עידן בן עברי – ת.ז. 304846348
עידן וייצמן – ת.ז. 312614175

סביבות פיתוח באינטרנט
עבודה 3.1

ID	Method Name	HTTP Method	Parameters	Returns	Explanations
1	/auth/login	POST	{ username: string, password: string }	{ Token }	the method is getting the user's data from the server. We use POST and not GET due to the fact that GET is not secured. We have no interest of enabling the password of any user to be visible.
2	/auth/register	POST	{ username: string, password: string, firstName: string, lastName: string, city: string, country: string, email: string, Categories: string, momOriginLastName: string, elementarySchoolName: string, favouriteColor: string, childhoodBFF: string }	{ ans: boolean }	Saves the new users data in the server. Thus uses POST method. The user must supply all the information and min of 2 answers for the last 4 questions and min 2 categories. Returns TRUE for a successful registration, else returns FALSE. The parameters "momOriginLastName, elementarySchoolName, favouriteColor, childhoodBFF" are four password recovery question. the categories should be in a string seperated by a comma. the options are Food, Culture, Shopping, Night_Life
3.1	/auth/recoveryQuestions	POST	{username}	[{question: answer}]	Receive the username and checks if it's a registered user. If he is, then return an array containg an object for each question and answer the user had answered during registration. If the username is not registered, then return object containing message 'username not found'
3.2	/auth/passwordRecovery	POST	{ username: string, question: string, answer: string }	{ password:string }	POST instead of GET for security. No DB change is made. The user must provide a valid answer and if it matches the question's answer in the DB the user's password will be returned. The user had answered more than one question when he registered so when he'll want to recover his password he will be presented with one of the questions he had answered.
4	/guest/randExplorePOIs/:minRating	GET	{ minRating: int }	[{ POI }, { POI }, { POI }] ********** POI = { name: string, picture: image, numViews: int, poiDescription: string, poiRank: float, category: string, reviews: [{ review_content: string, rankVal: int }] }	GET method returns three random points of interest with rating equal or above the minRating value. If no POI with matching rating than return message "No relevant POI's found" with error code 400. No DB change is made.

5	/guest/userRecommended_POIs/:username	GET	{ username: string }	[{ POI }, { POI }]	GET method looks for the user's favourite categories and find two matching popular points of interest. One for each category. No DB change is made.
6	/guest/userRecentSaves/:username	GET	{ username: string }	[{ POI }, { POI }]	GET method access the user's data and look for the last two points of interest he had saved. If two or more POI's found, return the last two. Else, display error message. No DB change is made.
7	/user/poi/getAll_POI	GET	~	POI [ ]	GET method returns an array containing all the points of interest available in the city. The array will be group by categories. No DB change is made.
8	/user/poi/removeFavouritePOI/:username/:poi_name	DELETE	{ username: string, poi_name: string }	{ ans: boolean }	POST method finds the user's data and delete the point of interest to the list of the user's favourite points. If the point had successfully deleted then return TRUE. Else, return FALSE, meaning no POI with this name in DB or this user did not saved the given POI. If parameters sent no correctly than returns 400 error code.
9	/user/getFavouritesNum/:username	GET	{ username: string }	{ ans: int }	GET method returns the number of favourites POI the user saved. No DB change is made.
10	/user/getUserFavourites/:username	GET	{ username: string }	POI [ ]	GET method returns the user's list of favourites points of interest. No DB change is made.

עידן בן עברי – ת.ז. 304846348
עידן וייצמן – ת.ז. 312614175

11	/user/savePOI_server	POST	{ username: string, favourites [:string] }	~	POST method saves the list of user's favourites in the DB. The elements in favourites array will be {poi_name: "", timeDate: "YYYY-MM-DD hh:mm:ss.sss"}. The user's favourites will be saved in the DB in the exact order he had placed them, according to the order he sorted them.
12	/guest/poi/addPOIreview	POST	{ poi_name: string, review_content: string, rankVal: int }	~	POST method finds the point of interest object by name. review_content may be an empty string if the user didn't enter a review content. However, the user must enter a numeric value between 1-5 for rankVal inorder to submit the review. If the user entered a poi_name which is not valid an error message "Point Of Interest - $poi_name does not exist." will be returned.
13	/guest/poi/addPOIview	POST	{ poi_name: string }	~	increment the poi views value by one. Return error code 400 if poi_name is not found.

<u>שינויים</u>

הוספנו לשרת את השורה 'app.use(CORS())' כיוון שנתקלנו בבעיה בדפדפן כרום. הבעיה נוצרת מזה שהדפדפן מזהה שהשליחה וקבלת הנתונים מתבצעת ממקורות שונים ולכן מזהה זאת פרצת אבטחה וחוסם את הגישה. השימוש ב-CORS מאפשר את העברת הנתונים.

1. בשיטה '/auth/login' הוספנו בדיקה שאם ה-username והסיסמא הם לא בדיוק אותו הדבר (אותיות קטנות\גדולות) אז תחזיר 403. כי קודם לא היה אכיפה לגבי אותיות גדולות/קטנות

2. בשיטה '/guest/randExplorePOIs/:minRating ' שינינו את השאילתה שתבצע LEFT JOIN כדי שתחזיר גם את כל השורות של נק עניין שאין להם ביקורות!
בנוסף שמרנו בDB את ה-URL של כל תמונה במקום קודם את הקובץ של התמונה כיוון שנתקלנו בקשיים לשלוף את התמונה מה-DB להציג אותה. SSMS שומר תמונות בצורה דחוסה ולא הצלחנו לשחזר את התמונה עצמה בעת השליפה מה-DB. לכן עברנו לשמור מחרוזת של URL של כל תמונה.

3. הוספנו את הפונקציה בנתיב '/auth/recoveryQuestions' אשר מקבלת שם משתמש, בודקת אם הוא אכן רשום במערכת. ומידת וכן מחזירה שאלה ותשובה לשחזור סיסמא של אותו המשתמש. השאלה אותה מחזירים נבחרת מצורה רנדומלית מסך השאלות שחזור סיסמא עליהן בחר המשתמש לענות בעת הרישום לאתר. נאלצנו להוסיף את הפונקציה הזו כיוון שלא ניתן היה אפשרי להשיג את שאלת שחזור הסיסמא קודם לכן. לפני כן, כתבנו רק פונקציה אשר מקבלת שאלה ותשובה ומאמתת אותן ומחזירה את הסיסמא של המשתמש במידה והתשובה תואמת את זו השמורה בבסיס הנתונים. אך לא ניתן היה לקבל את השאלה על מנת שניתן יהיה להציג אותה למשתמש.

4. בשיטה שמוחקת נק' מועדפת מהשרת, כתובת השיטה בשרת שונתה להיות '/user/poi/removeFavouritePOI/:username/:poi_name'. השינוי הוא כי השתמשנו ב-$http עבור בקשות HTTP. לפי ה-API של אותו service, לבקשות מסוג DELETE ניתן להעביר פרמטרים אך ורק מה-URI ולכן בוצע השינוי.

5. בשיטה '/user/getUserFavourites/:username ' בשרת היה כתוב נתיב:
/user/poi/getUserFavourites/:username וזה היה בטעות. בנוסף שינינו את הפונקציה ככה שתחזיר את השורות שבטבלת מועדפים ולא את כל הפרטים של הנק עניין המועדפות כי אנחנו צריכים את הזמן ותאריך שבו כל נק' נשמרה בשביל השמירה העתידית והחזרת שתי נק' העניין האחרונות שנשמרו.

עידן בן עברי – ת.ז. 304846348
עידן וייצמן – ת.ז. 312614175