MAXTECH

פשוט להיות מאסטר בהייטק

git למפתחים

בקרת גרסאות מתקדמת

ניהול קוד מקצועי ושיתוף פעולה בצוותי פיתוח

מודול ראשון: יסודות git והתקנה

מבוא למערכת בקרת הגרסאות git

1. הבנת עקרונות בקרת גרסאות משותפת (Distributed Version Control)
2. התקנה מקומית והגדרות ההתקנה הבסיסיות
3. יצירת מאגר גיט מקומי (Local Repository Initialization)
4. הכרת סביבת העבודה: Git Bash ועבודה בממשק שורת הפקודה
5. תהליך ההעלאה לשלב ההכנה (Staging Area Management)
6. הבנה מעמיקה של פקודת git add והאפשרויות השונות

מודול שני: מחזור הפיתוח והיסטוריית הפרויקט

ניהול קומיטים ותיעוד שינויים

1. מהו Commit משמעות וחשיבות בתהליך הפיתוח
2. התקנה ויישום של Git Graph לתצוגה גרפית
3. ניתוח מצב המאגר באמצעות git status - הבנה מתקדמת
4. עיון בהיסטוריית הפרויקט: שליטה ב-git log ואפשרויותיו
5. עריכת קומיטים בדיעבד: amend ו-allow-empty כטכניקות מתקדמות
6. ניווט והתמצאות בהיסטוריית הפרויקט

מודול שלישי: ניהול שינויים וטכניקות ביטול

שליטה מתקדמת בתהליך הפיתוח

1. פקודות קיצור דרך ואופטימיזציה של זרימת העבודה
2. שימוש ב-restore לביטול העלאות לשלב ההכנה
3. אסטרטגיות מגוונות לביטול קומיטים ושחזור מצבים קודמים

מודול רביעי: ניהול נקודות ציון ושמירה זמנית
Tags וטכניקות Stashing מתקדמות

1. יצירה וניהול של Tags לסימון גרסאות ונקודות ציון חשובות
2. הבנה מעמיקה של מערכת ה Tags ויישומיה בפיתוח מקצועי
3. שליטה בGit Stash: list, message, untracked, apply
4. טכניקות מתקדמות ב drop, clear, pop ואסטרטגיות שימוש

מודול חמישי: כלי עזר ותמיכה טכנית
אופטימיזציה של סביבת הפיתוח

1. הבנת פקודות מערכת echo, clear, ניהול CRLF
2. התקנה והיכרות עם Sourcetree כ GUI מתקדם
3. הצדקה ויתרונות השימוש בכלים גרפיים לניהול Git
4. מערכת העזרה המובנית --h , help לתמיכה טכנית מיידית

מודול שישי: פיתוח מקבילי – ניהול Branches
אסטרטגיות פיתוח מבוזר

1. יצירה ויישום של אסטרטגיית Branching מקצועית
2. ניווט בטוח ויעיל בין ענפי הפיתוח השונים
3. זיהוי ומניעת מעברים לא תקינים בין Branches
4. פיתוח Safe Branch Switching

מודול שביעי: איחוד קוד וניהול קונפליקטים
אסטרטגיות Merge וטכניקות פתרון

1. הבנה מעמיקה של פקודת merge
2. ניתוח מתקדם באמצעות Sourcetree לתצוגה גרפית של מיזוגים
3. הגדרה וזיהוי של מצבי Non-Conflict
4. יישום מקצועי של פתרון קונפליקטים (Merge Resolution)
5. טכניקות מתקדמות לפתרון קונפליקטים מורכבים

מודול שמיני: ניהול היסטוריה וביטול שינויים
טכניקות מתקדמות לשחזור ועריכת היסטוריה

1. ביטול שינויים בשלבים שונים של תהליך הפיתוח
2. יישום revert לביטול קומיטים בצורה בטוחה
3. מחיקת קומיטים מההיסטוריה - טכניקות ואזהרות
4. אסטרטגיית Revert and Stash לניהול שינויים מורכב
5. שימוש ב-reset --soft לשמירת עבודה תוך מחיקת קומיטים
6. הבנת אפשרויות reset --mixed , reset --hard ויישומיהן

מודול תשיעי: אינטגרציה עם GitHub
מעבר לפיתוח מבוזר בענן

1. יצירה ויישום של Repository מרוחק ב-GitHub
2. טכניקות העלאה מקצועיות באמצעות push
3. הגדרת והקמת Origin לחיבור מאובטח עם GitHub

מודול עשירי: זרימת עבודה בצוות פיתוח
שיתוף פעולה מקצועי ב-Remote Repositories

1. יצירה וניהול של Branches ב-GitHub והורדתם לפיתוח מקומי
2. אסטרטגיות Merge וניהול Stash בסביבת צוות
3. טכניקות Pull עם Stash לשמירת עבודה מקומית
4. יישום Clone להורדת פרויקטים מ GitHub והקמת סביבת עבודה

מודול אחד עשר: סינכרון ועבודת צוות מתקדמת
ניהול עדכונים ושיתוף פעולה רציף

1. שיטת Pull לקבלת עדכונים בין מפתחים
2. ניהול מספר קומיטים בתהליך Pull מורכב
3. יישום fetch לקבלת עדכונים מ GitHub ללא השפעה על הקוד המקומי
4. אסטרטגיית Merge After Fetch לשליטה מלאה בתהליך העדכון

מודול שנים עשר: ניהול קומיטים מתקדם ויצירת תיקונים
טכניקות מתקדמות להעברת שינויים בין ענפים

1. יצירת קבצי Patch - ייצוא שינויים לקובץ חיצוני
2. ביטול שינויים ברמת הקובץ הבודד
3. העברת קומיט בודד מענף אחר לענף הנוכחי (Cherry Pick)
4. מיזוג של שני קומיטים מענפים נפרדים
5. טכניקות מתקדמות להעברת קומיטים בין ענפים - המשך ויישום

מודול שלושה עשר: ניהול ענפים מתקדם ו-Rebase
טכניקות Branching ו-Rebase מקצועי

1. יצירת ענפי פיתוח מקומיט מסוים והעלאה ל-GitHub
2. מחיקת Branches מקומיים ומרוחקים וניהול הקשרים ביניהם
3. יישום Rebase ללא קונפליקטים
4. פתרון קונפליקטים בתהליכי Rebase מורכבים