, ואז הקורסים שתלויים בהם).

### :Backtracking and Pruning

Pseudocode – Course Scheduling with Backtracking and Pruning

Algorithm ScheduleCourses(courses, constraints):

```

bestSolution ← null
bestScore ← -∞
currentSchedule ← empty schedule

```

Sort courses by priority or number of constraints

Backtrack(0)

Procedure Backtrack(index):

If index == number of courses:

    score ← Evaluate(currentSchedule)

    If score > bestScore:

        bestScore ← score

        bestSolution ← copy(currentSchedule)

Return

course ← courses[index]

For each possibleSlot in AvailableSlots(course):

    If IsValid(course, possibleSlot, constraints):

        Assign(course, possibleSlot)

        Backtrack(index + 1)

        Unassign(course)

Else:

    Continue // Pruning – skip invalid option

הסבר:

הקדמה ל - Backtracking

1. מכין מבנה נתונים ריק שבו יבנה השיבוץ הנוכחי.
2. ממיין את הקורסים לפי עדיפויות (למשל: אילוצים, רמת חשיבות, כמות הגבלות).
3. מתחילה תהליך רקורסיבי של שיבוץ - קורס אחרי קורס.

מה קורה בתוך Backtracking :

1. בודק אם סימנו לשבץ את כל הקורסים:

○ אם כן - מחשב ציון לשיבוץ.

○ אם הציון טוב יותר מהפתרון הנוכחי טוב שנמצא עד עכשוו הוא שומר אותו כפתרון הטוב

ביתר.

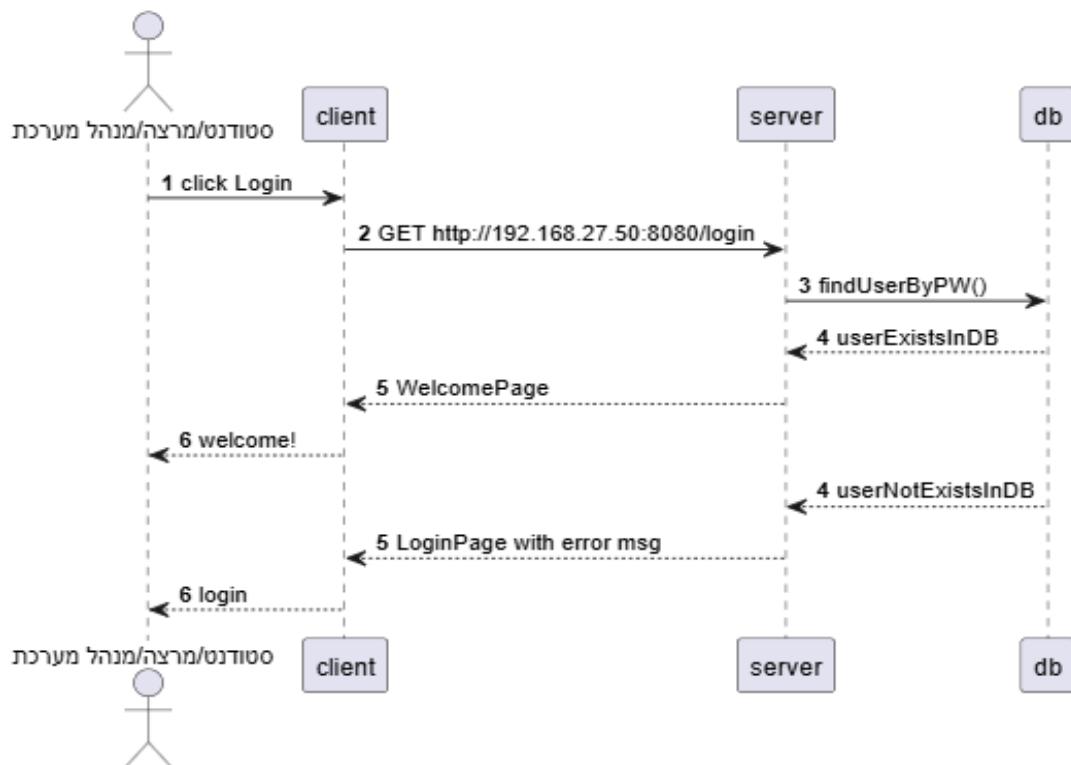
2. לוחץ את הקורס הבא בתור לפי האינדקס.
3. עובר על כל המשבצות האפשרות שהקורס יכול להיכנס אליו (ימים, שעות, וכו').
4. לכל משבצת בודק אם אפשר לשbez את הקורס שם לפי אילוצים:
  - אם המשבצת חוקית אז הוא משbez את הקורס.
  - נכנס רקורסיבית לקורס הבא.
  - כחווזרים מהרקורסיה - מסיר את השיבוץ כדי לנסוט אפשרות אחרת.
5. אם המשבצת לא חוקית (התנששות, אי זמינות, כפילות וכו) מדלג עליה מיד (Pruning) בלי נסוטות לחפש פתרון אחרת.

## 5. הליכים עיקריים בפתרון בעיה בטכנולוגיות הנדסה מתקדמות

**תרשיimi Sequence:**

להלן תרשימי Sequence שירחיבו מעט על הפעולות המרכזיות שהפרויקט עושה. פעולות הפרויקט כתובות בתרשימי Sequence ה - **Use-cases** שמוצג בסעיף 3.

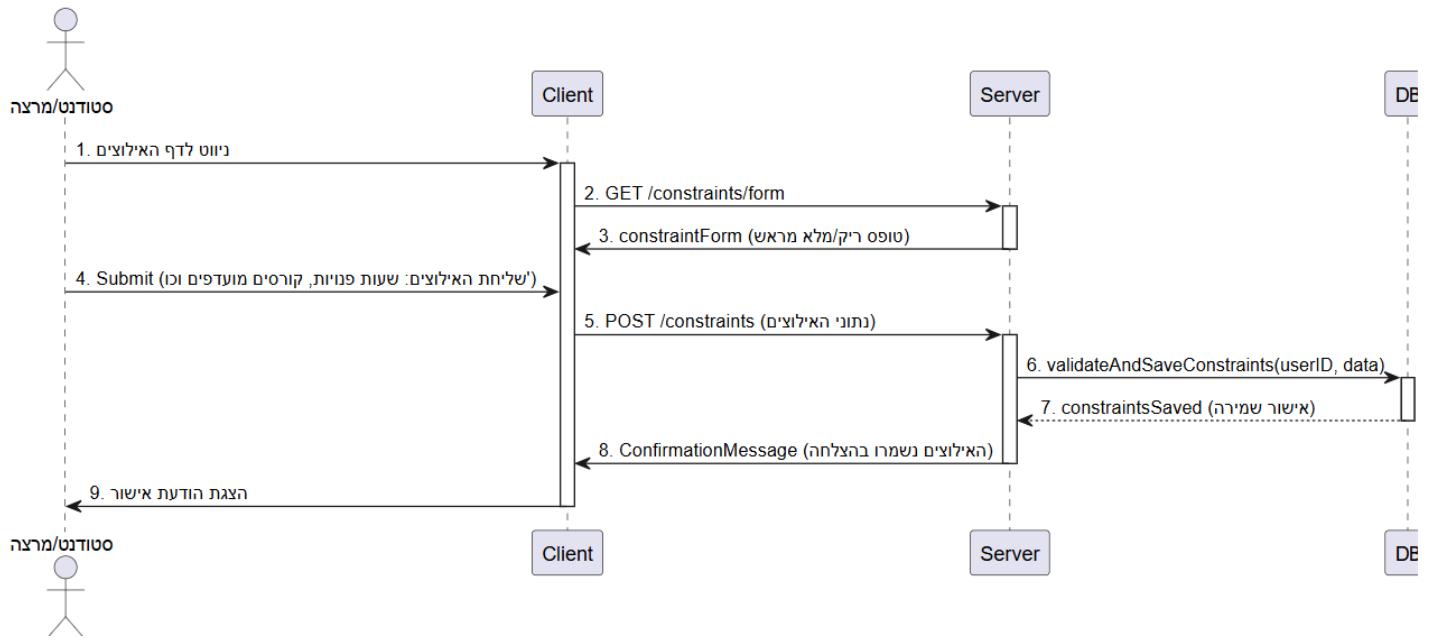
**תרשים ראשון - התחברות למערכת (Login):**



בתרשים זה אפשר לראות את הגדרת הכניסה למערכת (Login) המערב את כל המשתמשים הקיימים במערכת. בתרשימים זה אפשר לראות שני תרחישים מרכזיים שהם הצלחה ומעבר לעמוד (לאחר אימות), וכישלון שמקפץ הودעה בהתאם.

### תרשים שני - הזנת אילוצים (Submit Constraints)

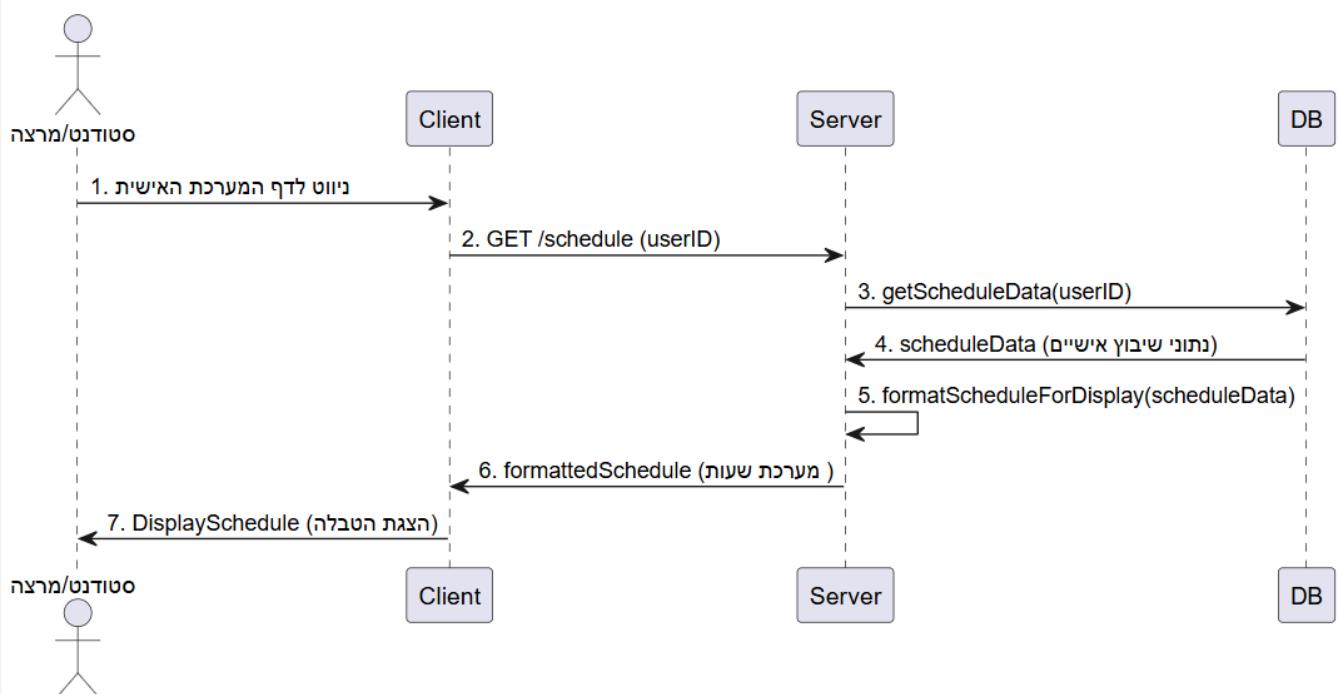
Sequence diagram (הזנת אילוצים - Submit Constraints)



בתרשים אפשר לראות שאינטראקציה להזנת אילוצים מוגדרת עבור משתמשים שהם סטודנט/מרצה. התהליך כולל בקשה לטופס האילוצים, מילוי ולאחר מכן שליחת הנתונים מהלוקח אל השרת, מבצע אימות ושמירה. התוצאה הסופית היא קבלת הודעת אישור על שמירה מוצלחת.

### תרשים שלישי - צפייה במערכת אישית (View Schedule)

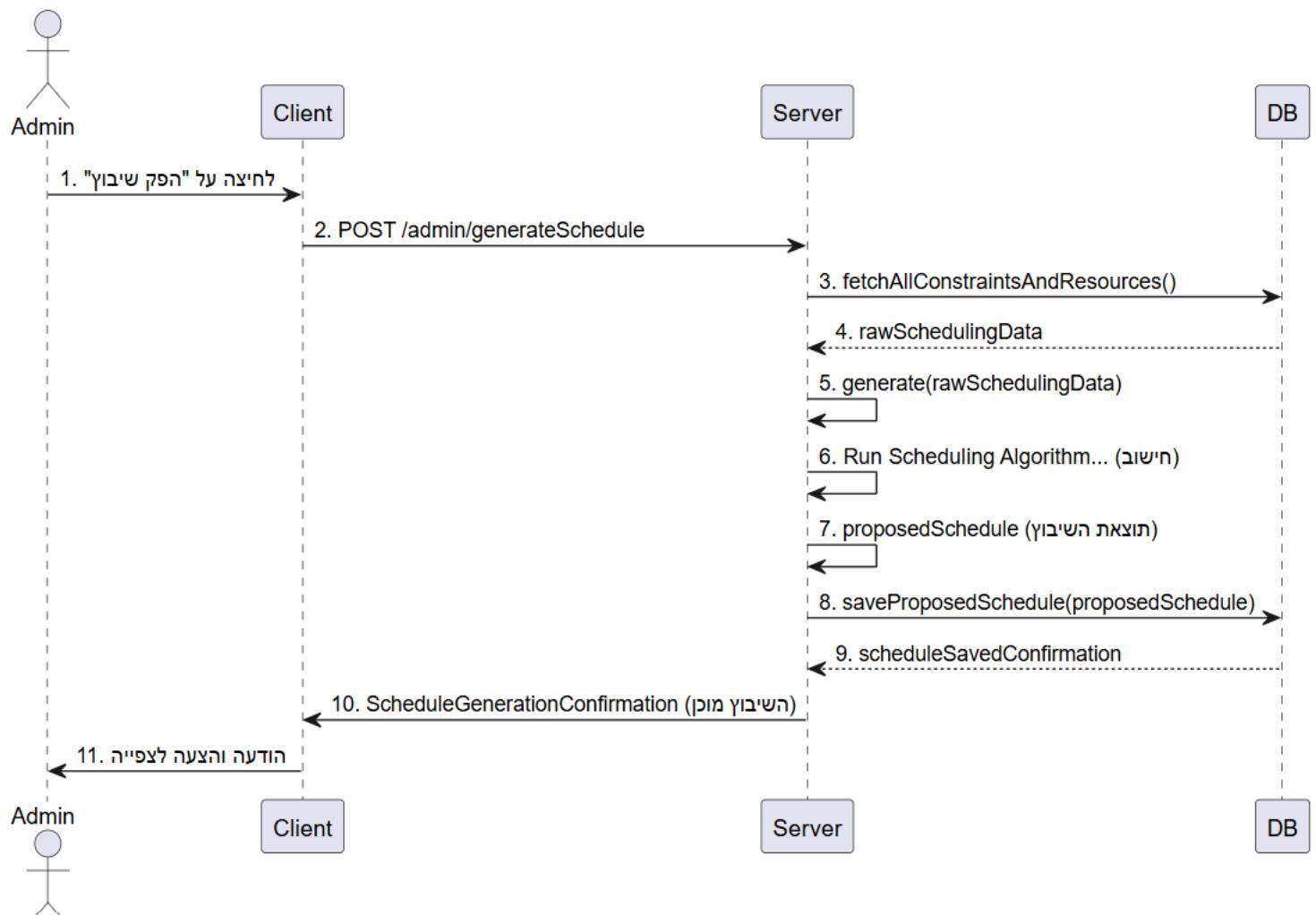
Sequence diagram (צפייה במערכת אישית - View Schedule)



בתרשים אפשר לראות שתהליך הצגת השעות האישית מוגדר עבור סטודנטים ומרצים, ומתאר את השלבים להחזרה והצגה של נתונים מותאמים אישית.

### תרשים רביעי - הפקת שיזבז (Generate Schedule)

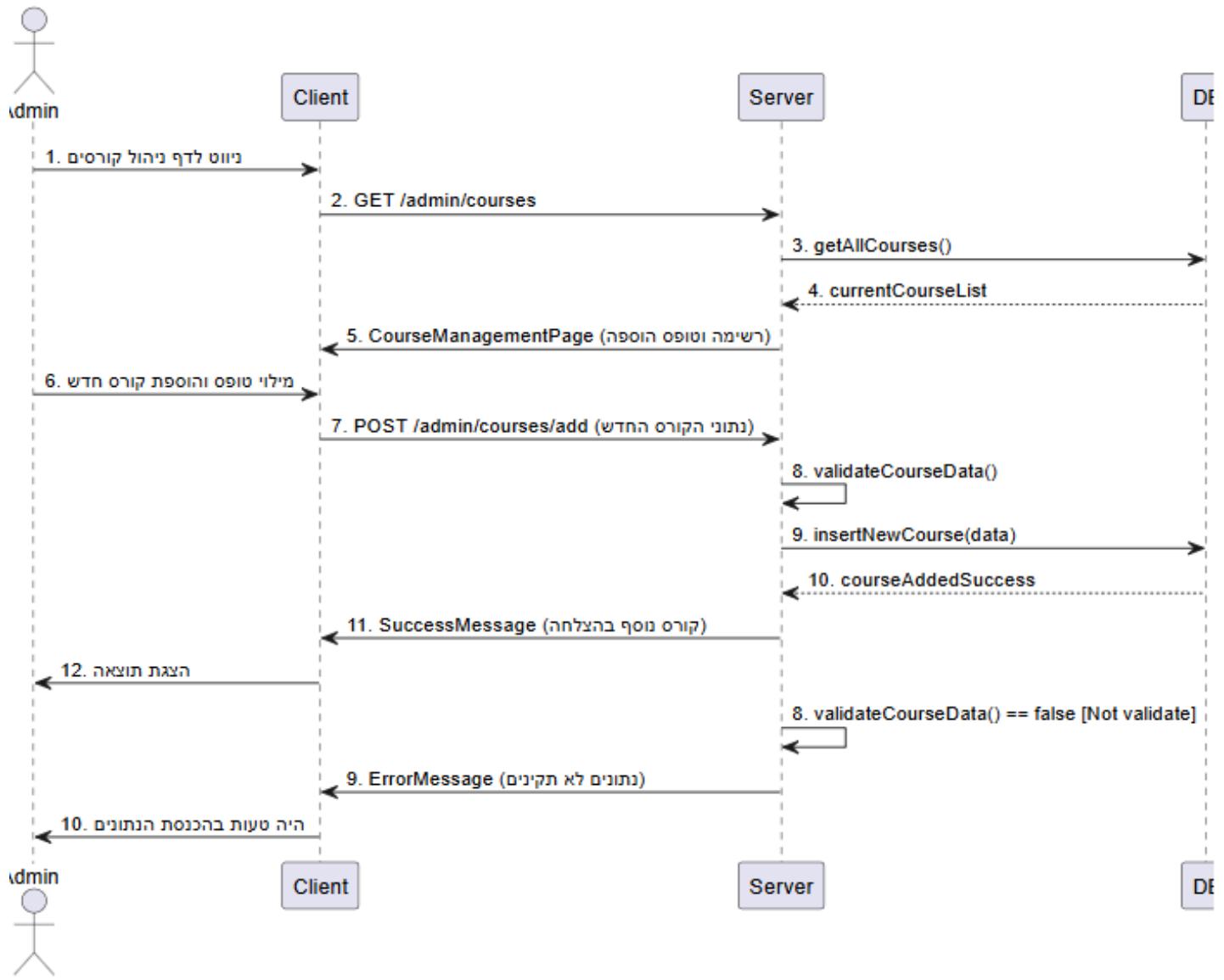
Sequence diagram (הפקת שיזבז - Generate Schedule)



בתרשים אפשר לראות שתהליך הפקת השיזבז מבוצע בלחיצת כפתור על ידי מנהל המערכת (Admin), ומציג באופן ישיר את כל החישובים שמתבצעים בתוך השרת. לאחר שהשרת אוסף את כל הנתונים הנדרשים (שלבים 3-4) הוא מבצע בעצמו את אלגוריתם השיזבז המורכב (שלבים 5-7), ולאחר קבלת תוצאת השיזבז המומלצת, הוא שומר אותה בסיס נתונים. לבסוף, המערכת מודיעה למנהל על השלמת התהליך וזמןות השיזבז לצפייה.

### תרשים חמישי - ניהול קורסים (Manage Resources):

ניהול קורסים - Sequence diagram



בתרשים אפשר לראות שההילך ניהול הקורסים מוגדר לפעולות מנהל המערכת (Admin), הוא מציג את השלבים להוספת קורס חדש למערכת. התהליך כולל בקשת רשימת קורסים הקיימים מה-DB, הצגת טופס הוספה למנהל ובעקבותיו שליחת נתוני הקורס החדש לשרת. לאחר קבלת הנתונים, השרת מבצע אימוט למסד הנתונים ומחזיר למנהל הודעה הצלחה או הודעה שגיאה בהתאם לתוצאות העימות.

**ארQUITקטורה:**

המערכת מבוססת על מודל **שרות-לקוח**, שהוא מודל עבודה נפוץ בפיתוח מערכות מחשوب מודרניות. במודל זה קיימת הפרדה ברורה בין שני גורמים עיקריים: הלוקח (Client) והשרות (Server). הלוקח הוא הרכיב דרכו המשתמש מתקשר עם המערכת, בדרך כלל באמצעות דפדפן או אפליקציה, ואחראי על הצגת המידע וקבלת קלט מהמשתמש. השירות, לעומת זאת, אחראי על ביצוע עיקר העבודה הלוגית של המערכת, עיבוד הנתונים, ניהול חוקי העסק והתקשורת עם מסד הנתונים.

התקשורת בין הלוקח לשירות מתבצעת באמצעות בקשות (Requests) ותשובות (Responses) לרוב בפרוטוקול HTTP. הלוקח שולח בקשה לשירות, השירות מעבד אותה בהתאם לוגיקה ומחזיר תשובה מתאימה. מודל זה מאפשר חלוקת עומסם, אבטחת מידע טובה יותר, תחזוקה נוחה ורחבת המערכת בעתיד ללא פגיעה בחוויה המשתמש.

כדי להבטיח מערכת יציבה, מודולרית וקלת לתחזוקה, המערכת מפותחת על פי **ארQUITקטורת שלוש השכבות**, אשר מפרידה בין רכיבי המערכת העיקריים בהתאם לאחריותם. הפרדה זו מאפשרת פיתוח מסודר, תחזוקה קלה, בדיקות ייעילות וסדרוגים עתידיים ללא תלות בין הרכיבים.

**1. שכבת התצוגה - frontend:** שכבת משק המשמשת אחראית על האופן שבו המשתמש רואה את המערכת ומתקשר אליה. זהה שכבה המוצגת ישירות למשתמש, והוא כוללת מסכים, טפסים, כפתורים, טבלאות, גרפים והודעות מערכת. תפקיד המרכזית הוא לספק חוותית שימוש נוחה, ברורה ואינטואיטיבית, תוך הצגת מידע בצורה ויזואלית מובנת.

השכבה מנהלת את כל אינטראקציות הקלט והפלט: קליטת נתונים מהמשתמש, הצגת תוצאות ותשובות מהמערכת, וביצוע בדיקות בסיסיות כגון אימויות קלט (לדוגמה: שדות חובה או פורמט תקין). עם זאת, שכבה זו אינה מבצעת לוגיקה מורכבת ואנייה ניגשת שירותים למסד הנתונים. היא מתקשרת עם שכבת הלוגיקה באמצעות קרייאות לשירותי השירות, וכן שומרת על הפרדה ברורה בין המשק לפונקציונליות הפנימית של המערכת.

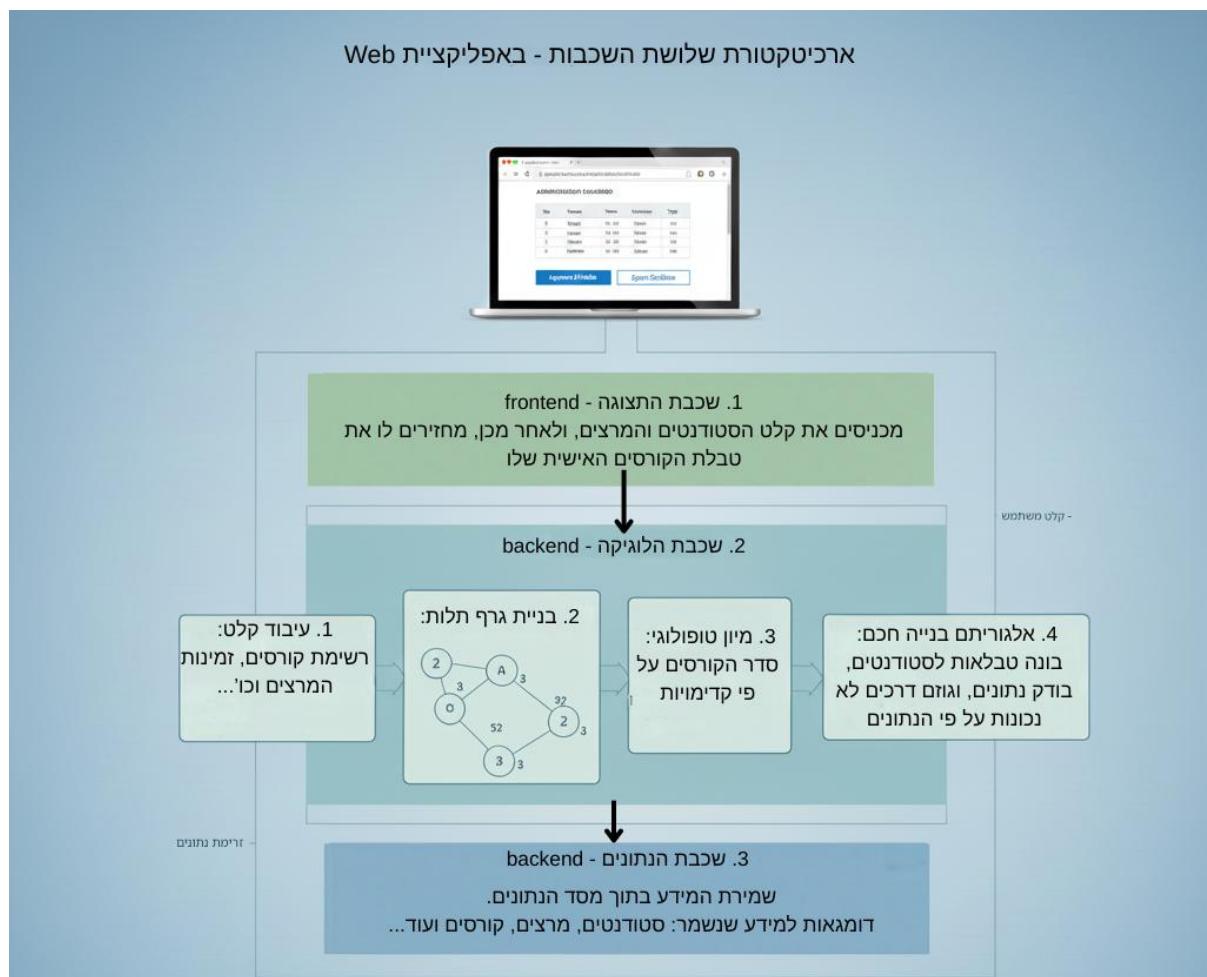
**2. שכבת הלוגיקה - backend:** שכבת הלוגיקה מהווה את לבת המערכת וממומשת בצד השירות. שכבה זו אחראית על עיבוד הנתונים המתקבלים מהлокוח, יישום חוקי העסק, קבלת החלטות והפעלת התהליכי המרכזים של המערכת. היא מגדרה כיצד המערכת אמורה להתנהג במצבים שונים, בהתאם לדרישות העסקיות.

שכבה זו משמשת כמתווך בין שכבת משק המשמשת לבין שכבת הנתונים. היא בודקת את תקיפות הנתונים, מבצעת חישובים, מנהלת זרימות עבודה ומחליטה אילו פעולות יש לבצע על מסד הנתונים. בנוסף, שכבת הלוגיקה העסיקית מטפלת בנושאים כגון הרשאות משתמשים, אבטחת מידע, טיפול בשגיאות וניהול תהליכי מורכבים. הפרדה זו מאפשרת שינוי של הלוגיקה העסיקית מבלי להשפיע ישירות על המשק או על מבנה הנתונים.

**3. שכבת הנתונים - backend:** שכבת הנתונים אחראית על ניהול ו אחסון המידע של המערכת. תפקידה הוא לבצע פעולות של שיליפה, הוספה, עדכון ומחיקה של נתונים (CRUD) מול מודל נתונים, תוך שימוש, עקביות ובטחת המידע. שכבה זו אינה מכילה לוגיקה עסקית, אלא מתמקדת בגישה ייעילה ומואבטחת נתונים.

השכבה מספקת משקל אחיד לגישה לנתונים עבור שכבת הלוגיקה העסקית, ומסתירה את פרטי המימוש של מודל הנתונים, כגון סוג המודד, מבנה האוסףים או שאילתות ספציפיות. גישה זו מאפשרת להחיליף או לשדרג את מודל הנתונים בעתיד עם מינימום שינויים בשאר חלקי המערכת, ותומכת בעבודה רציפה ובזמן אמת של המערכת כולה.

להלן תרשים המתאר את ארכיטקטורת שלושת השכבות:



בתרשים אפשר לראות שארQUITקטורת שלוש השכבות (Three-Tier Architecture) מוממשת באפליקציה Web, והיא מפidea בבחירה בין האחריות והתפקיד של כל שכבה במערכת.

השכבה הראשונה היא שכבת התצוגה, שכבה זו מייצגת תחת הספרה 1 בתרשים והיא ממוקמת המשמש. היא מייצגת על ידי ממשק ה - Web בדף, והוא מטפלת בקלט המשתמש (כגון הכנסת אילוצים) ומציגת את הנתונים, כמו טבלת הקורסים האישית.

השכבה השנייה היא שכבת הלוגיקה היא מייצגת תחת הספרה 2 בתרשים. היא ממוקמת ב - backend, והוא לב המערכת. שכבה זו אחראית על עיבוד נתונים הקלט וכוללת לוגיקה מורכבת כמו בניית גרפף תלויות, מון לפי קידימות ויישום אלגוריתם חכם לשיבוץ כדי לייצר את מערכת השעות הסופית.

השכבה השלישית והאחרונה שモפיעה תחת הספרה 3 היא שכבת הנתונים ממוקמת בתחתית, והוא אחראית אך ורק על אחסון ושליפת הנתונים (כגון נתונים סטודנטים, מרצים, קורסים) עבור שכבת הלוגיקה העסקית.

התרשימים מדגים כיצד המידע זורם בין השכבות, כאשר כל שכבה מטפלת בתחום האחוריות שלה בלבד, מה שבטייח מודולריות וקלות תחזוקה.

## 6. הליכים עיקריים בתחום למידת מכונה - לא רלוונטי

ה策עה זו אינה עוסקת בלמידה מכונה, لكن סעיף זה לא רלוונטי.

## 7. הליכים עיקריים במע' הפעלה/רשות/תקורת/abweטה - לא רלוונטי

ה策עה זו לא עוסקת במערכות הפעלה/רשות/תקורת/abweטה. لكن סעיף זה לא רלוונטי.

הערה חשובה: **התוצאות למודל Client-Server** מופיע בהרחבה בסעיף 5 של ה策עה.

## 8. תיאור פרוטוקולי תקשורת

המערכת מקיימת תקשורת רציפה בין הרכיבים השונים. כדי להבטיח אמינות,יעילות וabweטה, המערכת משתמשת בכמה פרוטוקולי תקשורת סטנדרטים:

**HTTP** - זהו פרוטוקול שכבת היישום המשמש להעברת "היפר-טקסט" ונתונים בין הלקוח לשרת. (HyperText Transfer Protocol) HTTP משמש כבסיס לתקשורת שרת-לקוח. המערכת תשתמש בפרוטוקול זה בשבייל להעביר מידע, משכבה התצוגה אל שכבת הלוגיקה ובנוסף משכבת הלוגיקה אל שכבת הנתונים.

**TCP/IP** הוא צמד הפרוטוקולים הבסיסיים והמרכזיים של רשת האינטרנט. פרוטוקול TCP (Transmission Control Protocol) מבטיח שכל חבילות הנתונים שנשלחות בין המערכות יגיעו בשלמותן ובסדר הנכון לעד, מה שהופך את התקשורת לאמינה.

לעומתו IP (Internet Protocol), אחראי על ניתוב חבילות הנתונים בין רשתות שונות, ובentity שהנתונים יגיעו מהמקור לכתובת היעד הנכונה.

**API REST** המערכת בונה על שהיא גישה סטנדרטית המאפשרת תקשורת עלייה בין רכבי הרשת. כל הנתונים המועברים במערכת, בין השירות ללקוח ודרך נקודות הקצה, מוצגים בפורמט JSON שיוסבר בסעיף הבא.

בנוסף, המערכת תשלב API חיצוני לצורך הוספה ערך נוסף וחווית משתמש משופרת. לדוגמה, באמצעות הטמעת API של תחזית מזג אויר, המערכת תציג המלצות לבוש פרקיות בסיסר הבית של המשתמש בהתאם לתחזית ימי הלימוד.

**JSON** הוא פורמט סטנדרטי להצגת והעברת נתונים, המבוסס על טקסט פשוט וקריא. הפורטט מאפשר לייצג מידע בצורה מסודרת באמצעות זוגות של מפתח וערך, וכן הוא נוח הן לкриאה על ידי אדם והן לעיבוד על ידי תוכנות JSON. הוא נפוץ מאוד בשימוש במערכות תוכנה מודרניות, ובעיקר משמש להעברת נתונים בין שירות לקוחות תקשורת בין יישומים.

## 9. פיתוחים עתידיים

- שילוב מערכת הودעות פנימית בזמן אמת (צ'אט שירות-לקוח).
- הצגת סטטיסטיות מתקדמות על ציוני קורסים וביצוע סטודנטים - הסטטיסטיות יוצגו למנהל המערכת בלבד.
- פיתוח אפליקציית מובייל משלימה עם גישה למערכת האישית של הסטודנט.

## 10. תיאור טכנולוגיות הנדסה

- שפת תכנות:
  - שפת התוכנות המרכזית בה נעשה שימוש היא **Java 21**. זהה שפת תכנות עילית, מונחית עצמים (Object Oriented Programming - OOP), הנחשבת לאחת השפות הנפוצות והיציבות ביותר בעולם. פיתוח התוכנה Java מאפשר בניית מערכות מורכבות, מאובטחות וקלות לתחזקה, ומתאימה במיוחד לפיתוח אפליקציות Full-Stack הכוללות גם הצד השירות (Backend) וגם הצד לקוח (Frontend). השפה תומכת בניהול זיכרון אוטומטי ועובדת בסביבות רבות משתמשים.

- מסגרות עבודה לתשתיות תוכנה (Frameworks) - אשר מהוות תשתיית מוכנה לפיתוח תוכנה. מסגרת עבודה היא אוסף של קוד, כלים וספריות שמספקים שלד ברור לאפליקציה, ובכך חוסכים זמן פיתוח ומונעים כתיבת קוד חוזר. ה - Framework מכתיב דרך עבודה מסודרת, כגון הפרדה לשכבות, חלוקה למודולים, ניהול נכון של בקשות, נתונים ולוגיקה.
- בצד השרת נעשה שימוש ב- **Spring Boot 3.5.8**, מסגרת עבודה לפיתוח מערכות Backend ב - Java. שפת העילית, Java מאפשרת הקמה מהירה של שרת HTTP מביסס Apache Tomcat, יצרת שירותים (Services) להפעלת הלוגיקה, עבודה עם מאגרי נתונים (Repositories) לביצוע פעולות CRUD. וכן בנית REST Controllers המטפלים בבקשתות ותגובה HTTP. היתרונות המרכזיים של Spring Boot הוא הפשטות והאוטומציה שהיא מספקת, לצד גמישות והתאמאה לפרויקטים בקנה מידה גדול.
- בצד הלוקוח נעשה שימוש ב- **Vaadin 24.9.5**, מסגרת עבודה המאפשרת פיתוח ממשקי משתמש עשירים שירות ב - Java, ללא צורך בכתיבה ידנית של HTML , JavaScript או CSS.
- Vaadin מספקת מגוון רחב של רכיבי UI מוכנים, כאשר הלוגיקה של הרכיבים רצתה בצד השרת והשינויים מסונכרנים אוטומטית לדף במערכות WebSockets ומנגנון PUSH כך ניתן לפתח ממשק משתמש מתקדם, אחדיד וקל לתחזקה.
- סביבות פיתוח משלבות (IDEs):
  - לצורך כתיבת הקוד וניהול הפרויקט נעשה שימוש בסביבת פיתוח משלבת (IDE) מסוג **VS Code 1.107.1**. זהה סביבת פיתוח מודרנית הכוללת עורק קוד מתקדם, קומפיילר, מנפה שגיאות, השלמות קוד אוטומטיות (IntelliSense) והציגת תיעוד API. בנוסף VS Code מאפשר ניהול פרויקטים באמצעות Maven ותוכנת Class Diagrams - Use Case, Sequence PlantUML ליצירת תרשימי.
- ספריות עזר:
  - ספריות העזר שבפרויקט עדין לא ידועות.
- מנגנים נוספים:
  - מסד הנתונים מנהל באמצעות **MongoDB Atlas**, שירות ענן המאפשר שליטה מלאה על הנתונים דרך הדפדפן. MongoDB Atlas מספק מנגנוני אבטחה מתקדמים, כגון ניהול משתמשים והרשאות באמצעות סיסמאות, והגבלה גישה ברמת הרשות באמצעות רשימות כתובות IP מורשות בלבד, ובכך מבטיח שמירה על שלמות ואבטחת המידע.

## 11. מסד נתונים

לצורך שמירת הנתונים, המערכת משתמשת במסד נתונים בענן מסוג **NoSQL**. בחירה זו נעשתה לאחר בחינת מספר חלופות, כאשר השיקול המרכזי היה גמישות, היכולת להתמודד עם מבנים דינמיים של שאלוני אישיות, לאחסן סוג מידע משתנים, ולעדכן נתונים בזמן אמיתי במשתוק ה- Web.

מסד נתונים בענן הוא מסד נתונים המאוחסן ומנווה על גבי תשתיות מרוחקות של ספק שירות, ולא על שרת מקומי. עבודה בענן מאפשרת זמינות גבוהה, גישה מאובטחת מכל מקום, גיבוי אוטומטי, וה坦אמנה קלה לעומסי שימוש משתנים, ללא צורך בתחזקה פיזית של שרתים. פתרון זה מתאים במיוחד למערכות מודרניות הפועלות דרך האינטרנט ודורשות עבודה רציפה וזמן אמיתי.

NoSQL הוא סוג של מסדי נתונים שאינם מבוססים על מבנה טבליי קשיח כמו במסדי נתונים רצינליים (SQL). במקום זאת, הוא מאפשר אחסון נתונים באוסףים גמישים כגון מסמכים, דבר המאפשר על עבודה עם מידע שאינו אחד או משתנה לארוך זמן. גישה זו מתאימה במיוחד למערכות בהן בניית הנתונים עשוי להיות פתוחה, כמו שאלונים, טפסים דינמיים ונתוני משתמשים מגוונים.

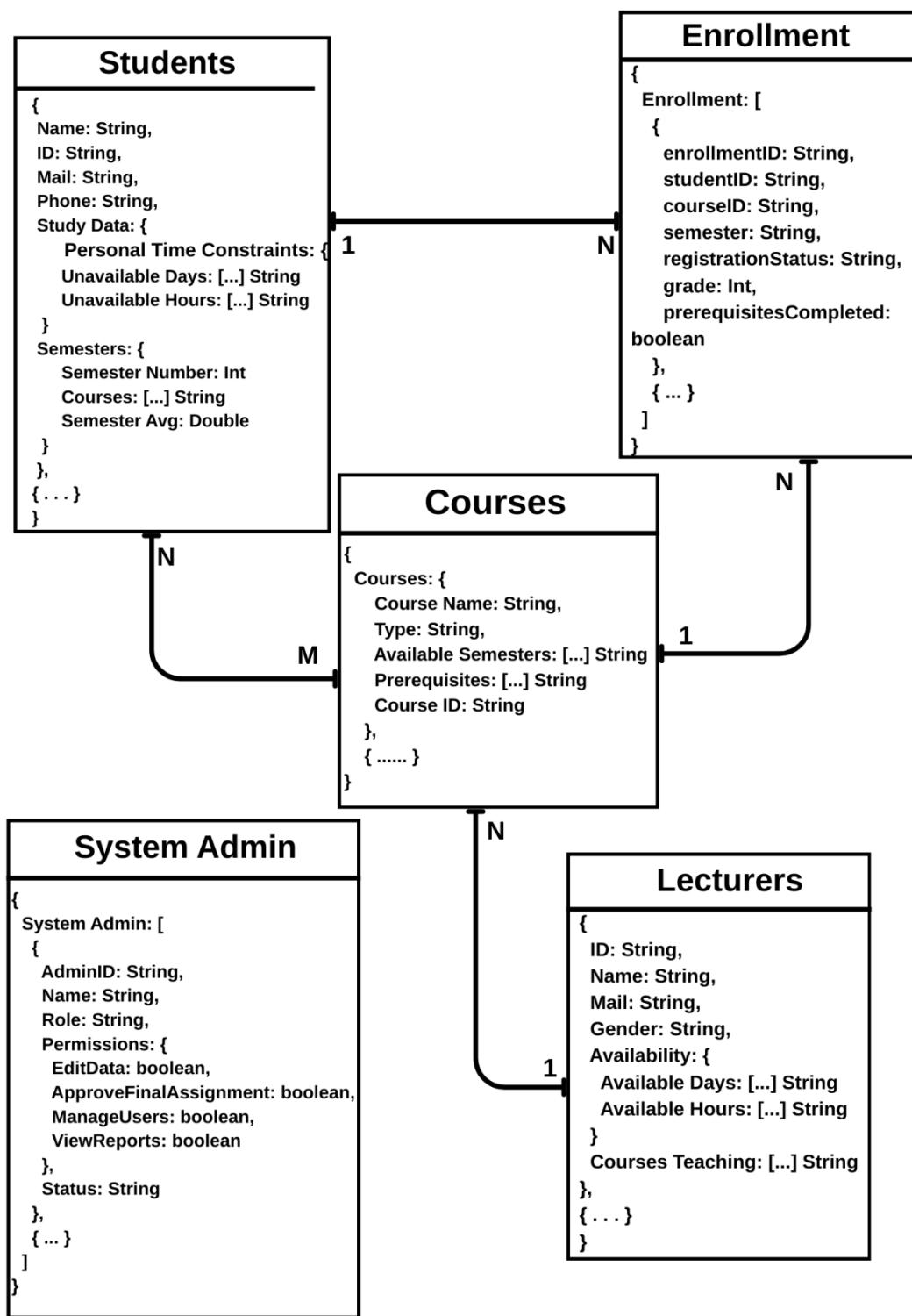
בפרויקט זה נעשה שימוש ב- **MongoDB**, מסד נתונים NoSQL מבוסס איסופים. הוא מאפשר עבודה נוחה עם נתונים מורכבים ודינמיים, ביצועים טובים בעבודה בזמן אמיתי, ואינטגרציה פשוטה עם מערכות Web ושרתים Backend. שילוב זה מספק פתרון יעיל, גמיש ומדרגי לניהול נתונים המערכת.

המידע שמאותחן במסד הנתונים מכיל את האוסףים הבאים:

- **סטודנטים Students** - מכיל את פרטי הסטודנטים, כמו שם, ת"ז, טלפון ומיל. אילוצי שעות אישיים, סמסטרים, קורסים וצינויים. בנוסף, המערכת תשמור ממוצע סמסטריאלי, ונתונים נוספים.
- **קורסים Courses** - כוללת פרטים על כל קורס: שם הקורס, סוג (חובה/בחירה), סמסטרים דודדים אפשריים, ותנאי קדם.
- **מרצים Lecturers** - מכילה מידע על המרצים, זמינותם, הקורסים בהם הם מלמדים והעדפות הוראה. כמובן שגם כאן ישמר מידע על המרצים, כמו שם, מין, ת"ז וכו'...
- **הרשות Enrollment** - קישור בין סטודנטים לקורסים בהם נרשםו, כולל סטטוס הרשמה, צינויים ומידע על השלמת דרישות קדם.
- **מנהל מערכת System admin** - מכילה מידע על משתמשי מערכת בעלי הרשות מיוחדות, כולל מנהל מערכת ואחרים שיכולים לעורוך נתונים או לאשר את השיבוץ הסופי.

השימוש במסד הנתונים זה מאפשר המערכת לסנן את המידע בזמן אמיתי, לבצע אלגוריתמים מורכבים על אילוצים והעדפות, ולשמור על גמישות גם כאשר הנתונים משתנים בתדריות גבוהה.

להלן תרשים ERD המתאר את מסד הנתונים של המערכת:



בתרשים ERD זה אפשר לראות את מבנה מסד הנתונים של המערכת ואת יחסיו הגומلين בין הישויות המרכזיות המרכיבות אותה. התרשים מציג באופן חזותי כיצד מושרים הסטודנטים, המרצים והמנהלים אל הקורסים ואל נתוני ההרשמה, וממחיש את ארגון המידע הנדרש לצורך ביצוע תהליכי השיזוף והניהול התקין.

## 12. פרטימ פורמליים

### לוחות זמנים:

שלבי עבודה	לסיים עד תאריך
בחירה פרויקט, חקירה ולמידה עמוק של נושא הפרויקט	<b>1.12.2025</b>
כתיבה והגשת הצעת הפרויקט לאישור משרד החינוך	<b>11.12.2025</b>
ימוש הקוד של האלגוריתם המרכזי, ביצוע בדיקות ושיפורים	<b>13.1.2026</b>
בנייה צד שרת	<b>3.2.2026</b>
בנייה מסד הנתונים ושלילובו	<b>17.2.2026</b>
בנייה צד לקוחות	<b>3.3.2026</b>
כתיבת ספר הפרויקט	<b>24.3.2026</b>
הגשת הפרויקט כולו (ספר + קוד) להגנה וקבלת ציון מגן	<b>7.4.2026</b>

### חתימת מנהה הפרויקט:


  
 ס. י. כהן  
 מנהל פרויקט

**חתימת הסטודנט:**

**חתימת רכז המגמה:**