Final React Exam - War Simulator

"סימולציית התקפה והגנה"

הסיפור:

בעולם של מתיחות גוברת, נולדת סימולציה חדשה שמאתגרת את השחקנים לקחת חלק בעימות וירטואלי בין כוחות ההגנה של ישראל לבין קואליציית ארגונים עוינים. כל שחקן נדרש לנווט את תפקידו באופן אסטרטגי: המגינים בוחרים אזורים בישראל (צפון, דרום, מרכז או יהודה ושומרון) ומשתמשים במערכות הגנה מתקדמות, כמו כיפת ברזל, כדי ליירט טילים וכטב"מים שנשלחים לעברם. מנגד, התוקפים מייצגים ארגונים כמו חיזבאללה או חמאס, שלכל אחד מהם יכולות תקיפה ומגבלות ייחודיות, בהתבסס על מקביליהם בעולם האמיתי. המתח מתגבר ככל שהשיגורים והיירוטים מתקיימים בזמן אמת דרך התראות חיות, וטוענים את השחקנים באווירת חירום ועימות בלתי פוסק. רק השחקנים המיומנים והמהירים ביותר יצליחו להגן על שטחם או לפרוץ את קווי ההגנה, ולהפוך כל החלטה לקריטית ומותחת.

מטרה:

פיתוח סימולציה שבה משתמשים משויכים לתפקיד הגנה (ישראל) או לתפקיד התקפה (ארגוני אויב). כל משתמש יקבל כלים ואפשרויות שונות בהתאם לתפקידו.

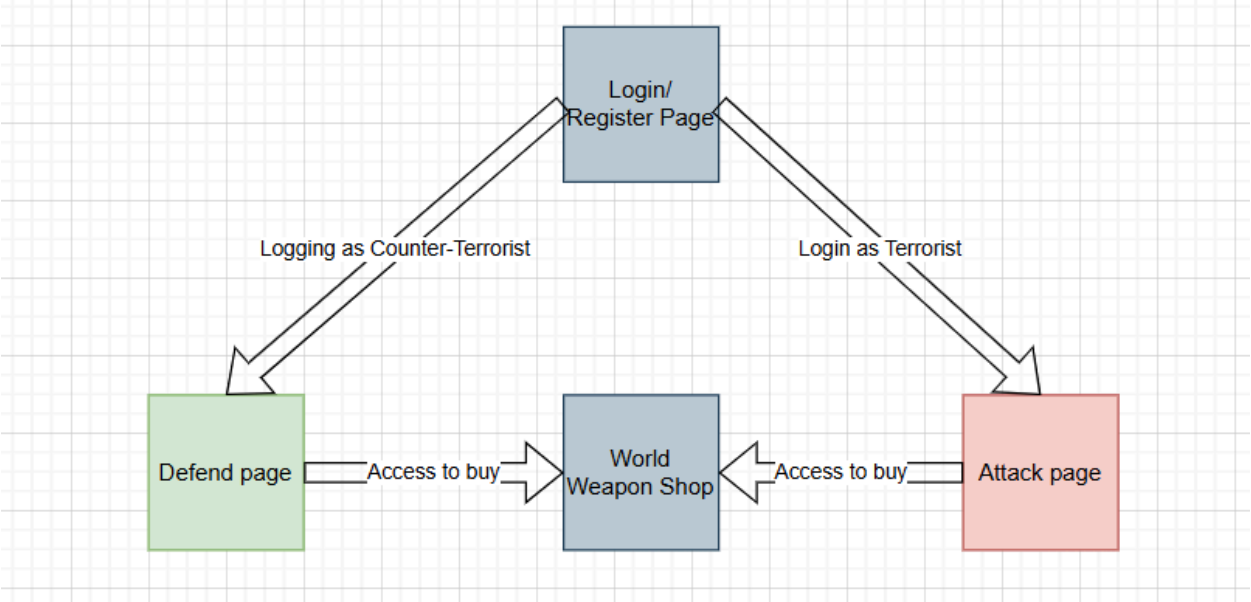
מבנה הפרויקט

1. **צד ההגנה (ישראל):**
   - מחולק לאזורים: צפון, דרום, מרכז ויהודה ושומרון.
2. **צד ההתקפה:**
   - מיוצג על ידי ארגונים (חיזבאללה, חמאס, חות'ים ומשמרות המהפכה).

Final React Exam - War Simulator

דרישות

ארכיטקטורה



1. אימות והקצאת משתמש:

- יש לממש מערכת הרשמה והתחברות באמצעות REST API.
- בעת ההרשמה, המשתמש בוחר ארגון (צה"ל או אחד אחד מאויבי ישראל המיודעים).
- במידה והמשתמש בחר בצה"ל, עליו לבחור גם את האזור. בבחירת האזור, זה ישפיע על אילו טילי הגנה המשתמש יראה.

2. Backend Setup:

- יצירת ניתוב (routing) עבור התחברות והרשמה.
- לכל משתמש יהיו:
  - שם משתמש
  - סיסמה
  - ארגון (צה"ל או ארגון התקפה)
  - אזור (אם מדובר בצה"ל)
- יש להשתמש ב-**organization.json** כדי לטעון נתוני התחלה לכל ארגון ואזור ולהזין אותם למסד הנתונים (בפעם הראשונה בלבד).
- יש להשתמש ב-**missiles.json** כדי לטעון נתונים על טילים ולהשתמש בהם כדי לוודא יכולות יירוט.

3. Frontend Pages:

- **דף הבית:**
  - יאפשר למשתמשים להתחבר או להירשם.
- **לוח הבקרה של ההגנה (למשתמשי צה"ל):**
  - הצגת כל כלי ההגנה הרלוונטיים בהתאם לאזור הנבחר.
  - לכל כלי הגנה יופיעו יכולות היירוט שלו (למשל, כיפת ברזל יכולה ליירט רקטות או כטב"מים).

Final React Exam - War Simulator

- בעת תקיפה לאזור של המשתמש:
  - הצגת התראה עם אפשרות ליירוט (אפשרות יירוט תופיע במסך בצורה של כפתור - 'X' אדום במסכים)
    - במידה ואין כלי יירוט מתאים, הצגת האיום ללא אפשרות יירוט
- כל תוצאה של יירוט מוצלח או כושל תוצג בלוח הבקרה ותישמר.

- **לוח הבקרה של ההתקפה (למשתמשי התקפה):**
  - לאחר כל שיגור, עדכון סטטוס הטיל ("פגע" או "לא פגע").

4. כללי התקפה והגנה:

- **הגנה:**
  - לכל אזור בישראל יש מערכות הגנה מסוימות. בעת תקיפה, המשתמש יקבל הודעת אזהרה ויוכל להפעיל את מערכת ההגנה המתאימה אם היא זמינה.
  - אם למשתמש אין יכולת יירוט למתקפה המסוימת או לחילופין נגמר המלאי של הטילים, הוא יראה את האיום ללא יכולת להגיב.
  - כל תוצאה של יירוט מוצלח או כושל תוצג בלוח הבקרה ותישמר.
  - לכל טיל יירוט יש גם זמן יירוט (speed) כמו שלטיל יש זמן פגיעה, ולכן עליכם לוודא כאשר אתם מנסים להגן שזמן היירוט קטן מזמן הפגיעה שנשאר לטיל.
- **התקפה:**
  - לכל ארגון יש מגבלות לגבי האזורים שאליהם הוא יכול לשגר. לדוגמה, לא לכל ארגון יש אפשרות לשגר טילים לכל האזורים.
  - המשתמש יכול לבחור את סוג הטיל ולשגר אותו לאזור זמין. לאחר השיגור, מערכת תעדכן את מצב הטיל (האם פגע או לא פגע).

5. פרוטוקול תקשורת:

- יש להשתמש ב-REST API עבור התחברות והרשמה.
- יש להשתמש ב-WebSockets עבור שיגורי טילים והתרעות בזמן אמת.

בונוס אופציונלי

- **חנות טילים:**
  - יצירת דף גישה מוגבל שניתן להגיע אליו רק מדפי ההגנה או ההתקפה.
  - הצגת טילים זמינים ואפשרות להוסיף כמות מסוימת מהם לארסנל של המשתמש, ללא צורך במערכת תשלום.
  - אין מגבלת תשלום, אפשר לקנות כמה טילים או הגנות שרוצים

טכנולוגיות מומלצות:

- **Frontend:** TypeScript, React
- **Backend:** Node.js with TypeScript, REST API, WebSocket

Final React Exam - War Simulator

גישה לפרויקט: חלוקת עבודה לבלוקים של שעתיים

יום 1: Backend ומסד נתונים

1. **הקמת מסד נתונים וטעינת נתוני התחלה (שעתיים):**
   - הגדרת סכמת הנתונים במסד הנתונים וטעינת נתוני התחלה מהקבצים `organization.json` ו-`missiles.json`.
2. **מימוש API בסיסי עבור הרשמה והתחברות (שעתיים):**
   - יצירת מסלולי API עבור רישום והתחברות למשתמשים.
3. **לוגיקת הקצאת תפקידים ואזורים (שעה):**
   - הגדרת החוקים למשתמשים בתפקידים שונים, כולל בחירת אזור עבור משתמשי צה"ל.
4. **API נתוני טילים (שעתיים):**
   - יצירת מסלולי API להצגת נתוני טילים ואפשרויות יירוט עבור המשתמשים.

יום 2: Frontend עם React ו-WebSocket

1. **הקמת Frontend ודף הבית (שעתיים):**
   - יצירת דף הבית, טפסי רישום והתחברות, וחיבור ל-API של ה-Backend.
2. **פיתוח לוחות בקרה להגנה והתקפה (שעתיים):**
   - פיתוח לוח בקרה מותאם למשתמשי ההגנה ולמשתמשי ההתקפה, כולל אפשרויות יירוט ושיגור.
3. **אינטגרציה עם WebSocket (שעתיים):**
   - הגדרת WebSocket ליצירת התראות בזמן אמת עבור יירוט ושיגור טילים.
4. **בדיקות סופיות, ניפוי באגים ותוספות אופציונליות (שעתיים):**
   - בדיקות וביצוע התאמות אחרונות.