

## הנחיות והמלצות לגבי הפרוייקט בסייבר

1. בראש וראשונה התוכנה צריכה לעבוד על פי הספסיפיקציות שקבעתם. במילים אחרות, שלא יהיה פער בין מה שמתואר בספר הפרוייקט ובין הפעלתו. אפשר גם לתת את המוטיבציה לכתיבת הפרוייקט הזה (בין אם רצון אישי, או משהו מועיל, מעניין או בעל פוטנציאל מסחרי, או להיטיב עם הכלל ולשעשע אנשים).
2. בעקרון, כפי שכבר הזכרנו לפני מספר חודשים, הפרוייקט אמור להכיל לפחות את שלושת האלמנטים של: ממשק גראפי, רשתות תקשורת בין מחשבים וקשר למערכות הפעלה (חלונות או אחרת). יתכן שישאלו אתכם מה אימצתם עבור כל אחד מאלמנטים אלה ותהיו מוכנים להשיב ולהראות. למי שאין את שלושת אלה, צריך להתכונן להגן על העבודה על בסיס של נושא 'מחקרי מעמיק' שאינו בתוכנית הלימודים. מה המקורות שאותם חקרתם (קישורים, שמות ספרים, פירסומים, קוד של אחרים וכו'). נסו לעשות את זה מסודר ולהראות איך למדתם את הנושא (גם אם זה דבר שקרה לפני יב'), לענות על מדוע בחרתם את הפלטפורמה, איזה יתרונות היא נותנת לעומת תכנות בכלים שלמדתם, עם דגש על תהליך לימוד מורכב שבעקבותיו הצלחתם להרים פרוייקט.
3. ספר הפרוייקט צריך לעזור למי שלא מכיר את הפרוייקט שלכם להבין את כל מה שהתוכנה אמורה לעשות. קחו בחשבון שהבוחן יקדיש כרבע שעה עד 20 דקות לעיין בספר לפני שיבוא לבחון אתכם. מה שמצריך גם הסבר תמציתי (ברמת מנהלים) וגם הסבר מפורט לכל דבר (אם לא יבין מהתמצית).
4. הקוד צריך להיות מתועד היטב. עדיף עודף תיעוד מאשר חוסר. אם הקוד בפייתון, כדאי להיצמד להנחיות של pep8. אם חלק מהקוד שכתבתם לדעתכם מכיל תיחכום לא פשוט, אפשר להסבירו באריכות. מומלץ כמו ב 'זן של פייתון' שהקוד יהיה ברור, פשוט, קריא ושלא יחזור על עצמו (עקרון ה-DRY או Don't Repeat Yourself). כמו שאמר איינשטיין: 'הכל צריך להיות פשוט ככל האפשר, אבל לא פשוט יותר'.
5. מומלץ מאד לכתוב את התוכנה בעזרת מחלקות ועצמים (תכנות מונחה עצמים), זה מעלה את ערך הפרוייקט בעיני הבוחנים. כמובן חלוקה למתודות/פעולות/פונקציות לוגיות שאינן גדולות מידי (סדר גודל של עמוד. במילים אחרות, שלא יהיה צורך לדפדף על מנת לקרוא את הפונקציה.
6. אם סיימתם את הפרוייקט ונניח שיצא לכם 200 או 300 שורות תוכנה בלבד, כדאי מאד לחשוב על תוספות שיספקו את הדרישה של 'פרוייקט מעמיק ומשמעותי'.

7. מאד חשוב לכלול דיאגרמות כגון תרשימי זרימה שמתארים את חלקי הפרוייקט, גם בגדול של שלבי בתכנון וגם של הקוד עצמו. תחפשו מה זה DFD (Data Flow Diagram) - זהו כלי של עיצוב תוכנה וגם אם לא השתמשתם בזה, אפשר לאחר שהקוד כתוב, לתאר אותו כדיאגרמה מסוג זה.
8. כדאי גם לשתף פעולה ביניכם בצורה שלכל אחד יהיה תלמיד אחר שישחק את 'פרקליט השטן'. מה הכוונה? שאחד מכם ינסה להיכנס לנעליו של הבוחן וינסה להעלות על 'המוקד' את הפרוייקט של חברו. לשאול שאלות קשות על מה שיש, מה שאין, למה יש, למה אין ואיך הייתם משנים לדבר כזה או אחר וכמה זמן זה היה לוקח. רוב הסיכויים שאם תלמיד אחר ימצא 'פאקים' אצלכם, הבוחן בוודאי ימצא אותם. אפשר לראות את התהליך כמה שנקרא בתעשית פיתוח תוכנה כ: Quality Assurance, או בקרת איכות. זה יכלול מצב שבו גם תציגו אחד לשני את הפרוייקט ולכן זו תהיה הכנה טובה לפני שתציגו לפני ולפני הבוחן. כמה שיהיה לכם יותר ניסיון בזה, הביצועים יעלו בהתאם.
9. בכל פרוייקט תוכנה בעולם ישנן אפשרויות הרחבה, שיפור ושיכלול. נסו להכליל את כל האפשרויות, אפילו המשוגעות ביותר שהייתם יכולים להוסיף לפרוייקט, אם היה לכם זמן אינסופי. זה יכול להיות תוסף חשוב בספר הפרוייקט. גם ביל ג'וי מסאן מיקרו-סיסטמס וגם ליינוס טורוולדס מגוגל שנחשבים למתכנתים הטובים בעולם (או בין הטובים) אחרי שכתבו משהו, מייד יחשבו על איך לשפרו (גירסה 1.1).
10. למי שחושש שהפרוייקט שלו אולי פשטני מידי, אני מציע לנסות ולכתוב ספר פרוייקט מפורט יותר ומקצועי יותר. אפשר גם להסביר חלק מהקוד, לצרף צילומי מסך וכדומה. הרעיון מאחורי זה פשוט. כמה שהספר יהיה עבה יותר, יפחתו הסיכויים שהבוחן יקרא הכל בעיון. מצד שני, הפרוייקט 'יראה' יותר גדול ממה שהוא באמת ולכן לא יוכל לתת ציון נמוך על עבודה שנראית משמעותית ועמוקה. זה כמו שלפני שידרוג הטלפון לגרסה הבאה של מערכת הפעלה אתם מקבלים מסמך בעל 80 עמודים (כתוב משפטית) וצריכים להסכים עם התנאים. מישהו פעם קרא את כל התנאים?
11. אם מישהו יחשוב על עוד טיפים ו/או המלצות שכדאי לאמץ ולא כתובות כאן, אשמח אם תוסיפו רעיונות משלכם.