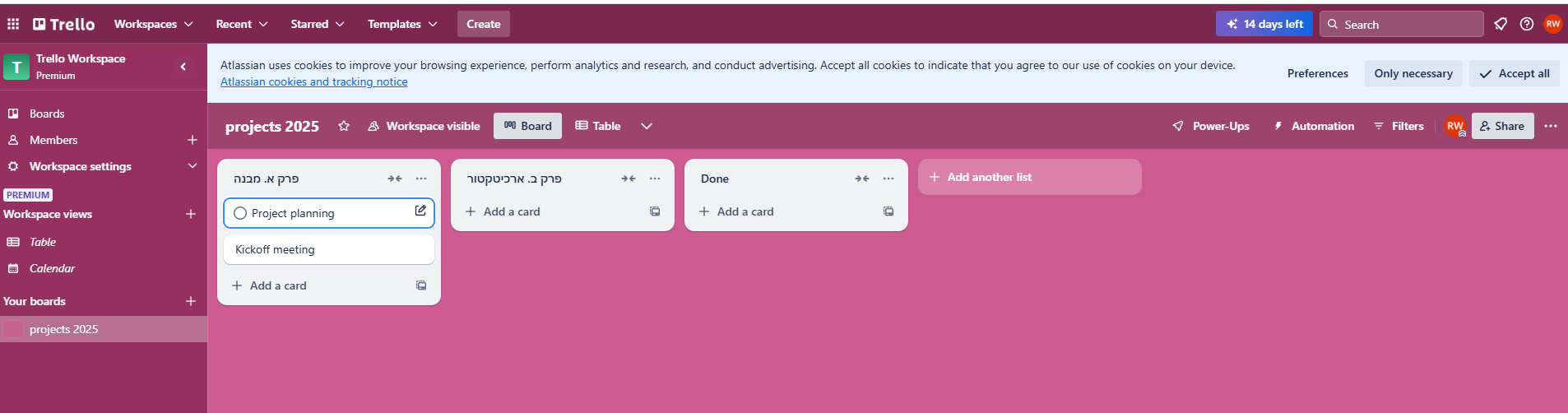
* 1. **הנחיית פרויקטים – מיון טיפים לפי כלים ושיטות עבודה**

**ח-1. כלים טכנולוגיים לפיתוח וניהול פרויקטים**

* **אני מלמד את התלמידים להשתמש בכלים נוספים בזמן פיתוח הפרויקט כמו למשל** GIT **לניהול הפיתוח ומקשר את החשיבות בשימוש ב-** GIT **לחיים האמיתיים, כמו למשל שניתן להסתכל על הקוד שכתבו ואז להתרשם מהם (צבא) וכלים נוספים כמו כלי** genAI**.**
* במעקב אחרי התלמידים, אני משתמשת בקובץ Google Sheet עם טבלה גדולה הכוללת עמודות שונות של דרישות החובה, ועוד כמה דברים שאני דורשת מהם. אחת לשבוע-שבועיים התלמידים מעדכנים את הקובץ וגם מראים לי מה הוסיפו.
* הנחייה אישית/קבוצתית: עליך לזכור שכל תלמיד הוא אינדיבידואל, בעל יכולת קוגניטיבית ומערכתית משלו, וגם אם הוא במגמת הנדסת תוכנה אין הדבר מעיד על הומוגניות מוחלטת בינו לבין חבריו למגמה. לפיכך המלצתי היא לבנות מסלול התקדמות אישית לכל תלמיד, כמובן תוך התחשבות בלוחות הזמנים המחייבים (מועדי בחינות של משרד החינוך, למשל). כלי שמאוד עוזר לי באופן אישי הוא גאנט מסודר, ובו טבלה בה מפורטות כל המשימות בחלוקה לפי נושאים, כאשר לכל תלמיד מוקדשת שורה, וכל תלמיד מסמן בתיבת הבחירה (צ'קבוקס) את המשימות שכבר ביצע. אני מוצאת שזהו כלי יעיל כיוון שהוא מאפשר לתלמידים לעקוב בעצמם אחר ההתקדמות שלהם, וגם להשוות את עצמם לקצב הכללי בכיתה בכדי להבין האם עליהם להדביק איזשהו פער. תלמידים מתקדמים מאותגרים יותר, למשל בהכוונה למציאת פתרונות בעצמם ברחבי הרשת, עידוד לכתיבת קוד גנרי יותר ושימוש בכלים שלא לימדתי בכיתה, לבחירתם.
* כדאי להגדיר להם מטרות מדוייקות קצרות טווח, משבוע לשבוע, שיראו את ההתקדמות ולהגדיר עוד משימה וכן הלאה.
* כל דבר שהכתה צריכה לבצע צריך להיות כתוב בצורה ברורה ונגישה לכלל התלמידים בקובץ גלובלי הכולל תאריכים מדויקים למתי זה ייבדק. יש לתרגל (בעיקר בחודשים הראשונים) עם התלמידים את הכניסה למסמך הגלובלי הזה ובדיקה מה נדרש שם. כל שיעור בחודשים הראשונים נפתח בלהסתכל מה צריך להיום, מה צריך למחר וכדומה. לאט לאט לתת לתלמידים לגדול ולהתרגל לעשות זאת בעצמם. בכתה י"ב גם כדאי לתת להם לרשום בעצמם את התאריכים של ההגשות השונות על מנת שיתרגלו לנהל בעצמם את התאריכים.
* ניהול זמנים נכון הוא קריטי להצלחת תלמיד בפרויקט, במיוחד עבור תלמידים שמתקשים לארגן את הלמידה שלהם. חשוב ללמד תלמידים חלוקה משימה לזמנים – הגדרת זמנים לפיתוח (קביעת זמן סיום), לתיקון באגים ולתיעוד. בצעו תכנון דו שבועי ומעקב לעזור לתלמיד לקבוע עדיפויות פונקציונלית בפרויקט.
* ריכוז החומר בתיקיות מסודרות, שם הקובץ שלי הוא שם השיעור ומיספור שקופץ ב- 10 במקרה וארצה להכניס נושא ביניים.
* שימוש בכלים טכנולוגיים כמו למשל github כדי ללמד את התלמידים את עקרונות הפיתוח.
* חשוב ללמד שימוש בכלים לניהול קוד (שימוש ב-GitHub לניהול קוד ולשיתוף פעולה(.
* תוכנת – Trello לניהול משימות, תיעוד התקדמות וקשיים. - בתחילת הפרויקט לאחר בחירת הנושא אנחנו מגדירים ביחד לוח זמנים ויעד לכל שלב בפרויקט. אנחנו משתמשים בטבלת ניהול זמנים המצורפת לתבנית של ספר הפרויקט עם תאריכי יעד ומשימות מוגדרות. כמו כן, ניתן להשתמש בכלי ניהול פרויקטים אינטרנטיים, למשל כמו Trello, התלמיד יכול לסמן התקדמות לפי צבעים, להוסיף הערות ולתעד קשיים שהוא נתקל בהם בדרך.

trello

* תלמידים מתקשים לרוב לא בגלל חוסר יכולת, אלא מחוסר ידע איך מתמודדים עם פרויקט תוכנה. חשוב להגדיר תוכנית עבודה ברורה, לחלק את הפרויקט למשימות קצרות וברורות עם הגדרת זמן ברורה. חשוב לוודא שהתלמיד יודע מה המשימה הבאה שלו ואיך למדוד את ההתקדמות שלו. חשוב ללמד שימוש בכלים לניהול קוד (שימוש ב-GitHub לניהול קוד ולשיתוף פעולה(.

**ח-2. תיעוד, ארגון ותיקי פרויקט**

* אני מבצע חזרה בבית לפני השיעור. כל החומר בשיעור מבוסס על מסמכי הוורד שנמצאים ב-classroom.
* אני מוסיף ללוח המבחנים שלהם מועדי הגשה: הצעת פרויקט , הקמת מסד נתונים, הצגת אמצע, סימולצית הגשה, הגשת ספר פרויקט. לקראת כל שלב, אני מזכיר להם את מועד ההגשה וכיצד ניתן לקדם את הפרויקט לקראתו. תלמיד שנתקע למשל עם באג "עקשן", אני מורה לו להניח לו ולהתקדם לפיצ'ר נוסף, שאפשר לפתח, מבלי שאותו באג ייתקע את הפרוייט, זאת כדי לעמוד בלוחות הזמנים.
* אני משתמש בקלאסרום באופן אינטנסיבי. תלמידים שלי מקבלים את כל המטלות דרך הקלאסרום ומעלים את הפתרונות כספריות דחוסות לבדיקה, הם הודרכו כיצד להעלות נכון פרויקטים דחוסים לבדיקה קלה נוחה גם עבורי. כנ"ל הדברים אמורים לגבי כל קבצי פרויקט הסיום שלהם כולל ספר פרויקט, יש להם טבלת זמנים לשלבי הפרויקט בהם הם נדרשים להעלות את קבצתי הפרויקט לבדיקה.
* במהלך השנה יש לכל תלמיד 30 הגשות, כל הגשה היא אישית ויש הגנה לכל מטלה – כלומר כל מטלה מוגשת ונבדקת למול התלמיד – כך שנדרשת הבנה מלאה ויכולת שינוי, ניתן להגיש הגשות חוזרות עד קבלת 100 ועד לגיליון ההערכה הבא כך כל תלמיד יודע בדיוק היכן עליו להשתפר.
* בפרויקט האישי העקיבה היא מעבר תלמיד תלמיד ולראות היכן הם נתקעים. אם מישהו נתקע – אני יושב איתם על הפרויקט ומנסה לפתור איתם את הבעיה – בצורה זו התלמידים לומדים איך אני פותר בעיות – debug, לשים breaking points , QA...
* חשוב שהתלמידים יתעדו בקובץ וורד תקלות וקשיים שהתמודדו איתם. בשלב מאוחר יותר, כשהם כותבים את ספר הפרויקט, הם שוכחים את כל המהמורות בדרך ולא מצליחים לשחזר את מהות התקלות ודרך פתרונן ולכן גם ברפלקציה, הם מתקשים לכתוב על חוויתם האישית.
* כמו לתלמידי כל המגמות, תמיד מסומנים בלוח המבחנים לפחות 2 מבחנים במחצית (מתכונת אחת לפחות) אני מנצלת את המועדים האלה ומתייחסת אליהם כ"מבחן" לכל דבר – על התלמידים להגיש לי ספר פרויקט + תיקיית פרויקט מלאה עם תיעוד – משתדלת לבחון אותם אישית כל תלמיד כמה דקות בכדי לדעת מה מצבו של כל אחד...
* קלסרום כיתתי - אני והתלמידים מאוד אוהבים לעבוד עם הקלסרום - זה כלי שמארגן את כל תהליך הלמידה וחומרי הלימוד בחלוקה לנושאים ותאריכי שיעור. במיוחד במקצוע של הנחיית פרויקטים זה כלי פדגוגי ממדרגה ראשונה עבור המורה ובעיקר עבור התלמידים שיש להם תמיד את האפשרות לאן להגיע ולהיות בעניינים.

**ח-3. בדיקות, דיבוג ולימוד שגיאות**

* הנחיית תלמיד שנתקע: לשאול את התלמיד שאלות מנחות, להוביל אותו לפתרון לבד. למשל: תראה לי איך ביצעת debug, תמצא את השגיאה בlog, העתקת לאינטרנט? חיפשת פיתרון?
* התמודדות עם תקלות – מיומנות חובה אצל כולם. אני נוהגת לתת לתלמידים קוד עם תקלות שונות. המשימה שלהם לתקן את כל התקלות כך שהקוד יעבוד כנדרש.
* זה לא מחשבים שעושים שגיאות, זה בני אדם שכותבים שגיאות. למד והדגם בדיקות והרצות קוד דחופות. אין לכתוב הרבה קוד במכה ולצפות שזה יעבוד. כותבים כל פעם שתיים/שלוש שורות ומריצים. כך יודעים היכן השגיאה.
* כתיבת קוד נכונה מצמצמת שגיאות. הדגם כתיבה רעה מול טובה. הסבר כיצד הכתיבה הזו מצמצמת שגיאות וכמובן בדוק שהתלמידים אכן מבצעים זאת. כאשר הם טועים, הצבע על המקום הבעייתי ובקש שהם יתקנו אותו.
* לא לפחד מבאגים! מסבירה לתלמידים שכנראה מישהו נתקל באותו באג לפניהם, וייתקל גם אחריהם ... ולכן חשוב לדעת כיצד לחפש את הפתרונות בגוגל או באמצעות בינה מלאכותית. מה שמוביל לטיפ האוף טופיק הקריטי ביותר בעיניי – אנגלית! כמה שיותר לתרגל ולשפר את האנגלית שלהם, כי זו השפה העיקרית בה מתרחשים הדברים ולכן חשוב לשכלל את מיומנויות השפה הזו בכדי לאפשר לעצמם התקדמות מקצועית מיטבית בתחום.
* ללמוד לתקן שגיאות הוא נושא חשוב לא פחות מכתיבת קוד. כאשר אתה מדגים קוד על המחשב מול התלמידים צור בכוונה שגיאות שונות בקוד ותן לתלמידים למצוא אותם. הקפד על גם על שגיאות קומפילציה וגם שגיאות אלגוריתמיות. למד את התלמידים למצוא שגיאות בעזרת הדפסות ובעזרת נקודות עצירה. למד אותם כיצד קוראים את השגיאות שסביבת הפיתוח מציגה והמידע הרב שניתן למצוא בסביבת הפיתוח (כמו ה call stack של איזה פעולה קראה לאיזה פעולה טרם השגיאה) וכדומה.
* למדו את התלמידים איך לדבג תוכנה, כולל עבודה עם כלי debug , מה שיאפשר לתלמיד יותר הבנה בחומר הנלמד.

**ח-4. שיטות הוראה ולמידה טכניות**

* איזון בין גמישות למסגרת ברורה: תהיה קשוח עם דדליינים, אבל גמיש עם הדרך להגיע אליהם – לוח זמנים ברור הוא קריטי, אבל אפשר לאפשר לתלמידים לבחור איך להגיע לכל שלב. לדוגמה, לאפשר להם לבחור אם להגיש תיעוד בכתב, בסרטון או במצגת.
* בכל נושא שימוש ב- GenAI , לפני הטמעת הקוד, אני מבקש מהם לעבור על שורות קוד חשובות, בהן יש שימוש במחלקות או מבנים לא מוכרים, לקרוא עליהם ולראות שהם מבינים את הפונקציונליות שלהם, זאת כדי שגם אם ההטמעה תצליח, חשוב שלא ידקלמו אח"כ את הקוד בהסבר מול הבוחן.
* בכל נושא שיש להטמיע, למשל: תפריט נפתח. לגשת למצגת, לעבור עליה קודם ולהבין את הרכיבים השונים. רק לאחר מכן להטמיע בתוך המערכת. בדרך כלל, אני כן נותן להם להתחיל לעבוד על העיצוב, משום שזה החלק הקל וכך הם יכולים לצלול ברכות יותר אל תוך העבודה.
* החינוך הטוב ביותר הוא כאשר אתה מבצע את הדבר ולא רק אומר זאת. הקפד לעמוד בסטנדרט שאתה דורש מהתלמיד.
* חשוב להגדיר תוכנית עבודה ברורה, חלוקה לתת משימות ברורות ומוגדרות היטב. כל תלמיד עובד בצורה שונה ולכן חלוקה ברורה למשימות, תעזור למקד את התלמידים.
* יצירת אווירה חיובית בכיתה: תן מקום לטעויות – חשוב שהתלמידים יבינו שטעויות הן חלק מהתהליך. אם תלמיד מפחד לטעות, הוא לא ינסה דברים חדשים. אתה יכול להדגים באג מכוון, ולשאול את הכיתה איך לתקן אותו.
* לגבי אתגרים ב-30%, שאלתי בתחילת השנה מה התחביבים של התלמידים ולדוגמא כשראיתי שחלק מהתלמידים לא מבצעים את הפרויקט עם מוטיבציה גבוהה, בניתי מערך שיעור חדש ששילב כדורגל (רוב הבנים בכיתה אוהבים את התחום), במקרה אחר התלמידים בנו עמוד למשחק שח-מט (תחביב של כמה בנות בכיתה). בניגוד לשנה שעברה שלקחתי נושא של חנות אינטרנטית והלבשתי את מירב תחומי הלימוד על נושא החנות, השנה אני מגוון מאוד. אני משתדל שהחלק הראשון של השיעור יתמקד בחזרה על החומר של השיעור הקודם והצגת החומר החדש. לפעמים אני גם מבצע את המשימה ב"לייב" כדי להראות לתלמידים את דרך המחשבה שלי. מתוך 3 שעות ברצף שיש לי, פרק הזמן הזה יכול לקחת בין 20 -40 דקות, שאר השיעור הוא תרגול וביצוע משימה. בפרויקט האישי אני מציג חומר חדש כל 3-5 מפגשים, כאן החלק הראשון של השיעור ייקח 50% ויותר מזמן השיעור.
* ללמד את התלמידים לתכנן לפני שמתחילים לקודד! לפני שניגשים לכתיבת תוכנה - חשוב ללמד תלמידים חשיבה מקדימה. לפני כל תרגיל לבקש מהתלמידים לאפיין ברמת מסכים ותרשים הזרימה את האלגוריתם.
* למדתי שכל דבר קטן שהתלמידים צריכים לבצע יוגדר ב-classroom כמשימה, אם לא יוגדר כך חלקם הגדול לא יבצעו אותו.
* תתחיל ללמד עם הצלחות לתלמידים. בחר חומרים קלים, ביצוע פרויקטים פשוטים שיאפשרו לתלמיד להצליח ולקבל ציונים גבוהים בהתחלה.

**ח-5. מתודולוגיות פיתוח והוראה**

* הגישה האג'ילית טובה יותר לעבודה בפרויקטים בהנדסת תוכנה, שלבים קצרים ומדורגים.
* ישנם נושאים שמשותפים לפרויקטים שונים, חשוב לכנס את התלמידים שזה נוגע אליהם וליצור קבוצת עבודה הכוללת הנחייה שלכם ועבודה משותפת של הקבוצה למימוש.
* חשוב לשבת עם כל תלמיד ובהתאם לפרויקט שלו להכין לו את מפת ההתקדמות האישית שלו, מבחינת למידה של הנושאים הספציפיים שהוא נדרש לפרויקט שלו והן לגבי שלבי ההתקדמות בפיתוח ומימוש הפרויקט שלו כולל תיעוד – ספר פרויקט.
* חשוב מאד ליצור אבני דרך ברורות. תלמידים חלשים, פעמים רבות, מרגישים אבודים, כי הם לא יודעים מאיפה להתחיל, איך לחפש מידע ואיך להתקדם באופן עצמאי. עבורם יש ליצור משימות קטנות עם הסברים ברורים ולוודא שהם מבינים כל שלב לפני שממשיכים הלאה. אפשר ליצור יחד איתם את סדר המשימות ולוודא שהם עומדים במשימה לפני מעבר למשימה הבאה. במקרה הצורך אפשר להצמיד לתלמיד חלש, תלמיד חזק יותר שילווה אותו.
* לא כל התלמידים חייבים להיות מתכנתים בעתיד – יש תלמידים שיבחרו ללמוד הנדסת תוכנה כי זה מסקרן אותם, אבל בסוף ילכו למקצועות אחרים. גם אם הם לא ימשיכו בפיתוח תוכנה, הם ירוויחו מיומנויות חשיבה ביקורתית, פתרון בעיות וניהול פרויקטים.