פרויקט סיום סייבר

SECRET DEALS

אורט ע״ש גוטמן

עדן דוננפלד

328074042

ריקי יפה, יונתן מירז ועידו מגנר

חלופת הגנת סייבר ומערכות הפעלה

תאריך הגשה 17.5.2023

תוכן עניינים

מבוא

ייזום –

תיאור ראשוני של המערכת –

נושא הפרויקט הוא תוכנת שרת-לקוח אשר מחפשת ומציגה כתוצאה את הדיל הכי משתלם ליעד שנבחר בטווח תאריכים, לפי מסננים שהלקוח בחר.

כל לקוח מזין את מקום המראה, מקום נחיתה, תאריך המראה, תאריך נחיתה וכמות נוסעים. כמו כן, קיימת אפשרות להזין מסנני טיסה ומסנני מלון.

מסנני טיסה –

- מספר עצירות
- סוג מושב
- מחיר מקסימלי

מסנני מלון –

- כמות חדרים
- כמות מבוגרים
- כמות ילדים
- מספר כוכבים מינימלי
- דירוג מלון מינימלי
- מחיר מקסימלי

בחרתי בפרויקט זה מכיוון שאני מאוד נהנית לטוס לחוץ לארץ, לחופשה כיפית ומהנה, אך תהליך החיפוש עבור החופשה האופטימלית יכול להיות ארוך ומייגע. לכן, הפרויקט שלי מקצר את התהליך והופך את החיפוש לקצר, חסכוני וממוקד יותר.

אתגרים שאני צופה שיהיו לי בפרויקט הם בעיקר בחלק של הלקוח, מאחר ובחרתי בתחום זר עבורי, אפליקציית אנדרואיד בשפת ג׳אווה.

הגדרת הלקוח –

מערכת זו מיועדת לכל אדם החפץ ומעוניין בנסיעה לחוץ לארץ, במחיר הטוב ביותר שניתן למצוא, לטיסה ומלון שהכי מתאימים ללקוח, לפי מסננים שהוא בוחר, בשביל חופשה חסכונית וכיפית.

הגדרת יעדים ומטרות –

המטרה העיקרית שלי בפרויקט היא ליצור מערכת של שרת-לקוח אשר מטפלת בלקוח ומציגה עבורו את הדיל של טיסה ומלון המשתלם ביותר עבורו. הלקוח הוא אפליקציית אנדרואיד אשר מעבירה לשרת את המסננים שהלקוח הזין עברו החופשה שלו, והשרת מחזיר שני קישורים, אחד עבור הטיסה הכי משתלמת וזולה, והשני עבור המלון המומלץ ביותר באתר הזמנת חופשות.

בעיות תועלות וחסכונות –

הבעיה שהפרויקט שלו נועד לפתור היא תהליך החיפוש הארוך ביותר והמייגע שהרבה אנשים נפגשים כאשר מחפשים עבורם חופשה. התועלת של פרויקט זה הוא קיצור משמעותי את תהליך החיפוש. לבסוף, המערכת מציגה עבור הלקוח, חופשה אופטימלית, והופכת את תהליך החיפוש לקצר, חסכוני וממוקד הרבה יותר.

סקירת פתרונות קיימים –

אתר google flights – https://www.google.com/travel/flights

אתר booking – https://www.booking.com/index.he.html

השוני בין המערכת המוצעת לבין אתרים אלו, היא שמערכת זו מציגה ללקוח דיל משתלם וחוסכת ממנו את החיפוש בעצמו. כמו כן, המערכת יעילה יותר מהאדם, היא בודקת את האופציות האפשריות ומשווה את המחירים, ולפי כך, חסכונית עבור הלקוח.

סקירת טכנולוגיית הפרויקט –

המערכת אינה טכנולוגיה חדשה ולא מוכרת. המערכת היא תוכנה שרת-לקוח אשר מייצרת קישורים ועושה web scraping לחילוץ פרטי הטיסה והמלון המשתלמים.

תיחום הפרויקט –

הפרויקט עוסק במימוש מערכת שרת-לקוח, כאשר הלקוח הוא אפליקציית אנדרואיד, והשרת הוא בפייתון. פרטי המשתמשים נשמרים במסד נתונים firebase של google. כמו כן, פרטי החיפוש, טיסות ומלון שהלקוח מזין, נשמרים במסד נתונים SQLite מוצפן בהצפנת rsa.

איפיון –

תיאור מפורט של המערכת –

המערכת היא תוכנת שרת-לקוח, שרת בפייתון, לקוח כאפליקציית אנדרואיד בג'אווה. התוכנה מבצעת רישום למערכת באפליקציה על ידי שם משתמש, אימייל וסיסמה ומכניסה את פרטיו למסד הנתונים firebase. לאחר מכן, מוצג עבור הלקוח, מסך הזנת פרטי חובה : מקום המראה, מקום נחיתה, תאריך המראה, תאריך נחיתה ומספר נוסעים. לחיצה על המשך תוביל להזנת פרטי רשות של העדפות טיסה : מספר עצירות, סוג מחלקה, מחיר מקסימלי עבור אדם, ושל העדפות מלון : כמות חדרים, כמות מבוגרים, כמות ילדים, כמות כוכבים מינימלית, דירוג מינימלי ומחיר מקסימלי. כעת, אחרי שהלקוח סיים להזין את כל הפרטים והעדפות שלו, נוצר חיבור socket עם השרת, והפרטים אלו מועברים לשרת. השרת מתחבר למסד נתונים SQLite ובו שלוש טבלאות : טבלה של חיפוש – פרטי החובה בשם search, טבלה של העדפות טיסה בשם flights וטבלה של העדפות מלון בשם hotels. כל המידע במסד נתונים זה נשמר מוצפן בהצפנת RSA. בעזרת המידע שהלקוח שלח, השרת יוצר שלושה קישורים, על ידי הצבת הפרטים בצורה המתאימה, אחד לטיסה, שני למלון ואחד נוסף לאטרקציות. שלושת הקישורים נשלחים ללקוח על ידי חיבור socket. האפליקציה בזמן אמת, מחלצת את שלושת הקישורים ומציגה באפליקציה שלושה כפתורים, אחד עם קישור לטיסה, השני עם קישור למלון והשלישי עם קישור לאטרקציות. לבסוף, המערכת מציגה עבור הלקוח באפליקציה את הדיל המשתלם ביותר לפי המסננים שהלקוח בחר.

פירוט יכולות המערכת –

כניסת משתמש קיים – מסך login, הרשמת משתמש חדש – מסך sign up.

הזנת פרטי חיפוש, טיסה ומלון – מסך search, flights ו – hotels.

מציאת הטיסה, המלון והאטרקציות על ידי יצירת קישורים מתאימים בצד השרת.

הגדרות –

Client-Server – ארכיטקטורת תוכנה אשר מגדירה את היחס בין תוכנות משתפות פעולה. המודל מחלק את המשימות או עומס העבודה בין ספק השירות או המשאבים – השרת, לבין מבקש השירות – הלקוח.

Database – דרך לאחסון נתונים במחשב בצורה שבה כל הנתונים מקושרים ביניהם, כך שניתן יהיה להשתמש בנתונים באופן יעיל. בסיס נתונים מאוחסן באמצעי אחסון נתונים, בדרך כלל על גבי דיסק קשיח, המאפשר גישה ישירה לנתונים.

הגישה לבסיס הנתונים נעשית באמצעות תוכנה ייעודית - מערכת לניהול בסיס נתונים. בסיס הנתונים בנוי לפי מודל לאחסון הנתונים, כמו מנגנונים פנימיים למיון ולחיפוש.

פירוט הבדיקות –

הכנסת מידע למסד הנתונים SQLite – הרצת צד השרת וצד הלקוח והכנסת הערכים למסד הנתונים לא מוצפנים, על מנת שאוכל לבדוק את תקינותם.

העברה של מידע מצד השרת לצד הלקוח ומצד הלקוח לצד השרת – בדיקות בזמן הרצת המערכת עם חיבור אינטרנט יציב.

בדיקות תקינות במסכי החיפוש באפליקציה – תקינות הערכים ובדיקה על ידי הרצה של נתונים שונים, נכונים ושגויים.

בדיקות תקינות בהרשמת משתמש חדש באפליקציה – תקינות הערכים ובדיקה על ידי הרצה של נתונים שונים, נכונים ושגויים לפי הקריטריונים.

תכנון וניהול לוח זמנים –

פעילות	**זמן התחלה מתוכנן**	**זמן סיום מתוכנן**	**זמן התחלה בפועל**	**זמן סיום בפועל**
צד לקוח – מסך כניסה, מסכי התחברות וחיבור למסד הנתונים	אמצע דצמבר	סוף ינואר	תחילת ינואר	סוף ינואר
צד לקוח – מסכי חיפוש, פרטי טיסות ופרטי מלון	סוף ינואר	אמצע פברואר	סוף ינואר	תחילת פברואר
חיבור שרת והעברת נתוני הלקוח לשרת	אמצע פברואר	תחילת מרץ	תחילת פברואר	סוף פברואר
צד שרת – יצירת קישורים מתאימים וחיבור למסד הנתונים	תחילת מרץ	סוף מרץ	סוף פברואר	אמצע מרץ
חיבור לקוח והעברת תוצאות השרת ללקוח	סוף מרץ	תחילת אפריל	אמצע מרץ	סוף מרץ
צד לקוח – הצגת התוצאות	תחילת אפריל	אמצע אפריל	סוף מרץ	תחילת אפריל

תיאור תחום הידע – פרק מילולי

צד הלקוח –

הרשמה למערכת –

כניסת משתמש קיים על ידי שם משתמש וסיסמה או הרשמת משתמש חדש למערכת על ידי שם משתמש, אימייל וסיסמה. ניתן לעבור בין המסכים. הרשמת משתמש חדש מובילה לאחר מכן לכניסת משתמש קיים עם שם המשתמש והסיסמה.

אוסף יכולות נדרשות :

ממשק משתמש – מסך כניסה למשתמש קיים ומסך הרשמה למשתמש חדש.

קליטת נתונים – שם משתמש וסיסמה (ואימייל למשתמש חדש).

בדיקת תקינות – תקינות הנתונים עם בסיס הנתונים firebase.

הצגת תשובה למשתמש – במידה ושם המשתמש / סיסמה שגויים מוצגת הודעה למשתמש.

הזנת פרטי חיפוש, טיסה ומלון –

לאחר כניסת המשתמש למערכת, מוצג עבורו שלושה מסכים, הראשון הוא מסך החיפוש, בו הוא מזין את פרטי החובה שהם : מקום המראה, מקום נחיתה, תאריך המראה, תאריך נחיתה וכמות נוסעים. לאחר מכן, על המשתמש אפשרות להזין העדפות טיסה : מספר עצירות, סוג מחלקה, מחיר מקסימלי לאדם, והעדפות מלון : כמות חדרים, מספר מבוגרים, מספר ילדים, כמות כוכבים מינימלית, דירוג מינימלי ומחיר מקסימלי.

אוסף יכולות נדרשות :

ממשק משתמש – מסכי הזנת פרטי החיפוש, העדפות טיסה והעדפות מלון.

קליטת נתונים – קליטת נתוני המשתמש.

בדיקת תקינות – חלק מהשדות הם רשימה של אפשרויות שניתן לבחור רק ממנה, בדיקת תקינות שמות, תאריכים ועוד.

שליחה לשרת – שליחת כל המידע לשרת באמצעות socket והכנסת המידע, מוצפן, למסד נתונים עם שלוש טבלאות.

הצגת תוצאה למשתמש –

לאחר סיום ביצוע השרת, הלקוח מקבל שלושה קישורים, אחד של הטיסה לפי העדפות המשתמש, השני של המלון לפי העדפות המשתמש והשלישי של אטרקציות ליעד הנבחר.

אוסף יכולות נדרשות :

ממשק משתמש – מסך הצגת התוצאות, ובו שני כפתורים על מנת לפתוח את הקישורים המתאימים שהתקבלו מהשרת.

קבלה מהשרת – קבלת שלושת הקישורים שהשרת יצר דרך שליחת המידע באמצעות socket.

צד השרת –

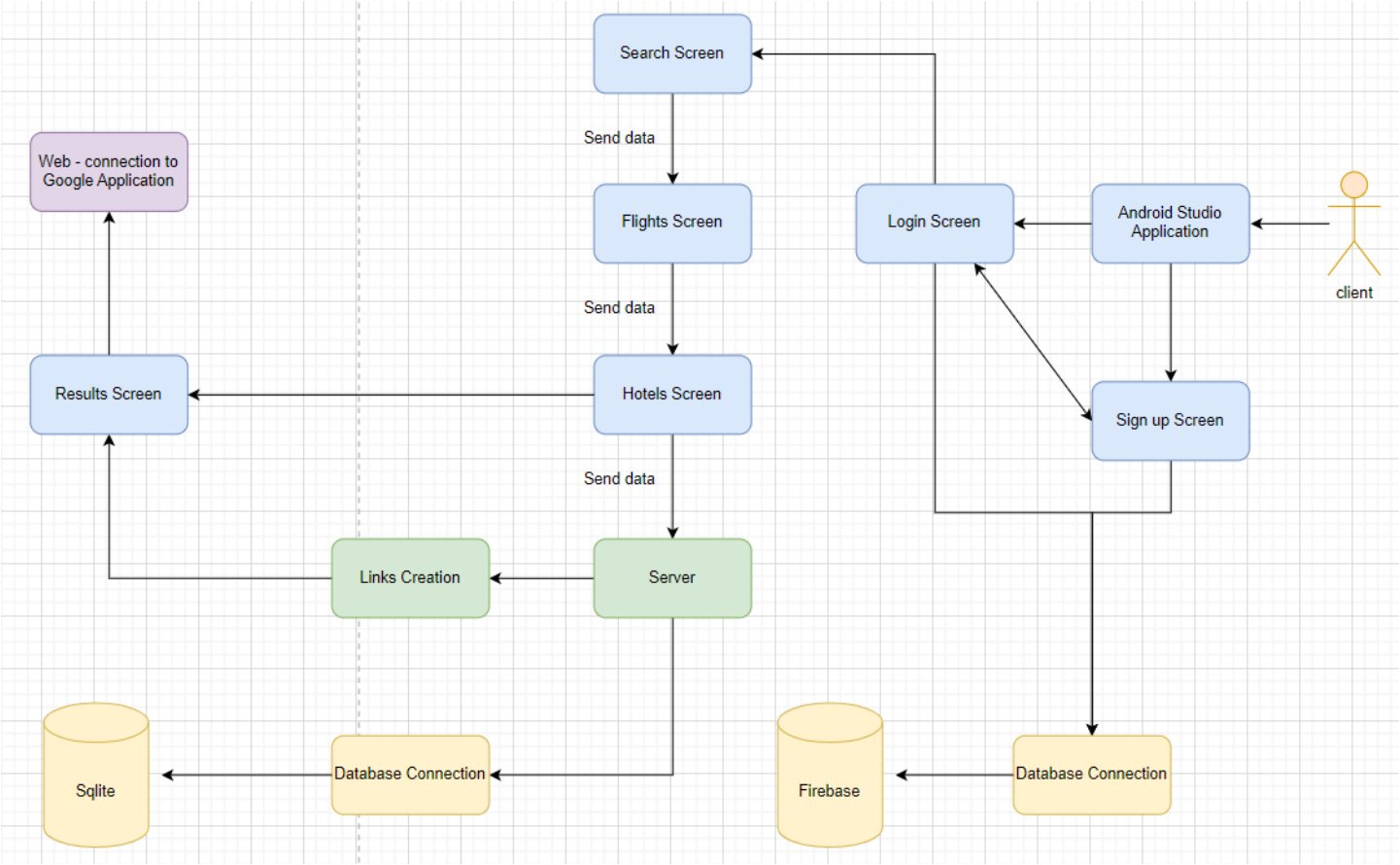
מציאת הטיסה והמלון המתאימים –

עיבוד הנתונים – קליטת הנתונים מלקוח, ועיבוד הנתונים ויצירת קישור לטיסה וקישור למלון על ידי הצבת המידע ב – url של אתר חיפוש kayak. בנוסף, המרה ושינוי של חלק מהמידע כך שיתאים לקישור.

הצפנה – הכנסת המידע שהתקבל מהלקוח למסד נתונים SQLite עם שלוש טבלאות, כאשר כל המידע שמוכנס הוא מוצפן בהצפנת RSA.

מבנה / ארכיטקטורה של הפרויקט

זרימת המידע במערכת –



תיאור סביבת הפיתוח –

צד לקוח – סביבת פיתוח Android Studio בשפת Java.

צד שרת – סביבת פיתוח PyCharm בשפת python 3.9.

תיאור פרוטוקול התקשורת –

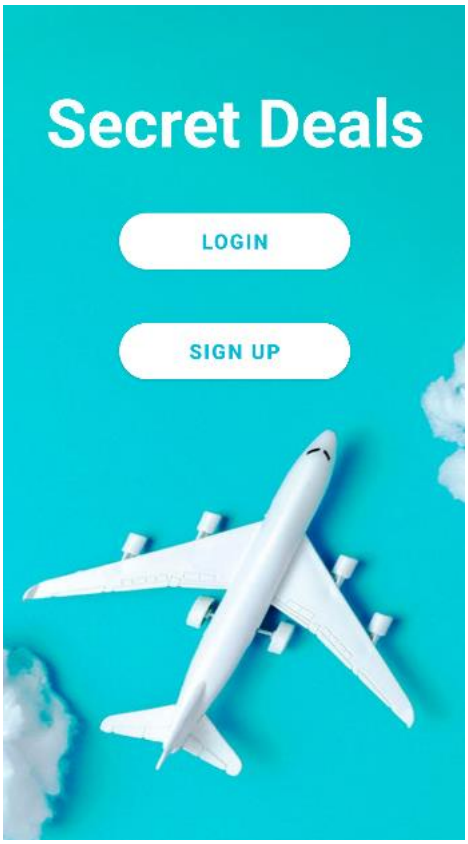
בפרויקט השתמשתי בפרוטוקול התקשורת TCP – Transmission Control Protocol. זהו פרוטוקול בתקשורת נתונים הפועל בשכבות התעבורה של מודל ה-OSI ובמודל ה-TCP/IP, ומבטיח העברה אמינה של נתונים בין שתי תחנות ברשת מחשבים באמצעות יצירת חיבור מקושר.

הודעה מלקוח לשרת – איסוף כל המידע בנוגע לפרטי החיפוש, הטיסה והמלון ושליחתם לשרת באמצעות socket. כל שדה מופרד על ידי רווח.

הודעה מהשרת ללקוח – שליחת הקישורים שהשרת יצר, ללקוח באמצעות socket. כל שדה מופרד על ידי רווח. קישור לטיסה, קישור למלון וקישור לאטרקציות.

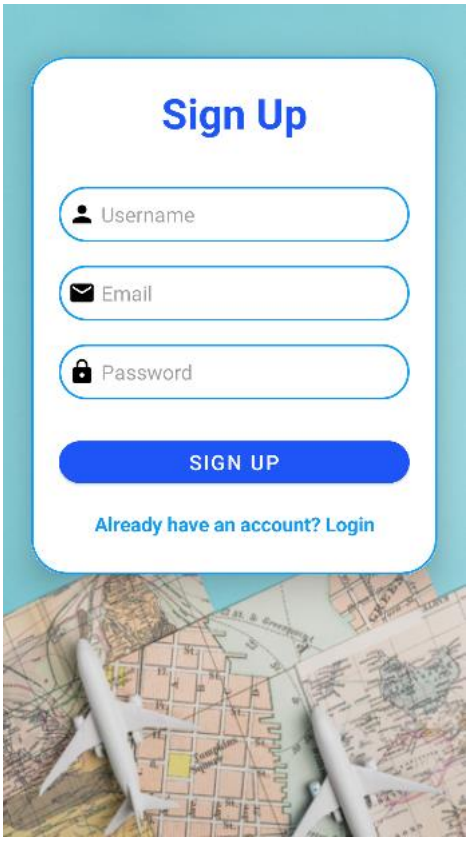
תיאור מסכי המערכת –

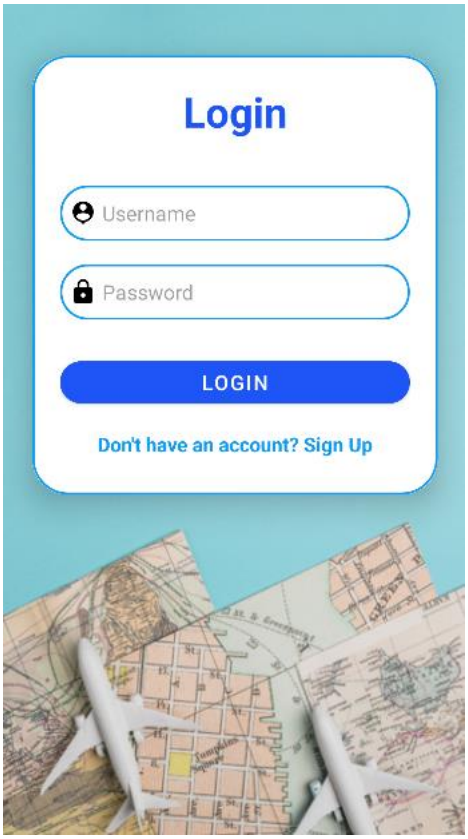
מסך כניסה :



מסך הכניסה לאפליקציה המכיל תמונת רקע, כותרת, ושני כפתורים המובילים להרשמה. אחד מוביל לכניסת משתמש קיים – Login. השני מוביל להרשמת משתמש קיים – Sign up.

מסכי הרשמה :





שני מסכי הרשמה, אחד הוא כניסה למשתמש קיים – Login המבקש שם משתמש וסיסמה מתאימים. הסיסמה מוסתרת ומופיעה כעיגולים שחורים. הבדיקה נעשית על ידי כניסה למסד הנתונים Firebase בשם users בו שמורים פרטי הלקוחות. לחיצה על הכפתור Login יוביל לכניסה לאפליקציה, למסך החיפוש.

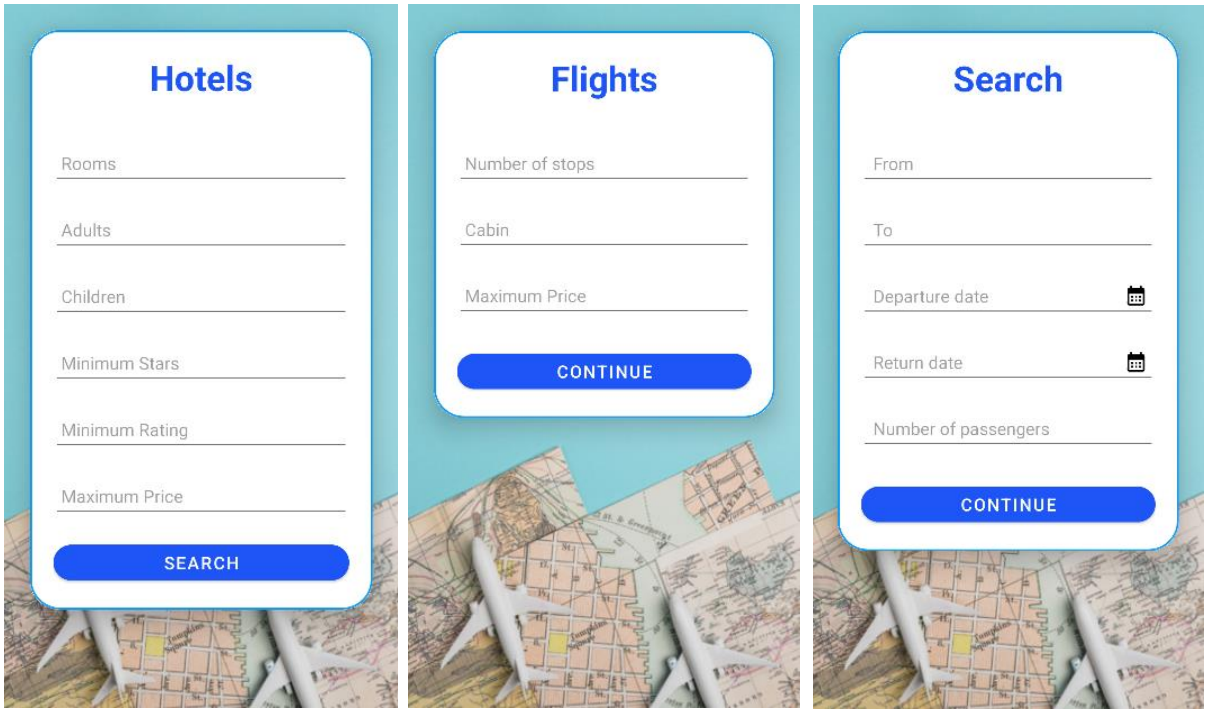
מסך ההרשמה השני הוא הרשמת משתמש חדש – Sign Up המבקש שם משתמש, אימייל וסיסמה. שם המשתמש צריך להיות בין 4-15 תווים, מכיל לפחות אות גדולה אחת, אות קטנה אחת וללא רווחים. כמו כן, האימייל צריך להיות תקין והסיסמה צריכה להיות בין 8-20 תווים, מכילה לפחות ספרה אחת, אות אחת גדולה, אות אחת קטנה וללא רווחים. במידה והפרטים תקינים, פרטיו של הלקוח מתווספים למסד הנתונים Firebase בשם users. לחיצה על הכפתור Sign Up יוביל למסך ההתחברות Login.

ניתן לעבור בין שני המסכים על ידי לחיצה על השורה מתחת להזנת הפרטים.

במסך Login – Don't have an account? Sign Up.

במסך Sign Up – Already have an account? Login.

מסכי חיפוש :



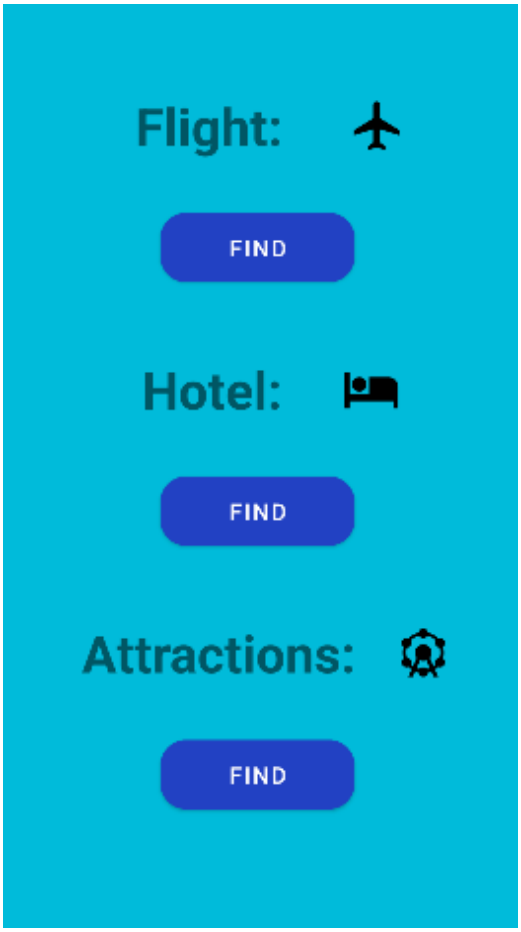
שלושה מסכי חיפוש.

המסך הראשון הוא חיפוש כללי Search, כל פרטיו הם פרטי חובה. יעד המוצא ויעד הנחיתה הם מתוך מאגר שמות שמור של ערים. לא ניתן לכתוב דבר שאינו במאגר. תאריך ההמראה ותאריך הנחיתה נבחרים מתוך לוח השנה שנפתח בלחיצה על תיבת הטקסט. תאריך ההלוך לפני תאריך החזור. מספר הנוסעים הוא מספר בלבד. במידה ואחד הפרטים לפחות שגוי, מופיעה הודעת שגיאה במקום השגוי. במידה וכל הפרטים מתאימים, לחיצה על הכפתור Continue תוביל למסך הבא של חיפוש הטיסה.

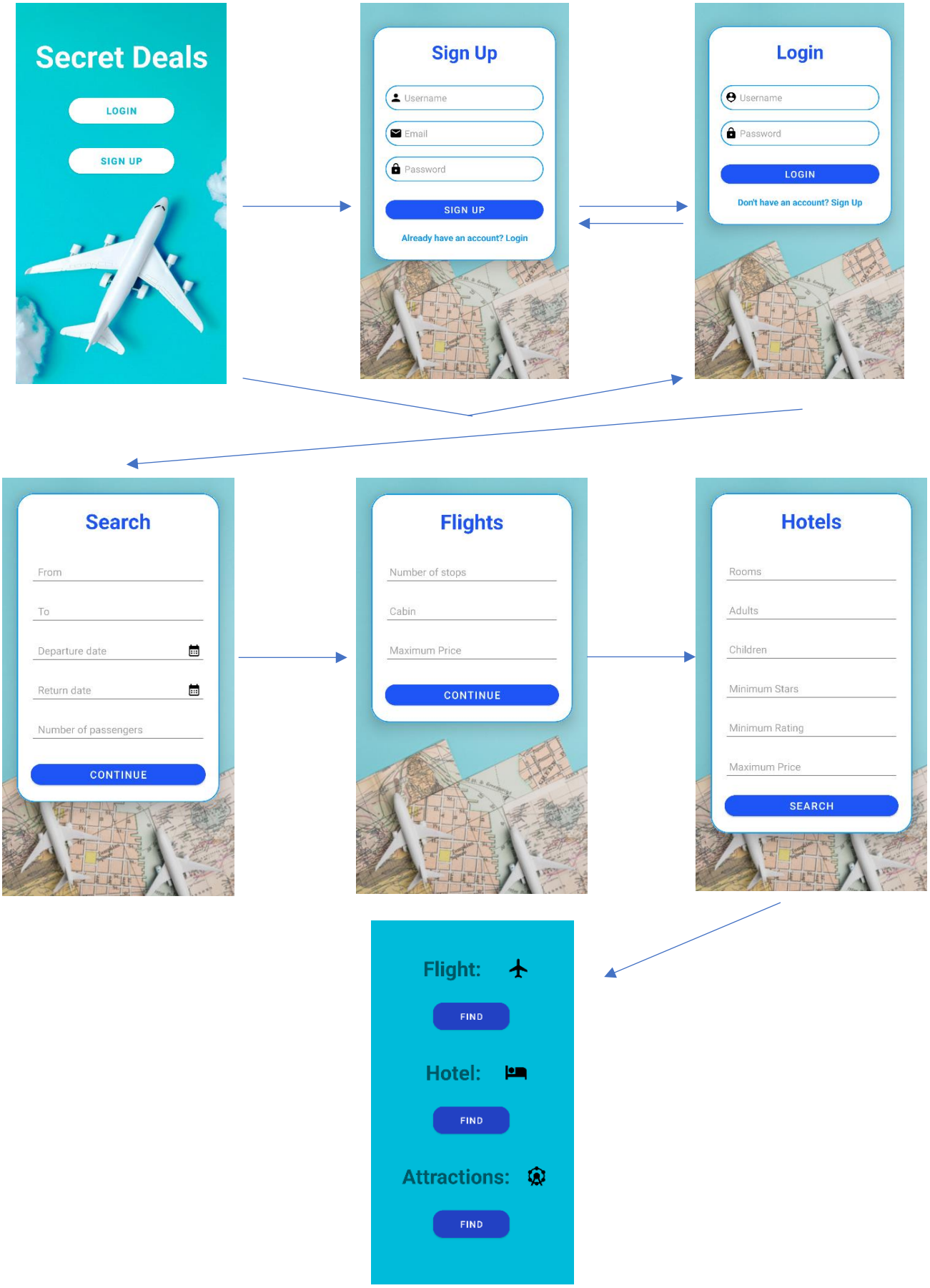
המסך השני הוא חיפוש הטיסה Flights, פרטיו הם רשות. מספר העצירות זה 0, 1 או 2+. סוג מחלקה זה Economy, Premium Economy, Business ו – First. מחיר מקסימלי הוא מספר בלבד. הודעת שגיאה תופיע אם יירשם דבר שאינו ברשימה. במידה וכל הפרטים מתאימים, לחיצה על Continue תוביל למסך הבא של חיפוש מלון.

המסך השלישי הוא חיפוש מלון Hotels, פרטיו הם רשות. מספר החדרים, מספר מבוגרים ומספר הילדים הם מספרים בלבד. כמות כוכבים מינימלית זה 2, 3, 4, 5. דירוג מינימלי זה 6, 7, 8, 9. מחיר מקסימלי זה מספר בלבד. הודעת שגיאה תופיע אם יירשם דבר שאינו ברשימה. במידה וכל הפרטים מתאימים, לחיצה על Search תוביל למסך הבא של התוצאות.

מסך תוצאות :



מסך תוצאות המקבל מהשרת שלושה קישורים, אחד לטיסה, השני למלון והשלישי לאטרקציות. לחיצה על כפתור Find מתחת לכותרת Flight פותחת את דפדפן google chrome עם הקישור המתאים לטיסה. לחיצה על כפתור Find מתחת לכותרת Hotel פותחת את דפדפן google chrome עם הקישור המתאים למלון. לחיצה על כפתור Find מתחת לכותרת Attractions פותחת את דפדפן google chrome עם הקישור המתאים לאטרקציות והסבר על העיר.


Secret Deals
LOGIN
SIGN UP
Sign Up
Username
Email
Password
SIGN UP
Already have an account? Login
Login
Username
Password
LOGIN
Don't have an account? Sign Up
Search
From
To
Departure date
Return date
Number of passengers
CONTINUE
Flights
Number of stops
Cabin
Maximum Price
CONTINUE
Hotels
Rooms
Adults
Children
Minimum Stars
Minimum Rating
Maximum Price
SEARCH
Flight:
FIND
Hotel:
FIND
Attractions:
FIND

תיאור מבני הנתונים :

מסד נתונים deals מסוג SQLite אשר כולל שלוש טבלאות :

SEARCH – פרטי החיפוש, פרטי חובה, כולם מחייבים ערך :

from_des – סוג טקסט.

to_des – סוג טקסט.

departure_date – סוג טקסט.

return_date – סוג טקסט.

passengers – סוג מספר שלם.

FLIGHTS – פרטי הטיסה, פרטי רשות, לא מחייבים ערך :

stops – סוג מספר שלם.

cabin – סוג טקסט.

price – סוג מספר שלם.

HOTELS – פרטי המלון, פרטי רשות, לא מחייבים ערך :

rooms – סוג מספר שלם.

adults – סוג מספר שלם.

children – סוג מספר שלם.

stars – סוג מספר שלם.

ranking – סוג מספר שלם.

price – סוג מספר שלם.

כל הערכים מוצפנים בהצפנת RSA.

דוגמה לערכים אפשריים :

SEARCH (12 rows)

```
SELECT * FROM 'SEARCH' LIMIT 0,30
```

Execute

from_des	to_des	departure_date	return_date	passengers
90,118,143,214,207,185,77,93,...	61,99,129,80,42,6,108,212,180...	71,142,192,240,135,128,51,42,...	134,66,158,221,219,189,229,1...	115,9,184,88,239,134,210,67,0...
105,181,49,183,174,255,128,2...	22,35,235,205,87,132,105,193,...	144,237,155,192,217,67,241,1...	0,242,22,63,16,195,122,41,132...	55,9,212,239,219,179,105,123,...
2,4,89,3,129,128,139,83,77,23...	110,135,180,134,82,117,121,13...	91,18,89,159,114,149,253,208,...	67,169,248,136,159,59,23,113,...	123,24,35,9,40,63,27,240,163,...
123,220,92,232,145,153,101,9...	100,128,124,66,142,164,190,2...	87,19,112,39,38,145,188,247,8...	123,40,245,207,177,227,96,67,...	6,158,72,82,91,105,154,47,72,...
139,224,26,31,226,78,46,178,9...	110,153,0,178,150,142,24,211,...	55,219,98,69,13,181,55,198,80...	16,204,196,111,242,58,201,242...	124,148,28,161,196,244,113,1...
125,210,8,55,67,197,9,61,110,...	146,100,13,233,170,46,254,14,...	103,225,53,93,236,136,106,11...	75,78,152,113,68,76,60,84,87,...	50,203,84,221,112,249,75,78,1...
71,120,249,57,188,88,71,125,1...	54,176,226,167,211,188,145,1...	24,2,1,42,13,149,25,35,83,180,...	59,242,42,141,31,47,114,249,2...	0,62,152,125,42,85,231,14,113...
161,166,161,234,1,134,18,117,...	67,242,102,185,211,141,175,2...	36,103,213,159,171,36,237,45,...	140,42,11,246,5,95,5,68,95,34,...	63,66,134,26,209,235,107,132,...
163,214,226,136,107,76,145,8...	34,210,39,162,143,209,144,14...	42,16,197,50,255,94,178,197,2...	19,89,200,168,138,87,29,189,1...	108,18,201,123,82,235,83,108,...
107,56,107,202,51,150,176,22...	31,190,202,155,65,170,43,219,...	140,103,80,243,24,219,11,140,...	189,60,58,158,234,220,92,169,...	134,40,48,4,197,35,101,190,11...
29,200,26,183,237,208,98,74,1...	24,84,163,196,231,133,124,68,...	32,72,132,73,114,77,24,193,83...	11,249,1,245,135,217,117,115,...	184,51,246,193,73,87,149,149,...
24,185,192,87,184,171,46,142,...	27,24,12,165,20,157,57,227,1,...	82,109,183,77,3,62,32,9,174,2...	6,221,21,241,254,132,84,133,2...	30,129,94,143,81,51,143,216,4...

SQLite הוא בסיס נתונים יחסי משובץ. כלומר : להבדיל מרוב בסיסי הנתונים היחסיים, SQLite אינו תהליך עצמאי נפרד המקבל קריאות מתהליכים נפרדים או מרוחקים, אלא ספריה הנקראת או מופעלת מתוך תהליך קיים.

מסד נתונים users מסוג Realtime Database – Firebase אשר כולל את השדות :

username – שם משתמש, מפתח ראשי, סוג string.

password – סיסמה, סוג string, מוצפנת בהצפנת hash.

email – כתובת אימייל, סוג string.

דוגמה לערכים אפשריים :

```
users
  Eden
    email: "edendonenfeld@gmail.com"
    password: "a5b7188ea66b929fda1d977dad53f61684a6f36e03d58c39e8e22cbfdffdcc83"
    username: "Eden"
  Eden21
```

Google Firebase היא תוכנת פיתוח יישומים הנתמכת על ידי Google, המאפשרת למפתחים לפתח יישומי iOS, אנדרואיד ואינטרנט. Firebase מספקת כלים למעקב אחר ניתוח נתונים, דיווח ותיקון קריסות אפליקציה, יצירת ניסוי שיווק ומוצר.

מסד נתונים בזמן אמת – מסד הנתונים של Firebase Realtime הוא מסד נתונים NoSQL המתארח בענן המאפשר אחסון וסנכרון נתונים בין משתמשים בזמן אמת. הנתונים מסונכרנים בין כל הלקוחות בזמן אמת ועדיין זמינים כאשר אפליקציה לא מקוונת.

תיאור האלגוריתמים המרכזיים :

מנוע חיפוש הטיסות, מלונות והאטרקציות :

הבעיה האלגוריתמית היא בהינתן פרטי החיפוש של המשתמש עבור חופשה, מה הטיסות והמלונות המומלצים עבורו לפי הנתונים שהזין. בנוסף לכך, הפניה לאטרקציות במקום הנחיתה.

אלגוריתם חיפוש הטיסות, מלונות והאטרקציות נעשה על ידי שימוש באתרים, והכנסת פרמטרים ל – url. השרת מקבל את כל המידע שהלקוח הזין בנוגע לחופשה שלו, ויוצר קישורים מתאימים לפי מה שהזין. אם הזין אם חלק מהנתונים, כל הנתונים וגם שום נתון. אחרי שתהליך יצירת הקישורים מסתיים, נשלחים שלושת הקישורים ללקוח, בו הם מוצגים על ידי לחיצת כפתור ופתיחת דפדפן google chrome עם הקישור המתאים לטיסה, מלון ואטרקציות ליעד.

בפרק מימוש הפרויקט פירטתי על האלגוריתמים והוספתי תדפיס קוד בהתאם.

סקירת חולשות ואיומים :

מסד הנתונים של הפרטים שהלקוח הזין על החופשה שלו מוצפנים בהצפנה אסימטרית RSA. בהצפנה א-סימטרית משתמשים בשני מפתחות – מפתח ציבורי ומפתח פרטי. המפתח הציבורי חשוף לכולם, והוא המפתח שיוצר את ההצפנה. על מנת לפענח את ההצפנה יש להיעזר במפתח הפרטי, שנמצא רק אצל המקבל.

בהרשמת משתמש חדש למערכת, הסיסמה מוכנסת למסד הנתונים Firebase מוצפנת בהצפנת hash, SHA-256.

כמו כן, שימוש בפרוטוקול אמין TCP, בדיקת אמינות הנתונים נעשה באמצעות חישוב Checksum. מחשב המקור מחשב פונקציה על המידע הנשלח ומוסיף אותו כחלק מהפתיח של חבילת ה-TCP. כאשר מחשב היעד מקבל את החבילה הוא משווה את תוכן הפתיח עם תוצאת חישוב חדשה של הפונקציה שהוא מבצע. לאחר אימות הנתונים שולח מחשב היעד אישור למקור על מנת להודיע לו שהחבילה הגיעה באופן תקין. אם הנתונים הגיעו בצורה לא תקינה מבקש מחשב היעד שידור חוזר של החבילה.

הפרוטוקול מחייב את המקור בתהליך הנקרא "לחיצת ידיים משולשת", התהליך הוא בעל שלושה שלבים ומתבטא בצורה הבאה:

1. המקור שולח הודעת SYN (סנכרון) כדי להודיע על רצונו לפתיחת התקשורת מול היעד.
2. במידה והיעד זמין ומחובר ויש ביכולתו לפתוח בשיחה הוא ישלח אל המקור הודעת SYN-ACK (אישור סנכרון) ובכך יודיע על זמינותו למקור.
3. המקור מקבל את הודעת התשובה מהיעד ושולח לו הודעת ACK (אישור), וכעת שני הצדדים פתחו בשיחה ומסוגלים להעביר נתונים.

מימוש הפרויקט

חלק א' –

צד הלקוח – אפליקציית אנדרואיד –

MainActivity.java – מסך הכניסה לאפליקציה.

תכונות :

כפתור התחברות למשתמש חדש signupButton. וכפתור התחברות למשתמש קיים loginButton.

פעולות :

onClick – ברגע שנלחץ הכפתור, הפעולה מתבצעת ומעבירה למסך הרלוונטי. הפעולה מקבלת עצם מסוג View ואינה מחזירה ערך.

SignupActivity.java – מסך התחברות למשתמש חדש.

תכונות :

תיבת טקסט לשם משתמש signupUsername, לאימייל signupEmail ולסיסמה signupPassword. טקסט לניתוב למסך התחברות למשתמש קיים loginRedirectText. כפתור signupButton. עצם מסוג FirebaseDatabase ועצם מסוג DatabaseReference.

פעולות :

checkUser – פעולה המקבלת שלושה מחרוזות – שם משתמש, אימייל וסיסמה. הפעולה בודקת את תקינות הערכים שהזין הלקוח בעת ההרשמה. אם קיימת בעיה, מוציאה הערת שגיאה עם הסבר והפעולה מחזירה את הערך false. אחרת, אם אין בעיות הפעולה מחזירה את הערך true.

isUpper, isLower, isDigit – פעולות עזר אשר מקבלות מחרוזת ומחזירות ערך true אם קיים אות גדולה, אות קטנה או ספרה, ומחזירות ערך false אם לא קיים.

hashPassword – פעולה המקבלת מחרוזת סיסמה ומחזירה את הסיסמה מוצפנת בהצפנת SHA-256.

LoginActivity.java – מסך התחברות למשתמש קיים.

תכונות :

תיבת טקסט לשם משתמש loginUsername ולסיסמה loginPassword. כפתור loginButton וטקסט לניתוב למסך התחברות למשתמש חדש signupRedirectText.

פעולות :

validateUsername – פעולה לא מקבלת ערך. בודקת את תקינות שם המשתמש, במידה ויש בעיה מוציאה התראת שגיאה עם הסבר ומחזירה את הערך false. אחרת, אם אין בעיות הפעולה מחזירה את הערך true.

validatePassword – פעולה לא מקבלת ערך. בודקת את תקינות שם המשתמש, במידה ויש בעיה מוציאה התראת שגיאה עם הסבר ומחזירה את הערך false. אחרת, אם אין בעיות הפעולה מחזירה את הערך true.

checkUser – פעולה לא מקבלת ערך. בודקת את תקינות המידע בעזרת הפעולות הקודמות ומכניסה את הפרטים למסד הנתונים. לא מחזירה ערך.

hashPassword – פעולה המקבלת מחרוזת סיסמה ומחזירה את הסיסמה מוצפנת בהצפנת SHA-256.

SearchActivity.java – מסך חיפוש.

תכונות :

תיבות טקסט למקום המראה from, מקום נחיתה to, תאריך המראה departureDate, תאריך נחיתה returnDate וכמות נוסעים numberOfPassengers. כמו כן, כפתור searchFlights, מחרוזת message ומערך של מחרוזות cities.

פעולות :

validateSearch – פעולה המקבלת חמישה מחרוזות, fromS, toS, detS, retS ו – passengersS. הפעולה בודקת את תקינות כל הפרטים. במידה ויש בעיה מוציאה הערת שגיאה עם הסבר ומחזירה את הערך false. אחרת, אם אין בעיות, מחזירה את הערך true.

showDatePickerDialogDet – פעולה לא מקבלת ערך. יוצרת עצם מסוג DatePickerDialog לתאריך ההמראה. הפעולה לא מחזירה ערך.

showDatePickerDialogRet – פעולה לא מקבלת ערך. יוצרת עצם מסוג DatePickerDialog לתאריך הנחיתה. הפעולה לא מחזירה ערך.

מחלקה :

InnerFirstDate – פעולת onDataSet מקבלת יום, חודש ושנה ומציבה לפי פורמט בתיבת הטקסט של תאריך ההמראה.

InnerSecondDate – פעולת onDataSet מקבלת יום, חודש ושנה ומציבה לפי פורמט בתיבת הטקסט של תאריך הנחיתה.

FlightsActivity.java – מסך חיפוש טיסות.

תכונות :

מחרוזת message, תיבות טקסט למספר עצירות stops, סוג מחלקה cabin ומחיר מקסימלי price. כפתור searchFlights. כמו כן, מערך מחרוזות numberOfStops ומערך מחרוזות cabinTypes.

פעולות :

validateFlights – פעולה המקבלת מחרוזת של מספר העצירות number_stops ומחרוזת של סוג מחלקה type_cabin. הפעולה בודקת את תקינות המידע. במידה ויש בעיה, מוציאה הערת שגיאה עם הודעה מתאימה ומחזירה את הערך false. אחרת, אם אין בעיות, מחזירה את הערך true.

HotelsActivity.java – מסך חיפוש מלונות.

תכונות :

מחרוזת message, תיבות טקסט כמות חדרים rooms, כמות מבוגרים adults, כמות ילדים children, כמות כוכבים stars, דירוג rating ומחיר מקסימלי price. כפתור searchHotels. כמו כן, מערך מחרוזות minimumStars ומערך מחרוזות minimumRating.

פעולות :

validateHotels – פעולה המקבלת מחרוזת של מספר הכוכבים number_stars ומחרוזת של דירוג number_rating. הפעולה בודקת את תקינות המידע. במידה ויש בעיה, מוציאה הערת שגיאה עם הודעה מתאימה ומחזירה את הערך false. אם אין בעיות, מחזירה את הערך true.

מחלקה :

Send – פעולת doInBackground יוצרת חיבור Socket בפורט 8000 ושולחת את המידע שנאסף במסך החיפוש, טיסות ומלון אל השרת, במצב אסינכרוני.

ResultActivity.java – מסך התוצאות.

תכונות :

כפתורים flightsLink, hotelsLink ו – attrLink. טקסט attrText. מחרוזות data, linkFlights, linkHotels ו – linkAttractions.

מחלקה :

Get – פעולת doInBackground יוצרת קישור Socket בפורט 1234 ומאזינה ומקבלת מידע מהשרת, קישורים לטיסות, מלון ואטרקציות.

HelperClass.java – מחלקת עזר.

תכונות :

מחרוזת username, מחרוזת email ומחרוזת password.

פעולות :

פעולות Get ו – Set לכל אחת מן התכונות.

צד השרת – שרת בפייתון –

city.py – קובץ המכיל מילון, כאשר המפתח שווה לשם העיר ושם המדינה, והערך שווה לקישור 3 האותיות של העיר. לדוגמה : AMS : Amsterdam, Netherlands. כמו כן, מכילה רשימה של ערים שאתר האטרקציות תומך בהן לשם בדיקה בקובץ השרת. בקובץ זה אין פעולות.

data.py – קובץ המטפל במסד הנתונים SQLite. הוא מתחבר אל מסד הנתונים deals.db בתיקייה הנוכחית. אם מסד הנתונים קיים, הוא מוחק את הערכים בו. אם מסד הנתונים לא קיים, הוא יוצר שלוש טבלאות חדשות SEARCH, FLIGHTS, HOTELS. בקובץ זה אין פעולות.

server.py – קובץ השרת. ראשית, יוצר socket ומאזין לתקשורת בפורט 8000. לאחר מכן, כאשר התקבל קישור והתקבלה ההודעה, מפרק את ההודעה ל – 3 רשימות. אחת לחיפוש search, השנייה לטיסות flights והשלישית למלונות hotels. כעת, השרת יוצר את הקישור לטיסות ומלונות על ידי הזרקת הפרמטרים ב – url של אתר www.il.kayak.com. כמו כן, נוצר קישור לאתר לאטרקציות על ידי שימוש ב – url של אתר www.aviewoncities.com. אם מקום הנחיתה אינו נמצא ברשימה מתוך קובץ city.py, נוצר קישור של אתר www.wikipedia.com על העיר. הקישורים שנוצרו נשלחים ללקוח, על ידי יצירת socket אשר מאזין לתקשורת ושולח את המידע. כל הנתונים של החיפוש, טיסות ומלון נכנסים למסד הנתונים deals.db מוצפנים בהצפנה א-סימטרית RSA. בהצפנה א-סימטרית משתמשים בשני מפתחות – מפתח ציבורי ומפתח פרטי. המפתח הציבורי חשוף לכולם, והוא המפתח שיוצר את ההצפנה. על מנת לפענח את ההצפנה יש להיעזר במפתח הפרטי, שנמצא רק אצל המקבל.

משתנים :

sock – משתנה מסוג socket.

port – מספר פורט להאזנת מידע מהלקוח.

IP – מחרוזת של כתובת ה – IP של השרת.

publicKey, privateKey – מפתח ציבורי ומפתח פרטי שנוצרו על ידי הצפנת RSA.

URL – מחרוזת המחזיקה את הקישור.

פעולות :

socket_get – פעולה אינה מקבלת ערך. היא יוצרת חיבור socket עם IP '0.0.0.0' בפורט 8000. כאשר מתבצע החיבור, המידע שנשלח מהלקוח לשרת נשמר ברשימה data וזה הערך שמוחזר.

data_info – פעולה מקבלת רשימה data של המידע שהתקבל מהלקוח. היא מחלקת את המידע לשלוש רשימות search, flights ו – hotels. בנוסף, הפעולה ממירה את השם של העיר לקיצור 3 אותיות לפי המילון שבקובץ city.py. הפעולה מחזירה את שלוש הרשימות האלו.

url_making_flights – פעולה המקבלת שתי רשימות search ו – flights, ויוצרת את הקישור לאתר הטיסות על ידי הזרקת פרמטרים ל – url של אתר www.il.kayak.com. הפעולה מחזירה את הקישור הזה כמחרוזת.

get_cities – פעולה המקבלת רשימה search, ויוצרת שתי מחרוזות, אחת של שם העיר כפי שצריך להיות בקישור של המלונות, והשנייה שם העיר כפי שצריך להיות בקישור של האטרקציות. הפעולה מחזירה את שתי המחרוזות האלו.

hotel_rating – פעולת עזר המקבלת מחרוזת rating, וממירה את מספר הדירוג למילה, לדוגמה : דירוג "7" שווה ל – good. הפעולה מחזירה את המחרוזת הזו.

url_making_hotels – פעולה המקבלת שתי רשימות search ו – hotels ומחרוזת של שם העיר hotel_city כפי שהוחזר מהשיטה get_cities. הפעולה יוצרת את הקישור לאתר המלונות על ידי הזרקת פרמטרים ל – url של אתר www.il.kayak.com. הפעולה מחזירה את הקישור הזה כמחרוזת.

url_making_attractions – פעולה המקבלת מחרוזת attr_city כפי שהוחזר מהשיטה get_cities. היא בודקת אם העיר נמצאת ברשימת הערים של האתר מתוך הרשימה בקובץ city.py, אם כן, יוצרת קישור לאתר www.aviewoncities.com עם מקום הנחיתה בקישור. אחרת, העיר לא נמצאת ברשימת הערים של האתר, נוצר קישור של אתר ויקיפדיה www.wikipedia.com על מקום הנחיתה. הפעולה מחזירה את הקישור שנוצר כמחרוזת.

socket_send – פעולה המקבלת שלוש מחרוזת של שלושת הקישורים url_flights, url_hotels ו – url_attractions. הפעולה יוצרת חיבור socket עם השרת בפורט 1234 ושולחת את שלושת הקישורים האלו לשרת. הפעולה אינה מחזירה ערך.

database_insertion – פעולה המקבלת את שלוש הרשימות search, flights ו – hotels. הפעולה מכניסה למסד הנתונים את הערכים לפי הרשימות. את הרשימה search לטבלת SEARCH, את הרשימה flights לטבלת FLIGHTS ואת הרשימה hotels לטבלת HOTELS.

main – הפעולה הראשית, בה מורצות כל הפעולות. אינה מקבלת ערך ואינה מחזירה ערך.

רצף קריאת הפעולות בשיטה הראשית – main :

```
def main():
    data = socket_get()
    search, flights, hotels = data_info(data)
    url_flights = url_making_flights(search, flights)
    hotel_city, attr_city = get_cities(search)
    url_hotels = url_making_hotels(search, hotels, hotel_city)
    url_attraction = url_making_attractions(attr_city)
    print("URL FLIGHTS: " + url_flights)
    print("URL HOTELS: " + url_hotels)
    print("URL ATTRACTIONS: " + url_attraction)
    socket_send(url_flights, url_hotels, url_attraction)
    database_insertion(search, flights, hotels)
```

חלק ב' –

מנוע חיפוש הטיסות, מלונות והאטרקציות :

צירפתי קטעי קוד מיוחדים.

ב – SearchActivity מתבצע חילוץ הפרטים והעברתם למסך הבא על ידי Intent.

```
searchFlights.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        String fromText = from.getText().toString().trim().replaceAll( regex: "\\s",  replacement: "");
        String toText = to.getText().toString().trim().replaceAll( regex: "\\s",  replacement: "");
        String depDateText = departureDate.getText().toString().trim();
        String retDateText = returnDate.getText().toString().trim();
        String numberPassengersText = numberOfPassengers.getText().toString().trim();

        if (validateSearch(fromText, toText, depDateText, retDateText, numberPassengersText)) {
            message = fromText + " " + toText + " " + depDateText + " " + retDateText + " " + numberPassengersText + " ";
            System.out.println("Yay!");
            Intent intent = new Intent( packageContext: SearchActivity.this, FlightsActivity.class);
            intent.putExtra( name: "MESSAGE", message);
            startActivity(intent);
        }
    }
});
```

לאחר מכן, העברת הפרטים על ידי Intent מ – FlightsActivity למסך הבא.

```
searchFlights.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        String numberOfStops = stops.getText().toString().trim();
        String cabinFlights = cabin.getText().toString().trim();
        String priceFlights = price.getText().toString().trim();

        if (validateFlights(numberOfStops, cabinFlights)) {
            Intent intent = getIntent();
            message = intent.getStringExtra( name: "MESSAGE");
            message += numberOfStops + " " + cabinFlights + " " + priceFlights + " ";
            Intent intent_2 = new Intent( packageContext: FlightsActivity.this, HotelsActivity.class);
            intent_2.putExtra( name: "MESSAGE2", message);
            startActivity(intent_2);
            System.out.println("Yay!");
        }
    }
});
```

כאשר מגיעים ל – HotelsActivity, כל המידע של חיפוש הטיסות והמלון נאסף ונשלח לשרת על ידי חיבור Socket בפורט 8000, במצב א-סינכרוני.

```
public class Send extends AsyncTask<Void,Void,Void> {
    Socket s;
    PrintWriter pw;
    protected Void doInBackground(Void...params) {
        try {
            s = new Socket( host: "172.20.10.2",  port: 8000);
            pw = new PrintWriter(s.getOutputStream());
            pw.write(message);
            pw.flush();
            pw.close();
            s.close();
        } catch (UnknownHostException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
        return null;
    }
}
```

כעת, הנתונים נשלחו לשרת על ידי חיבור Socket עם אותו פורט 8000.

```
"""
create a connection to the client, and receives the data
:return: data: list of all the user's input
"""
sock.bind((IP, port))
sock.listen(maxConnections)
print("Server started at " + IP + " on port " + str(port))

(client, address) = sock.accept()
print("New connection made!")

# Listing for message
running = True
data = []
while running:
    message = client.recv(1024).decode()
    if message != '':
        print(message)
        data = message.split(' ')
        # from to dep ret passengers stops cabin price rooms adults children stars rating price
        break
client.close()
sock.close()
# end of socket
return data
```

השרת יוצר קישור לטיסות וקישור למלון על ידי שימוש באתר www.il.kayak.com, והזרקת הפרמטרים שהתקבלו מהלקוח בקישור.

```
"""
making the url of the flights
:param search: list of all search information
:param flights: list of all flights information
:return: url_flights: string of the url created
"""
# start of url making
# flights
url_flights = f"https://www.il.kayak.com/flights/{search[0]}-{search[1]}/{search[2]}/{search[3]}"
if flights[1].lower().strip() != 'economy' and search[4] == '1':
    url_flights += f"/{flights[1]}?sort=price_a"
elif flights[1].lower().strip() == 'economy' and search[4] != '1':
    url_flights += f"/{search[4]}adults?sort=price_a"
elif flights[1].lower().strip() != 'economy' and search[4] != '1':
    url_flights += f"/{flights[1]}/{search[4]}adults?sort=price_a"
else:
    url_flights += '?sort=price_a'

if flights[0] != '' and flights[2] == '':
    url_flights += f"&fs=stops=~{flights[0]}"
elif flights[0] == '' and flights[2] != '':
    url_flights += f"&fs=price=-{flights[2]}"
elif flights[0] != '' and flights[2] != '':
    url_flights += f"&fs=price=-{flights[2]};stops=~{flights[0]}"
else:
```

```
"""
making the url of the hotels
:param search: list of all the search information
:param hotels: list of all the hotels information
:param hotel_city: string of the name of the city
:return: url_hotels: string of the url created
"""
# hotels
url_hotels = f"https://www.il.kayak.com/hotels/{hotel_city}/{search[2]}/{search[3]}/{hotels[1]}adults?sort=rank_a"
if hotels[2] != '0':
    url_hotels = f"https://www.il.kayak.com/hotels/{hotel_city}/{search[2]}/{search[3]}/{hotels[1]}adults/{hotels[2]}ch
if hotels[0] != '1':
    url_hotels = f"https://www.il.kayak.com/hotels/{hotel_city}/{search[2]}/{search[3]}/{hotels[1]}adults/{hotels[2]}ch
if hotels[3] != '':
    url_hotels += f"&fs=stars={hotels[3]}"
    if hotels[5] != '':
        url_hotels += f";price=-{hotels[5]}"
    if hotels[4] != '':
        rate = hotel_rating(hotels[4])
        url_hotels += f";extendedrating={rate}"  # noqa
elif hotels[5] != '':
    url_hotels += f"&fs=price=-{hotels[5]}"
    if hotels[4] != '':
        rate = hotel_rating(hotels[4])
        url_hotels += f";extendedrating={rate}"  # noqa
```

יצירת קישור לאטרקציות על ידי שימוש באתר aviewoncities.com.

```python
def url_making_attractions(attr_city):
    """
    making the url of the attractions
    :param attr_city: the destination city
    :return: url_attraction: string of the url created
    """
    # attractions
    url_attraction = ""
    if attr_city in attr_cities:
        if attr_city == "Washington D.C.":
            attr_city = "washington"
        attr_city = attr_city.lower()
        if len(attr_city.split(' ')) > 1:
            attr_city = attr_city.replace(' ', '-')
        url_attraction = f"https://aviewoncities.com/{attr_city}"
    else:
        url_attraction = f"https://en.wikipedia.org/wiki/{attr_city}"
    return url_attraction
```

שליחת הקישורים חזרה אל הלקוח על ידי חיבור Socket בפורט 1234.

```python
"""
creates a connection to server and sends the links created
:param url_flights: url of the flights
:param url_hotels: url of the hotels
:param url_attraction: url of the attractions
"""
# Create a socket object
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Bind the socket to a local address and port
sock.bind(('0.0.0.0', 1234))

# Listen for incoming connections
sock.listen()

# Accept a connection from a client
client, address = sock.accept()
print('Connected to', address)

# Send data to the client
client.send(f"{url_flights} {url_hotels} {url_attraction}".encode())

# Close the connection when finished
client.close()
```

קבלת המידע שנשלח משרת במסך התוצאות באותו פורט 1234, במצב א-סינכרוני.

```
public class Get extends AsyncTask<Void,Void,Void> {
    @Override
    protected Void doInBackground(Void... voids) {
        try {
            // Create a socket connection to the Python server
            Socket socket = new Socket( host: "172.20.10.2",  port: 1234);

            // Read the incoming data from the socket
            InputStream in = socket.getInputStream();
            byte[] buffer = new byte[1024];
            int bytesRead = in.read(buffer);
            String line = new String(buffer,  offset: 0, bytesRead);
            System.out.println("Received data: " + line.trim());
            data = line.trim();
            String[] links = data.split( regex: " ");
            linkFlights = links[0];
            linkHotels = links[1];
            linkAttractions = links[2];

            // Close the socket when finished
            socket.close();
        } catch (IOException e) {
            // Handle any exceptions that may occur
            e.printStackTrace();
        }
```

חלק ג' –

מסמך בדיקות מלא :

הכנסת מידע למסד הנתונים – SQLite –

מטרת הבדיקה היא לבדוק את אופן הכנסת המידע שהתקבל מהלקוח למסד הנתונים SQLite. בכל ההרצות, הוכנס המידע לשלוש טבלאות. בתחילה, המידע שהוכנס לא היה מוצפן על מנת שאוכל לבדוק ולראות את המידע. לאחר בדיקות רבות, המידע הוכנס בצורה נכונה ושיניתי את ההכנסת מידע, להכנסת מידע מוצפן באמצעות הצפנת RSA.

העברה של מידע מצד השרת לצד הלקוח, ומצד הלקוח לצד השרת –

מטרת הבדיקה היא לבדוק את העברת המידע מהשרת ללקוח, וההפך, מהלקוח לשרת. העברת המידע נוצרת על ידי חיבור socket בפורט מסוים. הלקוח יוצר קשר עם השרת ושולח לו את המידע שהלקוח הזין. לאחר מכן, השרת יוצר קשר עם הלקוח ושולח לו את הקישורים. בדיקות רבות של העברת המידע בזמן חיבור אינטרנט יציב.

בדיקות תקינות במסכי החיפוש באפליקציה –

מטרת הבדיקה לבדוק את תקינות המידע שהמשתמש מזין בעת החיפוש באפליקציה, במסכי search, flights ו – hotels.

במסך search, כל הפרטים הם חובה. מקום ההמראה ומקום הנחיתה צריכים להיות רק מתוך המאגר שנפתח כשמקלידים (Autocomplete). התאריכים צריכים להיות אחרי התאריך של אותו היום, ותאריך הנחיתה אחרי תאריך ההמראה. מספר הנוסעים הוא מספר בלבד.

במסך flights, כל הפרטים הם רשות. מספר עצירות זה 0, 1 או 2+, לא ניתן להזין ערך אחר. סוג מושב גם צריך להיבחר מתוך הרשימה שנפתחת כאשר מקלידים. מחיר מקסימלי הוא מספר בלבד.

במסך hotels, כל הפרטים הם רשות. כמות חדרים, כמות מבוגרים וכמות ילדים הם מספרים בלבד. כמו כן, מספר כוכבים מינימלי נבחר מתוך הערכים 2, 3, 4, 5 ומספר דירוג מינימלי נבחר מתוך 6, 7, 8, 9. מחיר מקסימלי הוא מספר בלבד.

ביצעתי בדיקות רבות על מנת לבדוק את כל התנאים האלו, על ידי הצבת נתונים שונים, בדיקת גבולות, הזנת חלק מהנתונים, נתונים שגויים ועוד על מנת לבדוק את כל האפשרויות. כמו כן, בדקתי את הערות השגיאה שהתקבלו בהרצה בעקבות הזנת פרטים לא לפי התנאים.

בדיקות תקינות בהרשמת משתמש חדש באפליקציה –

מטרת הבדיקה לבדוק את תקינות המידע שהמשתמש מזין בעת הרשמה למערכת באפליקציה, במסך signup.

במסך זה, שם המשתמש צריך להיות בין 4-15 תווים, מכיל לפחות אות גדולה אחת, אות קטנה אחת וללא רווחים. כמו כן, האימייל צריך להיות תקין והסיסמה צריכה להיות בין 8-20 תווים, מכילה לפחות ספרה אחת, אות אחת גדולה, אות אחת קטנה וללא רווחים. במידה והפרטים תקינים, פרטיו של הלקוח מתווספים למסד הנתונים Firebase בשם users.

כל הבדיקות נעשו על ידי שימוש בספריית regex – רצף תווים שמגדיר תבנית חיפוש. בהגדרתו הכללית ביותר פירושו ביטוי בשפה רגולרית, שמוגדרת כשתי מחרוזות או יותר הכפופות לתקנות תחביר מסוימות.

ביצעתי בדיקות רבות על מנת לבדוק את כל התנאים להרשמת משתמש חדש למערכת באפליקציה, בדקתי איך הנתונים התקבלו במסד הנתונים Firebase, הזנתי נתונים שלא עונים על הדרישות ובדקתי את הערות השגיאה שהתקבלו בהרצה בעקבות הזנת פרטים לא לפי התנאים.

מדריך למשתמש

עץ קבצי המערכת –

צד לקוח –

```
C:.
│   AndroidManifest.xml
│
├───java
│   └───com
│       └───example
│           └───app2
│                   FlightsActivity.java
│                   HelperClass.java
│                   HotelsActivity.java
│                   LoadingActivity.java
│                   LoginActivity.java
│                   MainActivity.java
│                   ResultActivity.java
│                   SearchActivity.java
│                   SignupActivity.java
```

```
└───res
    ├───drawable
    │       bg2.png
    │       bg4.png
    │       bg5.jpg
    │       ic_baseline_attractions_24.xml
    │       ic_baseline_calendar_month_24.xml
    │       ic_baseline_email_24.xml
    │       ic_baseline_flight_24.xml
    │       ic_baseline_hotel_24.xml
    │       ic_baseline_lock_24.xml
    │       ic_baseline_person_24.xml
    │       ic_baseline_person_pin_24.xml
    │       ic_launcher_background.xml
    │       lavender_border.xml
    │       line_text.xml
    │
    ├───drawable-v24
    │       ic_launcher_foreground.xml
    │
    ├───layout
    │       activity_flights.xml
    │       activity_hotels.xml
    │       activity_loading.xml
    │       activity_login.xml
    │       activity_main.xml
    │       activity_result.xml
    │       activity_search.xml
    │       activity_signup.xml
```

צד שרת –

```
└───client_server
    │   city.py
    │   data.py
    │   deals.db
    │   server.py
```

התקנת המערכת –

סביבה נדרשת וכלים נדרשים :

בשביל השרת :IDLE לשפת python גרסה 3.9, והורדת הספריות socket, sqlite3 ו – rsa.

בשביל הלקוח : תוכנת Android Studio, והורדת Emulator, או חיבור למחשב של טלפון עם מערכת ההפעלה של אנדרואיד.

נדרש חיבור אינטרנט יציב על מנת ליצור חיבור בין הלקוח והשרת, על ידי sockets.

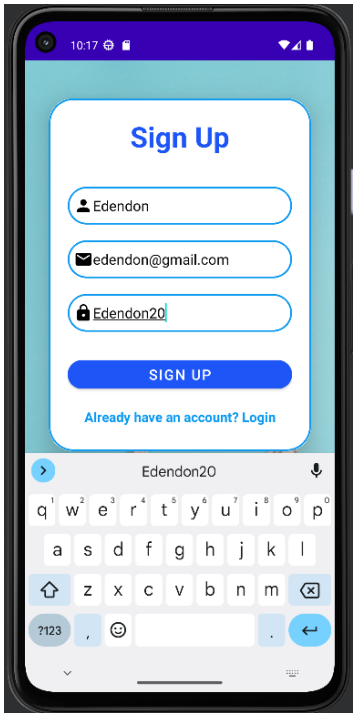
מיקומי קבצים :

קבצי השרת ומסד הנתונים של השרת נמצאים ב – SecretDeals/client_server/. הקבצים הם :

city.py, data.py, server.py ו – deals.db.

קבצי האפליקציה נמצאים ב – App2/app/src/main שם נמצאת תיקיית java אשר בה יש את קבצי java. כלומר, ה – Activities. כמו כן, שם נמצאת תיקיית res אשר בה יש את קבצי העיצוב xml. וכל תוספות העיצוב (Vector Assets ועוד).

דוגמה להרצה ראשונה של המערכת :

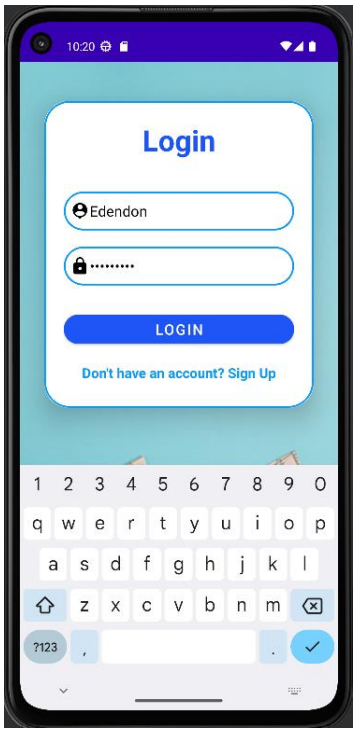


ראשית, מריצים את צד השרת – קובץ server.py.

לאחר מכן, פותחים את האפליקציה, כאשר נפתח מסך הכניסה, לוחצים על כפתור ה – Sign up על מנת להירשם למערכת.

הזנת פרטים של המשתמש, שם משתמש, אימייל וסיסמה, לפי הדרישות. דוגמה לקלט נכון :

בסיום, לוחצים על כפתור Sign up.

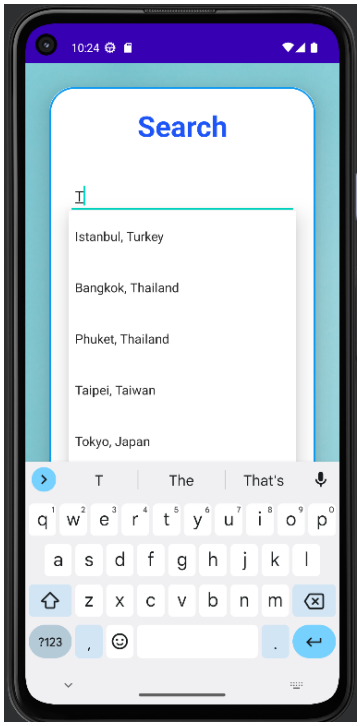


כעת, עושים Login עם פרטי המשתמש.

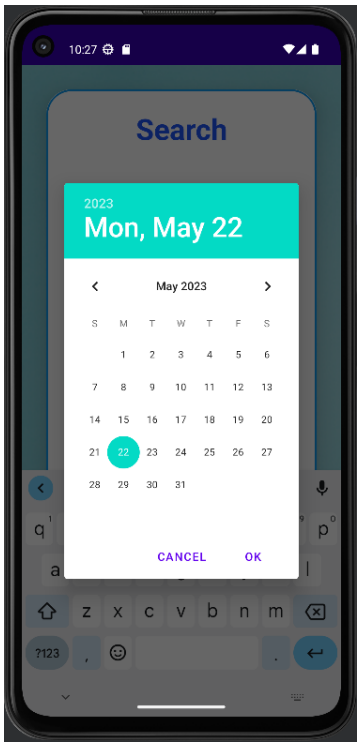
הסיסמה במצב זה מוסתרת – מצב password.

בסיום, לוחצים על כפתור Login.

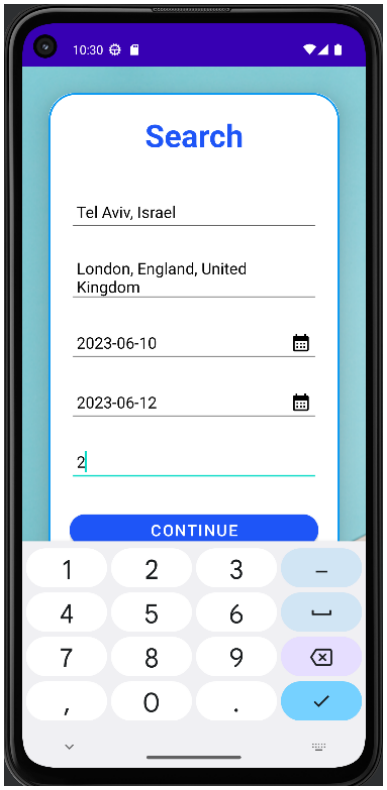
בשלב זה, המשתמש מחובר למערכת, ומתחיל תהליך החיפוש. על המשתמש לבחור מקום המראה, מקום נחיתה, תאריך המראה, תאריך נחיתה ומספר נוסעים, אלו הם פרטי חובה וממלאים אותם לפי הדרישות.



כאשר מתחילים להקליד מקום המראה או מקום נחיתה, יש השלמה אוטומטית. לדוגמה :

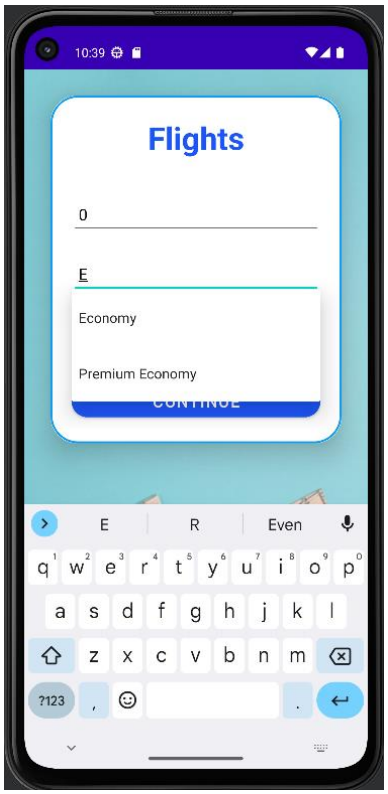


כמו כן, לחיצה על תיבת הטקסט של תאריך ההמראה או תאריך הנחיתה, פותחת לוח שנה שניתן לבחור ממנו תאריכים.



דוגמה לחיפוש מתל אביב, ישראל ללונדון, אנגליה, מהתאריך 10.6.2023 עד לתאריך 12.6.2023 לשני נוסעים.

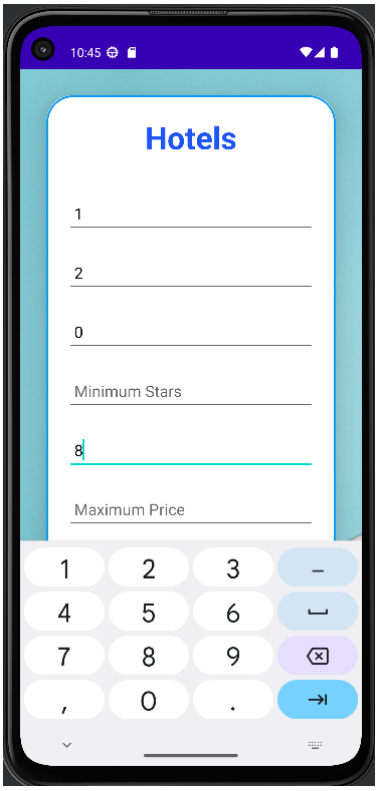
בסיום, לוחצים על כפתור Continue.



המשתמש הגיע אל חיפוש הטיסות, אלו הם פרטי רשות.

המשתמש יכול להזין מספר עצירות, סוג מחלקה ומחיר מקסימלי של טיסה. גם כאן, יש השלמה אוטומטית למספר העצירות ולסוג המחלקה.

בסיום, לוחצים על כפתור Continue.



המשתמש הגיע אל חיפוש המלונות, אלו הם פרטי רשות.

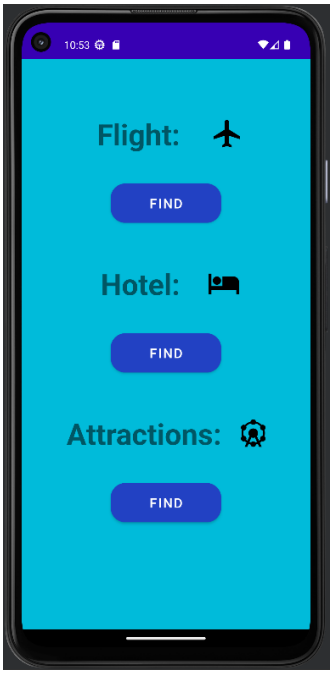
המשתמש יכול להזין כמות חדרים, כמות מבוגרים, כמות ילדים, מספר כוכבים מינימלי, דירוג מינימלי ומחיר מקסימלי.

גם כאן, יש השלמה אוטומטית למספר הכוכבים ולדירוג.

בסיום, לוחצים על כפתור Search.

כל הנתונים שהמשתמש הזין עוברים לשרת. פלט השרת :

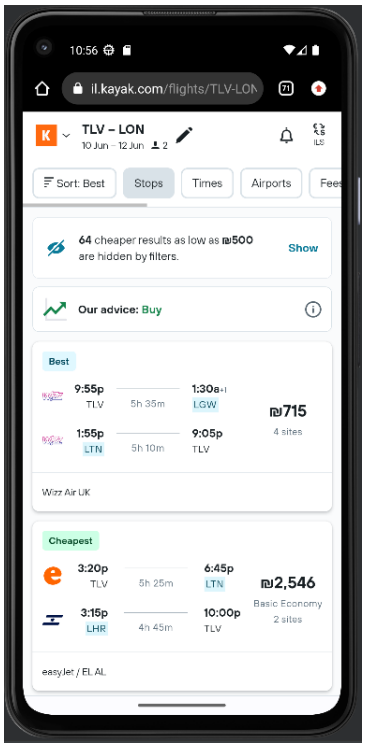
```
Server started at 0.0.0.0 on port 8000
New connection made!
TelAviv,Israel London,England,UnitedKingdom 2023-06-10 2023-06-12 2 0 Economy  1 2 0  8
['TelAviv,Israel', 'London,England,UnitedKingdom', '2023-06-10', '2023-06-12', '2', '0', 'Economy', '', '1', '2', '0', '', '8', '']
Return:  LON
Departure:  TLV
['TLV', 'LON', '2023-06-10', '2023-06-12', '2']
['0', 'economy', '']
['1', '2', '0', '', '8', '']
Found: London, England, United Kingdom
URL FLIGHTS: https://www.il.kayak.com/flights/TLV-LON/2023-06-10/2023-06-12/2adults?sort=price_a&fs=stops=~0
URL HOTELS: https://www.il.kayak.com/hotels/London,England,UnitedKingdom/2023-06-10/2023-06-12/2adults?sort=rank_a&fs=extendedrating=great
URL ATTRACTIONS: https://aviewoncities.com/london
Connected to ('192.168.1.31', 59833)
Opened database successfully
Inserted values into tables

Process finished with exit code 0
```

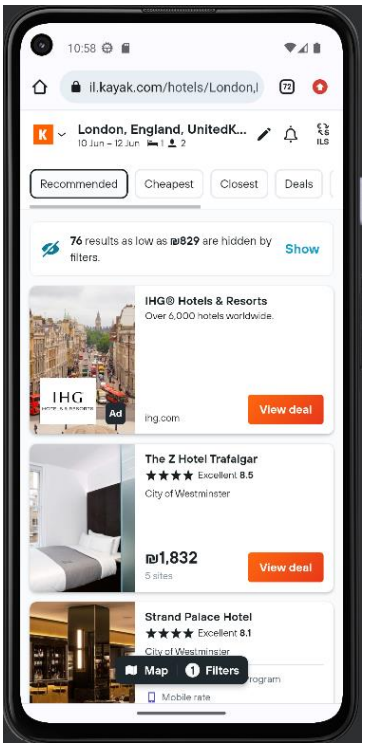


בצד הלקוח, מסך התוצאות נפתח, בו קיבל את תוצאות השרת.

לחיצה על כפתור פותח את דפדפן Google Chrome עם הקישור המתאים.

לדוגמה, קישור לטיסה, למלון ולאטרקציות :





רפלקציה

כבר בתחילה, חשוב לי לציין כי מאד נהניתי מהעבודה על הפרויקט. תהליך העבודה היה מאתגר מרתק ומעניין. במהלך העבודה נתקלתי במספר קשיים ואתגרים אך גם הצלחות והרבה סיפוק.

החשיפה לפרויקט זה הציגה בפניי רעיונות חדשים ומעניינים אשר העניקו לי זווית ראייה חדשה ופתיחות מחשבתית. במהלך העבודה על הפרויקט למדתי רבות על מגוון תחומים ואף העשרתי את ידיעותיי בנושאים שונים.

נחשפתי לתכנות אפליקציות בסביבת אנדרואיד. זהו נושא חדש עבורי, שלא למדנו אותו בשיעורים בבית הספר, או בלימודיי לתואר. לקחתי על עצמי אחריות ללמוד את התחום בעצמי. נהניתי מתהליך הלמידה העצמאי והמרתק. בעקבות כך, נחשפתי גם למסד הנתונים של Google, הנקרא Firebase, אופן הקישור לאפליקציה, התנהלות עם מסד הנתונים והכנסת מידע חדש.

בנוסף לכך, הרחבתי את ידיעותיי בתכנות שרת-לקוח, השכלתי ולמדתי בנוסף לידע שהיה לי, על הקשר בין שני הצדדים, על התקשורת, תכנות ה – sockets ומעבר המידע מצד הלקוח לצד השרת, לא רק בשפת Python, אלא גם בשפת Java.

במהלך עבודתי נתקלתי גם בקשיים שהייתי צריכה להתגבר עליהם. לדוגמא, בחירת הנושא לפרויקט. בתחילה, חשבתי על רעיון של חיזוי, אך מצד שני היה לי חשוב לבצע פרויקט שיעניין אותי יאתגר וייתן לי ערך מוסף. לכן בחרתי ברעיון של טיסות ונסיעות לחו"ל.

קושי נוסף שעלה במהלך הדרך הינו עמידה בלוח זמנים. מלבד בית ספר ומבחני בגרות, בנוסף, אני לומדת גם במכללה האקדמית נתניה שנה רביעית לתואר במדעי המחשב מה שמותיר לי זמן מועט. בעקבות זאת, תכננתי מראש לוח זמנים קפדני ומדויק על מנת למקסם את הזמן הפנוי ולהקדישו לביצוע הפרויקט.

מסקנותיי מביצוע פרויקט זה הן כי בתחילת הדרך חשוב מאד לעבוד על פי תכנון מסודר ומפורט, וכך גם למדתי לנהל באופן נכון ומיטבי את הזמן.

כמו כן, ביצוע הפרויקט דרש ממני להתמודד עם מגוון נושאי לימוד חדשים, כל זאת פיתוח מיומנויות של יכולת לימוד עצמי וחשיבה עצמאית. חשוב לי לציין כי, היכולות האלה תרמו לי רבות להתקדם ולבצע את הפרויקט.

שאלת חקר – האם ביצוע הפרויקט בקבוצה היה מניב תוצאה טובה יותר?

לגבי משאבים נוספים לא חושבת שהייתי משנה משהו בפרויקט.

פרויקט זה העניק לי ידע רב ואף נתן לי כלים לחיים, ביצוע הפרויקט בצורה הטובה ביותר דרש ממני שעות רבות של קריאה, למידה עצמאית וחיפוש פתרונות. כאשר נתקלתי בקשיים נדרשתי לעיתים לחשיבה מחוץ לקופסא וזאת על מנת להבין איך להתקדם.

בזכות פרויקט זה שיפרתי באופן משמעותי את יכולות הלמידה העצמאית שלי, אני מרגישה שלמדתי להיות יותר פרואקטיבית בחיפוש אחר מידע וכך גם העמקתי והרחבתי מאד את הידע שלי בנושאים היו חדשים לגמרי עבורי.

במבט לאחור, ביצוע הפרויקט היה שונה ומעניין יותר מכל העבודות שקיבלנו בבית הספר. הפרויקט דרש שעות רבות של עבודה ומחקר, העבודה התבססה על ידע בסיסי שנלמד בבית הספר אולם, רוב הפרויקט התבסס בעיקר על לימוד עצמי, חיפוש מידע באתרים מהימנים וסרטוני הסברה באופן עצמאי, משמעת עצמית גבוהה, יכולת עמידה בזמנים והצבת יעדים תוך הקפדה על עמידה בלוח זמנים מה שגרם לי לתחושת מסוגלות גבוהה.

בעקבות ביצוע פרויקט זה קיבלתי כלים חדשים, ופיתחתי מיומנויות אשר יעזרו לי בעתיד. רכישת ידע חדש ומעשיר או מסוגלות לישיבה של שעות רבות על מנת להצליח. קיבלתי תובנות לחיים שאני יכולה ומסוגלת להצליח, הצבת יעדים והשגתם.

חשוב לי מאד להוקיר תודה ענקית למורה שלי ריקי ולחונכים יונתן ועידו, שייעצו, עזרו ותמכו לאורך כל הדרך ובזכותם האמנתי שאני יכולה לבצע את הפרויקט בצורה הטובה ביותר ואני מקווה שעשיתי כך.

ביבליוגרפיה

"הוסף את Firebase לפרויקט Android שלך" – אתר firebase.google.

https://firebase.google.com/docs/android/setup?hl=he

"Login and SignUp Page in Android Studio using Firebase Realtime Database with Profile" – androidknowledge website.

https://androidknowledge.com/login-signup-android-studio-firebase-realtime/

" How to Encrypt and Decrypt Strings in Python?" – kabilan – geeksforgeeks website.

https://www.geeksforgeeks.org/how-to-encrypt-and-decrypt-strings-in-python/

"Java Regular Expressions" – w3schools website.

w3schools.com/java/java_regex.asp

נספחים

קטעי קוד מיוחדים נוספים :

הרשמת משתמש חדש למערכת – בדיקות תקינות של שם המשתמש לפי קריטריונים, אימייל נכון, סיסמה לפי קריטריונים. השתמשתי בספריות regex.Matcher ו – regex.Pattern, ויצרתי ביטוי רגולרי בהתאם לדרישות.

```
// checks username using regex
String regex_username = "^(?=.*[a-z])(?=.*[A-Z])(?=\\s+$).{4,15}$";
// It contains at least 4 characters and at most 15 characters.
// It contains at least one upper case alphabet.
// It contains at least one lower case alphabet.
// It doesn't contain any white space.
Pattern pattern_username = Pattern.compile(regex_username);
Matcher matcher_username = pattern_username.matcher(username);
String message_username = "";
if (!matcher_username.matches()) {
    if (username.length() < 4 || username.length() > 15)
        message_username = "Username must be at least 4 characters and at most 15 characters";
    else if (!isUpper(username))
        message_username = "Username must contain at least one upper case alphabet";
    else if (!isLower(username))
        message_username = "Username must contain at least one lower case alphabet";
    else if (!username.equals(username.replaceAll( regex: " ", replacement: "")))
        message_username = "Username must not contain any white space";
    else
        message_username = "Username not valid";
    signupUsername.setError(message_username);
    return false;
}
```

```
// checks email using regex
String regex_email = "^(.+)@(.+)$";
Pattern pattern_email = Pattern.compile(regex_email);
Matcher matcher_email = pattern_email.matcher(email);
if (!matcher_email.matches()) {
    signupEmail.setError("Email not valid");
    return false;
}
```

```java
// check password using regex
String regex_password = "^(?=.*[0-9])(?=.*[a-z])(?=.*[A-Z])(?=\\S+$).{8,20}$";
// It contains at least 8 characters and at most 20 characters.
// It contains at least one digit.
// It contains at least one upper case alphabet.
// It contains at least one lower case alphabet.
// It doesn't contain any white space.
Pattern pattern_password = Pattern.compile(regex_password);
Matcher matcher_password = pattern_password.matcher(password);
String message_password = "";
if (!matcher_password.matches()) {
    if (password.length() < 8 || password.length() > 20)
        message_password = "Password must be at least 8 characters and at most 20 characters";
    else if (!isUpper(password))
        message_password = "Password must contain at least one upper case alphabet";
    else if (!isLower(password))
        message_password = "Password must contain at least one lower case alphabet";
    else if (!isDigit(password))
        message_password = "Password must contain at least one digit";
    else if (!password.equals(password.replaceAll( regex: " ",  replacement: "")))
        message_password = "Password must not contain any white space";
    else
        message_password = "Password not valid";
    signupPassword.setError(message_password);
    return false;
```

קטע קוד נוסף, הוא במסך החיפוש, הוספתי אופציה לבחור תאריך מלוח שנה שנפתח על ידי לחיצה על תיבת הטקסט. לוח השנה הוא לתאריך ההמראה ולתאריך הנחיתה. השתמשתי בספריות רבות, ביניהן : app.DatePickerDialog, widget.DatePicker, text.DateFormat, util.Calendar, util.Date.

```
private void showDatePickerDialogDep() {
    DatePickerDialog departureD = new DatePickerDialog(
            context: this, new InnerFirstDate(),
            Calendar.getInstance().get(Calendar.YEAR),
            Calendar.getInstance().get(Calendar.MONTH),
            Calendar.getInstance().get(Calendar.DAY_OF_MONTH)
    );
    departureD.show();
}

private void showDatePickerDialogRet() {
    DatePickerDialog returnD = new DatePickerDialog(
            context: this, new InnerSecondDate(),
            Calendar.getInstance().get(Calendar.YEAR),
            Calendar.getInstance().get(Calendar.MONTH),
            Calendar.getInstance().get(Calendar.DAY_OF_MONTH)
    );
    returnD.show();
}
```

```java
public class InnerFirstDate implements DatePickerDialog.OnDateSetListener{
    @Override
    public void onDateSet(DatePicker view, int i2, int i1, int i) {
        String i_text = Integer.toString(i);
        String i1_text = Integer.toString( i: i1+1);
        if (i_text.length() == 1)
            i_text = "0" + i_text;
        if (i1_text.length() == 1)
            i1_text = "0" + i1_text;

        String date = i2 + "-" + i1_text + "-" + i_text;
        departureDate.setText(date);
    }

}
```

```java
public class InnerSecondDate implements DatePickerDialog.OnDateSetListener{
    @Override
    public void onDateSet(DatePicker view, int i2, int i1, int i) {
        String i_text = Integer.toString(i);
        String i1_text = Integer.toString( i: i1+1);
        if (i_text.length() == 1)
            i_text = "0" + i_text;
        if (i1_text.length() == 1)
            i1_text = "0" + i1_text;

        String date = i2 + "-" + i1_text + "-" + i_text;
        returnDate.setText(date);
    }
}
```