משרד התעשיה המסחר והתעסוקה

מה"ט המכון הממשלתי להכשרה בטכנולוגיה ובמדע.



## המגמה: תוכנה נז'

פרויקט גמר

הנושא: LHOTEL

##### **המגיש/ה: ­­­\_\_\_מאור שטרן\_\_\_**

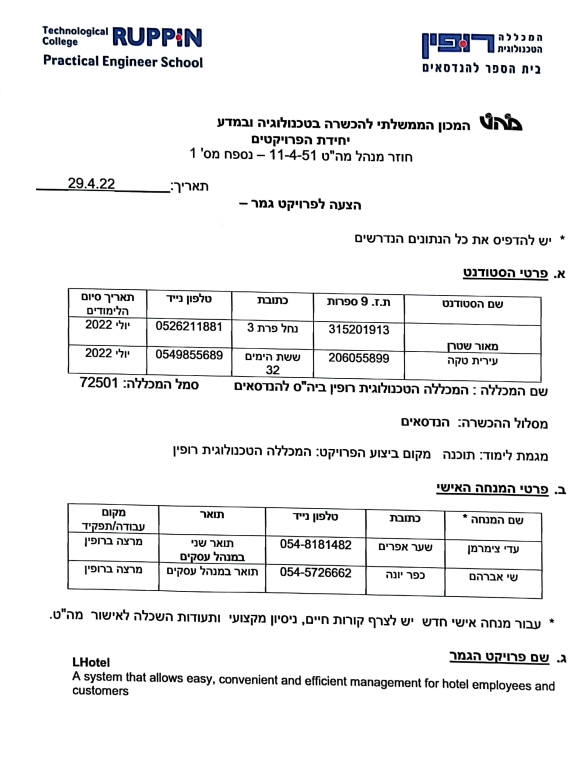
##### **ת.ז.:\_\_\_\_\_\_315201913\_\_\_\_\_**

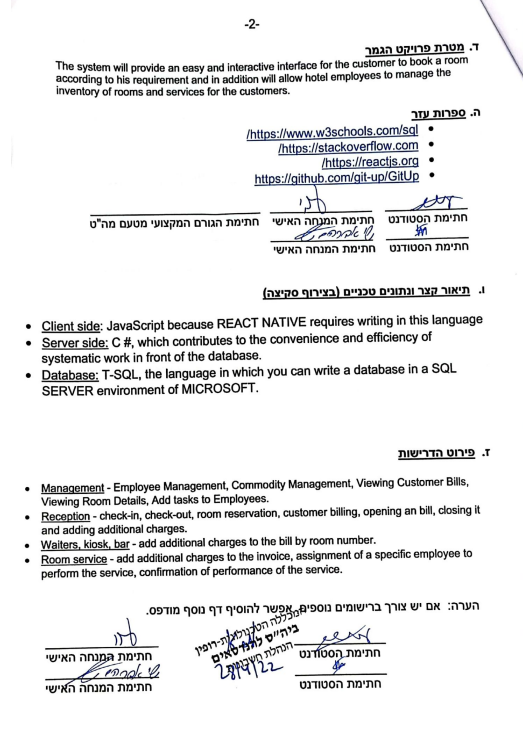
##### **המגיש/ה: ­­­\_\_\_\_\_\_עירית טקה\_\_\_**

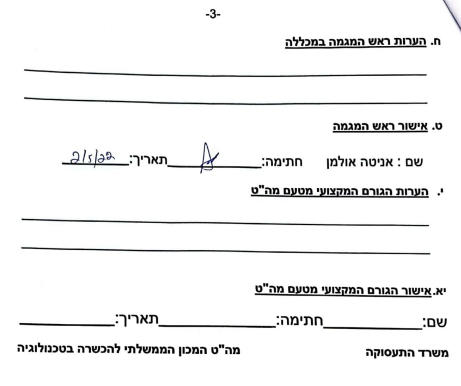
##### **ת.ז.:\_\_\_\_\_\_\_\_206055899\_\_\_\_\_**

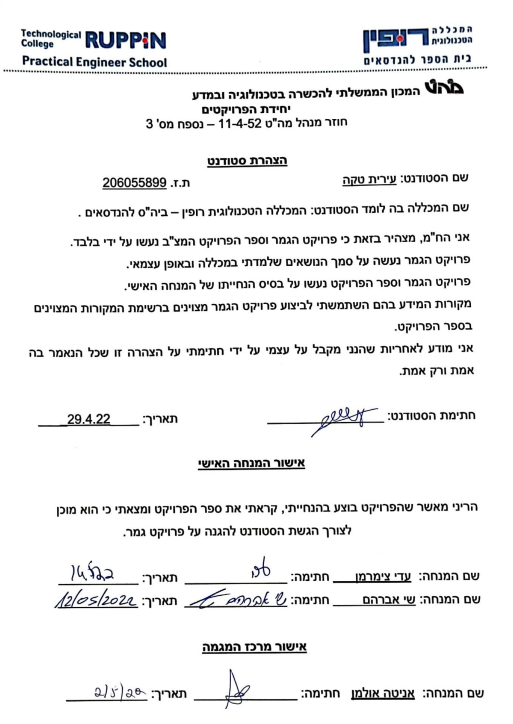
**המנחה: עדי צימרמן**

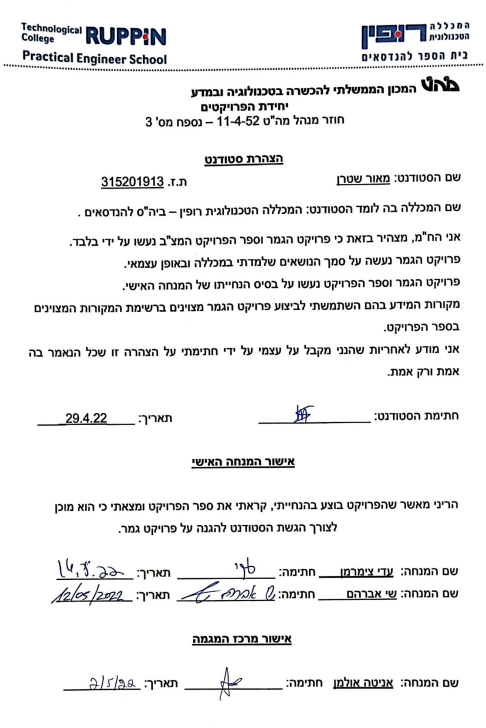
**המנחה:\_\_\_\_אברהם שי\_\_\_\_\_\_**











1. **שם הפרויקט:**

Hotel management system

A system that allows easy, convenient and efficient management for hotel employees and customers

1. **רקע**
   1. **תיאור ורקע כללי**

A complete system that allows management and monitoring of the full rooms, and their association for dedicated marketing purposes, provides control and supervision over all hotel activities.

* 1. **מטרות המערכת**

The system will provide an easy and interactive interface for the customer to book a room according to his requirement and in addition will allow hotel employees to manage the inventory of rooms and services for the customers.

1. **המצב בשוק והבעיות**
   1. **סקירת מצב קיים בשוק**

Booking - the best travel app with deals for booking hotels and accommodation units for vacations, booking accommodation including hotels, apartments and resorts.

Advantages:

1. Information Security and Privacy - Booking's servers are secure and keep customers' information secure.

2. Direct booking with the hotel - The advantage of direct booking is that the chance of cancellations and / or changes of any kind is almost non-existent. This also reduces the commissions that have to be paid to the giant companies for packages.

Disadvantages:

1. Since the reservation is directly in front of the hotel, it is advisable to check the recommendations on the hotel itself and the credit card charge. There are hotels that sometimes overlap, make mistakes or just charge different charges than what you have chosen.

2. Booking is one of the leading sites in the field of consumer pressure in e-commerce, sometimes even under the sign of "illegal".

Portel – Protel's intuitive hotel management system helps hotels enhance their guest experience. Continuous and smooth communication between departments. Accounting functions for monitoring any financial transaction. All this and more - whether in the cloud or in a hotel. Booking engine allows hotels to tailor the visibility of their booking engine to their exact needs. An app that offers hotels an easy solution for advertising their branded app, driven directly by protel's hotel system

Advantages:

1. You do not pay for non-arrival or cancellations, there is only a one-time token payment paid 30 days after booking so there is no down payment.

2. A cloud-based system that protects the system in the event of information loss.

3. Protel, as an authorized integration partner and PMS provider, provides Google hotel promotion with the availability, rates and any other information you would like to include.

Disadvantages:

1. Due to the sensitive customer identity holders and the full financial detail of the hotel the hotel management system requires a very strong data protection system, which can be very harmful in case of hacking and retrieving sensitive data or changes.
2. The screen is hard to read on a lap top as the lettering is all light grey. Even when switching to the dark background it is a little difficult to read. It seems there are a lot of clicks to get one function completed. Although I love the aesthetics of the program, but I find it difficult to use. The program is constantly down or scrolling or pausing. There seem to be lots of issues with accessibility and a long lag time.
   1. **בעיות במצב הקיים**

Currently there is no system that provides all the services and all the required and expected services from a large system.

There is an interface that focuses only on the management of the hotel itself and there are apps that focus only on customer orders and today there is no system that combines both, which forces customers to mess with different apps and hotel management with different interfaces.

1. **מה הפרויקט אמור לחדש או לשפר**

* Convenient user interface
* Free of charge system
* Online access of employees to the system according to their role
* Adding additional charges per room without the need for payment
* More efficient employee and task management

1. **דרישות מערכת ופונקציונאליות**
   1. **דרישות מערכת**

* Smartphone / tablet
* Internet connection
* information system
* granting permissions by position
* setting up inactivity for an employee who leaves
  1. **דרישות פונקציונאליות**
* Management - Employee Management, assignment of a specific employee to perform the service, Commodity Management, Viewing Customer Bills, Viewing Rooms.
* Reception - check-in, check-out, room reservation, customer billing, opening an bill, closing it and adding additional charges.
* Room service - add additional charges to the invoice, confirmation of performance of the service.
* Customer - registration to the system , viewing and ordering products ,reservations ,check in / out, room service request

1. **בעיות צפויות במהלך הפיתוח ופתרונות**

|  |  |  |
| --- | --- | --- |
| Type of problem | Description | Solution |
| Operational | Retrieving records from the database takes longer than expected | Displays to the user that retrieval takes longer than expected |
| Operational | Updating records in the database is not performed | The record update will be written to the database log via transaction. Also, each action will be written to a log table using triggers and checked every 24 hours |
| Loads | User load on the server may cause it to crash | Resource allocation and CDN (Content Delivery Network) activation. In the extreme case, we will limit the amount of active users at any given moment. (Under the responsibility of the server administrator) |

1. **פתרון טכנולוגי נבחר**
   1. **טופולוגית הפתרון**

As part of a final project, the solution we offer an application which will be accessible through the user mobile phone/ tablet.

The application will connect to the C # server over the Internet as customized in the client-server model.

* 1. **טכנולוגיות בשימוש**

Client side: react native because a solution can be exported for android and iOS systems.

Server Side:

* WEB API.
* RESTFUL API that enables CRUD creation efficiently.

Database: The database will be SQL that allows working with a relational database with a clear schema.

* 1. **שפות הפיתוח.**

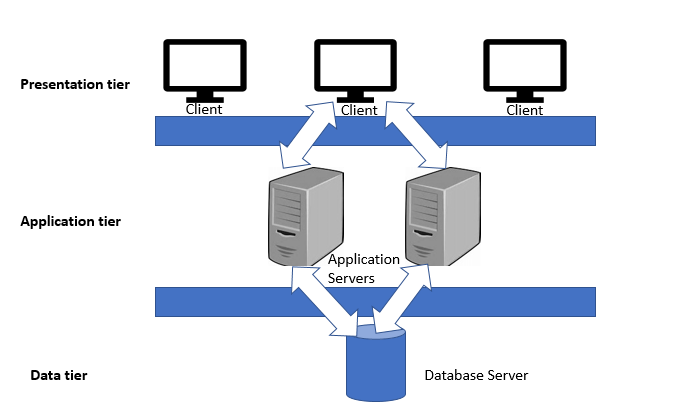
Client side: JavaScript because REACT NATIVE requires writing in this language

Server side: C #, which contributes to the convenience and efficiency of systematic work in front of the database.

Database: T-SQL, the language in which you can write a database in a SQL SERVER environment of MICROSOFT.

* 1. **תיאור הארכיטקטורה הנבחרת**

The selected architecture is a client server model



* 1. **חלוקה לתכניות ומודולים**

**­­**

* 1. **סביבת השרת**

The development environment will be in front of a local server and database.

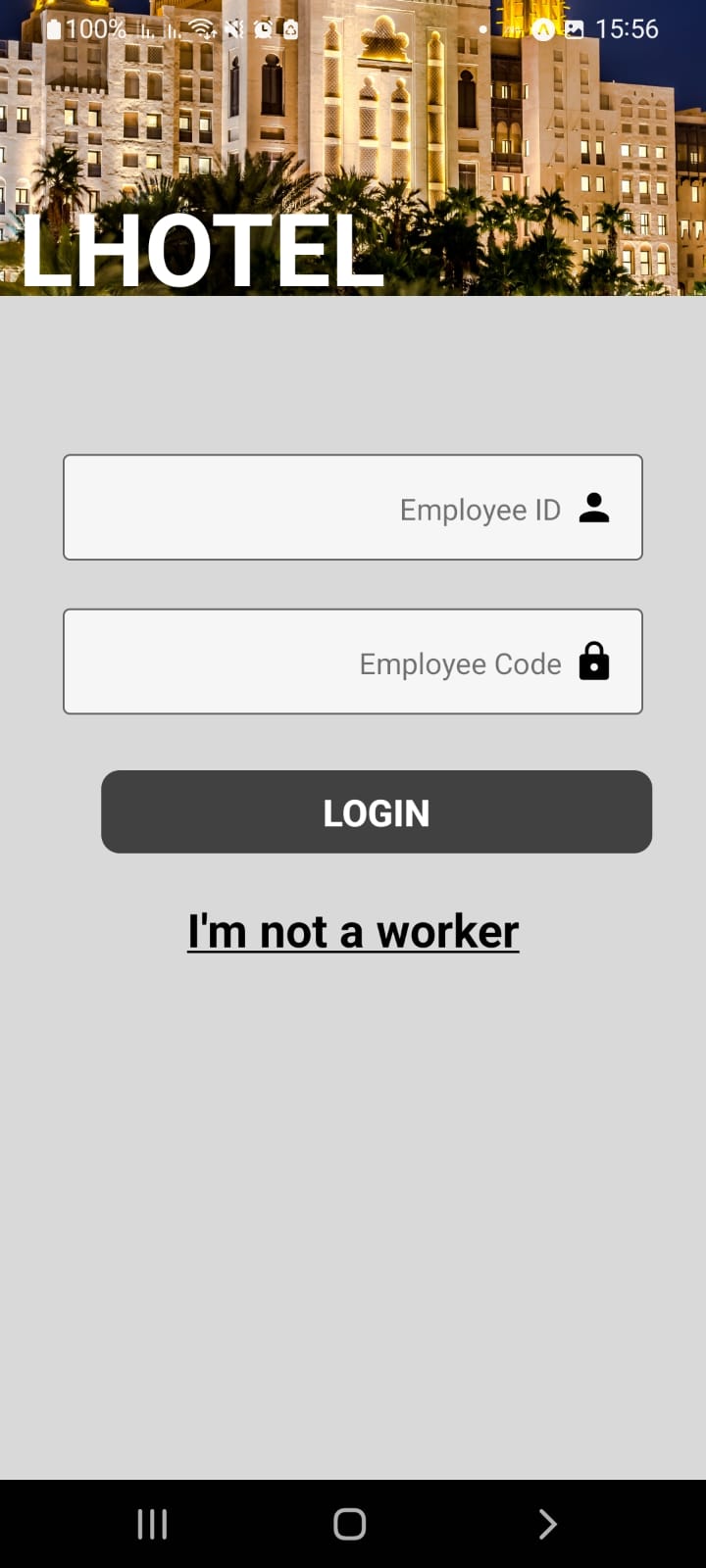
The production environment will be hosted by "My Windows Hosting", which enables .NET work, and Microsoft's SQL Server

* 1. **ממשק המשתמש/לקוח – GUI**

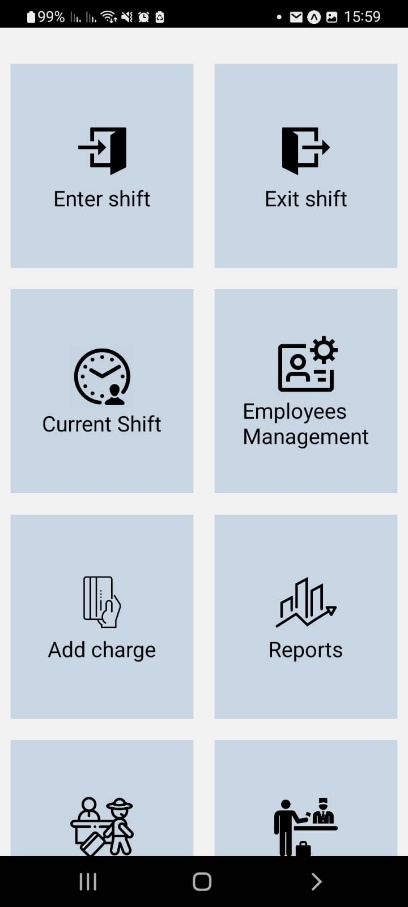
**home screen**



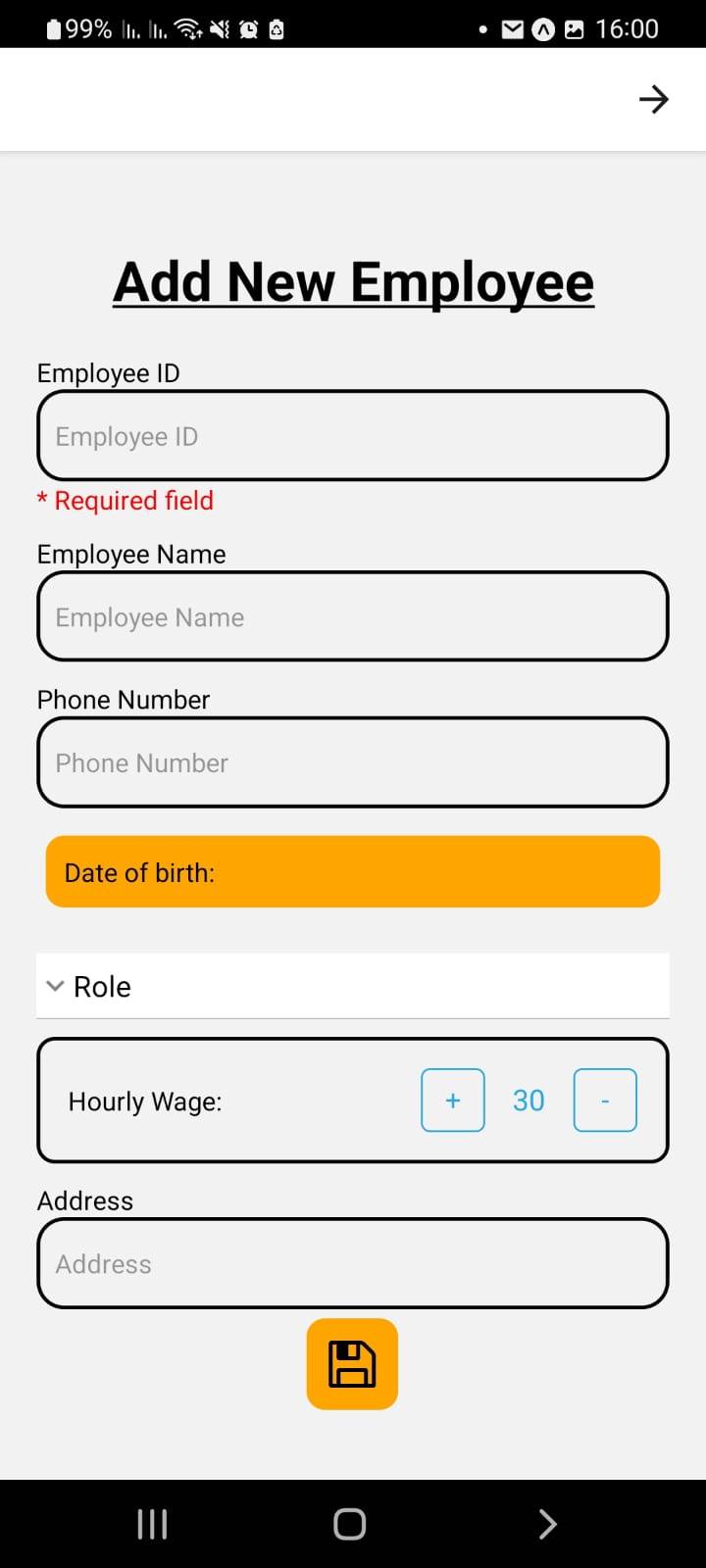
**Login - worker**



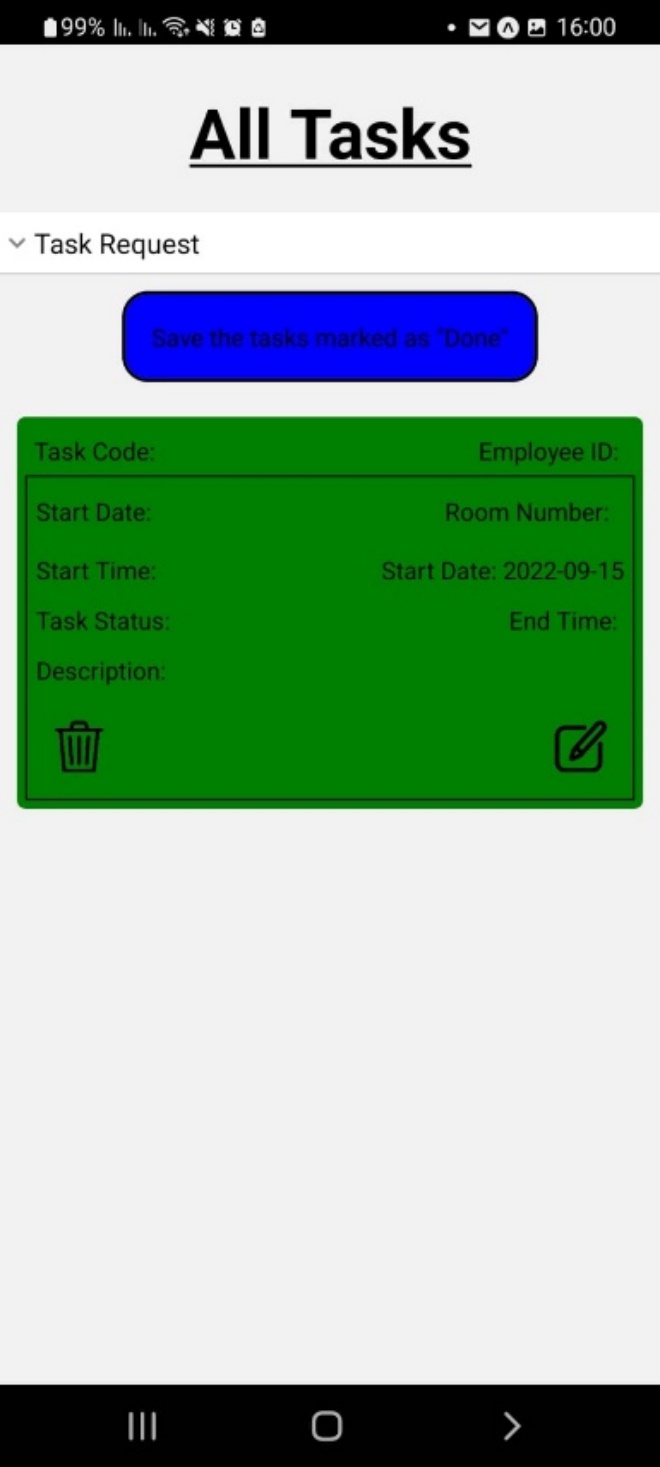
**Worker menu**



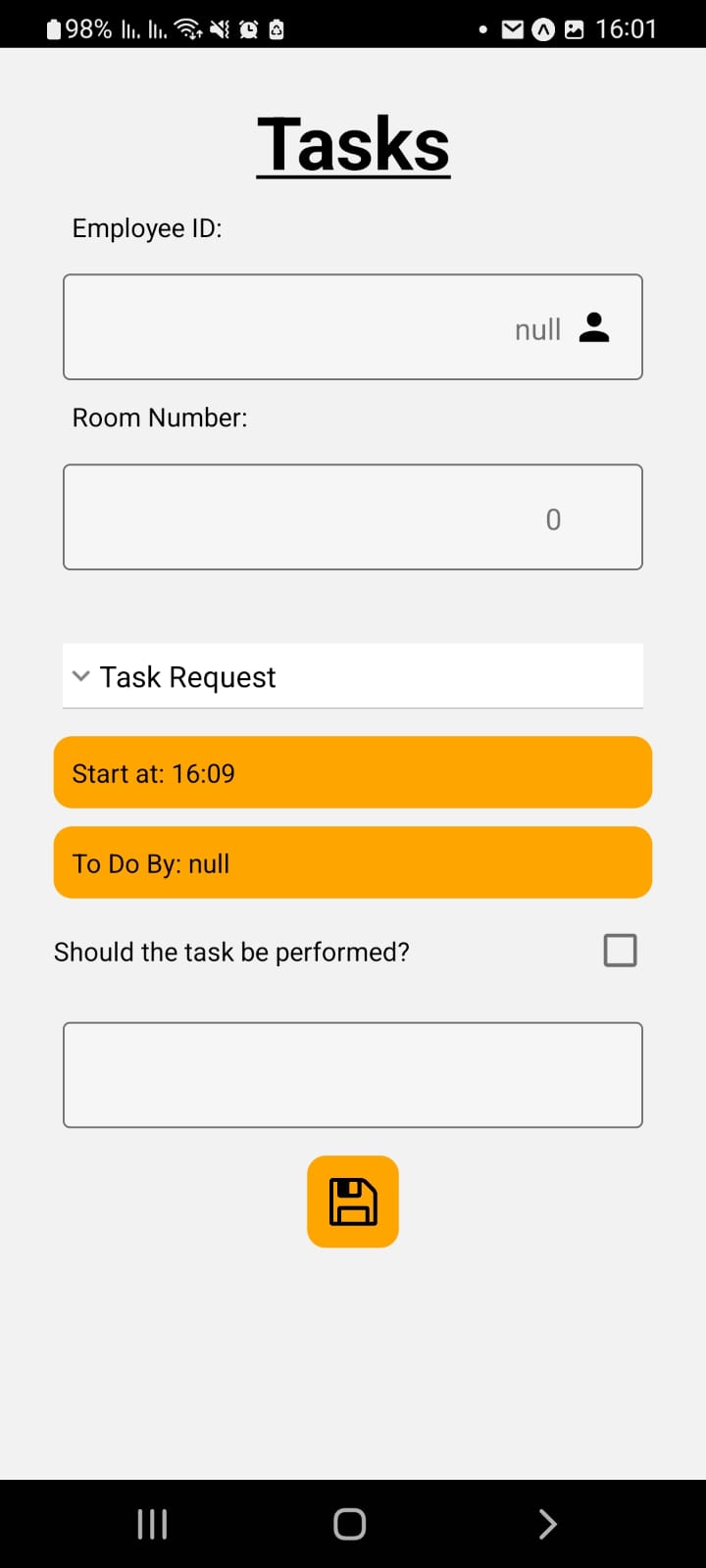
**new employee Add**



**All tasks**



**Update task**

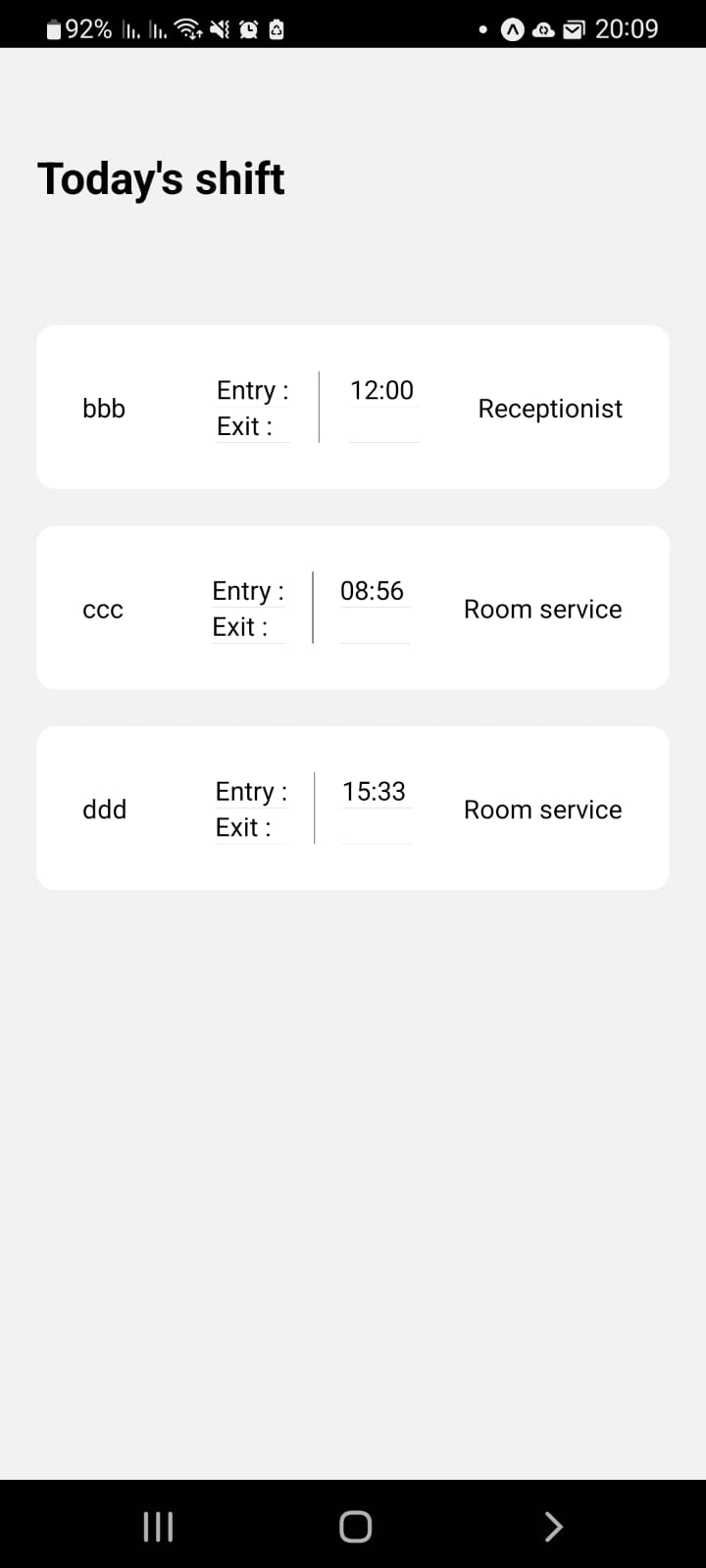


**Employees management**

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

**Today's shift**



**Add charge**

תמונה שמכילה שולחן

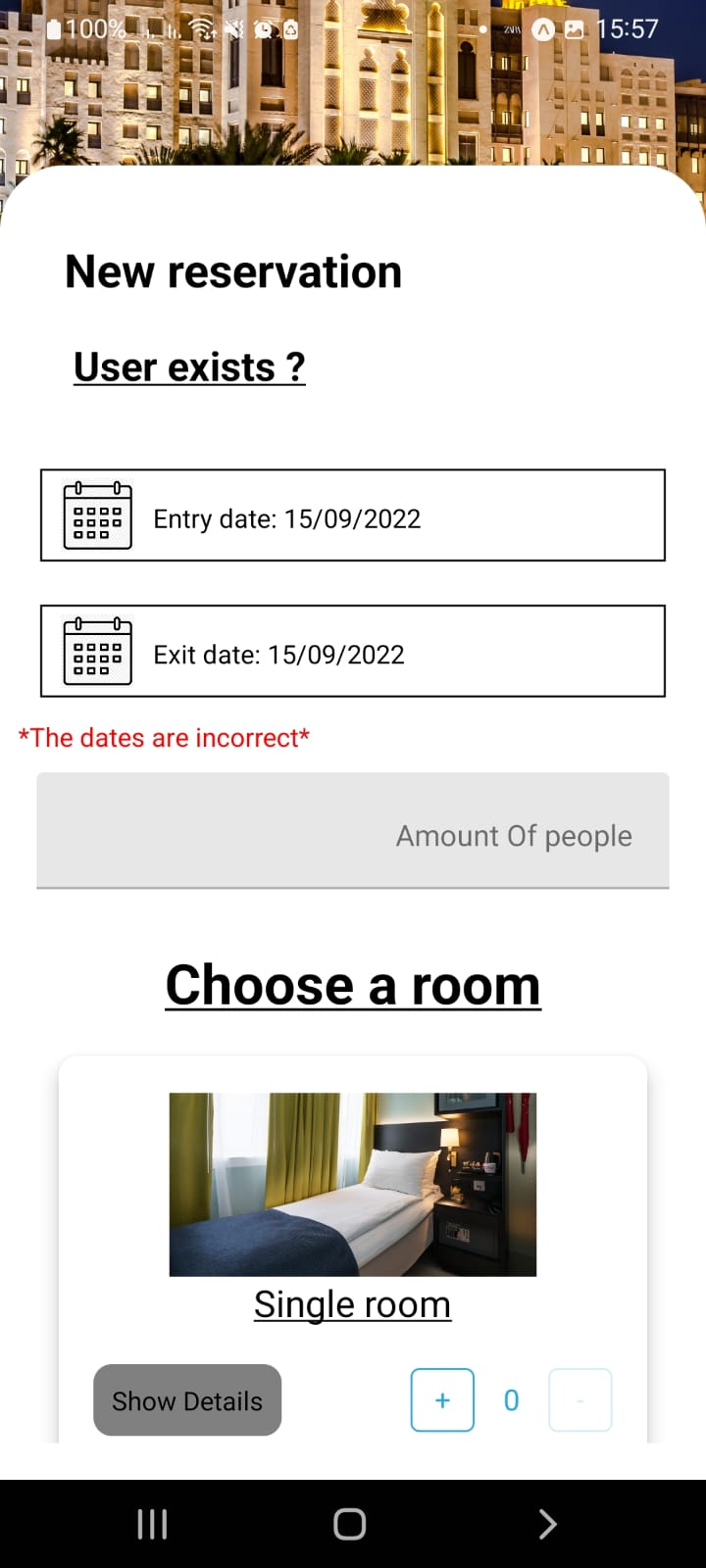
התיאור נוצר באופן אוטומטי

**Check in- main**

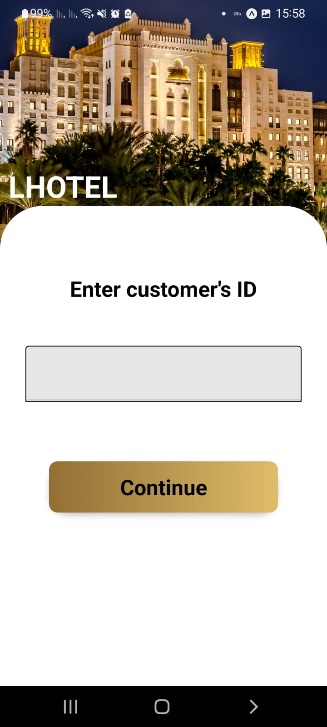
תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

**New reservation**



**Existing reservation search**



**Check in – confirmation**

תמונה שמכילה טקסט

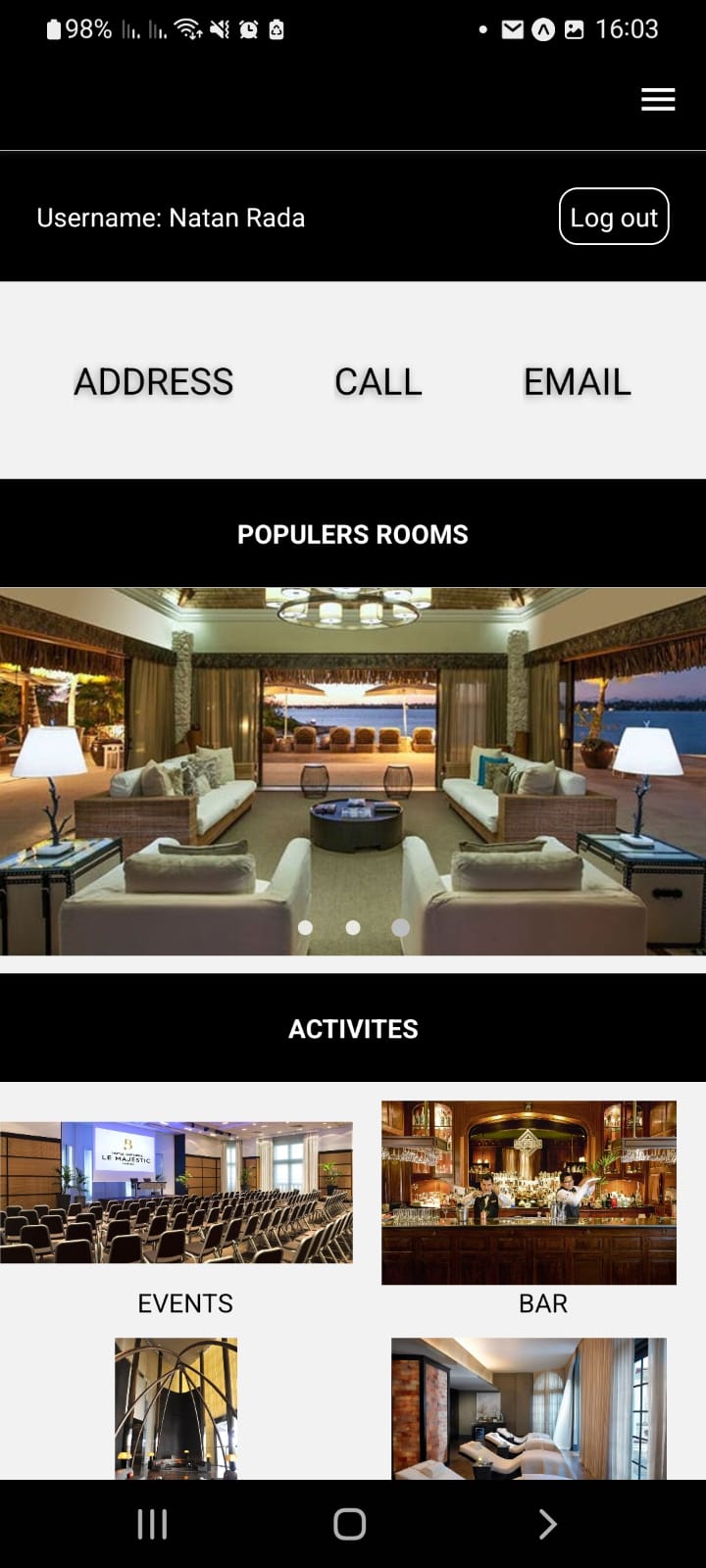
התיאור נוצר באופן אוטומטי

**Check out**

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

**Customer home**



**Login - customer**

תמונה שמכילה טקסט

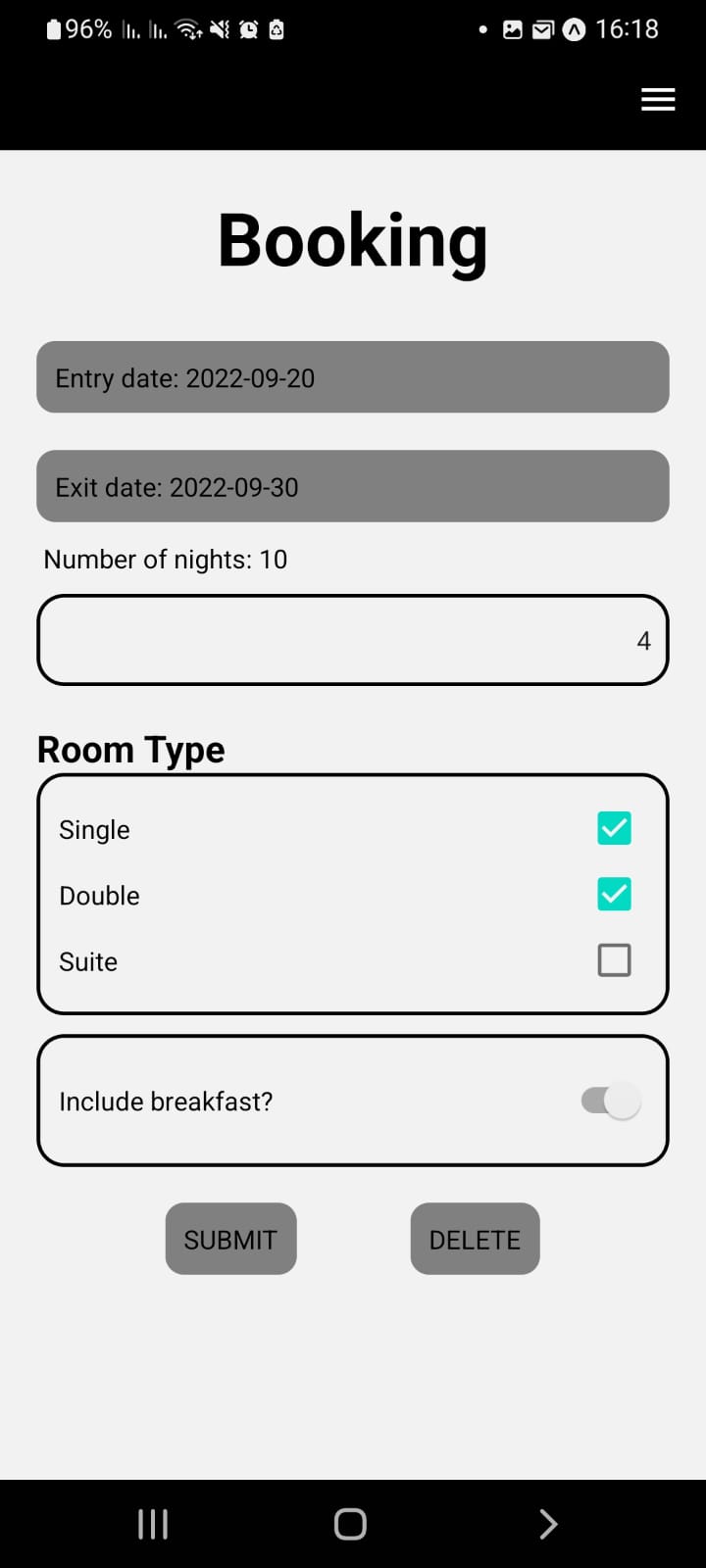
התיאור נוצר באופן אוטומטי

**Registration**

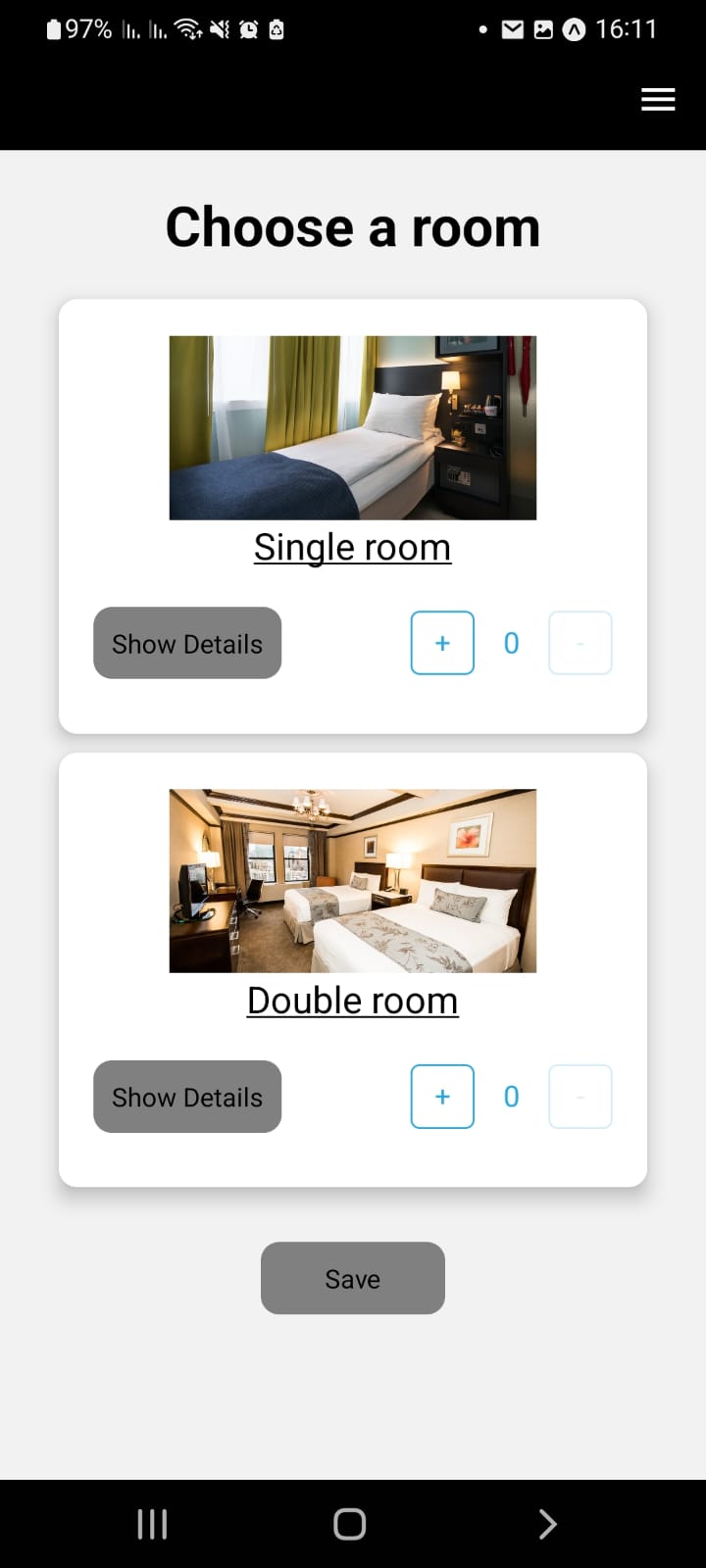
תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

**Booking**



**Save rooms**



**Payment**

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

**Order confirm**

תמונה שמכילה שולחן

התיאור נוצר באופן אוטומטי

* 1. **ממשקים למערכות אחרות/API**

**Irrelevant**

**7.9. שימוש בחבילות תוכנה**

Client side: **Material UI / Bootsrap / Xpro** - design libraries that contain ready-made design elements, to shorten working time and development

Server side:

* **Cors** - a package that allows the connection between the local server and the application during development.
* **Expo** - a shell for Native React that allows access to the components of a mobile device.
* **SQLCLIENT** - A package that allows connection between the database and the WEB API.

1. **שימוש במבני נתונים וארגון קבצים** 
   1. **מבני הנתונים**

Diagram, schematic

Description automatically generated

**Dictionary of tables**

**Table name: לקוחות (Customers)**

|  |  |  |  |
| --- | --- | --- | --- |
| **תיאור** | **מאפיינים** | **טיפוס** | **שם השדה** |
| ID Customer | Numbers only  Not null  Primary key | Int | \_ID Customer |
| Customers types | Numbers only  Not null  foreign key | Int | Customer\_Type |
| First name | Letters  Not null | Nvarchar(30) | First\_Name |
| Last Name | Letters | Nvarchar(30) | Last\_Name |
| Maill | Numbers and letters  Not Null | Nvarchar(100) | Mail |
| Password | Numbers and letters  Not Null | Nvarchar(30) | Password |
| Phone Number | Numbers and letters | Nvarchar(30) | Phone\_Number |
| Cardholder Name | Letters only | Nvarchar(30) | Card\_Holder\_Name |
| CreditCard Date | Letters only | Nvarchar(5) | Credit\_Card\_Date |
| 3digit | Numbers only | int | Three\_Digit |
| Credit\_Card\_Number | Numbers only | Nvarchar(16) | Credit\_Card\_Number |

**Table name: סוגי לקוחות (Customers\_Types)**

|  |  |  |  |
| --- | --- | --- | --- |
| **תיאור** | **מאפיינים** | **טיפוס** | **שם השדה** |
| Customers types | Numbers only  Not null  Primary key | Int | Customers\_Type |
| Description | Letters | Nvarchar(30) | Description |

**Table name: קטגוריה (Category)**

|  |  |  |  |
| --- | --- | --- | --- |
| **תיאור** | **מאפיינים** | **טיפוס** | **שם השדה** |
| Category Number | Numbers only  Not null  Primary key | Int | Category\_Number |
| Description | Letters | Nvarchar(30) | Description |

**Table name: חדרים (Rooms)**

|  |  |  |  |
| --- | --- | --- | --- |
| **תיאור** | **מאפיינים** | **טיפוס** | **שם השדה** |
| Room number | Numbers only  Not null  Primary key | Int identity(1,1) | Room\_Number |
| Room type | Letters  Not null | Nvarchar(30) | Room\_Type |
| Price Per Night | Numbers only  Not null | int | Price\_Per\_Night |
| Details | Letters  Not null | Nvarchar(100) | Details |

**Table name: חדרים ללקוח (Customers\_Rooms)**

|  |  |  |  |
| --- | --- | --- | --- |
| **תיאור** | **מאפיינים** | **טיפוס** | **שם השדה** |
| Room number | Numbers only  Not null  Primary key  foreign key | Int | Room\_Number |
| Bill number | Numbers only  foreign key | Int | Bill\_Number |
| Customer ID | Numbers only  Not null | Int | Customer\_ID |
| Entry Date | Date  Not null | date | Entry\_Date |
| Exit Date | Date  Not null | date | Exit\_Date |
| Amount of people | Numbers only | int | Amount\_Of\_People |
| Room Status | letters only  Not null | Nvarchar(30) | Room\_Status |

**Table name: חשבון (Bill)**

|  |  |  |  |
| --- | --- | --- | --- |
| **תיאור** | **מאפיינים** | **טיפוס** | **שם השדה** |
| Bill number | Numbers only  Not null  Primary key | Int identity(1,1) | Bill\_Number |
| Customer ID | Numbers only  Not null  Primary key  foreign key | Int | Customer\_ID |
| Employee ID | Numbers only  Not null  Primary key  foreign key | Int | Employee\_ID |
| Credit Card number | Numbers only  Not null | Nvarchar(16) | Credit\_Card\_Number |
| Bill\_Date | Date only  Not null | Date | Bill\_Date |
| Bill\_Status | letters only  Not null | Nvarchar(10) | Bill\_Status |

**Table name: מוצרים (Products)**

|  |  |  |  |
| --- | --- | --- | --- |
| **תיאור** | **מאפיינים** | **טיפוס** | **שם השדה** |
| Product Code | Numbers only  Not null  Primary key | Int identity(1,1) | Product\_Code |
| Category Number | Numbers only  Not null  foreign key | Int | Category\_Number |
| Description | Letters | Nvarchar(30) | Description |
| Price per unit | Numbers only  Not null | DECIMAL(10,2) | Price\_Per\_Unit |
| Discount Precentage | Numbers | DECIMAL(10,2) | Discount\_Precentage |

**Table name: פרטי קבלה (Bill\_Details)**

|  |  |  |  |
| --- | --- | --- | --- |
| **תיאור** | **מאפיינים** | **טיפוס** | **שם השדה** |
| Bill number | Numbers only  Not null | Int | Bill\_Number |
| Customer ID | Numbers only  Not null  foreign key | Int | Customer\_ID |
| Bill\_Date | Date only  Not null | Date | Bill\_Date |
| Room number | Numbers only  Not null  foreign key | Int | Room\_Number |
| Purchas date | Date only | Date | Purchas\_Date |
| Product Code | Numbers only  Not null  Foreign key | Int | Product\_Code |
| Amount | Numbers only  Not null | Int | Amount |
| Purchas Time | Time only | Time | Purchas\_Time |
| Purchas\_Method | Letters only  Not null | Nvarchar(20) | Purchas\_Method |

**Table name: מטלות (Tasks\_Type)**

|  |  |  |  |
| --- | --- | --- | --- |
| **תיאור** | **מאפיינים** | **טיפוס** | **שם השדה** |
| Tasks number | Numbers only  Not null  Primary key | Int | Tasks\_Number |
| Task Name | Letters  Not null | Nvarchar(30) | Task\_Name |

**Table name: עובדים (Employee)**

|  |  |  |  |
| --- | --- | --- | --- |
| **תיאור** | **מאפיינים** | **טיפוס** | **שם השדה** |
| Employee ID | Numbers only  Not null  Primary key | Int | Employee\_ID |
| Employee Code | Numbers only  Not null | Int identity(1,1) | Employee\_Code |
| Employee Name | Letters  Not null | Nvarchar(30) | Employee\_Name |
| Phone number | Numbers only | Nvarchar(30) | Phone\_Number |
| Worker code | Number  Not null  foreign key | int | Worker\_Code |
| Hourly wage | Numbers only  Not null | Int | Hourly\_Wage |
| Birth Date | Date only | date | Birth\_Date |
| Address | Letters | Nvarchar(30) | Address |

**Table name: סוגי עובדים ((Employees\_Types**

|  |  |  |  |
| --- | --- | --- | --- |
| **תיאור** | **מאפיינים** | **טיפוס** | **שם השדה** |
| Worker code | Numbers only  Not null  Primary key | Int | Worker\_Code |
| Description | Letters  Not null | Nvarchar(30) | Description |

**Table name: משימות לעובדים ((Employee\_Task**

|  |  |  |  |
| --- | --- | --- | --- |
| **תיאור** | **מאפיינים** | **טיפוס** | **שם השדה** |
| Employee ID | Numbers only  Not null  Primary key  Foreign key | Int | Employee\_ID |
| Task Number | Numbers only  Not null  Primary key  Foreign key | Int | Task\_Number |
| Start Date | Date  Not null  Primary key  Foreign key | date | Start\_Date |
| Start Time | Date  Not null  Primary key | date | Start\_Time |
| End Date | Date  Not null | date | End\_Date |
| Task Status | Letters | Nvarchar(30) | Task\_Status |
| Description | Letters | Nvarchar(30) | Description |

**Table name:רכישת סחורה(Purchase\_Of\_Goods)**

|  |  |  |  |
| --- | --- | --- | --- |
| **תיאור** | **מאפיינים** | **טיפוס** | **שם השדה** |
| Product Code | Numbers only  Not null  FOREIGN key | Int | Product\_Code |
| Price per unit | Numbers only  Not null | DECIMAL(10,2) | Price\_Per\_Unit |
| Amount | Numbers only  Not null | Int | Amount |
| Purchas date | Date only  Not null | Date | Purchas\_Date |
| Sum Total | Numbers only  Not null | DECIMAL(10,2) | Sum\_Total |

**Table name:משמרות(Shifts)**

|  |  |  |  |
| --- | --- | --- | --- |
| **תיאור** | **מאפיינים** | **טיפוס** | **שם השדה** |
| Employee ID | Numbers only  Not null  Primary key  Foreign key | Int | Employee\_ID |
| Date | Date only  Not null  Primary key | Date | Date |
| Employee Code | Numbers only  Not null | int | Employee\_Code |
| Worker code | Number  Not null | int | Worker\_Code |
| Entrance\_Time | Time only  Not null | Time | Entrance\_Time |
| Leaving\_Time | Time only | Time | Leaving\_Time |

**Table name: תיעוד רכישות (Purchases\_Documentation)**

|  |  |  |  |
| --- | --- | --- | --- |
| **תיאור** | **מאפיינים** | **טיפוס** | **שם השדה** |
| Bill\_Number | Numbers only | Int | Bill\_Number |
| Customer\_ID | Numbers only | Int | Customer\_ID |
| Bill\_Date | Date only | Date | Bill\_Date |
| Room\_Number | Numbers only | int | Room\_Number |
| Room\_Type | Letters only | nvarchar(30) | Room\_Type |
| Price\_Per\_Night | Numbers only | Int | Price\_Per\_Night |
| Amount\_Of\_People | Numbers only | Int | Amount\_Of\_People |
| Number\_Of\_Nights | Numbers only | Int | Number\_Of\_Nights |
| Payment\_Method | Letters only | nvarchar(30) | Payment\_Method |

* 1. **שיטת האחסון**

Tabular database on top of microsoft sql server 2019

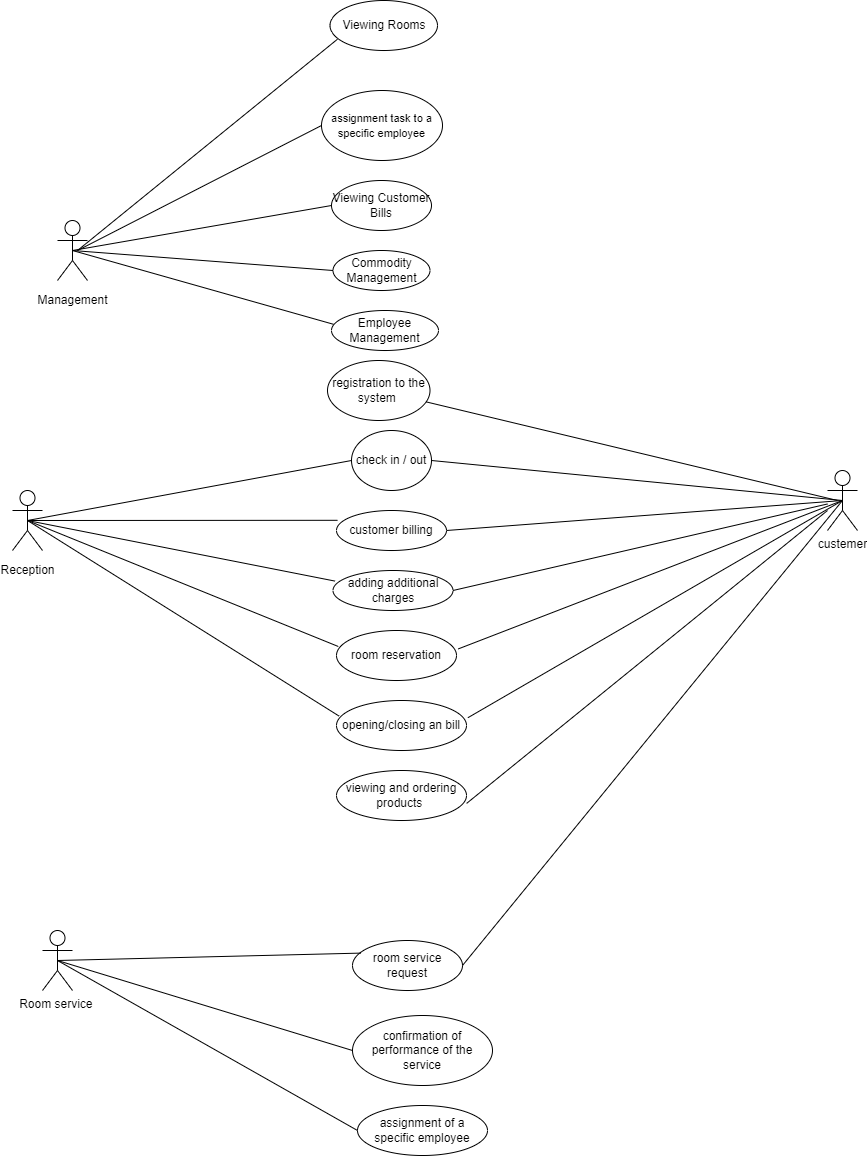
* 1. **מנגנוני התאוששות**

When examining the recovery mechanisms in an SQL-based database we embed transactions for each procedure of updating, deleting and creating a new record.

Using transactions allows security in that the record is written first to the log file and only after verifying that there is no problem is it written to the main database file.

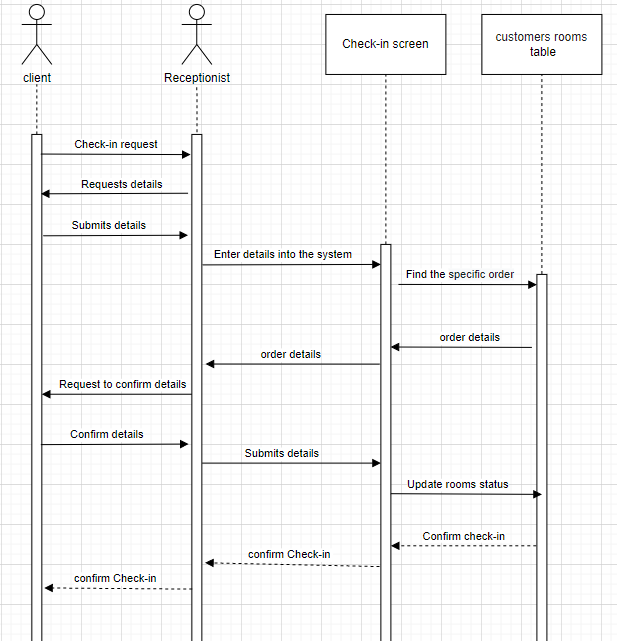
Also, the extent and there is a fall in the database, the use of transactions saves the information and when the database recovers the record is rewritten.

1. **תרשימי מערכת מרכזיים** 
   1. **Use Case**

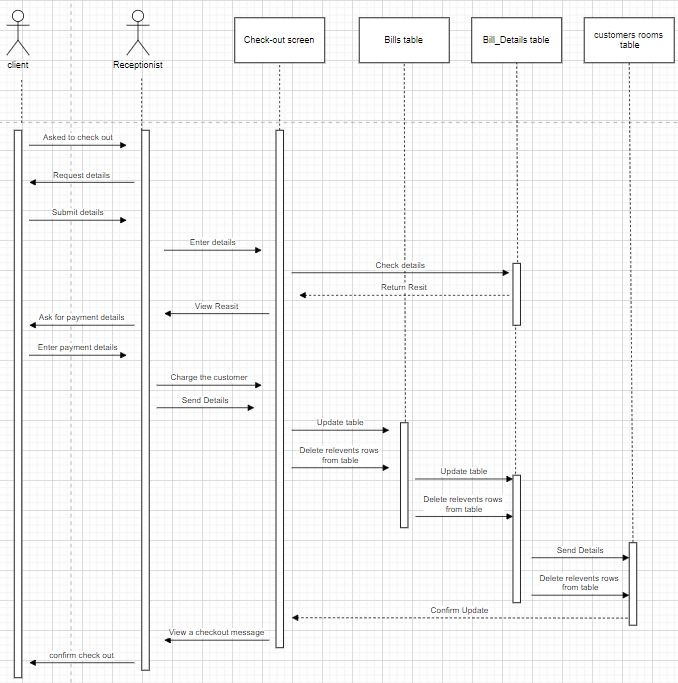


**Sequence Diagram 9.2**

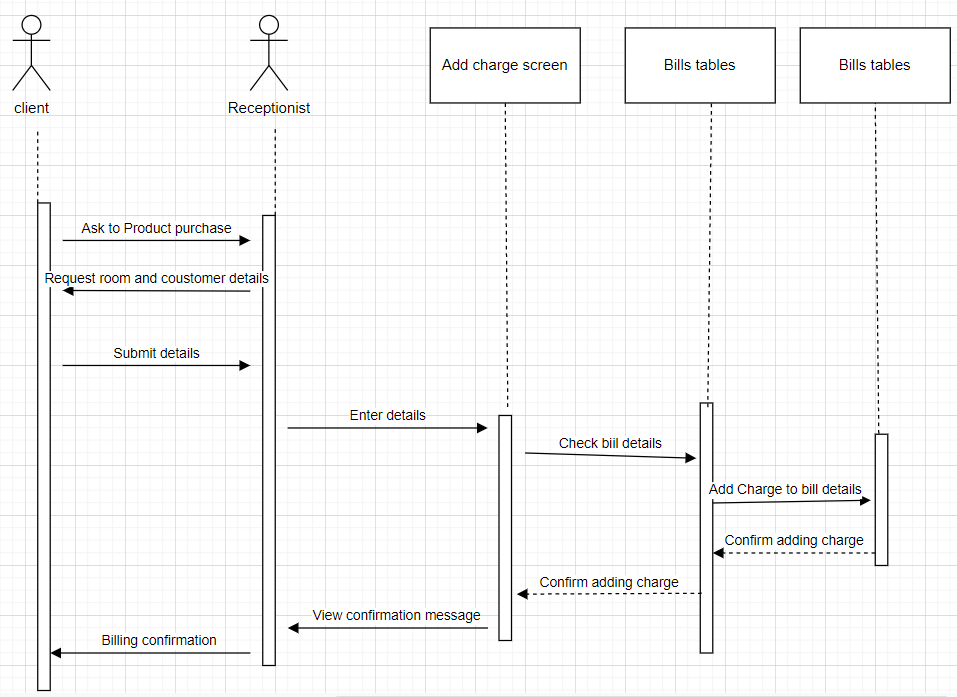
**check in**

****

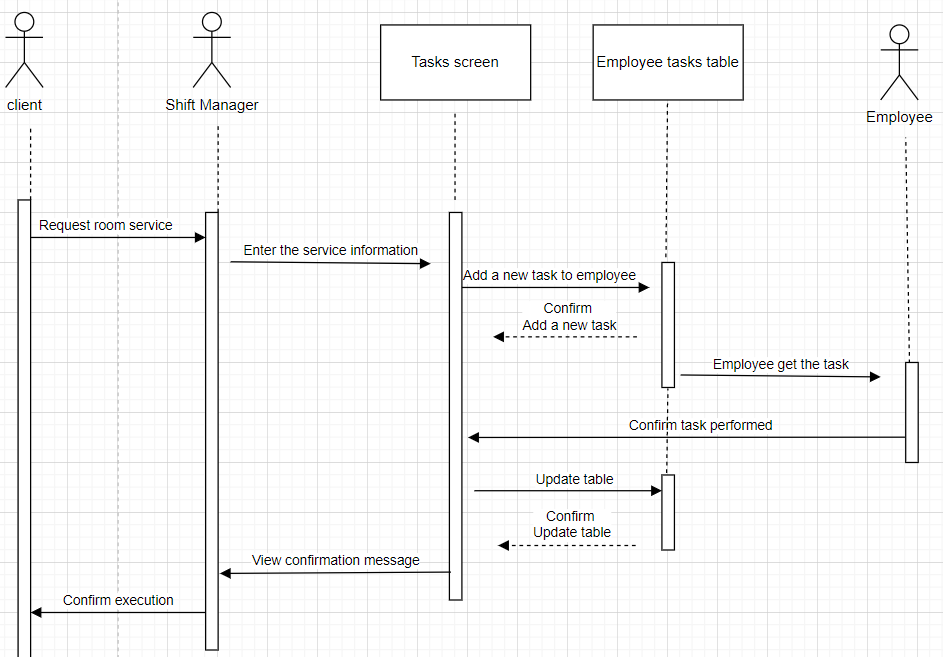
**check out**

****

**Add a charge**

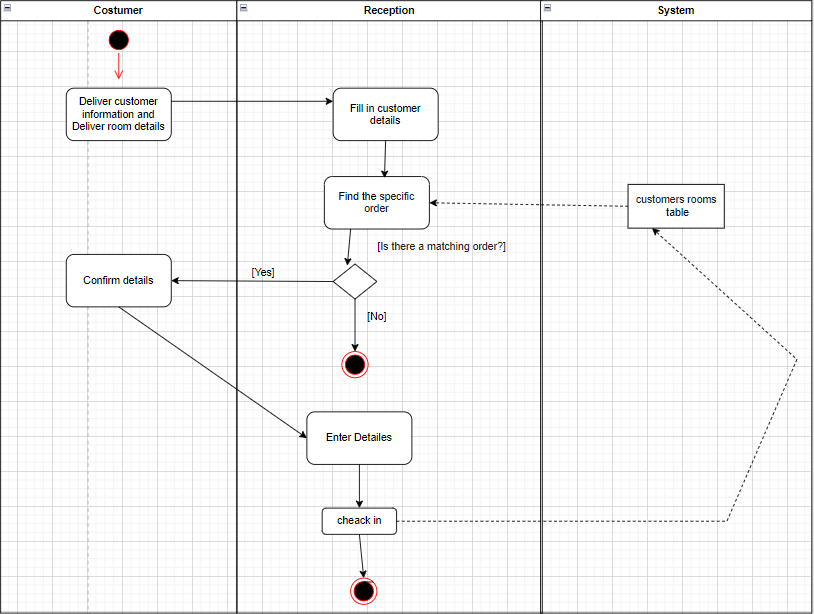
****

**Performing tasks**

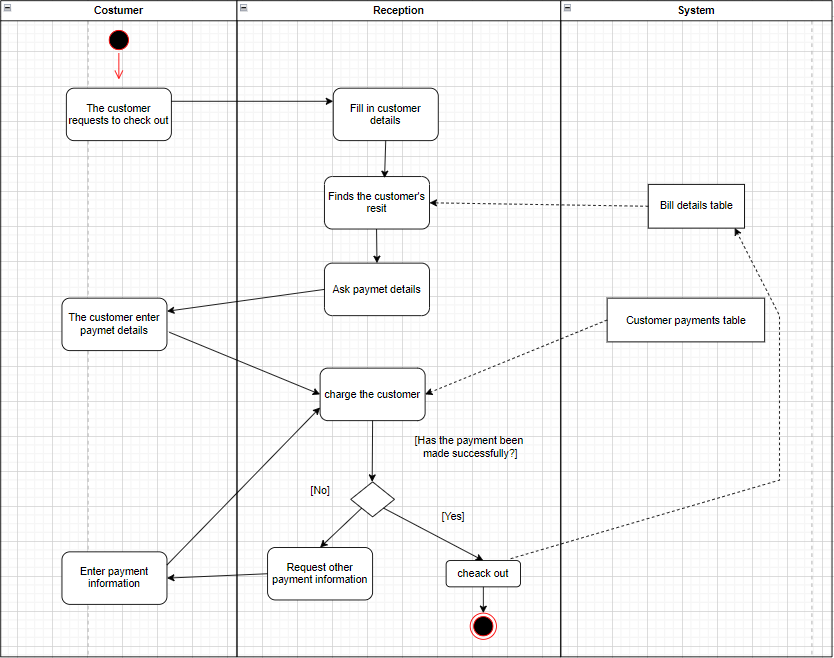


**9.3Data Flow**

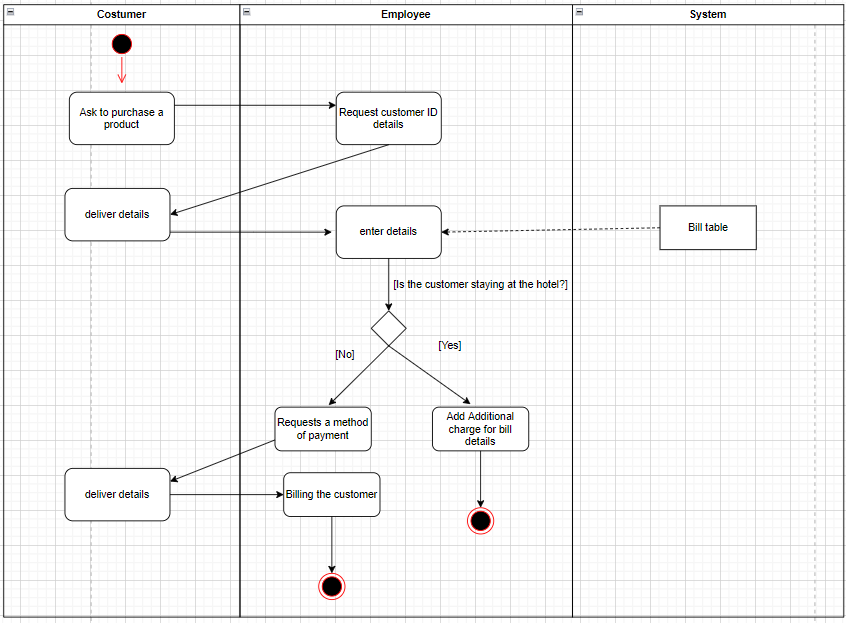
**check in**

****

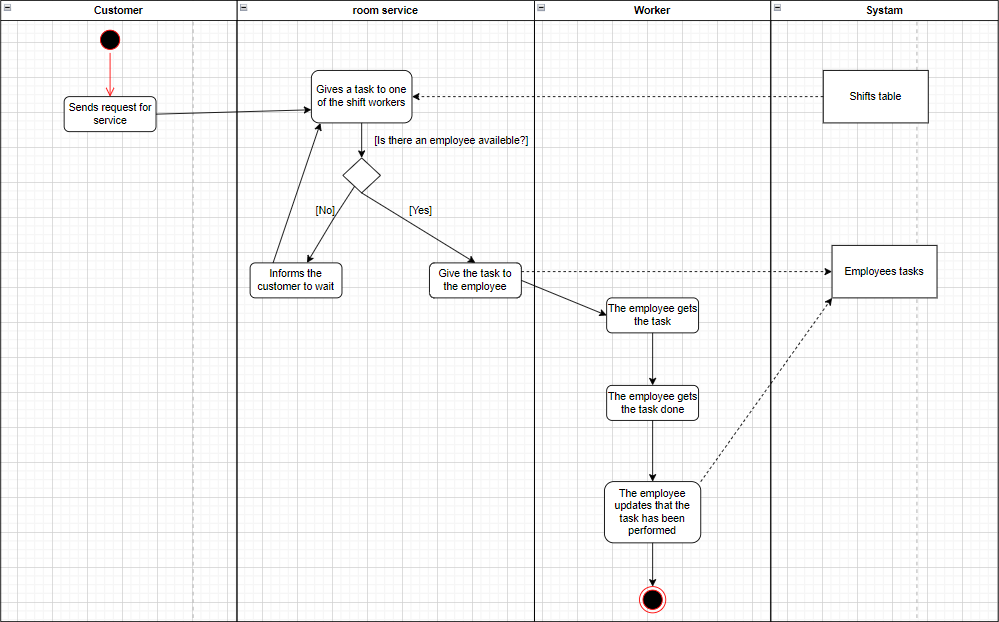
**check out**

****

**Add a charge**

****

**Performing tasks**

****

1. **תיאור המרכיב האלגוריתמי – חישובי**

**10.1 הבעיה שבא לפתור**

* **In order to book a room without an online system, the customer needs to call the reception representative and check with him which rooms are available.**

**If two customers call 2 different reception representatives, since the system is not online, a situation can arise in which the same room is reserved for 2 different customers.**

* The solution we offer is a solution where a customer can pre-book a room through the online system which in the booking process saves the room for 10 minutes, if the booking is canceled he does not save the room and another customer can book it.
* **In order to manage a large group of employees, each group in charge of something has a group manager who is responsible for giving tasks to each employee, but when the group has to perform tasks in different places in the hotel, it will be really difficult for the person in charge to communicate with the employees and get a snapshot.**
* The solution we offer is to connect all the employees to an online system which allows the person in charge to send a task for the employees, where the employees will be notified of a new task to perform and see the details of the task. When the employee completes the task he will signal that the task has been completed, and the supervisor will be notified that the task has been completed successfully.

This allows both employees and the person in charge to report on the state of the tasks efficiently, assign new tasks and get a clear and up-to-date picture of the tasks.

**10.2 איסוף מידע וניתוחים סטטיסטיים**

* Which month of the year has the largest number of visitors.
* The best-selling product in each category.
* Quantity of room service requests.
* Quantity of a particular product purchased.
* Regulating income and expenses.
* Regulation of the quantity of products purchased in the store.

1. **אבטחת מידע**

**Users Permissions:**

* super admin - Manage database tables.
* Shift manager - managing tasks for employees, managing employees, managing customers, viewing old / closed invoices.
* Receipt worker - saving and closing invoices, adding charges to customers, check in / out, billing customers.
* Room service worker - confirmation of performing tasks, viewing tasks.
* Customers - creating a new order, viewing the order details, ordering products.

Checking permissions calls during login, when the user enters the login information a verification is made against the FireBase, if the user exists and the details are correct, the database will retrieve the user and return his details.

At the customer level, when an answer is received, we direct the user according to his definition (whether customer or employee) to the page that suits him, where he will have accessibility that only such a user is able to perform.

1. **משאבים הנדרשים לפרויקט**

**12.1 מספר שעות המוקדש לפרויקט, חלוקת עבודה בין חברי הצוות**

1000 hours, teamwork throughout the project.

**12.2 ציוד נדרש**

laptop, smartphone

**12.3 תוכנות נדרשות**

Microsoft SQL Management, Visual Studio Code

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

* SQL link to a react project
* Encrypt passwords with Expo Crypto
* Create dynamic graphs

**12.5 ספרות ומקורות מידע**

* <https://www.w3schools.com/sql/>
* <https://stackoverflow.com/>
* <https://reactjs.org/>
* <https://www.youtube.com/>
* <https://github.com/git-up/GitUp>
* <https://reactrouter.com/>
* <https://www.google.com/>
* <https://getbootstrap.com/>
* <https://mui.com/getting-started/installation/>
* <https://regex101.com/>

1. **תכנית עבודה ושלבים למימוש הפרויקט**

* Characterization – 10/06/2022
* System design and coding – 31/08/2022
* Tests and repairs – 9/2022
* Submission of a project – 10/09/2022
* Project protection – 10/2022

1. **בדיקות**

**14.1 בדיקות תהליכיות**

Test ID: Register a new user.

The purpose of the test: Create a new user for the system.

Prerequisites: Client without existing account, Internet, active site.

|  |  |  |
| --- | --- | --- |
| **Step description** | **Desired outcome** | **Result obtained** |
| Reaching the registration screen | The user reaches the registration page successfully | The user reaches the registration page successfully |
| Filling out the registration form | All required fields are filled with no errors | All required fields are filled with no errors |
| Submit the registration form | Another user to the customer base successfully | Another user to the customer base successfully |
| Registration Confirmation | The customer receives a certificate that is successfully added to the database | The customer receives a certificate that is successfully added to the database |

**14.2 בדיקות יחידה**

Test ID: Check the validity of the fields on the registration form.

The purpose of the test: Make sure that all the input that the

customer has entered is in the correct format.

Prerequisites: Client without existing account, Internet, active site.

|  |  |  |
| --- | --- | --- |
| **Step description** | **Desired outcome** | **Result obtained** |
| Fill in a username | Username begins with a capital letter, no spaces, no special characters | Username begins with a capital letter, no spaces, no special characters |
| Fill in an email | Fill in an email in the correct email format, an email that does not exist in the system | Fill in an email in the correct email format, an email that does not exist in the system |
| Fill in a password | Contains a capital letter, contains a small letter, at least one special character, length between 4-12 characters | Contains a capital letter, contains a small letter, at least one special character, length between 4-12 characters |
| Fill in a password confirmation | The field must contain the same password entered in the "Password" field | The field must contain the same password entered in the "Password" field |
| Fill in a phone number | The field must match the format of a valid international phone number | The field must match the format of a valid international phone number |
| Agreeing to terms of service | The field must be marked with confirmation that the user agrees to the terms of service | The field must be marked with confirmation that the user agrees to the terms of service |
| Delete | All fields are emptied of content | All fields are emptied of content |
| Save | Checking all required fields is complete and submitting the form | Checking all required fields is complete and submitting the form |

1. **בקרת גרסאות**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Component** | **Component name** | **Version** | **Date** | **Changes** |
| Specification document | Specification document | 1.0 | 01/04/2022 | Original document |
| Specification document | Specification document | 2.0 | 04/04/2022 | Functionality update,  Screenshot update,  Updating charts,  Updating technologies,  Additional sections 14-16  General update |
| Specification document | Specification document | 3.0 | 16/05/2022 | Adding headings to section 7.7  Update dates in section 13 |
| Specification document | Specification document | 3.0 | 12/04/2022 | DSD update  Update the SQL tables  Correction of the Sequence Diagram and Data Flow |
| Specification document | Specification document | 4.0 | 04/07/2022 | DSD update, update tables |
| Specification document | Specification document | 5.0 | 17/08/2022 | sequence diagrams update,  Data Flowes updatem,  DSD update,  Tables update |
| Specification document | Specification document | 6.0 | 06/09/2022 | Updating the DSD and list of tables to be up to date with the database tables |
| Database | DSD | 1.0 | 04/04/2022 | Original DSD |
| Database | DSD | 2.0 | 06/04/2022 | Improving the existing database and correcting errors |
| Database | SQL | 3.0 | 15/05/2022 | Creating procedures required for server activity |
| Database | SQL | 4.0 | 04/07/2022 | Updating tables will better suit the needs of the system, adding procedures |
| Database | SQL | 5.0 | 22/08/2022 | Updating tables will better suit the needs of the system, adding procedures |
| Database | SQL | 6.0 | 06/09/2022 | Adding an archival table that holds the history of all the hotel's purchases, creating procedures that create the relevant reports |
| Database | SQL | 7.0 | 13/09/2022 | A change in the procedures that will be suitable for obtaining the desired information |
| LHOTEL Servers | Code | 1.0 | 01/08/2022 | Establishing servers for dedicated API requests for receiving information from the database |
| LHOTEL Servers | Code | 2.0 | 21/08/2022 | Adjusting the way the information is pulled in a way that will allow more coordinated work with the database |
| LHOTEL Servers | Code | 3.0 | 06/09/2022 | Changing functions and classes in order to have a match between classes and the information returned from the database |
| LHOTEL Servers | Code | 4.0 | 13/09/2022 | Changing the functions in a way that will be compatible with the information received from the SQL |
| LHOTEL APP | Code | 1.0 | 31/07/2022 | source code |
| LHOTEL APP | Code | 2.0 | 22/08/2022 | Creating screens suitable for the type of user (customer/employee),  Creating a link of the application with the server |
| LHOTEL APP | Code | 3.0 | 25/08/2022 | Updating how to navigate between screens |
| LHOTEL APP | Code | 4.0 | 01/09/2022 | Creating dedicated screens for every action the user will need to operate the application |
| LHOTEL APP | Code | 5.0 | 13/09/2022 | Adding the API requests to the relevant pages |
| Management side | Code | 1.0 |  | source code |

1. **קטעי קוד**

**SQL**

-- פרוצדורה אשר מבצעת את שמירת החדרים ללקוח

create proc SaveRoomReservation

@id int,

@Card\_Holder\_Name nvarchar(30), -- יצירת פרמטרים הדרושים לביצוע הזמנה

@Credit\_Card\_Date nvarchar(5),

@Three\_Digit int,

@Credit\_Card\_Number nvarchar(16),

@Employee\_ID int,

@Counter\_Single int,

@Counter\_Double int,

@Counter\_Suite int,

@Entry\_Date date,

@Exit\_Date date,

@Amount\_Of\_People int

as

begin tran

exec UpdateCustomerCredit @id,@Card\_Holder\_Name,@Credit\_Card\_Date,@Three\_Digit,@Credit\_Card\_Number --- הפעלת פרוצדורה המעדכנת פרטי כ.אשראי ביטחון של לקוח

DECLARE @Bill\_Date as date = (select Bill\_Date from Bill where Customer\_ID = @id and Bill\_Status = 'Open')

DECLARE @date as date = GETDATE()

if NOT EXISTS (select \* from Bill where Customer\_ID = @id and Bill\_Status = 'Open')--- עדכון פרטים בחשבונית פתוחה במידה וקיימת אחרת יצירת חשבונית חדשה ללקוח

exec AddNewBill @Employee\_ID, @id,@Credit\_Card\_Number ,@date,'Open'

else

exec AlterBill @id, @Bill\_Date ,@Employee\_ID, @Credit\_Card\_Number,'Open'

set @Bill\_Date = (select Bill\_Date from Bill where Customer\_ID = @id and Bill\_Status = 'Open')

DECLARE @bill\_number as int = (select Bill\_Number from Bill where Customer\_ID = @id and Bill\_Status = 'Open')

DECLARE @room\_number as int

while @Counter\_Single > 0 ----- הקצאה ושמירה של חדרים ללקוח לפי כמות החדרים שמעוניין , סוג ,כמות ובאם סטטוס החדר פנוי

begin

set @room\_number = (SELECT MIN(Room\_Number) AS Room\_Number FROM dbo.AvailableRooms() WHERE Room\_Type = 'Single room')

IF NOT EXISTS(select Room\_Number from [dbo].[Customers\_Rooms] where Room\_Number = @room\_number)

exec AddNewCustomerRooms @room\_number, @bill\_number,@id,@Bill\_Date,@Entry\_Date,@Exit\_Date,@Amount\_Of\_People,'Reserved'

else

exec AlterCustomerRoom @room\_number, @bill\_number,@id,@date,@Entry\_Date,@Exit\_Date,@Amount\_Of\_People,'Reserved'

set @Counter\_Single = @Counter\_Single - 1

end

while @Counter\_Double > 0

begin

set @room\_number = (SELECT MIN(Room\_Number) AS Room\_Number FROM dbo.AvailableRooms() WHERE Room\_Type = 'Double room')

IF NOT EXISTS(select Room\_Number from [dbo].[Customers\_Rooms] where Room\_Number = @room\_number)

exec AddNewCustomerRooms @room\_number, @bill\_number,@id,@Bill\_Date,@Entry\_Date,@Exit\_Date,@Amount\_Of\_People,'Reserved'

else

exec AlterCustomerRoom @room\_number, @bill\_number,@id,@date,@Entry\_Date,@Exit\_Date,@Amount\_Of\_People,'Reserved'

set @Counter\_Double = @Counter\_Double - 1

end

while @Counter\_Suite > 0

begin

set @room\_number = (SELECT MIN(Room\_Number) AS Room\_Number FROM dbo.AvailableRooms() WHERE Room\_Type = 'Suite')

IF NOT EXISTS(select Room\_Number from [dbo].[Customers\_Rooms] where Room\_Number = @room\_number)

exec AddNewCustomerRooms @room\_number, @bill\_number,@id,@Bill\_Date,@Entry\_Date,@Exit\_Date,@Amount\_Of\_People,'Reserved'

else

exec AlterCustomerRoom @room\_number, @bill\_number,@id,@date,@Entry\_Date,@Exit\_Date,@Amount\_Of\_People,'Reserved'

set @Counter\_Suite = @Counter\_Suite - 1

end

if (@@error !=0) ----הצגת הודעת שגיאה במידה ומתקבלת שגיאה

begin

rollback tran

print 'error'

return

end

commit tran

go

create trigger AddRoomToDetails ---- טריגר להכנסת רשומה חיוב על חדר חדשה לטבלת פרטי חשבונית של לקוח , מופעל כאשר ססטוס חדר בטבלת ההזמנות משתנה למאוכלס

( (לקוח ביצע צ'ק אין

on [Customers\_Rooms] for update

as

if exists (select Room\_Number from inserted where [Room\_Status] = 'Occupied')

begin

insert [dbo].[Bill\_Details]

select Bill\_Number, Customer\_ID, Bill\_Date, Room\_Number,Entry\_Date,8,1,convert(time,getdate()),'Credit'

from inserted

end

go

create proc CheckOut

@id int,

@Exit\_Date date --------- יצירת פרמטרים הדרושים לביצוע צ'ק אאוט ללקוח

as

begin tran

exec AddPurchases\_Documentation @id -----הרצה של פרוצדורה השומרת את כל הרכישות של הלקוח לפי ת.ז בטבלת תיעוד רכישות

DECLARE @bill\_Number AS int = (select Bill\_Number from [dbo].[Customers\_Rooms]

where [Customer\_ID] = @id and Exit\_Date = @Exit\_Date GROUP BY Bill\_Number) --- הצלבה של הנתונים שהתקבלו כקלט אל מול הנתונים השמורים במסד הנתונים

ועדכון של סטטוס חשבונית סגור אם אומתו

UPDATE [dbo].[Bill] SET Bill\_Status = 'Close'

where Customer\_ID=@id and Bill\_Number = @bill\_Number and Bill\_Status = 'Open'

exec DeleteBill\_Detail @id,@bill\_Number ---- הרצה של פרוצדורות המוחקות נתוני לקוח בטבלאות הרלוונטיות

exec DeleteCustomerRoom @id , @bill\_Number

if (@@error !=0) ----הצגת הודעת שגיאה במידה ומתקבלת שגיאה

begin

rollback tran

print 'error'

return

end

commit tran

go

**C#**

public static List<Shift> GetWorkersOnShift() // פונקציה המחזירה רשימה של משמרות

{

try

{

SqlDataReader reader = SQLConnection.ExcNQReturnReder(@"exec GetWorkersOnShift"); //השמת תוצאת הרצת הפרוצדורה

// באובייקט רידר

if (reader == null || !reader.HasRows)

return null;

List<Shift> employees = new List<Shift>();

while (reader.Read())//הגדרת רשימה מסוג משמרת ,יצירה והכנסה של אובייקטים מסוג משמרת מנתוני הרידר אליה

{

employees.Add(new Shift()

{

EmployeeID = (int)reader["Employee\_ID"],

EmployeeCode = (int)reader["Employee\_Code"],

EmployeeName = (string)reader["Employee\_Name"],

Description = (string)reader["Description"],

Date = (DateTime)reader["Date"],

EntranceTime = (string)reader["Entrance\_Time"],

LeavingTime = null

});

}

return employees;//החזרה של הרשימה

}

catch (Exception e)// הצגת שגיאה במידה והתקבלה

{

Console.WriteLine(e.Message);

return null;

}

finally//סגירת החיבור אל מסד הנתונים

{

SQLConnection.CloseDB();

}

}

}

public static List<Task> GetTaskById(int id)// פונקציה המחזירה רשימת משימות של עובד

{

try

{

string str = $@"exec GetTask\_ById {id}";// הגדרת הפקודה להרצת הפרוצודורה הרלוונטית

str = str.Replace("\r\n", string.Empty);//סידור תחביר הפקודה ע"י מחיקה של רווחים וירידות שורה מיותרים

SqlDataReader reader = SQLConnection.ExcNQReturnReder(str); // השמת הנתונים המוחזרים ממסד הנתונים בעקבות הרצת הפרוצדורה לאובייקט רידר

if (reader == null && !reader.HasRows)//הצגת הודעה מתאימה אם לא קיימים נתונים מוחזרים NULLוהחזרת

{

Console.WriteLine("There is no such employee in the system");

return null;

}

List<Task> tasks = new List<Task>();

while (reader.Read())//באם קיימים נתונים מוחזרים יצירת אובייקטי משימה מהם והכנסתם לרשימה

{

tasks.Add(new Task()

{

TaskCode = (int)reader["Task\_Code"],

EmployeeID = (int)reader["Employee\_ID"],

TaskName = (string)reader["Task\_Name"],

RoomNumber = (reader["Room\_Number"] != DBNull.Value)// תנאי אם מקוצר לטובת השמת ערך דיפולטיבי במידה ולא קיים ערך

? (int)reader["Room\_Number"] : -1,

StartDate = (DateTime)reader["Start\_Date"],

StartTime = (string)reader["Start\_Time"],

EndTime = (reader["End\_Time"] != DBNull.Value)

? (string)reader["End\_Time"] : null,

TaskStatus = (string)reader["Task\_Status"],

Description = (string)reader["Description"]

});

}

return tasks;// החזרה של הרשימה

}

catch (Exception e) // הצגת שגיאה במידה והתקבלה

{

Console.WriteLine(e.Message);

return null;

}

finally //סגירת החיבור אל מסד הנתונים

{

SQLConnection.CloseDB();

}

}

public class SQLConnection // מחלקה לטובת התנהלות מסודרת ויעילה מול מסד הנתונים

{

public static string conStr = @"data source=SQL5108.site4now.net;initial catalog=db\_a79b5b\_proj13;user id=db\_a79b5b\_proj13\_admin;password=XXNEA6q6VbvATG6g";

public static SqlConnection sqlConnection = new SqlConnection(conStr); // יצירת אובייקט חיבור למסד הנתונים

public static void CloseDB() // פונקציה לסגירת החיבור עם מסד הנתונים

{

try

{

sqlConnection.Close();

}

catch (Exception e)

{

Console.WriteLine(e.Message);

}

}

public static void OpenDB() // פונקציה לפתיחת החיבור עם מסד הנתונים

{

try

{

sqlConnection.Open();

}

catch (Exception e)

{

Console.WriteLine(e.Message);

}

}

public static int ExeNonQuery(string str) //פונקציה להחזרת כמות השורות המושפעות בעקבות הרצה של הפקודה המתקבלת כפרמטר

{

try

{

OpenDB();

SqlCommand comm = new SqlCommand(str, sqlConnection);

int result = comm.ExecuteNonQuery();

CloseDB();

return result;

}

catch (Exception e)

{

Console.WriteLine(e.Message);

return 0;

}

}

public static SqlDataReader ExcNQReturnReder(string command) //פונקציה להחזרת אובייקט המכיל נתונים בעקבות הרצה של הפקודה המתקבלת כפרמטר

{

try

{

SqlCommand comm = new SqlCommand(command, sqlConnection);

OpenDB();

SqlDataReader reader = comm.ExecuteReader();

return reader;

}

catch (Exception e)

{

sqlConnection.Close();

Console.WriteLine(e.Message);

return null;

}

}

**REACT NATIVE**

  const CreateUser = () => { // פונצקיה היוצרת אובייקט חדש מסוג משתמש ומחזירה אותו

        var Hashes = require('jshashes')

        let SHA1 = new Hashes.SHA1().b64\_hmac(id, password) // הצפנת סיסמת משתמש לפי מפתח ת.ז שלו והשמה במשתנה

        let user = {

            calssName: User,

            fields: {

                CustomerID: id,

                FirstName: first\_name,

                LastName: last\_name,

                Mail: email,

                password: SHA1,

                PhoneNumber: phone

            }

        }

        return user

    }

const GetDBCustomerById = async () => { // פונקציה אסינכרונית לטובת קבלת נתוני משתמש לפי ת.ז המתקבל במידה וקיים

                                          //  ומילוי אוטומטי שלהם בטופס הזמנה חדשה

    const requestOptions = {

      method: 'POST',

      body: JSON.stringify({

        "Customer\_ID": IDCheck,

      }),

      headers: { 'Content-Type': 'application/json' }

    };

    let result = await fetch('http://proj13.ruppin-tech.co.il/GetDBCustomerById', requestOptions);

    let user = await result.json();

    if (user !== null) {

      setCustomerID(IDCheck);

      setMail(user.Mail);

    setFirstName(user.FirstName);

      setLastName(user.LastName);

      setPhoneNumber(user.PhoneNumber);

      myContext.setIsUserExist(true)//עדכון סטטוס משתמש אם קיים בסטייט הגלובאלי

    }

  }

const CheckIn\_Without\_Existing\_User = async()=>{ // פונצקיה לביצוע צ'ק אין ללקוח שלא קיים לו משתמש במערכת

    let newCustomer = {                           // יצירת אובייקט מסוג לקוח

      className: Customer,

      fields: {

        CustomerID: curr.CustomerID,

        CustomerType: curr.CustomerType,

        FirstName: curr.FirstName,

        LastName: curr.LastName,

        Mail: curr.Mail,

        Password : curr.CustomerID,

        PhoneNumber:  curr.PhoneNumber,

        CardHolderName: curr.CardHolderName,

        CreditCardNumber: curr.CreditCardNumber,

        CreditCardDate: curr.CreditCardDate,

        ThreeDigit: curr.ThreeDigit,

        EmployeeID: curr.EmployeeID,

        AmountOfPeople: curr.AmountOfPeople,

        CounterSingle: curr.CounterSingle,

        CounterDouble: curr.CounterDouble,

        CounterSuite: curr.CounterSuite,

        ExitDate: curr.ExitDate,

        EntryDate: curr.EntryDate,

      },

    };

const requestOptions = { //API יצירת בקשת

  method: 'POST',        //לטובת יצירת  משתמש חדש שמירת הזמנתו וביצוע צ'ק אין

  body: JSON.stringify(newCustomer),

  headers: { 'Content-Type': 'application/json' }

};

let result = await fetch('http://proj13.ruppin-tech.co.il/CheckIn\_Without\_Existing\_User', requestOptions);

let reservationResult = await result.json();

if (reservationResult)

console.log(reservationResult);

}

const CheckIn\_With\_Existing\_User = async()=>{  // פונצקיה לביצוע צ'ק אין ללקוח שקיים לו משתמש במערכת

 let customer = {                                 // יצירת אובייקט מסוג לקוח

    className: Customer,

    fields: {

      CustomerID: curr.CustomerID,

      CustomerType: curr.CustomerType,

      CardHolderName: curr.CardHolderName,

      CreditCardNumber: curr.CreditCardNumber,

      CreditCardDate: curr.CreditCardDate,

      ThreeDigit: curr.ThreeDigit,

      AmountOfPeople: curr.AmountOfPeople,

      EmployeeID: curr.EmployeeID,

      CounterSingle: curr.CounterSingle,

      CounterDouble: curr.CounterDouble,

      CounterSuite: curr.CounterSuite,

      ExitDate: curr.ExitDate,

      EntryDate: curr.EntryDate,

    },

  };

  const requestOptions = { // API  יצירת בקשת

    method: "POST",         //לטובת יצירת  לקוח חדש שמירת הזמנתו וביצוע צ'ק אין

    body: JSON.stringify(customer.fields),

    headers: { "Content-Type": "application/json" },

  };

  let result = await fetch(

    "http://proj13.ruppin-tech.co.il/CheckIn\_With\_Existing\_User",

    requestOptions

  );

  let data = await result.json();

  if (data !== null) {

    console.log(JSON.stringify(data));

    return;

  }

}

  const CheckIn = () => { // פונצקית צ'ק אין ראשית , בודקת אם קיים משתמש רשום בהתאם לערכו בגלובל סטייט

    {                     // ומפעילה את הפונקציית צ'ק אין הרלוונטית

      myContext.isUserExist ? CheckIn\_With\_Existing\_User(): CheckIn\_Without\_Existing\_User()

      alert("You have checked in successfully !");

    }

  };