משתתפים :

* דורון בן זקן
* דור מור
* איתי רזון

**חלופות**

**החלופות האפשריות שלנו מתחלקות באופן הבא:**

1. **אלגוריתמים וספריות לזיהוי פנים**
2. **Backend services technology**
3. **Frontend technology**

**קריטריונים להשוואה:**

|  |  |  |  |
| --- | --- | --- | --- |
| **קריטריון** | **משמעות** | **משקל** | **טווח מדרג** |
| רמת קושי | מורכבות הפתרון בהתאם לניסיון המעשי שלנו | 30% | 1 – קשה |
| 10 – קל |
| זמן פיתוח עד תוצאה סופית | זמן שדרוש להשלמת הפיתוח מהשלב הראשון ועד להשקה בשוק | 15% | 1 – חודשים עד שנה |
| 10 – עד שבועיים |
| התאמה למטרות | עד כמה הפתרון עונה על הצרכים והדרישות העיקריות | 40% | 1 – לא מתאים |
| 10 – התאמה מלאה |
| דרישות לפיתוח | רכיבי תוכנה וענן שנעשה בהם שימוש והצורך בלמידה עצמית | 10% | 1 – דרישות רבות |
| 10 – דרישות מינימליות |
| עלות כספית | עלויות פיתוח ומשאבים | 5% | 1 – דרושים משקיעים |
| 10 – אין עלות |

**1. השוואה אלגוריתמים וספריות לזיהוי פנים:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **פיתוח מודל זיהוי פנים משלנו** | **זיהוי פנים מבוסס ענן** | **Dlib** | **OpenCV** |
| **רמת קושי**  **30%** | **פיתוח מודל עצמאי דורש ידע מתקדם במודלים של למידת מכונה, עיבוד תמונה, אלגוריתמים, ותכנות מתקדם. זהו פרויקט בעל אתגרים רבים.**  **ציון: 1** | **זיהוי פנים דרך שירותי ענן פחות מורכב, דורש יותר ידע בתקשורת API, אך השירותים מוכנים לשימוש.**  **ציון: 3** | **Dlib הוא קוד פתוח ולכן גמיש וזול אך, דורש** **ידע טכני מתקדם בנושאים כמו עיבוד תמונה ולמידת מכונה ובמערכות גדולותDlib עשוי להיתקל בקשיים מבחינת סקלביליות.**  **ציון: 3** | **OpenCV הוא קוד פתוח ולכן גמיש וזול, מציע כלים רבים לעיבוד תמונה וביניהם זיהוי פנים, בעל מהירות ביצוע גבוהה ותומך במגוון פלטפורמות.**  **ציון: 5** |
| **זמן פיתוח עד תוצאה סופית**  **15%** | **פיתוח ואימון המודל עצמאית דורש זמן רב ולא בר התכנות במסגרת הזמן הדרוש.**  **ציון: 1** | **זיהוי פנים מבוסס ענן מאפשר פיתוח מהיר יותר בזכות שירותים מוכנים לשימוש.**  **ציון: 7** | **Dlib מציע פתרונות מוכנים לשימוש, אך עדיין דורש זמן נוסף לפיתוח ולהתאמה אישית.**  **ציון: 5** | **OpenCV תומך בזיהוי פנים, אך ידרוש יותר זמן להתאמות אישיות בשיטה ידנית.**  **ציון: 7** |
| **התאמה למטרות**  **40%** | **פתרון מותאם אישית מבטיח את ההתאמה המלאה למטרות ודרישות ספציפיות של הפרויקט.**  **ציון: 7** | **זיהוי פנים מבוסס ענן הוא פתרון טוב, אך יש מגבלות בהתאמה אישית.**  **ציון: 7** | **Dlib הוא פתרון חזק שמתאים לפרויקטים מותאמים אישית, אך יש לו מגבלות במידת ההתאמה.**  **ציון: 7** | **OpenCV מציע כלים גמישים, אך לא תמיד מתאים בצורה אופטימלית לכל פרויקט.**  **ציון: 7** |
| **דרישות לפיתוח**  **10%** | **יצירת מודל עצמאי דורשת דרישות גבוהות, כולל ידע במודלים של למידת מכונה, עיבוד תמונה, ואלגוריתמים.**  **ציון: 2** | **פתרון מבוסס ענן דורש מינימום דרישות פיתוח, אך עדיין יש צורך בידע נרחב על שירותי הענן והשימוש בהם.**  **ציון: 6** | **Dlib דורש התקנה של ספריית קוד פתוח ותשומת לב להתאמות אישיות.**  **ציון: 5** | **OpenCV דורש התקנה של הספרייה והבנה בסיסית של עיבוד תמונה, אך דרישותיו נמוכות יותר.**  **ציון: 6** |
| **עלות כספית**  **5%** | **פרויקט עצמאי דורש משאבים כספיים גבוהים לפיתוח המודל ,לתיוג הנתונים ולאימון המודל(נדרשת חומרה וזמן ריצה רב).**  **ציון: 2** | **פתרונות מבוססי ענן דורשים עלויות די גבוהות עבור שימוש בשירותים מוכנים, שיטת תשלום לפי שימוש.**  **ציון: 5** | **Dlib הוא קוד פתוח ואין עליו עלויות ישירות, אך עדיין יש צורך במשאבים לפיתוח.**  **ציון: 6** | **קוד פתוח פשוט לשימוש עם קהילת משתמשים רחבה וחינמית, תמיכה מקיפה בפתרון בעיות.**  **ציון: 7** |
| **ציון משוקלל** | **3.55** | **5.6** | **5.25** | **6.3** | |

**בהתבסס על הציונים המשוקללים, ניתן לראות כי OpenCV הוא הפתרון המומלץ ביותר עבור הפרויקט הזה עם ציון משוקלל של 6.3, בשל יכולתו להציע גמישות, ביצועים גבוהים, תמיכה רחבה במודולים שונים, ועלות כספית נמוכה.  
זיהוי פנים מבוסס ענן הוא הפתרון השני המומלץ, הודות לפשטות השימוש ויכולת הפיתוח המהירה, אך הוא פחות מתאים בהתאמה אישית ובדרישות הפיתוח.  
Dlib מספק פתרון חזק אך דורש מיומנויות טכניות גבוהות ואינו מתאים תמיד לפתרונות בקנה מידה גדול.  
פיתוח מודל זיהוי פנים עצמאי דורש את ההשקעה הרבה ביותר בהיבט של זמן, משאבים, ועלות כספית, ולכן הציון המשוקלל הנמוך ביותר.**

**מקורות:**

* **[1] -** **Face recognition: challenges, achievements and future directions**<https://ietresearch.onlinelibrary.wiley.com/doi/full/10.1049/iet-cvi.2014.0084>
* **[2] - Microsoft Azure Face API**<https://learn.microsoft.com/en-us/azure/ai-services/computer-vision/overview-identity>
* **[3] - Dlib Documentation**<http://dlib.net/>
* **[4] - OpenCV Documentation**<https://docs.opencv.org/4.10.0/>

**2. השוואה Backend services technology**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **C ++** | **Java** | **Python** |
| **רמת קושי**  **30%** | **C++ היא שפה עוצמתית אך קשה ללמידה, במיוחד כשמדובר בניהול זיכרון, עיבוד נתונים וכתיבה יעילה של קוד. דורשת הבנה מעמיקה במודלים טכניים מתקדמים.**  **ציון: 6** | **Java היא שפה ברורה יחסית, דורשת הבנה בעקרונות תכנות מונחה עצמים (OOP) ועקרונות נוספים. יש לנו ניסיון בסיסי עם Java.**  **ציון: 7** | **Python היא שפה פשוטה לשימוש. סינטקס פשוט וברור, עם קהילה תומכת וספריות עשירות שמפשטות את תהליך הפיתוח. יש לנו ניסיון עם Python וגם הספרייה OpenCV זמינה בשפת תכנות זו.**  **ציון: 9** |
| **זמן פיתוח עד תוצאה סופית**  **15%** | **זמן הפיתוח בC++ הוא ארוך יותר, בגלל הצורך בהגדרה מדויקת של כל פרט ואלגוריתם.**  **ציון: 4** | **זמן הפיתוח בJava מהיר יותר יחסית, בזכות ספריות מוכנות ושירותים מתקדמים, אבל לא בהכרח מהיר כמו Python.**  **ציון: 7** | **זמן הפיתוח ב Python מהיר מאוד, בעיקר בזכות הפשטות שבה ניתן לכתוב קוד ולהתמקד בניתוח בעיות ופתרונן. יש גם ספריות עשירות המזרזות את תהליך הפיתוח.**  **ציון: 9** |
| **התאמה למטרות**  **40%** | **C++ מתאימה במיוחד לפיתוח מערכות low level וליישומים הדורשים ביצועים גבוהים, כמו משחקים ואפליקציות low level.**  **ציון: 7** | **Java מצוינת למערכות מבוזרות ויישומים גדולים, אך פחות מותאמת לפרויקטים קטנים או לשינויים מהירים.**  **ציון: 8** | **Python מתאימה כמעט לכל סוגי הפרויקטים: אפליקציות ווב, מדע נתונים, למידת מכונה, אוטומציה ועוד. גמישה ומאפשרת פתרונות מותאמים אישית בצורה פשוטה.**  **ציון: 9** |
| **דרישות לפיתוח**  **10%** | **C++ דורשת הבנה עמוקה של ניהול זיכרון, אלגוריתמים ויכולת עבודה עם מערכות מורכבות.**  **ציון: 6** | **Java דורשת ידע בהבנה מעמיקה של עקרונות OOP ותכנון ארכיטקטוני, אך לא נדרשת המורכבות של C++.**  **ציון: 7** | **Python דורשת הבנה בסיסית בלבד של תכנות ודרישות נמוכות לפיתוח. אידיאלית לפרויקטים קטנים עם אילוצים בזמן.**  **ציון: 9** |
| **עלות כספית**  **5%** | **לא רלוונטי**  **ציון: 1** | **לא רלוונטי**  **ציון: 1** | **לא רלוונטי**  **ציון: 1** |
| **ציון משוקלל** | **5.85** | **7.1** | **8.6** |

* **Python היא השפה בעלת הציון הגבוה ביותר בשל פשטותה, זמינות ספריות עשירות והקלות שבה ניתן לפתח פרויקטים, במיוחד במקרים של אילוצים בזמן.**
* **Java דורשת יותר זמן וידע לעומת Python, אך היא עדיין מציעה פתרונות טובים עבור מערכות מבוזרות ויישומים גדולים, עם ציון משוקלל של 7.1.**
* **C++ מציעה ביצועים גבוהים למערכות low level אך דורשת זמן רב יותר לפיתוח וידע טכני מתקדם, עם ציון משוקלל של 5.85.**

**במקרים שבהם הזמן והמגבלות הטכניות הם גורמים חשובים, Python מציעה את הפתרון האופטימלי ביותר.**

**מקורות:**

* **[5] –** **C ++ Documentation**<https://en.cppreference.com/w/>
* **[6] – Java Documentation**<https://docs.oracle.com/en/java/>
* **[7] - Python Documentation**<https://docs.python.org/>

**3. השוואה Frontend technology:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Vue** | **Angular** | **React** |
| **רמת קושי**  **30%** | **Vue היא פשוטה וגמישה עם עקומת למידה נוחה, אך עשויה להיות פחות גמישה בפרויקטים גדולים.**  **ציון: 8** | **Angular דורשת למידה של TypeScript ותבניות מורכבות. מתאימה למפתחים מנוסים.**  **ציון: 7** | **React דורשת הבנה ב-JavaScript ובתכנות מבוסס קומפוננטות. יש תמיכה רחבה בקהילה ותיעוד מעולה.  יש לנו ניסיון עם React.**  **ציון: 9** |
| **זמן פיתוח עד תוצאה סופית**  **15%** | **Vue מציעה פיתוח מהיר, אך, תדרוש מאיתנו זמן לימוד ארוך יותר ופחות מתאימה לפרויקטים מורכבים מאוד.**  **ציון: 8** | **Angular דורשת זמן רב יותר בשל מורכבותה, אך מספקת פתרונות מובנים רבים.**  **ציון: 7** | **React מאפשרת פיתוח מהיר בזכות פשטות וגמישות, ומציעה מגוון כלים מוכנים.**  **ציון: 9** |
| **התאמה למטרות**  **40%** | **Vue מתאימה לפרויקטים קטנים ובינוניים, אך פחות בפרויקטים גדולים או צוותים גדולים.**  **ציון: 8** | **Angular מצוינת למערכות מורכבות, אך פחות מתאימה לפרויקטים קטנים.**  **ציון: 8** | **React מתאימה לכל סוגי הפרויקטים, כולל קטנים, גדולים, ויישומים מורכבים. גמישות בבחירת כלים.**  **ציון: 9** |
| **דרישות לפיתוח**  **10%** | **Vue דורשת ידע בסיסי, אך עשויה להיות מסורבלת בפיתוח מערכות מורכבות.**  **ציון: 8** | **דורשת הבנה ב-TypeScript, ארכיטקטורה מתקדמת ותכנון מדויק, מתאים יותר לצוותים מנוסים.**  **ציון: 7** | **דורשת ידע בסיסי ב-JavaScript וניהול state, עם גישה לפתרונות פשוטים וגמישים.**  **ציון: 8** |
| **עלות כספית**  **5%** | **לא רלוונטי**  **ציון: 1** | **לא רלוונטי**  **ציון: 1** | **לא רלוונטי**  **ציון: 1** |
| **ציון משוקלל** | **7.65** | **7.1** | **8.5** |

* **React ממשיכה להוביל בבירור בזכות פשטות, גמישות והתאמה למגוון רחב של פרויקטים, עם תמיכה חזקה מצד הקהילה וכלים עשירים.**
* **Angular מתאימה למערכות מורכבות, אך דורשת השקעה רבה בלמידה ובפיתוח.**
* **Vue מציעה פתרון פשוט ונוח לפרויקטים קטנים ובינוניים, אך עשויה להיות מסורבלת לפרויקטים גדולים ומורכבים.**

**המסקנה הברורה: אם המטרה היא לעבוד ביעילות, גמישות גבוהה ותמיכה מצוינת, React היא הבחירה האידיאלית.**

**מקורות:**

* **[8] –** **Vue Documentation**<https://vuejs.org/guide/introduction.html>
* **[9] – Angular Documentation**<https://angular.io/docs>
* **[10] - React Documentation**<https://legacy.reactjs.org/docs/getting-started.html>