תרגיל מסכם

יש לבנות יישום לניהול **ספר טלפונים** המאפשר צפייה, הוספה, עדכון ומחיקה של אנשי קשר.
את היישום יש לפתח באמצעות MongoDB, Express/NodeJS ו AngularJS.

מדריך

חלק א1 - באקאנד

1. יש לייצור תיקייה ריקה בשם phonebook שבה נייצור את כל קבצי היישום.
2. יש לייצור קובץ (ריק) בשם server.js שיהיה אחראי על צד השרת.
3. יש לייצור קובץ package.json באמצעות הרצת הפקודה npm init מתוך שורת הפקודה.
4. יש לייצור שרת באמצעות Express.
   על מנת לייבא את Express יש לרשום: npm install express –save
   לאחר מכן, יש לייבא את המודול באמצעות require("express").
   לדוגמה: var express = require('express')

5. יש לייצור מחוון (רפרנס) לשרת אקספרסס.
   לדוגמה: var server = express();

6. יש לייצור תת תיקייה בשם public שתהווה תיקייה לקבצים סטטיים, בא נשמור את קצבי הפרונאנד.

7. יש להגדיר את התיקייה public כתיקייה סטטית.
   לדוגמה: server.use(express.static('public/'))
8. יש לוהסיף middleware המתעד את הפניות לשרת באופן הבא:
   תאריך ושעת הפנייה
   שיטת הפנייה (POST, GET...)
   ה URL המבוקש

   רמז: יש להגדיר פונקצייה המקבלת שלושה פרמטרים request, response, next ולאחר הגדרת הפונקציה, יש להעביר את השם שלה (ללא הסוגריים) למתודה use של server.

   לדוגמה: server.use(mylogger)

   זאת בהנחה שהגדרנו את הפונקצייה mylogger קודם לכן, באופן הבא:
   (לא לשכוח להריץ את next() בסוף הפונקצייה, אחרת השרת ייתקע ולא יגיב כמצופה)

```
function mylogger(request, response, next) {


}
```

9. יש להפעיל את השרת, שיקשיב לפורט 8000.
   לדוגמה server.listen(8000)

חלק א2 – פרונטאנד

1. יש לייצור את index.html שמהווה את הדף הראשי של היישום.
   את הקובץ יש לשמור בתוך התיקייה public.
2. יש לייצור את הקובץ app.js בתוך התיקייה public\js.
3. יש להוסיך קישור לפריימוורק של אנגולר 1.5
4. יש להוסיף קישור לקובץ app.js בתוך index.html.
5. יש להגידר **מודול** בשם phonebook (app.js) ולקשר אותו עם ng-app בקובץ index.html
6. יש להגידר משנתה בשם aContacts המהווה מערך של אובייקטים של אנשי קשר.
   מומלץ להכניס מספר אובייקטים לשם בדיקה.
7. יש לייצור controller בשם ctrl_List אשר באחריותו להציג את תכון המערך בטבלה.
   לשם כך, יש להשתמש ב ng-controller, ng-repeat בקובץ index.html
   במקביל, יש להגדיר את ה controller ב app.js
8. יש לגרום ל ctrl_List לרשום הערה ב console של הדפדפן, בכל פעם שהוא מופעל.

חלק ב'

9. יש לייצור את aContacts ב server.js
   אפשר ורצוי להעתיק את אותו האחד שיצרתם ב app.js

10. יש להגדיר ניתוב בשרת אשר מחזיר את המערך aContacts אם וכאשר פונים לשרת בכתובת getContacts/ בשיטה GET.

    להזכיר: server.get('/getContacts', function(req,res) { });

11. יש להיכנס ל app.js (אנגולר) ולשנות את המערך aContacts למערך ריק (ללא נתונים).
    כלומר: var aContacts = [ ]

12. יש לגרום ל ctrl_List לטעון את aContacts בנתונים המתקבלים מהשרת.
    רמז: יש לגרום ל controller לפנות לשרת באמצעות $http ע"י פנייה ל getContacts/ בשיטת get.

תזכורת *- $http זה service שצריך לבקש במפורש (dependency injection)*
תזכורת *- $http.get() הוא Promis כלומר, יש להשתמש ב then().*

תזכורת *– then(). מקבלת שני פרמטרים, כאשר הראשון הוא callback למקרה של הצלחה. הפרמטר השני, אופציונלי, פונקציית callback למקרה של כשלון.*

חלק ג'

13. יש להפעיל את שרת בסיס נתונים של MongoDB

תזכורת – יש להפעיל את שרת ה MongoDB באמצעות הפקודה **mongod**
את הפקודה הנ"ל יש לרשום בשורת הפקודה (Command Prompt).
יש להשאיר את החלון פתוח כל עוד רוצים להשאיר את בסיס הנתונים זמין ופעיל.

14. יש להיכנס לשורת הפקודה של MongoDB באמצעות הפקודה **mongo**

15. יש לייצור בסיס נתונים חדש בשם phoneBookDB ולהוסיף אליו collection בשם contacts ולהוסיף אליו את אנשי הקשר המופיעים במערך aContacts.

הכוונה להוספה ידנית באמצעות db.contacts.insert()

16. יש להתקין את המודול של mongodb ל NodeJS על מנת לאפשר גישה לבסיס נתונים.

להתקנת המודול באמצעות ה npm יש לרשום את הפקודה הבאה ב Command Prompt:

npm install mongodb –save

על מנת לייבא את המודול לקוד, יש להוסיף ל server.js את השורה הבאה:

var mongodb = require('mongodb');

17. יש לייצור פונקציה אסינכרונית בשם getContacts (המקבלת callback) שתפקידה יהיה לפנות למסד נתונים ולהביא משם את רשימת אנשי הקשר, להמירם למערך ולהחזיר אותו באמצעות ה callback, לא לפני שסוגרים את החיבור לדטה בייס.

18. יש לעדכן את הניתוב getContacts/ של השרת(server.js) , על מנת שיביא את אנשי הקשר מהבסיס נתונים, בעזרת הפונקציה getContacts.

שימו לב – היות והפונקציה getContacts נבנתה כפונצקיה אסינכרונית, היא אמורה לקבל פונקציה שתטפל בתשובה, כלומר, תעביר את התשובה לדפדפן.

חלק ד'

19. יש להגדיר ניתוב בפרונטאנד המאפשר חלוקת היישום למסכים הבאים:
    a. צפייה ברשימה כאשר ניגשים ל /
    b. הוספת איש קשר חדש כאשר ניגשים ל add/
    c. עדכון איש קשר קיים כאשר ניגשים ל edit/:id/

לשם כך יש להשתמש ב ngRoute אותו יש לייבא ל index.html ולרשום כ dependency בעת הגדרת המודול של אנגולר (app.js).

20. יש לייצור view המאפשר הוספת איש קשר חדש ולייצור לו controller בשם ctrl_Add.

21. יש להוסיף פונקציה ב ctrl_Add שתפקידה לשדר את האיש קשר החדש אל השרת, לכתובת addContact/ בשיטת POST.

22. יש לייצור ניתוב בשרת המאפשר קליטת איש קשר חדש בכתובת addContact/
    יש לקלוט את הניתוב ב POST, כלומר
    server.post('/addContact',function(req,res) { })
    ניתוב זה יקלוט את איש הקשר החדש ויוסיף אותו לבסיס נתונים.
    אתגר - תשובה של הצלחה תחזיר (לדפדפן) את האובייקט החדש.

קליטת פרמטרים ב post מתוך השרת מתאפשרת בעזרת שימוש ב body-parser אותו יש להתקין באמצעות npm ולייבא כמודול לתוך server.js.

```
var bodyParser = require('body-parser');

server.use( bodyParser.json() );

var postedObject = request.body;
```

חלק ה'

23. באופן זהה יש לייצור את תהליך העדכון של רשומה קיימת.
יש לייצור ניתוב באנגולר לצורך עדכון, כמו גם בצד שרת.
העדכון בפועל יתבצע בבסיס נתונים.

24. יש לייצור תהליך של מחיקת איש קשר.