

תרגיל בית – שאלות פתוחות

שלב – 1

3.

את קוד המחשבון עצמו ניתן למצוא באפליקציה בקובץ `PhysicsCalculator.py`. הפרדתי אותו בכוונה ועשיתי לו `import` כדי שיהיה ניתן למצוא את קוד החישוב בקלות.

4.

ראשית כדי לוודא שהמוצר שבנינו מוסר תשובות נכונות ללקוח ניתן לחשב על דף עם מחשבון כמה חישובים ולראות שהם תואמים לחישובי המוצר שבנינו. בנוסף, קיימים כמה מקרי קצה אשר כדאי לשים לב אליהם.

דבר ראשון אשר צריך להתייחס אליו הוא זווית הזריקה.

את הזווית אני מודד מהכיוון החיובי של ציר נגד כיוון השעון.

אני אפשרתי באפליקציה כל זווית בין 0 ל 360 מעלות כך שאם הזווית היא בין 0 ל 90 או בין 270 ל 360 העצם ייזרק ימינה והמרחק יהיה חיובי ואם הזווית תהיה בין 90 ל 270 מעלות העצם ייזרק שמאלה והמרחק אשר יתקבל יהיה שלילי.

דבר נוסף, הוא גודל המהירות. פיזיקלית, המהירות היא וקטור ולכן יש לה זווית וגודל ואין משמעות לגודל שלילי כי הוא בערך מוחלט. (לכן הגבלתי באפליקציה לגודל חיובי בלבד)

בנוסף, הייתי צריך לטפל במקרה קצה בו המהירות ההתחלתית היא 0, במקרה זה זווית הנחיתה תמיד תהיה -270 (90-) וגם צריך לוודא שלא מחלקים ב0 בחישוב זמן הנחיתה.

מקרה נוסף אשר חשבתי עליו אך לא התייחסתי אליו בקוד כדי להשאירו ברור וקריא הוא מקרה בו הגובה הינו שלילי. אני לא אפשרתי קלט שלילי בקליטת הגובה אך ניתן גם ליצור מודל אשר מאפשר קליטת גובה שלילי.

מודל כזה ייחשב באופן דומה את התנועה הבליסטית אך יצטרך להודיע כי אין פתרון במקרים מסוימים בהם הזווית או המהירות לא מספיק "גדולים" כדי להגיע אל הקרקע (גובה 0) מהגובה השלילי בהם התחילו.

עוד דבר אשר אינו מוגדר היטב הינו מקרה בו יש זווית התחלתית אך אין מהירות ואין גובה. כלומר, אין תנועה בכלל, אני הנחתי שבמצב זה ניתן להגיד כי זווית הנחיתה היא עדיין 270 מעלות. (ניתן באותו אופן גם להגיד כי במצב כזה זווית הנחיתה שווה לזווית השיגור כי אין למצב זה באמת משמעות מאחר ואין שום תנועה)

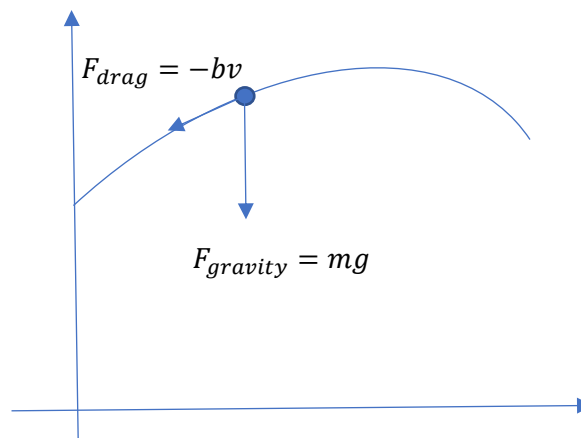
5.

כפי שנאמר בשאלה, המודל הפיזיקלי אותו אנו ממשים אינו מתאר בצורה מדויקת לחלוטין את התנהגות מציאותית של גוף אשר נע באוויר עם מהירות זווית התחלתית.

דבר זה נובע בעיקר מהחיכוך עם האוויר, לכן בכדי לתאר בצורה מדויקת יותר את התנהגות הגוף נצטרך להתחשב בכוח החיכוך עם האוויר.

את כוח החיכוך עם האוויר קשה לתאר ולחשב כי הוא מורכב מהרבה גורמים כמו צורת הגוף, גודל הגוף מסת הגוף דחיסות האוויר ועוד. לכן, בשביל שנוכל בכל זאת להתחשב בכוח החיכוך עם האוויר נסתכל על מודל מקורב עם קבוע b בו כוח החיכוך שווה $F_{drag} = -bv$ כאשר ניתן לדעת את b .

ראשית נשרטט סקיצה של המערכת ונראה איזה כוחות פועלים לכל כיוון.



על הגוף פועלים כוח המשיכה, וכוח החיכוך עם האוויר אשר פועל בניגוד לכיוון התנועה.

כדי לנתח את תנועת הגוף נצטרך ראשית לפרק את כוח החיכוך עם האוויר לרכיבו בציר ה x ובציר ה y .

נסמן:

$$F_{dx} = -bv_x$$

$$F_{dy} = -bv_y$$

לכן נקבל כי שקול הכוחות בציר ה x הוא:

$$\Sigma F_x = -bv_x$$

ושקול הכוחות בציר ה y הוא:

$$\Sigma F_y = -bv_y - mg$$

כעת ניתן לחשב באמצעות משוואות דיפרנציאליות ופתרונות מוכרים ולהגיע למשוואות המיקום בציר הx ובציר הy

$$x(t) = \frac{v_{0,x}m}{b} (1 - e^{(-\frac{tb}{m})})$$

$$y(t) = h + v_T t + (e^{(-\frac{tb}{m})} - 1)(v_T \frac{m}{b} - v_{0,y} \frac{m}{b})$$

כאשר v_T הינה המהירות הטרמינלית – המהירות המתקבלת כשאר כוח המשיכה שווה לכוח החיכוך עם האוויר, ונחשבה באופן הבא:

$$mg - bv_T = 0$$

$$mg = bv_T$$

$$v_T = \frac{mg}{b}$$

כעת באמצעות משוואות אלה, אם נרצה למצוא את המרחק שיעבור גוף ששוגר עם מהירות זווית מסוימת נחשב את הזמן שלוקח להגיע לגובה 0 באמצעות המשוואה השנייה ואת הזמן שנמצא נציב במשוואה הראשונה וכך נקבל את x שהוא המרחק האופקי שעבר הגוף.

בנוסף, אם מהירות הרוח היא קבועה אז היא תגרום להסטה של הגוף בווקטור $V_{wind}t$ כאשר V_{wind} הינה מהירות הרוח

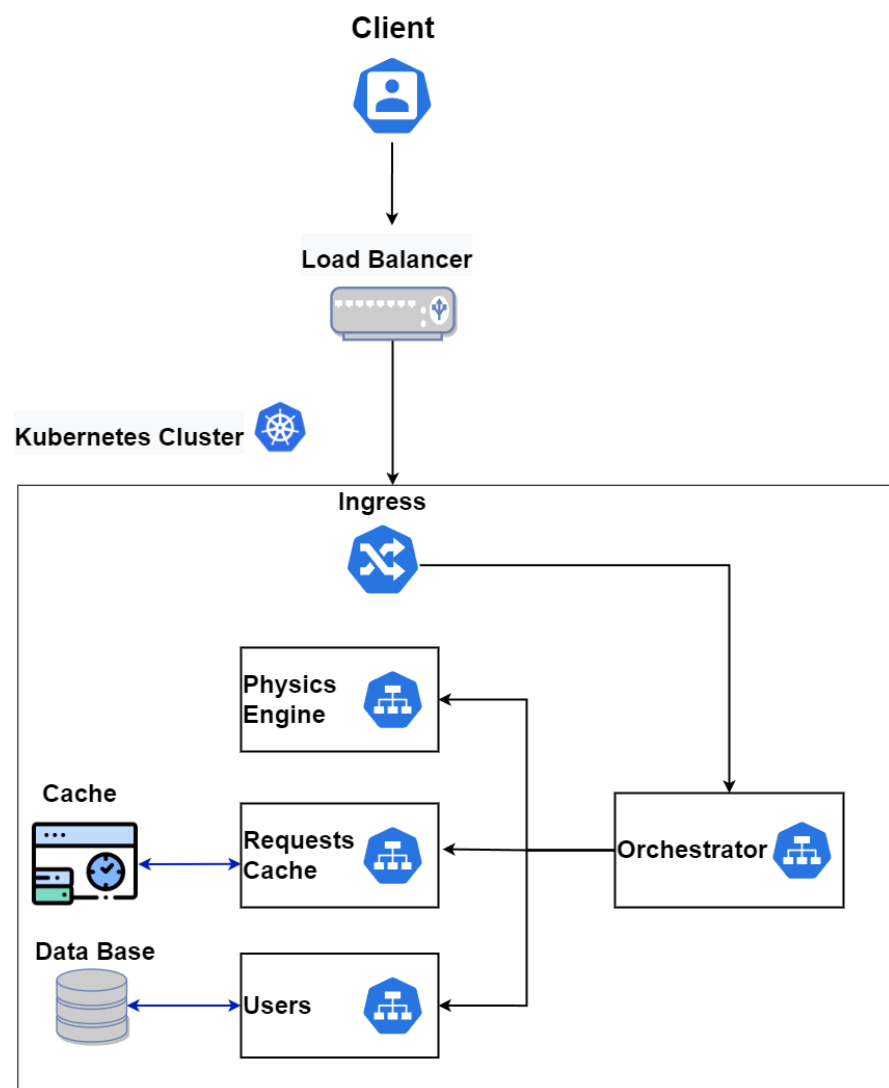
שלב - 2

4.

רעיון בניית המערכת:

המערכת תהיה מורכבת מ-4 microservices אחד אחראי על רישום וכניסת משתמשים, אחד אחראי על המנוע הפיזיקלי, ואחד אחראי על שמירת החישובים האחרונים ב-cache ובנוסף עוד אחד שינהל את הקשר בין כולם. זה יהיה microservices בשם orchestrator עליו יורחב בהמשך.

דיאגרמה להצגת סקיצה של ארכיטקטורת המערכת.



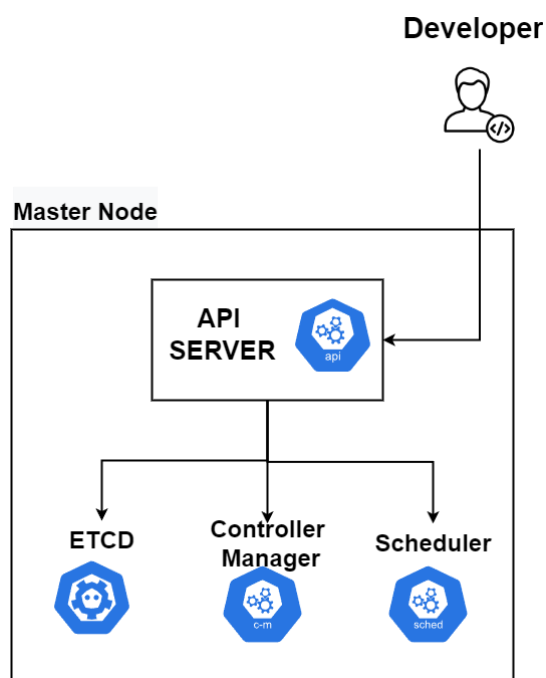
הדיאגרמה שהכנתי מורכבת מכמה חלקים. ראשית המשתמש פונה אל ה Load balancer אשר ימצא לו שרת מתאים (שרות פנוי בשעות העומס). לאחר מכן, בתוך השרת, נגיע אל ה Ingress אשר ידע להפנות אותנו אל ה microservice של ה Orchestrator. זה ינהל את הבקשות שמגיעה אליו באופן הבא:

ראשית, ה Orchestrator ישלח בקשה אל microservice בשם Requests Cache אשר מחובר אל Cache אשר מכיל בקשות חישובים קודמים ואת התשובה שלהם וה - Requests Cache ישלח בקשה לבדיקה האם החישוב הנדרש נמצא ב cache. אם כן, הוא יוחזר עם התשובה משם אל הלקוח. אם לא, ה orchestrator יפנה אל microservice בשם Physics Engine אשר מכיל את מנוע הפיזיקה ויחשב את התשובה בעצמו.

בנוסף, יכולות גם להגיע בקשות להתחברות והרשמה במערכת שלנו. לכן, קיים גם microservice בשם Users אשר יהיה אחראי על בקשות מסוג זה והוא מקושר לבסיס נתונים של המשתמשים הקיימים. כך לדוגמא, בעת בקשת כניסת משתמש ה Orchestrator יעביר את הבקשה עם הנתונים אל ה microservice של Users והוא יבדוק עם בסיס הנתונים אליו הוא מקושר אם המשתמש אכן קיים ויחזיר תשובה בהתאם.

התיאור לעיל הינו תיאור ב high level ואינו מתאר באופן מלא את המערכת אלא רק מציג את רעיון הבנייה וגם אינו כולל את צד המפתח.

אם נרצה להתחשב (לא יודע אם זה חלק מהדרישות) גם בצד המפתח אז נוכל להוסיף את הדיאגרמה של Master Node סטנדרטי:



ונקשר את ה REST API עם ה Kubelet אשר מותקן בכל node ואחראי כי כל containers שלנו רצים כמו שצריך.

נסתכל בקצרה על הרכיבים של ה Master Node.

ראשית יש את ה- API SERVER אשר עומד במרכז ה Master Node ומנהל את כל התקשורת בין הרכיבים והבקשות שמגיעות מה kubectl. למשל בקשה ליצירת pod

חדש. ה API SERVER יקבל בקשה ליצור pod חדש, יעדכן את ה ETCD עליו ויצור pod חדש לטיפול על ידי ה Scheduler. (יורחב עוד בהסבר על Scheduler)

הרכיבים הנוספים הם ה ETCD אשר משמש לאחסון הנתונים על מצב ה cluster שלנו וקונפיגורציות נוספות.

ה Controller manager אשר אחראי על מצב ה Cluster ויכול לעשות בו שינויים לפי הצורך. הוא מורכב מכמה תהליכי controllers כמו Replication controller אשר אחראי על ניטור מצב ה pods במערכת. למשל, אם pod אחד נפל, ה kubelet יודיע לו API ש pod אחד קרס, ה API ראשית יעדכן זאת ב ETCD ואז יודע ל Control manager אשר ישלח פקודה לפתוח pod חדש. (ויש controllers נוספים כמו node controller)

וה- Scheduler אשר אחראי על מציאת/השמת node מתאים ל pods חדשים שנוצרו לרוץ עליו. ה Scheduler פועל בשני שלבים. השלב הראשון הוא filtering, ראשית, הוא מסנן את ה Nodes אשר אינם מתאימים כלל לפי מספר דרישות (למשל, אין להם מספיק זיכרון).

השלב השני הוא Scoring, שלב זה הוא בעצם לבחור בעדיפות הכי טובה מבין ה nodes אשר עומדים בדרישות המינימליות. ה Scheduler "מדרג" את ה Nodes לפי מספר מאפיינים שונים ואז בוחר לעשות Binding לאחד הטוב ביותר.

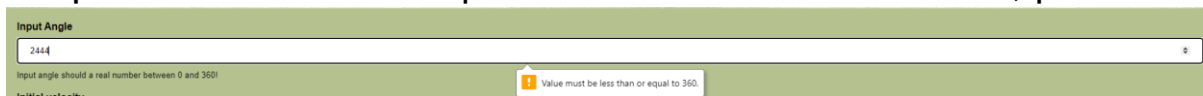
5. האפליקציה אשר מימשתי בשלב 3 מכילה בעמוד המחשבון הפיזיקלי את מוצר תוכנה זה. מתקבלות בקשות http לחישוב ומוחזרות תגובות http עם התשובה והצגתה בעמוד html, ראו בהמשך.

6. שאלה: מה על המערכת לעשות אם נשלחה בקשת חישוב לא צפוי ?

ראשית נשים לב כי כבר באפליקציה עצמה הגבלנו את בקשות החישוב רק לבקשות הגיוניות אשר נדע להתמודד איתם אך אם למרות זאת הגיעה בקשת חישוב שאינה צפויה אז יוחזר קוד שגיאה. במקרה זה, יתאים קוד שגיאה 400 - Bad Request (400) - כך גם מימשתי אצלי בקוד.

אצלי באפליקציה:

ראשית הגדרתי את הקלטים בתור מספרים ולכן לא ניתן להכניס טקסט או סימנים שונים. בנוסף, אם למשל נכניס זווית גדולה מ 360 נקבל הודעת שגיאה מהדפדפן -



דוגמאות נוספות:

אם לא נכניס קלט בכלל נקבל:

Initial height

height

Please fill out this field.

ואם נכניס גובה שלילי נקבל:

Initial height

-99

Value must be greater than or equal to 0.

בנוסף, מלבד בדיקות הקלט בצד בלקוח, מתבצעות בדיקות תקינות גם בצד השרת. אם הנתונים אשר קלטתי מהמשתמש אינם תקינים יוחזר קוד שגיאה 400 וגם הוספתי טקסט למסך:

Bad Request: Unexpected calculation request

מתוך ה inspect :

Request URL: http://127.0.0.1:5000/index
 Request Method: POST
 Status Code: 400 BAD REQUEST
 Remote Address: 127.0.0.1:5000
 Referrer Policy: strict-origin-when-cross-origin

7. בהתחשב בסיווג המערכת, כיצד נוכל להגן על הנתונים שזורמים במהלך התקשורת ?

כדי להגן על הנתונים שזורמים במהלך התקשורת נבצע מספר שלבים. שלב ראשון, נבצע כניסת למערכת באמצעות שם וסיסמא (ואפשר גם להוסיף טוקן). כך, נמנע באופן ראשוני מאנשים שאינם מורשים גישה למערכת.

בנוסף לכך, לכל משתמש תהיה את "הסמכות" שלו וכך תהיה לו גישה רק לחלקים מסוימים בערכת והוא יהיה מוגבל בפעולות ובשינויים שהוא יכול לעשות. באופן זה, גם אם תדלוף סיסמא של משתמש מסוים עדיין לא יהיה ניתן להשתלט על המערכת.

שלב שני, בדיקה ווידוא הבקשות. כל בקשה במערכת צריכה להיבדק ולהיות מסווגת כ"חברתית" "חברתית אך אינה חוקית" או "פוגענית". וכך ניתן למנוע מבקשות "פוגעניות" מלהגיע בכלל לשכבת המידע באפליקציה.

שלב שלישי אשר נועד בכדי להגן מפני "האזנות" והתקפות מסוג man in the middle, נעביר את הנתונים באופן מוצפן באמצעות פרוטוקול TLS, כלומר, נשתמש ב HTTPS במקום ב HTTP וכך גם אם מישהו יצליח להאזין למידע שמועבר הוא לא יוכל לפענח אותו והמידע יהיה חסר ערך.

מעבר לכך, נצטרך לשים לב כי בכל תגובה שאנו שולחים יש רק פרטים חיוניים ואין פרטים אשר יכולים לשמש נגדנו או לחשוף מידע חסוי.

בנוסף לכל אלו, כדי להגן על בסיס הנתונים של המשתמשים שלנו, נעביר כל סיסמא שמגיעה דרך פונקצית hash כך, גם אם תוקף יצליח לדלות נתונים מבסיס הנתונים שלנו הוא עדיין לא יוכל להיכנס כמשתמש לאפליקציה (הוא יראה בנתונים רק רצף תווים בלתי ניתנים לפיצוח ולא את הסיסמא של המשתמש – כך גם מימשתי אצלי באפליקציה)

לסיום אחרי שעברנו על כל השלבים אשר ציינתי לעיל, כדאי לבצע בדיקות אבטחה שונות כדי לוודא שאין פרצות אבטחה שלא חשבנו עליהם מראש.

8. ישנו סיכוי כי 2 או יותר משתמשים יבקשו את אותו החישוב (אותו קלט). כיצד ניתן לייעל את המערכת לאור זאת ?

כדי לייעל את המערכת ולחסוך ממנה חישובים חוזרים של אותו הקלט ניתן לשמור את החישובים הקודמים וכאשר מגיעה בקשה חדשה לבדוק בחישובים אשר שמרנו אם היא נמצאת ואם כן, להחזיר את התשובה שכבר חישבנו קודם לכן. ישנם שתי דרכים לעשות זאת,

דרך אחת היא באמצעות שימוש בבסיס נתונים. אנחנו יכולים לשמור כל בקשה שמגיעה ואת תוצאת החישוב שלה בבסיס הנתונים. בדרך זאת נוכל לשמור הרבה בקשות שיצטברו ואז עבור כל בקשה חדשה שתגיע נבדוק אם כבר חישבנו אותה בבסיס הנתונים שלנו ואם כן אז נשלח את התוצאה מבסיס הנתונים ונחסוך את החישוב. אולם, יש כאן tradeoff על הזמן אשר נדרש כדי לבצע חיפוש בבסיס הנתונים עבור כל בקשה חדשה שתגיע.

דרך נוספת היא באמצעות שימוש ב memory cache. היתרון של דרך זאת הוא שהגישה אליו היא יותר מהירה וכך נחסוך זמן בחיפוש בחישובים הקודמים בכל בקשה חדשה שתגיע אך ה tradeoff הוא כמות האחסון אשר ניתן לשמור בו ונצטרך לקבוע מדיניות פינוי (Eviction Policy) אשר תתאים לצורכי האפליקציה. (בסרטוט המערכת מופיעה דרך זאת)

9. אנו צופים כי עלולים להיות פרקי זמן בהם השימוש במערכת יהיה אינטנסיבי (המון משתמשים בבת אחת). כיצד נוכל לשפר את המערכת כך שתוכל לתת מענה מהיר למשתמשים רבים ?

כדי להתמודד עם פרקי זמן בהם השימוש במערכת יהיה אינטנסיבי קיים במערכת שלנו ה load balancer אשר יקבל בקשה וידע לנתב את הבקשה לשרת מתאים –

אחד מכמה שרתים אשר אנחנו נעשה להם deployment . באופן זה, לא כל העומס ייפול על שרת בודד ונוכל להתמודד עם פרקי הזמן בהם יש עומס על המערכת.

שלב – 3

4. על מה נשים דגש בעת פיתוח ממשק המשתמש?

בפיתוח הממשק נשים דגש על חווית משתמש נוחה ופשטות הכנסת הקלט כדי שהמשתמש יוכל להכניס את הנתונים מהר ככל האפשר ולא יבזבז על זה זמן חשוב. בנוסף, ניתן להציע למשתמש אפשרויות נפוצות או קיצורים שונים להכנסת הקלט.

5. איך נפתח את הממשק? באילו טכנולוגיות? מאילו אבני בניין נבנה את ממשק המשתמש?

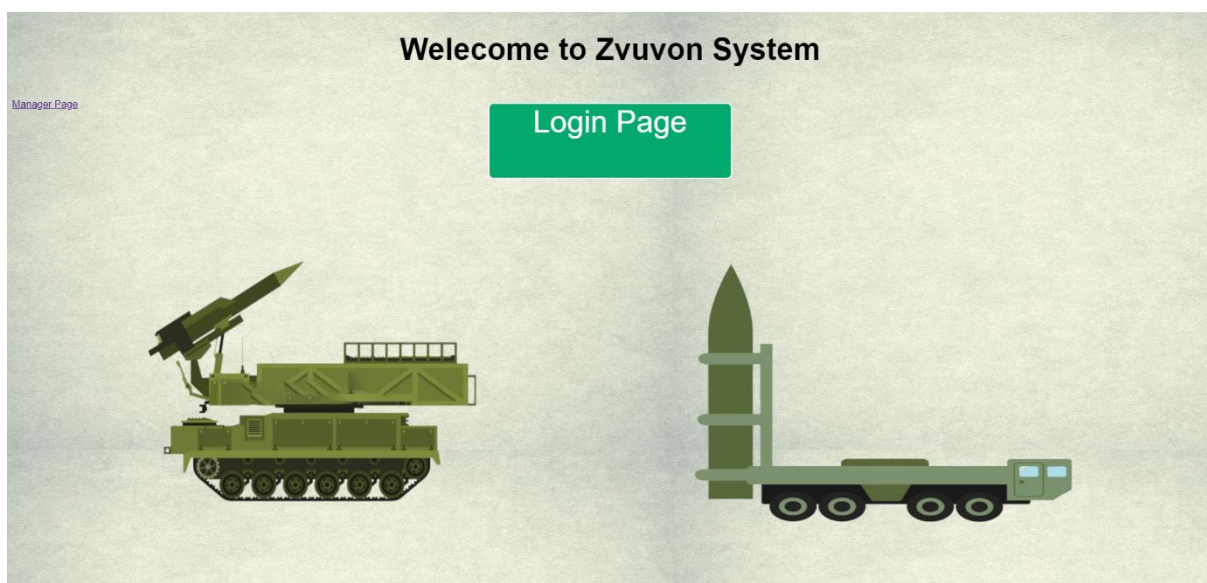
הממשק שלי מוצג כעמוד HTML אותו פיתחתי בעזרת Python Flask , CSS , HTML. באופן כללי, ממשק משתמש (מתקדם יותר כמובן ממה שאני עשיתי) יפותח גם עם כלים של front-end כמו React.js ו-Angular בנוסף HTML , JS , CSS וספריות נוספות.

צילומי מסך מהאפליקציה אשר מימשתי:

(במימוש האפליקציה שמתי דגש והקדשתי את רוב זמני על הפונקציונליות ופחות על הנראות ולכן האפליקציה באמת לא נראית טוב, מצטער 😞)

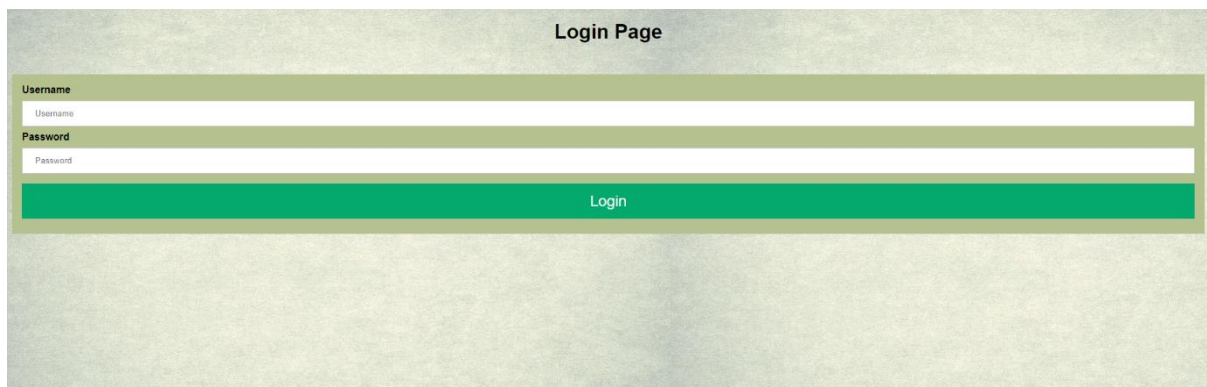
מסך בית - מסך הנחיתה של האפליקציה.

מציג קישור לעמוד הכניסה המאובטחת.



כניסת משתמש – משתמש מזין שם וסיסמא אשר נבדקים בבסיס הנתונים ואם הם נכונים הוא מועבר לעמוד מחשבון הפיזיקה של המערכת.

(קיים שימוש ב hash לסיסמאות – קודם עושים hash לסיסמא ורק אז משווים אותה לסיסמא מבסיס הנתונים כי גם הסיסמאות בבסיס הנתונים עברו hash)



Username

Username

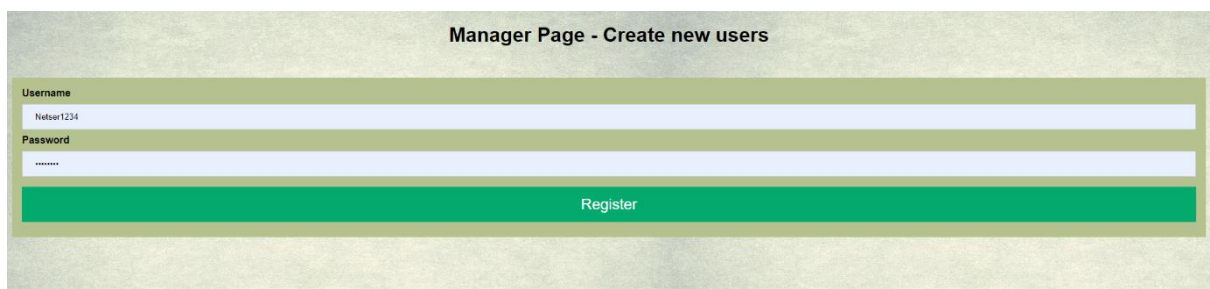
Password

Password

Login

עמוד מנהל להוספת משתמשים (כרגע שמתי אליו קישור קטן בצד שמאל למאלה ממסך הבית כדי שיהיה ניתן להגיע אליו בקלות לצורכי בחינת האפליקציה אך ברור שזה לא יהיה ככה במערכת אמיתית כי אסור שתהיה גישה לעמוד המנהל ממסך הבית).

בעמוד זה המנהל מכניס שם משתמש וסיסמא והם נכנסים לבסיס הנתונים וכך הוא בעצם יוצר משתמש מורשה חדש.



Manager Page - Create new users

Username

Heter1234

Password

Register

מחשבון הפיזיקה של המערכת – המשתמש יזין זווית התחלתית, מהירות התחלתית וגובה התחלתי וילחץ על submit. כך הוא בעצם שולח בקשת http מסוג post עם הנתונים ומקבל בחזרה תגובת http עם מידע אשר מוצג לו בעמוד. (מוצג בעמוד הבא)

Physics Calculator

Input Angle

Input Angle

Input angle should be a real number between 0 and 360!

Initial velocity

velocity

Initial height

height

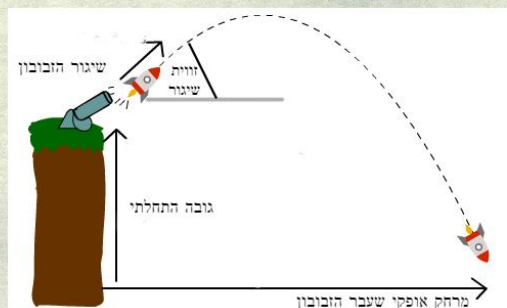
Submit

תוצאות – יוצגו למשתמש: זמן הטיסה הכולל, מהירות הנחיתה בציר האנכי ובציר האופקי, זווית הנחיתה, והמרחק האופקי הכולל שעבר הגוף (לפי לנתונים אשר הגיעו מתגובת ה-http)

Results

Time of flight :	1.50174154 Seconds
Landing horizontal velocity:	0.70710678 m/s
Landing vertical velocity:	-14.02497772 m/s
Landing angle:	-87.11372388 degrees
Distance traveled:	1.06189163 Meters

Flight Visualization



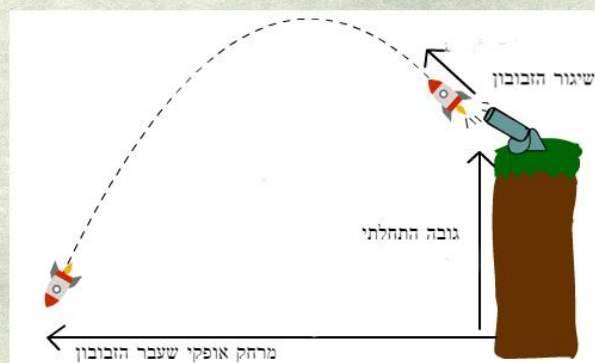
אם המרחק אשר עבר הגוף הינו שלילי ציור התעופה של הזבובון גם ישנה כיוון.

הערה: בציור זה לא כתבתי "זווית השיגור" כי זאת אינה זווית השיגור כפי שנמדדת במודל שלנו. הזווית עליו אנחנו מדברים במודל שלנו היא זווית יחסית לכיוון החיובי של ציר ה-x נגד כיוון השעון וכאן הזווית שנראית יחסית לקרקע היא לכיוון השלילי של ציר ה-x עם כיוון השעון.

Results

Time of flight :	6.04763728 Seconds
Landing horizontal velocity:	-24.04163056 m/s
Landing vertical velocity:	-35.28569115 m/s
Landing angle:	235.73174411 degrees
Distance traveled:	-145.39506124 Meters

Flight Visualization



ואם הגוף לא עבר מרחק ציור התעופה יראה כך:

Results

Time of flight :	1.42784312 Seconds
Landing horizontal velocity:	0.0 m/s
Landing vertical velocity:	-14.00714104 m/s
Landing angle:	270 degrees
Distance traveled:	0.0 Meters

Flight Visualization



הערה: הוספתי קובץ טקסט בשם requirements.txt אשר מכילות את הדרישות להרצת האפליקציה.

כדי להתקין בפשטות ניתן לכתוב בCMD

```
pip install -r requirements.txt
```