פרויקט גמר

**הנדסאי שם המגמה**

בנושא

תפריט חכם

מגישים: ג'וליאן תותרי

מנחה: רון להב

2025

תארי ך: 18.01.2025

חוזר מנהל מה"ט – 51-4-11 נספח מס( 1 'הצעה לפרויקט גמר)

**א. פרטי הסטודנט**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **תאריך סיום**  **הלימודים** | **טלפון נייד** | **כתובת** | **ת.ז. 9 ספרות** | **שם הסטודנט** |
|  | 0504575435 | 6 רח' 5056 | 214659385 | ג'וליאן תותרי |

**שם המכללה** המכללה הטכנולוגית נוף הגליל  **סמל המכללה :** 72209

**מסלול ההכשרה:** הנדסאים.

**מגמת לימוד:** הנדסת תוכנה-סייבר

**מקום ביצוע הפרויקט:** המכללה הטכנולוגית נוף הגליל

**ב. פרטי המנחה האישי**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **מקום עבודה/תפקיד** | **תואר** | **טלפון נייד** | **כתובת** | **שם המנחה \*** |
| ראש המגמה | Bsc / Mba | 0523203282 | יקינטון 23 חיפה | רון להב |



\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

חתימת הסטודנט חתימת המנחה האישי חתימת הגורם המקצועי מטעם מה"ט

# שם הפרויקט

תפריט חכם (smart menu). "ביקור חכם במסעדה ללא המתנה למלצר"

# רקע

## **תיאור ורקע כללי**

מערכת לניהול עבודת המלצרים והמטבח מול הלקוח במסעדה.

המערכת מציגה תפריט של מסעדה ומאפשרת ללקוח להזמין אוכל\שתיה מהתפריט, ומאפשרת גם לבקש מהמלצר שירות לפי הצורך למשל "לבקש חשבון", ובנוסף לזה תאפשר ללקוח לשלם את החשבון דרך כרטיס אשראי. הרעיון הוא שהכול מתנהל דרך האפליקציה בלי להמתין שהמלצר יגיע לשולחן על מנת לקבל את בקשות הלקוח

## **מטרות המערכת**

**1**. מטרת המערכת לשפר את עבודת המלצר במסעדה ולקצר את זמן ההמתנה למלצר. בכך מזרז את ביצועו שגורם להקטנת זמני המתנה של הלקוחות במסעדה.

**2**. המטרה היא להפוך את תפריט המסעדה לתפריט דיגיטלי, אבל לא רק אלא גם רוב פעולות המסעדה לפעולות דיגיטלית וטכנולוגית יותר מתקדמות במקום פעולות ידניות, בעיקר פעולות המלצר.

# סקירת מצב קיים בשוק, אלו בעיות קימות

* [Must Have Menu](https://www.musthavemenus.com/category/Restaurant-Menu.html)
* [*Web Restaurant*](https://www.webstaurantstore.com/guide/813/menu-buying-guide.html)- היום קיים מגוון רחב של מערכות כמו *(MSUT HAVE MENU && WEB RESTURANT)* השימושיות בתוך המסעדות מערכות אלו לא ניתן לבציע הזמנה של אוכל\שתיה דרכן אלא צריך לבצע מול המלצר, ובנוסף לזה לא ניתן לסגור את החשבון ולבצע תשלום דרך כרטיס אשראי.
* [Menu](https://www.webstaurantstore.com/guide/813/menu-buying-guide.html) - מערכת "Menu" לא מאפשרת ללקוחות להזמין אוכל דרכה אלא רק לצפות בתפריט של המסעדה.
* נכון להיום קיים בעיה נפוצה ברוב המסעדות היא איטיות השירות של המסעדה שזה בגלל מערכת ניהולית לא יעילה, גורם ללקוחות לצאת מהמסעדה ללא חזרה.
* רוב המסעדות עדיין עובדים עם תפריט נייר שקשה לבעל המסעדה להוסיף או לשנות פריט בתוכו.

# מה הפרויקט אמור לחדש או לשפר

הפרויקט משנה את סגנון עבודת המסעדות לסגנון דיגיטלי וטכנולוגי מתקדם יותר והפרויקט מאפשר למסעדות להיות מעודכנים תמיד, לטובת המסעדנים והלקוחות. כמו כן הפרויקט משפר את עבודת ה online והתאום בין הגורמים השונים שבמסעדה. המערכת מאפשרת למצע צ'ט בין הסועד לעובד ייעודי במסעדה לצורך התייעצות על מנות וסיוע בבחירת המנות.

# דרישות מערכת ופונקציונאליות

## **דרישות מערכת** – סביבת הטמעה ושימוש. שרידות, ביצועים\ התמודדות עם עומסים.

* המערכת תתנהל בצורה מקוונת, כלומר מתחברים לאתר האינטרנט של המערכת לכן חובה להיות מחובר לאינטרנט.
* הנתונים שנשמרים בענן המערכת עושה להם גיבוי בזמנים קבועים כך ניתן לשמור על הנתונים מהאבוד.
* המערכת תופעל בענן בעל כוח רב לארח מספר ענק של משתמשים בו זמני, בנוסף מאגרי הנתונים יתארחו על גבי ענן בעל קיבולת זיכרון גבוהה וגמישה.

## **דרישות פונקציונאליות – רשימת דרישות המשתמש מהמערכת, מהן הפעולות בהן נדרשת המערכת לתמוך.**

|  |  |  |  |
| --- | --- | --- | --- |
| **מס** | **תיאור דרישה** | **FR** | **NFR** |
| **1** | **משתמשים** |  |  |
| 1.1 | המערכת תאפשר קליטת משתמש חדש באמצעות ממשק טלפוני . | V |  |
| 1.1.1 | בזמן קליטת משתמש חדש  המערכת תקלוט הפרטים הבאים  1) שם פרטי  2) מספר טלפון  3) מייל  4) סיסמה לחשבון הלקוח | V |  |
| 1.1.2 | המערכת תאפשר ללקוח קיים לראות ולערוך את החשבון שלו במערכת. | V |  |
| 1.2 | המערכת תצפין את סיסמת המשתמש דרך שימוש באלגוריתם Hashing של BCrypt | V |  |
| 1.3 | המערכת תאמת בקשות משתמש לשרת דרך   JWT (JSON Web Token | V |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **2** | **מערכת** |  |  |
| 2.1 | לאחר כניסה למערכת, המערכת תחשוף בפני המשתמש את שולחנות המסעדה והסטטוס שלהם  **(Occupied\Empty)** | V |  |
| 2.2 | המערכת תאפשר למשתמש לסמן את עצמו כיושב על שולחן דרך כפתור באפליקציה | V |  |
| 2.3 | במידה ונמצא שולחן פנוי עבור הסועד / סועדים ולחצן "סמן כיושב" המערכת תעביר את המשתמש לתפריט המסעדה לשם בקשת האוכל | V |  |
| 2.4 | אחרי הזמנת ארוחות מהתפריט המשתמש יועבר לעמוד תשלום | V |  |
| 2.5 | המערכת תסמן שולחן כפנוי לאחר גמר תשלום ואישור המשלם שהוא פינה את השולחן | V |  |
| 2.6 | בתום ביצוע ההזמנה , המלצרים והשף של המסעדה יכולים לראות את ההזמנה והיא תצא לביצוע במטבח. | V |  |
| 2.7 | המערכת תאפשר למטבח לסמן את ההזמנה כמוכנה לשילוח לשולחן בתום ההכנה שלה. | V |  |
| 2.8 | המערכת תשתמש בשקעים (Sockets) לשם עדכון סטטוס הכנת הארוחה בזמן אמת כשמלצר או שף מעדכן שהארוחה מוכנה היא תעודכן בזמן אמת דרך פונקציה בצד הלקוח שמאזינה לתרחיש | V |  |
| 2.9 | המערכת תשתמש בשקעים (Sockets) לשם עדכון סטטוס ישיבה על שולחן בזמן אמת למשל אם לקוח עוזב את השולחן שהוא עליו השולחן שלו יסומן כ"פנוי" לשאר הלקוחות בזמן אמת דרך תרחיש אשר נמצא באפליקציה של כל לקוח | V |  |
| 2.10 | המערכת תאפשר לבטל הזמנה כל עוד ההזמנה לא הוגשה לשולחן | V |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **3** | **מלצר** |  |  |
| 3.1 | המערכת תאפשר לבצע קריאה מהירה למלצר בעזרת פונקציות מוגדרות מראש   1. להחליף סכו"ם – להביא סכו"ם חדש. 2. להביא קנקן מים. 3. להביא כיסא תינוק 4. לנקות את השולחן. 5. להביא מפיות נוספות. 6. הבאת רוטב נוסף. (הקלדת שם הרוטב) 7. הבאת תבלינים (הקלדת שם התבלין). 8. להחליף מנה – במקרה של טעות במנה או אי-שביעות רצון. 9. להביא חשבון – אם התשלום מתבצע מול המלצר. 10. להזיז מאוורר/מזגן – אם ניתן לשליטה ידנית. 11. **טעות בחשבון** – אפשר להפנות את הלקוח לשירות תמיכה דרך האפליקציה. 12. **לדווח על אלרגיות** – ניתן להציג מידע תזונתי על המנות ישירות באפליקציה. 13. **להזמין אריזת טייק-אווי** – אפשר לאפשר ללקוח להזמין אריזה ישירות באפליקציה ולגשת לעמדת האיסוף. 14. **להזמין נר/עוגה לאירוע** – ניתן להוסיף אופציה לחגיגות וליידע את הצוות ישירות 15. **לבקש להעביר שולחן** – אם יש שולחן פנוי, ניתן לבצע החלפה דרך המערכת ללא צורך במלצר. 16. **לבקש חשבון נפרד** – המערכת תאפשר חלוקת תשלום דיגיטלית לכל סועד בנפרד | V |  |
| 4 | **בסיס נתונים** |  |  |
| 4.1 | במערכת קיים בסיס נתונים השומר את נתונים הבאים:   1. נתונים של המשתמשים. 2. נתונים של המסעדה. 3. הזמנות קודמות של משתמשים. 4. פרטי שולחנות המסעדה (כמה אנשים יכולים לשבת וכו..) |  | V |
| 4.2 | פרטי משתמשים רגישים כמו סיסמאות, מספרי כרטיסי אשראי, מס' ת"ז של בעלי כרטיסי יוצפנו ע"י ספריית הצפנה | V |  |
| 5 | **בעל מסעדה** |  |  |
| 5.1 | לבעל מסעדה יהיו יכולות הוספה ומחיקה של עובדים, שולחנות, ומנות | V |  |
| 5.2 | בעל מסעדה יכול להוסיף עובד כל עוד שאותו עובד לא נמצא בבסיס הנתונים (המייל של העובד צריך להיות ייחודי) | V |  |
| 5.3 | בעל מסעדה יכול למחוק עובד כל עוד אותו עובד אינו מחובר לטרמינל שלא (כלומר כל עוד שהעובד אינו במשמרת) | V |  |
| 5.4 | כשמנהל מוחק שולחן מספר השולחנות משתני לשקף את כמות השולחנות האמיתי שנמצאים במסעדה. הסבר:  נניח שהמסעדה כוללת 5 שולחנות ממוספרות מ 1-5 אם בעל בא למחוק את שולחן 2 שולחן 2 נמחק שולחן 3 הופך לשולחן 2 שולחן 4 ל-3 וכו... | V |  |
| 5.5 | בשל הפונקציה בסעיף 5.4 ולשם מחיקת שולחן בצורה מוגנת כל השולחנות אחרי אותו שולחן צריכים להיות רכים כך שאין בהם אף לקוח ואף מלצר אחרת המחיקה תצור התנהגות לא צפויה שח התוכנה | V |  |

|  |  |  |  |
| --- | --- | --- | --- |
| 6 | תפריט מוצר |  |  |
| 6.1 | מוצרים במסעדה יחולקו לפי קטגוריות 1. מנות ראשונות  2. מנות עיקריות  3. קינוחים.  4. משקאות קרים\חמים  5. אלכוהול |  | V |
| 6.2 | כל מוצר יכלול   1. את מרכיבי המנה 2. מחיר 3. קטיגורית מנה 4. המלצות כלליות |  | V |

# בעיות צפויות במהלך הפיתוח ופתרונות:

|  |  |
| --- | --- |
| **תיאור הבעיה(6.1)** | **פתרון אפשרי(6.2)** |
| אינטגרציה של אוסף של APIs בעיקר Mailjet לשליחת מייל וAbstractAPI על מנת לבדוק אם כתובת מייל נמצאת | כמוס פונקציונאליות של שליחת מיילים דרך שימוש במחלקה המשלבת את פונקציונאליות שתי השירותים יחד |
| כישלון חיבור ל-MongoDB אחרי שלקוח נכנס יכול להיות בעיתי לפונקציונאליות של התוכנה | יישום caching לנתונים אשר חשובים ללקוח בעיקר הארוחות בתפריט וה-State האחרון של שולחנות המסעדה. בנוסף לכך השארת אותו State בצד שרת חשוב כי לפעמים כישלון חיבור ל-MongoDB לא יהיה תוצאה משגיאה בשרת |
| סוג החיבור בין צד הלקוח לצד השרת הוא HTTP כברירת מחדל | שימוש בתעודות TLS/SSL על מנת לשמור על הצפנת מידע שיוחלף בין שרת ללקוח |
| גישה ל-APIs דורשת מפתחות API שהתוכנה צריכה להכיל. אבל העלאת מפתחות עם הקוד ל-Github עלולה לחשוף את אותם מפתחות לשאר העולם | לשים את כלל המפתחות בקובץ JSON אחד אשר נקרא Secrets.json ואי העלאתו ל-Github דרך הוספת שמו לקובץ ה-.gitignore שלנו |

# פתרון טכנולוגי נבחר:

## **טופולוגית הפתרון:**

****

## **טכנולוגיות בשימוש**:

* Visual Studio Code תוכנה לעריכת קוד
* MongoDB Compass תוכנה לוויזווליזאציאת בסיס נתונים
* MailjetAPI ספריית עזר לשם שליחת מיילים למשתמשים

## **שפות הפיתוח:**

**צד לקוח:**

* **TypeScript**: שפת תכנות המוסיפה טיפוסיות ל-JavaScript ומאפשרת כתיבת קוד אמין ומובנה.
* **React Native**: ספריה לתכנות ממשקי משתמש לאיפון ולאנדרואיד בחרתי אותה בגלל פילוסופיאת ה-Write once, run everywhere אם לא בחרתי אותה אז היה נדרש ממני לכתוב את ה-UI לכל סוג פלפון בשפה לחוד
* **CSS**: לעיצוב ועיצוב דפי אינטרנט.

**צד שרת:**

* **C#**: שפת תכנות מונחית עצמים המשמשת לפיתוח צד שרת. השתמשתי בשפה זו בשל מהירותה
* **MongoDB**: מערכת לניהול מסדי נתונים גמישה ומודולרית, שמתאימה לעבודה עם כמויות גדולות של נתונים.

## **תיאור הארכיטקטורה הנבחרת:**

בחרתי בארכיטקטורת שרת-לקוח עם שרת מקומי לכל מסעדה המשתמשת באפליקציה בכדי לצבור על מחירי ה-Hosting ולהתמודד עם היכולות הפיננסיות של בעלי המסעדות. בחרתי העיקר ב-#C לשרת בשל מהירות השפה ובחרתי ב-React Native כספריה על גבי TypeScript לשם גמישותה עם סוגי מכשירים טלפונים שונים ויכולותיה הרחבות לתכנת UI לאותם מכשירים

## **חלוקה לרכיבים ומודולים:**

* **מחלקת Socket Service:** מחלקה המספקת תרחישים לניהול העברת מידע בין משתמשים ב-Session כגון אם משתמש ישב על שולחן התרחיש יסמן את השולחן כ-"לו פנוי" אצל שאר המשתמשים. השירות הזה גם מספק פונקציונאליות למלצר לארח שולחן לבדוק דרישות משתמש וכו...
* **מחלקת Owner Controller:** מחלקה שמנהלת את פעולות בעל המסעדה אשר לא צריכות חיבור ל-Session כגון מחיקת ארוחות מהתפריט, הוספת ארוחות ופעולות CRUD נוספות על מלצרים ושולחנות
* **מחלקת Security Manager:** מחלקה עם אחריות על הנפקת TOKENS לכל משתמש בנוסף להצפנת נתונים רגישים לפני שמירתן במסד הנתונים כמו סיסמאות.

## **סביבת השרת:**

בחרתי בארכיטקטורת שרת-לקוח עם שרת מקומי לכל מסעדה המשתמשת באפליקציה בכדי לצבור על מחירי ה-Hosting ולהתמודד עם היכולות הפיננסיות של בעלי המסעדות.

## **ממשק המשתמש/לקוח – :GUI**

**עמוד בית (משתמש)**



**תמונה שמכילה טקסט, צילום מסך, גופן

תוכן בינה מלאכותית גנרטיבית עשוי להיות שגוי.Table taken** **Table not taken.**



**תפריט אוכל (משתמש)**

**A screenshot of a phone screen

AI-generated content may be incorrect.**

**עמוד בית (מנהל)**



**תפריט משתמש – הזמנות שרות מהיר קריאה למלצר**

**שליחת הודעה מהירה למלצר (צד לקוח)**



**בקשת סועד (עמוד מלצר)**  **ממשק מלצר – בקשה של סועד.**

בקשה מהסועד להוסיף להזמנה פריט מהתפריט.

## **ממשקים למערכות אחרות : API/**

הפרויקט משתמש ב-API של **RSA** לצורך תקשורת בין השרת ללקוח ולהבאת נתונים של ערכים מתוזמנים, וכן לצורכי אבטחה ובקרה על הגישה למידע רגיש.

## **שימוש בחבילות תוכנה:**

**React**: ספרייה ליצירת ממשק משתמש אינטראקטיבי עם רכיבים.

**Axios**:ספרייה לביצוע בקשות HTTP ולניהול התשובות מהשרת.

**Mongoose** : ספרייה ליצירת מודלים ולניהול הקשר עם מסדי נתונים ב-MongoDB ב-Node.js.

**Express.js** : פריימוורק ליצירת אפליקציות רשת ושרתים ב-Node.js.

**CORS** : מנגנון אבטחה לניהול בקשות ממקורות שונים ולמניעת בעיות גישה בין דומיינים שונים.

# שימוש במבני נתונים וארגון קבצים :

## **ERD**



# תרשימי מערכת מרכזיים:

## **Use Case**



## **Sequence Diagram:**

## **Data Flow**

#### **Context Diagram**



#### **DFD0**



#### **DFD1 for P1 – Find Table**



#### **DFD1 for P5 – Waiter**



#### **מילונים**

#### **מילון ישויות:-**

|  |  |  |
| --- | --- | --- |
| **קוד** | **שם** | **תיאור** |
| E1 | User | ישות המתארת משתמש שסועד במסעדה בעלת האפליקציה |
| E2 | Owner | ישות בעל מסעדה |

#### **מילון תהליכים:-**

|  |  |  |
| --- | --- | --- |
| **קוד** | **שם** | **תיאור** |
| P1 | Find Table | תהליך אשר מנהל הבאת השולחנים מה-Session storage או ממסדי הניתונים במקרה והסועד הוא הסועד הראשון שהתחבר |
| P 2,6 | Meal handler | תהליך המאפשר פעולות CRUD על ארוחות יחד עם אפשרות ללקוח להזמין ארוחות |
| P3 | Special Request | תהליך אשר נותן ללקוח לשלוח הודעה למלצר בדבר אשר הוא דורש כגון "חשבון" |
| P4 | Payment | תהליך שמחזיר תוצאת תשלום על ארוחה "שולם בהצלחה\סירוב" אם בעל הכרטיס שילם באשראי |
| P5 | Waiter Handler | פעולה המאפשרת לבעל מסעדה לבצע פעולות CRUD על מלצרים |
| P5.1 | Add Waiter | תהליך שמתאר פונקציות של בעל מסעדה |
| P5.2 | Remove Waiter | תהליך שמתאר פונקציות של מלצר |
| P5.3 | Get Waiters | תהליך שמתאר פונקציות של בעל מסעדה |
| P7 | Table handler | פעולות CRUD על שולחנות |

#### **מילון מאגרי מידע:-**

|  |  |  |
| --- | --- | --- |
| **קוד** | **שם** | **שדות** |
| D1 | Tables | Id tableNumber  Capacity  waiterId  userId  isOccupied  isWindowSide |
| D2 | Order | TanleNumber  Meals  Quantities  TotalPrice |
| D3 | Waiter | Id name email password date phone |

#### **מילון זרימות מידע:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **קוד** | **שם הזרימה** | **מקור** | **יעד** | **שדות זרימה** |
| F1 | העברה לעמוד בית | 1E | 1P | User\_Id |
| F2 | שליפת שולחנות | 1P | 1D | - |
| F3 | נתוני שולחנות | 1D | 1P | Table\_Id  tableNumber numOfSeats  isWindowSide  isOccupied |
| F4 | נתוני שולחנות | 1P | 1E | Table\_Id  tableNumber numOfSeats  isWindowSide  isOccupied |
| F5 | נתוני ארוחות | E1 | P2 | mealName quantity  tableNumber |
| F6 | נתוני משתמש | P2 | P2 | userId |
| F7 | נתוני ארוחות | P2 | 4P | userId |
| F8 | נתוני משתמש | P2 | E1 | mealName quantity  tableNumber |
| F9 | נתוני הודעת בקשה מיוחדת | 1E | P3 | Messege  sid |
| F10 | פרטי תשלום | 1E | 4P | userId  orderId |
| F11 | אחסון פרטי תשלום | 4P | D2 | userId  orderId |
| F12 | הודעת סיום תשלום | 4P | 1E | - |
| F13 | נתוני מלצר | E2 | P5 | Name  Email  Phone  password |
| F14 | סיום הוספה\מחיקה | P5 | E2 | userId  Name  Email  Phone  password |
| F15 | פרטי ארוחה | E2 | P6 | mealName  price |
| F16 | סיום הוספה\מחיקה | P6 | E2 | mealId  mealName  price |
| F17 | נתוני שולחן | 2E | P7 | Capacity  isWindowside |
| F18 | הודעת הוספה\מחיקה | P7 | 2E | tableNumber  capacity  isWindowside  isOccupied |
| F16.1 | חיבור לסשן מסעדה | E1 | P1 | sid |
| F17.1 | שליפת נתוני שולחנות | P1 | D1 | tableNumber  capacity  isWindowside  isOccupied |
| F18.1 | נתוני שולחנות | D1 | P1 | tableNumber isWindowSide isOccupied  capacity |
| F19 | נתוני שולחנות | P1 | E1 | tableNumber  capacity  isWindowside  isOccupied |
| F20 | נתוני מלצר | E2 | P5.1 | userId  Name  Email  Phone  password |
| F21 | הודעת סיום הוספה | P5.1 | E2 | userId  Name  Email  Phone  password |
| F22 | נתוני מלצר | E2 | P5.2 | userId |
| F23 | הודעת סיום מחיקה | P5.2 | E2 | - |
| F24 | בקשת כל פרטי מלצרים | E2 | P5.3 | TOKEN |
| F25 | נתוני מלצרים | P5.3 | E2 | userId  Name  Email  Phone  password |

# . תיאור המרכיב האלגוריתמי – חישובי



## 

## **איסוף מידע וניתוחים סטטיסטיים (אנליטיקות)**

|  |  |
| --- | --- |
| **מידע** | **מטרתו** |
| נתוני לקוח, מלצר ובעל כמו כתובת ד"אל וסיסמא | נתונים אלו חשובים לשם אימות משתמשיחם והחלפת מידע בעיקר ללקוח שמקבל קבלות במייל |
| נתוני SESSION | נתוני ה-SESSION הם מי שעוזר לנו לראות איזה שולחן הוא פנוי\ ריק הוא מי שנותן ללקוח לשבת על שולחן ולמלצר לשרת שולחן מסוים והוא גם חשוב לפעולות כמו מחיקת מלצר כי חיבור מלצר למערכת (מציאותו במשמרת) מונעת מן הבעל למחוק אותו |
| נתוני ארוחות | ארוחות הם המוצרים העיקריים אשר שירותינו מספק לשם נתינת היכולת למשתמש להזמין הארוחות חייבות להיות שמורות במסד נתונים |
| נתוני שולחנות | השולחנות במסעדה הם הדבר אשר נותן ללקוח את הגישה להזמין (כתוצאה של ישיבתו) ולמלצר את היכולת לשרת את אותו לקוח |

## 

# . תיאור/התייחסות לנושאי אבטחת מידע

הנתונים תחת סכנה בפרויקט זה בצורה עיקרית היו סיסמאות משתמשים לשם כך השתמשתי בספריית bcrypt לשם ביצוע Hashig כך שאוכל לאחסן את הסיסמה של המשתמש בצורה בלתי קריאה ב-MongoDB

שינית השתמשתי ב-JSON Web Token (JWT) על מנת להפריד בין משתמשים ברמות גישה שונות. למשל משתמש רגיל אסור לו לגשת לפונקציות של בעל אז היה ניתן לתת לכל משתמש Role שהשרת יעיין כשהוא יאמת Token מסוים

# משאבים הנדרשים לפרויקט:

## **מספר שעות המוקדש לפרויקט**

|  |  |  |
| --- | --- | --- |
| שעות | מטרה | מבצע/ים |
| 50 | חיפוש ו למידה על ספריות API ו תכניכות לתכנון הפרוייקט | תותרי ג'וליאן |
| 100 | כתיבת ו ארגון קוד לשרת C# | תותרי ג'וליאן |
| 50 | כתיבת קוד ל-client (react native) | תותרי ג'וליאן |
| 100 | בדיקה ו-debug לתכנה ו לשרת ו ל- CLIENT | תותרי ג'וליאן |

## **ציוד נדרש**

החומרה הנדרשת עבור הפרויקט :

* פלפון חכם עם מערכת הפעלה Android או IOS
* מעבד עובד ברמת intel core i3

## **תוכנות נדרשות**

Visual Studio Code

MongoDB

dotnet

Fork

Git/github

PostMan

## **ידע חדש שנדרש ללמוד לצורך ביצוע הפרויקט**

* MVC
* MongoDB
* Mailjet
* Sockets

## **ספרות ומקורות מידע**

* YouTube
* אתר Geeks4Geeks.com
* תיעוד Mailjet: https://dev.mailjet.com/content/guides

# תכנית עבודה ושלבים למימוש הפרויקט

|  |  |  |
| --- | --- | --- |
| 1. | קבלת החלטה על נושא הפרויקט והתנעה ראשונית | 12/2024 |
| 2. | כתיבת הצעה / ניתוח פרויקט וממשקים | 12/09/2025 |
| 3. | Code Review 1 | 12/09/2025 |
| 4. | ועדה פנימית - Code Review 2 | 15/10/2025 |
| 5. | מסירת ספרי פרויקט / המתנה לבחינה | 15/10/2025 |

# תכנון הבדיקות שיבוצעו

|  |  |
| --- | --- |
| תיאור הבדיקה | תגובה צפויה |
| לחיצה על כפתור שולחן | סימון שולחן כ-"לא פנוי" והעברה לדף תפריט |
| לחיצה על כפתור הוספה לארוחה בתפריט | הוספת הארוחה להזמנה |
| לחיצה על Icon בקשות למלצר | העברה לדף בקשות ממלצר |
| לחיצה על כפתור "Wait Table" בטרמינל מלצר | המלצר מסומן כמטפל בשולחן ושאר פוהקציות השולחן כגון "צפייה בהזמנה" יופיעו |
| לחיצה על כפתור "מחק מלצר" בטרמינל הבעל | הודעת הצלחת מחיקה מופיעה אם המלצר עובד אחרת הודעת כישלון |
| לחיצה על כפתור "הוספת ארוחה" ע"י בעל | הארוחה נכתבת אם היא לא נמצאת |
| כפתור "סמן הזמנה כמוכנה" נלחץ ע"י מלצר | תוצג הודעה מתאימה למלצר ותועבר התרעה ללקוח המזמין |
| לחיצה על כפתור "עזוב שולחן" ע"י משתמש | פונקציות השולחן כמו לחצן בקשות ממלצר נעלמים |
| לחיצה על כפתור "צפיה בהזמנה" ע"י מלצר | המלצר מועבר לדף ההזמנה ורואה את פרטיה |

|  |  |
| --- | --- |
|  |  |
|  |  |
| חתימת הסטודנט | חתימת המנחה האישי |

ג. **הערות ראש המגמה**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ד**. אישור ראש המגמה**

|  |  |  |
| --- | --- | --- |
| להב רון | תמונה שמכילה שרטוט, אומנות ילדים, ציור, כתב יד  תוכן בינה מלאכותית גנרטיבית עשוי להיות שגוי. |  |
|  |  |  |
| שם | חתימה | תאריך |

ה. **הערות הגורם המקצועי מטעם מה''ט**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ו**. אישור הגורם המקצועי מטעם מה''ט**

**שם:**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ **חתימה:**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ **תאריך:**\_\_\_\_\_\_\_\_\_\_\_\_

הצהרת סטודנט

שם הסטודנט: ג'וליאן תותרי ת.ז: 214659385

שם המכללה בה לומד הסטודנט: המכללה הטכנולוגית נוף הגליל.

אני הח"מ, מצהיר בזאת כי פרויקט הגמר וספר הפרויקט המצ"ב נעשו על ידי בלבד.

פרויקט הגמר נעשה על סמך הנושאים שלמדתי במכללה ובאופן עצמאי.

פרויקט הגמר וספר הפרויקט נעשו על בסיס הנחייתו של המנחה האישי.

מקורות המידע בהם השתמשתי לביצוע פרויקט הגמר מצוינים ברשימת המקורות המצוינים בספר הפרויקט .

אני מודע לאחריות שהנני מקבל על עצמי על ידי חתימתי על הצהרה זו שכל הנאמר בה אמת ורק אמת .

תמונה שמכילה טיפוגרפיה, גופן, קליגרפיה, כתב יד

תוכן בינה מלאכותית גנרטיבית עשוי להיות שגוי.

חתימת הסטודנט: תאריך: 19/09/2025

אישור המנחה האישי

הריני מאשר שהפרויקט בוצע בהנחייתי, קראתי את ספר הפרויקט ומצאתי כי הוא מוכן לצורך הגשת הסטודנט להגנה על פרויקט גמר .

שם המנחה: רון להב חתימה:  תאריך:19/09/2025

**אישור ראש המגמה**

הריני מאשר שספר הפרויקט מוכן לצורך הגשת הסטודנט להגנה על פרויקט הגמר .

שם ראש המגמה: רון להב חתימה : A close up of a signature

AI-generated content may be incorrect. תאריך:19/09/2025

הבעת תודה

פרויקט זה היווה עבורי הזדמנות נדירה לבחון וליישם את הידע הנרחב והמגוון,

אותו רכשתי במהלך שנות לימודי במגמת הנדסת תוכנה במכללה הטכנולוגית נוף הגליל .

חשוב לי, להביע את תודתנו הרבה והערכתנו לכל אלו אשר תמכו , סייעו ועזרו לי לכל אורך הדרך.

ברצוני להודות :

למנחה ו ראש המגמה רון להב- על הנחיית הפרויקט, והעזרה המקצועית לאורך הדרך, תודה על הסבלנות והשקעת הזמן, תוך דגש על הקידום המקצועי שרכשתי דרכך.

**מ"ר שרארי פיראס-** על הדרכתו והובלת יעוץ לספריות אשר בהן אוכל להשתמש ב-React Native ועל הליווי שלי ושל שאר התלמדים בקורס ה-Full Stack.

**מ"ר שרארי איאס-** על העברתו ליעוץ כלפי בניית השרת ועל שיפור היכולת שלי ושל שאר התלמידים לפתור בעיות אלגוריתמיקה ותכנות בשעות התרגול אשר הוא העביר במכללה.

תקציר

מאז השעה בה סמארטפונים יצאו לעולם ראינו עלייה בשאיפת עסקים שונים להפוך את עצמם לעסקים דיגיטליים לו ברמה חלקית וזאת בגלל סיבה עיקרית אחת נוחות. נוחות מהירות וגישה קלה הם המוטו של המאה ה-21 בשל כך יצרתי את אפליקציית "Bite Byte" אפליקציית "Bite Byte" מובילה 3 שירותים בקופסה אחת והן כנ"ל

השירות הראשון הינו שירות ללקוחות מסעדה אשר דרכו ניתן לסמן את עצמך כיושב על שולחן, להזמין אוכל, ולתקשר עם המלצר שלך לגבי הצרכים שלך תוך כדי זמן ישיבתך במסעדה.

השירות השני הוא שירות טרמינל מלצר שנותן לעובדי המסעדה לצפות בדרישות הלקוחות כמו הזמנות והודעות מהירות שהסועד יכול לשלוח למלצר על מנת להביע את דרישותיו. דבר זה מהווה הקלה לעבודת המלצר.

השירות השלישי והאחרון הוא טרמינל בעל מסעדה. טרמינל בעל המסעדה נותן לו אוסף של פעולות כמו הוספת\הורדת עובדים, הוספת\הורדת ארוחות, והוספת\הורדת שולחנות

תוכן

[1. שם הפרויקט 3](#_Toc210150830)

[2. רקע 3](#_Toc210150831)

[**2.1.** **תיאור ורקע כללי** 3](#_Toc210150832)

[**2.2.** **מטרות המערכת** 3](#_Toc210150833)

[3. סקירת מצב קיים בשוק, אלו בעיות קימות 3](#_Toc210150834)

[4. מה הפרויקט אמור לחדש או לשפר 3](#_Toc210150835)

[5. דרישות מערכת ופונקציונאליות 4](#_Toc210150836)

[5.1. **דרישות מערכת** – סביבת הטמעה ושימוש. שרידות, ביצועים\ התמודדות עם עומסים. 4](#_Toc210150837)

[**5.2.** **דרישות פונקציונאליות – רשימת דרישות המשתמש מהמערכת, מהן הפעולות בהן נדרשת המערכת לתמוך.** 4](#_Toc210150838)

[6. בעיות צפויות במהלך הפיתוח ופתרונות: 9](#_Toc210150839)

[7. פתרון טכנולוגי נבחר: 9](#_Toc210150840)

[**7.1.** **טופולוגית הפתרון:** 9](#_Toc210150841)

[7.2. **טכנולוגיות בשימוש**: 9](#_Toc210150842)

[**7.3.** **שפות הפיתוח:** 10](#_Toc210150843)

[**7.4.** **תיאור הארכיטקטורה הנבחרת:** 10](#_Toc210150844)

[**7.5.** **חלוקה לרכיבים ומודולים:** 10](#_Toc210150845)

[**7.6.** **סביבת השרת:** 10](#_Toc210150846)

[**7.7.** **ממשק המשתמש/לקוח – :GUI** 11](#_Toc210150847)

[**7.8.** **ממשקים למערכות אחרות : API/** 15](#_Toc210150848)

[**7.9.** **שימוש בחבילות תוכנה:** 15](#_Toc210150849)

[8. שימוש במבני נתונים וארגון קבצים : 15](#_Toc210150850)

[**8.1.** **ERD** 15](#_Toc210150851)

[9. תרשימי מערכת מרכזיים: 16](#_Toc210150852)

[**9.1.** **Use Case** 16](#_Toc210150853)

[**9.2.** **Sequence Diagram:** 17](#_Toc210150854)

[**9.3.** **Data Flow** 18](#_Toc210150855)

[10. . תיאור המרכיב האלגוריתמי – חישובי 24](#_Toc210150856)

[**10.1.** **איסוף מידע וניתוחים סטטיסטיים (אנליטיקות)** 25](#_Toc210150857)

[11. . תיאור/התייחסות לנושאי אבטחת מידע 25](#_Toc210150858)

[12. משאבים הנדרשים לפרויקט: 26](#_Toc210150859)

[**12.1.** **מספר שעות המוקדש לפרויקט** 26](#_Toc210150860)

[**12.2.** **ציוד נדרש** 26](#_Toc210150861)

[**12.3.** **תוכנות נדרשות** 26](#_Toc210150862)

[**12.4.** **ידע חדש שנדרש ללמוד לצורך ביצוע הפרויקט** 26](#_Toc210150863)

[**12.5.** **ספרות ומקורות מידע** 26](#_Toc210150864)

[13. תכנית עבודה ושלבים למימוש הפרויקט 27](#_Toc210150865)

[14. תכנון הבדיקות שיבוצעו 27](#_Toc210150866)

[15. רשימת דפים 34](#_Toc210150867)

[16. מבוא 39](#_Toc210150868)

# רשימת דפים

1. דף כניסה
2. דף בחירת שולחן / דף שיחת מלצר
3. תצוגת תפריט
4. דף הודעות מהירות
5. דף תפריט
6. דף כניסה לצוות/בעלים
7. דף תפעול בעל מסעדה
8. דף הוסף מלצר
9. דף הסר מלצר
10. דף הוסף ארוחה
11. דף הסר ארוחה
12. דף הוסף הודעה מהירה
13. דף הסר הודעה מהירה
14. דף הוסף שולחן
15. דף הסר שולחן
16. דף הצג שולחן תפוס (מלצר)
17. דף הצג הזמנות אוכל (מלצר)
18. דף הצג הזמנות/הודעות מהירות
19. כניסה ראשונה

(1 דף כניסה 2) דף בחירת שולחן / דף שיחת מלצר



תמונה שמכילה טקסט, צילום מסך, תוכנה, סמל מחשב

תוכן בינה מלאכותית גנרטיבית עשוי להיות שגוי.3)תצוגת תפריט 4)דף הודעות מהירות 5) דף תפריט

תמונה שמכילה טקסט, צילום מסך, תוכנה, גופן

תוכן בינה מלאכותית גנרטיבית עשוי להיות שגוי.תמונה שמכילה טקסט, צילום מסך, תוכנה

תוכן בינה מלאכותית גנרטיבית עשוי להיות שגוי.

6) דף כניסה לצוות/בעלים 7) דף תפעול בעל מסעדה

תמונה שמכילה טקסט, צילום מסך, גופן, עיצוב

תוכן בינה מלאכותית גנרטיבית עשוי להיות שגוי.

8) דף הוסף מלצר 9) דף הסר מלצר

תמונה שמכילה טקסט, צילום מסך, תוכנה, תכונות מולטימדיה

תוכן בינה מלאכותית גנרטיבית עשוי להיות שגוי.תמונה שמכילה טקסט, צילום מסך, עיצוב

תוכן בינה מלאכותית גנרטיבית עשוי להיות שגוי.

10) דף הוסף ארוחה 11) דף הסר ארוחה

תמונה שמכילה טקסט, צילום מסך, תוכנה, גופן

תוכן בינה מלאכותית גנרטיבית עשוי להיות שגוי.

תמונה שמכילה טקסט, צילום מסך, תוכנה, תכונות מולטימדיה

תוכן בינה מלאכותית גנרטיבית עשוי להיות שגוי.12) דף הוסף הודעה מהירה 13) דף הסר הודעה מהירה



14) דף הוסף שולחן 15) דף הסר שולחן

תמונה שמכילה טקסט, צילום מסך, גופן, תוכנה

תוכן בינה מלאכותית גנרטיבית עשוי להיות שגוי.

תמונה שמכילה טקסט, צילום מסך, עיצוב

תוכן בינה מלאכותית גנרטיבית עשוי להיות שגוי.16) דף הצג שולחן תפוס (מלצר) 17) דף הצג הזמנות אוכל (מלצר)

תמונה שמכילה טקסט, צילום מסך, תוכנה, תכונות מולטימדיה

תוכן בינה מלאכותית גנרטיבית עשוי להיות שגוי.

18) דף הצג הזמנות/הודעות מהירות (מלצר) 19)כניסה ראשונה

תמונה שמכילה טקסט, צילום מסך, תוכנה, עיצוב

תוכן בינה מלאכותית גנרטיבית עשוי להיות שגוי.תמונה שמכילה טקסט, צילום מסך, תוכנה, גופן

תוכן בינה מלאכותית גנרטיבית עשוי להיות שגוי.

# מבוא

בעידן הדיגיטלי המודרני, מסעדות רבות מחפשות דרכים לייעל את עבודתן ולהעניק ללקוחות חווית שירות מהירה, נוחה ויעילה יותר. בעיה מרכזית הקיימת כיום ברוב המסעדות היא איטיות השירות וזמני המתנה ממושכים, הנובעים מתהליכי עבודה ידניים וחוסר ניצול של כלים טכנולוגיים מתקדמים.

פרויקט זה מציע מערכת **"תפריט חכם" (Smart Menu)**, אשר נועדה לנהל את הקשר בין הלקוח, המלצר והמטבח בצורה מקוונת ודיגיטלית. המערכת מציגה תפריט דיגיטלי ומאפשרת ללקוח לבצע פעולות מגוונות – החל מסימון ישיבה על שולחן, דרך הזמנת אוכל ושתייה, שליחת בקשות מהירות למלצר, ועד תשלום חשבון בכרטיס אשראי – והכול דרך האפליקציה בלבד, ללא צורך בהמתנה למלצר.

באמצעות המערכת מושגות מספר מטרות מרכזיות:

1. **קיצור זמני ההמתנה במסעדה** – הפחתת התלות במלצרים לביצוע פעולות בסיסיות והעברת השליטה לידי הלקוח.
2. **ייעול תהליכי העבודה** – הפיכת פעולות ידניות רבות לפעולות דיגיטליות, מה שמאפשר למלצרים להתמקד בשירות אישי ואיכותי יותר.
3. **שיפור חווית המשתמש** – יצירת ממשק נגיש, מהיר ואינטואיטיבי, שמבטיח ללקוח חווית שירות חדשנית.
4. **שילוב טכנולוגיות מתקדמות** – שימוש בכלים כגון Sockets לעדכון בזמן אמת, מנגנוני אבטחת מידע (כגון הצפנת סיסמאות ו־JWT), ובסיס נתונים גמיש לניהול מידע עסקי ולקוח.

ייחודיות המערכת היא בכך שהיא אינה מסתפקת בהצגת תפריט בלבד, אלא מעניקה פתרון כולל לניהול כלל הפעולות במסעדה – החל מבקשת שירות פשוטה ועד לניהול מלאי ותיאום בין הגורמים השונים. בכך היא מביאה בשורה חדשה לעולם המסעדנות, ומאפשרת לשלב בין חווית סועד מתקדמת לניהול מסעדה יעיל ורווחי יותר.

# קוד

* צד כניסה ללקוחות ולעובדים
* שפת: React Native, TypeScript
* לקוחות מזינים אימייל וסיסמה ונכנסים לפריסת הטאבים; צוות מזין אימייל וסיסמה ונכנס לטרמינל הצוות; למשתמש ללא חשבון קיימת אפשרות הרשמה.

import { ThemedText } from '@/components/ThemedText';

import { ThemedView } from '@/components/ThemedView';

import React, { useState } from 'react';

import { StyleSheet, View } from 'react-native';

import { useNavigation } from '@react-navigation/native';

import Logo from "@/components/ui/Logo";

import CurvedButton from '@/components/ui/CurvedButton';

import axios from 'axios';

import ip from '@/Data/Addresses';

import AsyncStorage from '@react-native-async-storage/async-storage';

import ThemedInput from '@/components/ThemedInput';

import { NavigationProp } from '@/Routes/NavigationTypes';

/\*\*

 \* A login screen for customers and staff to log in to the system.

 \* Customers can log in with their email and password to access the tabs layout.

 \* Staff can log in with their email and password to access the staff terminal.

 \* If the user does not have an account, they can sign up.

 \*

 \* @remarks

 \* This component uses the `useNavigation` hook from `@react-navigation/native` to

 \* navigate to the tabs layout or the signup screen.

 \*

 \* @returns A JSX element representing the login screen.

 \*/

export default function Login() {

  const [email, setEmail] = useState<string>('');

  const [password, setPassword] = useState<string>('');

  const navigation = useNavigation<NavigationProp>();

  const styles = StyleSheet.create({

    headerImage: {

      color: '#808080',

      bottom: -90,

      left: -35,

      position: 'absolute',

    },

    container: {

      flex: 1,

      justifyContent: 'center',

      alignItems: 'center',

      padding: 20,

      gap: 20,

    },

    emailText: {

      fontSize: 16,

      fontWeight: 'bold',

      marginBottom: 10,

    },

    signupContainer:{

      flexDirection: 'row',

      justifyContent: 'center',

      alignItems: 'center',

      gap: 20,

    },

    staff:{

      flexDirection: 'row',

      justifyContent: 'space-around',

      alignItems: 'center',

      width: '90%',

      backgroundColor:"transparent"

    },

    text:{

      fontSize: 25,

      fontWeight: 'bold',

      height: 'auto',

      width: 'auto',

    },

  });

  /\*\*

   \* Handles the login process for customers and staff.

   \* Performs validation and sends a request to the server to

   \* authenticate the user.

   \* If the user is authenticated, sets the token and user data in

   \* AsyncStorage and navigates to the tabs layout.

   \* If the user is not authenticated, displays an error message.

   \*/

  const handleLogin = async() => {

    // Perform login validation (optional)

    if (email && password) {

      // Send login request to the server

      try{

        const res = await axios.post(`http://${ip.julian}:5256/api/user/login`,{

            email: email,

            password: password

        })

        if(res&& res.status===200){

            await AsyncStorage.setItem('token', res.headers['x-auth-token']);

            await AsyncStorage.setItem('user', JSON.stringify(res.data.user));

            const user =await AsyncStorage.getItem('user')

        }

        // Navigate to the tabs layout

        navigation.reset({

          index: 0,

          routes: [{ name: 'Tabs' }],

        });

      }

      catch(e:any){

        alert(e.message);

      }

    } else {

      alert('Please enter email and password!');

    }

  };

  return (

    <ThemedView style={styles.container}>

      <ThemedText style={styles.text}>Customer Login</ThemedText>

      <Logo />

      <ThemedInput

        type="email-address"

        value={email}

        placeholder="Email"

        action={(text)=>setEmail(text)}

        />

      <ThemedInput

        placeholder='Password'

        type="password"

        value={password}

        action={(text) => setPassword(text) }

      />

      <CurvedButton

        title="Login"

        action={handleLogin}

        style={{ backgroundColor: '#34baeb' }}

      />

      <View style={styles.signupContainer}>

        <ThemedText style={styles.emailText}>Don't have an account?</ThemedText>

        <CurvedButton

          title="Sign Up"

          action={() => navigation.navigate('Signup')}

          style={{backgroundColor:"rgb(134, 0, 175)"}}

          />

      </View>

      <View style={styles.staff}>

          <ThemedText style={styles.emailText}>Are you staff?</ThemedText>

          <CurvedButton

            title="Staff Terminal"

            action={()=>navigation.navigate('Terminals')}

            style={{backgroundColor:"rgb(0, 175, 122)"}}

            />

      </View>

    </ThemedView>

  );

}

* צד הרישום הראשון
* שפת: React Native, TypeScript

import { ThemedText } from '@/components/ThemedText';

import { ThemedView } from '@/components/ThemedView';

import React, { useState } from 'react';

import { TextInput, StyleSheet, Button, TouchableOpacity,Platform  } from 'react-native';

import DateTimePicker from '@react-native-community/datetimepicker';

import { useTheme } from '@react-navigation/native';

import Logo from "@/components/ui/Logo";

import axios from 'axios';

import ip from '@/Data/Addresses'

import CurvedButton from '@/components/ui/CurvedButton';

import ThemedInput from '@/components/ThemedInput';

import ShowMessageOnPlat from '@/components/ui/ShowMessageOnPlat';

/\*\*

 \* Signup component

 \*

 \* @return {JSX.Element} - Signup component

 \*/

export default function Signup() {

    const [name, setName] = useState('');

    const [email, setEmail] = useState('');

    const [password, setPassword] = useState('');

    const [dob, setDob] = useState(''); // Store as YYYY-MM-DD

    const [dobDate, setDobDate] = useState(new Date(2000, 0, 1)); // Default to Year 2000

    const [phone, setPhone] = useState('');

    const [showDatePicker, setShowDatePicker] = useState(false);

    const { colors, dark: isDark } = useTheme();

/\*\*

 \* Handles the change of the date selection from the date picker.

 \* Updates both the display date and the formatted date of birth.

 \*

 \* @param {any} event - The event triggered by the date picker.

 \* @param {Date} [selectedDate] - The date selected by the user.

 \*                                If a date is selected, the function updates the

 \*                                state with the formatted date in 'YYYY-MM-DD' format.

 \*/

    const handleDateChange = (event:any, selectedDate?:Date) => {

        if (selectedDate) {

            setDobDate(selectedDate);

            // ✅ Fix: Use toLocaleDateString() to prevent timezone issues

            const formattedDate = selectedDate.toLocaleDateString('en-CA'); // YYYY-MM-DD format

            setDob(formattedDate);

        }

        setShowDatePicker(false); // Hide picker after selecting

    };

    /\*\*

     \* Handles the sign up request by sending a post request to the backend

     \* with the user's details. If the request is successful, it shows an alert

     \* with the response message. If there is an error, it shows an alert with

     \* the error message.

     \*

     \* @function

     \* @async

     \* @returns {undefined}

     \*/

    const handleSignup = async() => {

        try{

            const res = await axios.post(`http://${ip.julian}:5256/api/user/signup`,{

                name:name,

                email: email,

                password: password,

                date: dob,

                phone: phone

            })

            if(res && res.status===200){

                ShowMessageOnPlat(res.data);

            }

        }

        catch(e:any){

            alert(e.message);

        }

    };

    const styles = StyleSheet.create({

        container: {

            flex: 1,

            justifyContent: 'center',

            alignItems: 'center',

            padding: 20,

            gap: 20,

          },

        header:{

            fontWeight: 'bold',

            fontSize: 25,

        },

        input: {

            height: 40,

            width: '95%',

            borderWidth: 0.5,

            borderRadius: 10,

            backgroundColor: 'transparent',

            paddingHorizontal: 10,

            marginBottom: 20,

            color: isDark ? "white" : "black",

            borderColor: isDark ? "white" : "black",

          },

        button: {

            marginTop: 20,

            backgroundColor: 'rgb(134, 0, 175)',

        },

        opacity:{

            width: '100%',

           justifyContent: 'center',

           alignItems: 'center',

        }

    });

    return (

        <ThemedView style={styles.container}>

            <Logo />

            <ThemedText style={styles.header}>Sign up</ThemedText>

            <ThemedInput

                type="text"

                 placeholder="Name"

                action={(text) => setName(text)}

                value={name}/>

            <ThemedInput

                placeholder="example@domain.com"

                type={"email-address"}

                action={(text) => setEmail(text)}

                value={email}

            />

            <ThemedInput

                placeholder="Password"

                type="password"

                action={(text) => setPassword(text)}

                value={password}

            />

               <TouchableOpacity style={styles.opacity} onPress={() => setShowDatePicker(true)} >

                <TextInput

                    style={styles.input}

                    placeholder="Date of Birth"

                    placeholderTextColor={isDark ? "white" : "black"}

                    value={dob}

                    editable={false} // Prevent manual typing

                />

            </TouchableOpacity>

            {/\* Show picker when `showDatePicker` is true \*/}

            {showDatePicker && (

                <DateTimePicker

                    value={dobDate} // Default date

                    mode="date"

                    display={"spinner"} // 🔥 Now "spinner" on Android

                    onChange={handleDateChange}

                    maximumDate={new Date()} // Prevent selecting future dates

                />

            )}

            <ThemedInput

              placeholder='Phone Number'

                type="phone-pad"

                action={(text) => setPhone(text)}

                value={phone}

            />

            <CurvedButton

                title="Signup"

                action={async() => await handleSignup()}

                style={ {backgroundColor: "rgb(134, 0, 175)"}}

            />

        </ThemedView>

    );

}