**המרכיבים ותפקידיהם:**

לקוח- web +mobile:

מספק את ממשק המשתמש עבור לקוחות לאינטראקציה עם מערכת הקניות. משתמשים יכולים לבחור קטגוריות, לבחור מוצרים ולהוסיף אותם לעגלת הקניות שלהם.

**Client:** שימוש ב React.js (reack hooks/Redux).

**Server:** שימוש ב API (יצויין בסעיף הבא), ביצוע לוגיקה עסקית של שימוש ב AWS Lambda ו/או Services.

API-

משמש כנקודת הכניסה לכל בקשות HTTP של הלקוח. הוא מטפל בניתוב בקשות, חיבור וטיפול בפרוטוקול ה HTTP. הוא גם מספק תכונות אבטחה כמו בדיקת בקשות ואימות.  
יצירת פרוייקט API של .net core והגדרות AWS ב startup. ההגדרות תבוצענה גם בקבצי appsettings.json.  
הגדרות קונפיגורציית AWS כולל Buckets בקבצי C#.  
**בשימוש של פונקציות Lambda** – שירות זה מאפשר להריץ קוד על השרתים של אמזון ללא הצורך להגדיר שרת.  
כל פונקציה תטפל ב business logic שלה (Category, ShoppingCart ,Product, Customer וכו').  
פונקציות lambda מתקשרות עם בקשות ה CRUD (create, read, update, delete).  
**בשימוש של Interface**- כל interface יטפל ב business logic שלו, כשיהיה Class שירש ממנו ויממש את הפונקציות שבו.

DB- מסד הנתונים

RDS (Relational Database Service)- העברת בסיס הנתונים SQL אשר נמצא בשרת המקומי ל RDS של AWS. יהיו מספר טבלאות:  
א. טבלת Category המכילה את הקטגוריות ← לעתיד. כעת לא רלוונטי.

ב. טבלת Product המכילה את המוצרים.

ג. טבלת -ProductCategory טבלת Reference המקשרת בין 2 הטבלאות הרשומות מעלה ← לעתיד. כעת לא רלוונטי.

ד. טבלת Customer- המכילה את הלקוחות.

ה. טבלת ShoppingCart המקשרת בין הלקוח למוצר הנבחר ← לעתיד. כעת לא רלוונטי.

ו. טבלת CustomerProduct המכילה את שם הפריט והקטגוריה אותם בחר המשתמש.

Authentication & User Management-

Authentication-

Cognito- שימוש ב Cognito AWS לאימות וניהול משתמשים- sign up, sign in and tokens.

Authorization-

שימוש ב JWT tokens

Load Balancer-

מספק תעבורה של קריאות נכנסות מהרשת על פני שרתי קצה מרובים כדי להבטיח שאף שרת בודד לא הופך לכבד ומוצף, מה שמשפר את זמינות המערכת ומהימנותה.

שרתי אינטרנט- -EC2 Instances Elastic Compute Cloud

שרת וירטואלי של אמאזון, המאפשר להריץ את האפליקציה באמצעות ה AWS. שירות זה מאפשר להגדיל את עומסי העבודה למספר בלתי מוגבל של מכונות וירטואליות (VMs).  
יכיל את ה- logic של ה-frontend וה-backend של האפליקציה. מופעים מרובים מבטיחים זמינות גבוהה ויכולת התאמה להיקפים רבים של תעבורה.

אחסון נתונים- Amazon S3

מספק אחסון עמיד ביותר ומותאם להיקפים מאוד גדולים של אחסון.

Static Assets - מאחסן נתונים סטטיים כמו תמונות, לוגו, קבצי CSS, קבצי js וגיבויים.

Secure Access – שימוש בכתובת URL ידועות מראש לצורך גישה מאובטחת.

ניטור ורישום- CloudWatch, ELK Stack

מספק ניטור בזמן אמת, רישום והתראה. AWS CloudWatch אוסף ועוקב אחר מדדים, מנטר קבצים אנליטיים ומגדיר התראות. ניתן להשתמש ב ELK (Elasticsearch, Logstash, Kibana) לניתוח אנליטיקה מתקדמת.  
ניטור- אוסף ומנטר נתונים.  
מעקב- מספק מעקב מן הקצה אל הקצה על מנת לנתח ביצועים, נפילות ובעיות ומספק התראות יזומות של המערכת.  
ניתן להשתמש גם ב - AWS X-Ray על מנת לדבג ולנתח את הביצועים של האתר.

Configuration Management (Ansible/Chef/Puppet)-

מנהל את תצורת המערכת כדי להבטיח עקביות בסביבות שונות. כלים כמו Ansible, Chef או Puppet יכולים להפוך את התצורה לאוטומטית ולהבטיח תאימות.

Security (IAM, VPC, Security Groups)- אבטחה

מבטיח שהמערכת מאובטחת מפני גישה לא מורשית.   
AWS IAM (ניהול זהות וגישה) מנהל גישה של המשתמשים והרשאות המשתמשים.   
VPC (Virtual Private Cloud) מבודד את הרשת, על מנת לבנות רשת וירטואלית בתוך AWS.

הצפנת נתונים במעבר והאחסון. עבור נתונים רגישים ניתן להשתמש ב AWS KMS.  
קבוצות האבטחה שולטות בתעבורה הנכנסת והיוצאת.

גיבוי ושחזור נתונים-

מבטיח שהנתונים יגובו באופן קבוע וניתן לשחזר אותם במקרה של אובדן נתונים או כשל במערכת. נעשה שימוש בגיבויים אוטומטיים ובתכונות המסופקות על ידי RDS ו-S3.

Caching- AWS ElastiCache (Redis or Memcached)

מיועד לאחסן נתונים שניגשים אליהם הרבה ב cache.

Session Management- שומר ב cache את ה user session ואת הנתונים שניגשים אליהם פעמים רבות.

ביצועים- משפר את הביצועים של האתר ע"י צמצום הפעמים בהם ניגשים ל DB.

**דגשים:**

אוטומציה:

1. שימוש ב CD/CI לפריסה אוטומטית.
2. כלי ניהול תצורה להגדרת סביבה עקבית ואוטומטית.
3. התאמה אוטומטית של המופעים בהתבסס על עומס וטעינה הדפים.

הגדרות:

הגדרות AWS כמו CloudFrontPrivateKeyPath, Profile, ProfilesLocation שונות בין סביבות שונות (dev, qa, prod).

ניהול תצורה:

1. שימוש ב-Ansible/Chef/Puppet כדי להבטיח שכל המופעים מוגדרים באופן עקבי.
2. שימוש בקבצי קונפיגורציה עפ"י גרסאות, כדי לשמור על מעקב וחזרה אחורה במידת הצורך.

על ידי הטמעת ארכיטקטורה זו, מערכת הקניות יכולה להשיג זמינות גבוהה, מדרגיות, אבטחה ותחזוקה בענן.

[Frontend (Web & Mobile)]

↓

[API Gateway] <--> [AWS Cognito]

↓

[Lambda Functions] ---> [RDS (SQL)]

↓ ↓

[S3 Storage] [ElastiCache]

↓

[CloudWatch & X-Ray]