**API מעודכן**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Method Name** | **HTTP Method** | **Parameters** | **Returns** | **Explanation** |
| 1 | /Users/LogIn | POST | {“UserName”, “Password”} | Token | Login cannot pass in GET because it not secured. |
| 2 | /Users/Register | POST | JSON userDetails  {“username”,”Password”,  ”FirstName”,“LastName”,  ”City”,”Country”,”Email”,  “FavoriteCategories”, “questionID“,  questionsAns”} | Token | The user’s details are stored at the DB.  True=success  False=fail |
| 3 | /Users/RestorePassword | POST |  | Token | Using POST because GET is not secured for the user name and the user’s answers transfer together. |
| 4 | /POI/ (Midelware) | Use??? | Token |  | Using for verify the user token.  Return a massage if the authentication of the token failed |
| 5 | **/Users/POIName/:POI\_Name** | GET |  |  |  |
| 6 | **/Users/PopularPOI** | GET | - | {“POI\_Name”} | Returns the name of all popular POI **above threshold rank** (60%) from the DB. No changes at the DB. |
| 7 | /POI/Category/: Category\_Name | GET | {“ Category\_Name”} | JSON of POIs  {“POI\_Name”, “Category\_Name, “Num\_Of\_Views”,  ”Rank\_Per”, “POI\_Picture”, “Review\_Desc\_1”, “Review\_Desc\_2”, “Review\_Date\_1”, “Review\_Date\_2”} | Return list of all the POI by the given category. Using GET because no changes in the DB IS required. |
| 8 | /POI/RemovePOIFromUserFav | DELETE | { “POI\_Name”} | - | Delete POI from the user favorites list in the DB. |
| 9 | /POI/AddPOIToUserFavorites | POST | } “POI\_Name”} | - | Using POST because we need to add the POI to user favorites in the DB.  Return “FALSE” if failed |
| 10 | /POI/GetUser2PopularPOI | GET | - | JSON of 2 POIs  {“POI\_Name”, “Category\_Name, “Num\_Of\_Views”,  ”Rank\_Per”, “POI\_Picture”, “Review\_Desc\_1”, “Review\_Desc\_2”, “Review\_Date\_1”, “Review\_Date\_2”} | Return the user two most popular POI by the user favorites categories (one from each category) |
| 11 | /POI/GetUserAllFavPOI | GET | - | JSON of all the user favorites POIs  {“POI\_Name”, “Category Name, “Num\_Of\_Views”,  ”Rank\_Per”, “POI\_Picture”, “Review\_Desc\_1”, “Review\_Desc\_2”, “Review\_Date\_1”, “Review\_Date\_2”} | Return list of all the user favorites POI. using GET because there is no information to deliver to the server or to change in the DB |
| 12 | **/Users/GetAllPOI** | GET | - | {“POI\_Name”} | Return the Name Of all the POIs Using GET because no changes in the DB IS required. |
| 13 | /POI/POIRank | POST | {“POI\_Name”, “Rank”} | - | Using POST because we need to update the POI rank in the DB. |
| 14 | /POI/POIReview | POST | {“POI\_Name”, “Review”} | - | Using POST because we need to update the POI review in the DB. |
| 15 | /POI/RecentSavedPOI | GET | {“User\_Name”} | JSON of 2 POIs  {“POI\_Name”, “Category\_Name, “Num\_Of\_Views”,  ”Rank\_Per”, “POI\_Picture”, “Review\_Desc\_1”, “Review\_Desc\_2”, “Review\_Date\_1”, “Review\_Date\_2”} | Returns the user’s 2 last saved POI from the DB.  True=if found 2 POI  False=2 POI didn’t found |
| 16 | /POI/SaveAllFavPOIByUser | POST | {“POI\_Array: [“POI\_Name”, ”date”]} | - | Using post in order to add new favorites POI to the user favorites list. |
| 17 | /POI/SetlFavPOIOrderByUser | POST | {“POI\_Array: [“POI\_Name”, ”POI\_Order”]} | - | Using post in order to set the order of the favorites POI of the user. |
| 18 | /Users/QuestionIdByUserName/:userName | GET |  |  | Return the Question id of the user authentication question by his user Name |
| 19 | /Users/AllCategories | GET |  | {“Categorie\_Name”} | Return the Name Of all the Categories. Using GET because no changes in the DB IS required. |

**הערה:** כל בקשה של משתמש רשום (GET,POST,DELET....) מגיעה עםToken **בHEADER**, כלל הבקשות האלו מגיעות לModule – POI ומגיעות קודם כל ל Midelwareלטובת פיענוח של הtoken ובדיקת הנכונות על מול הDB.  
לכן הפרמטר token צויין רק בMidelware אך מגיעה בכל בקשה לPOI.  
כמו כן, הuser name אינו מועבר כפרמטר בבקשות אלו מפוענח מתוך הtoken.

**קישור לreposetory בgitHUB:** <https://github.com/ofirdvi/London>  
  
**שינויים שבוצעו כחלק ממימוש צד הלקוח ונימוקים עבורם**כגישה כללית וכחלק מהבנת תהליך הפיתוח שהינו תהליך איטראטיבי אשר במקרים רבים מצריך ביצוע שינויים על מנת לבצע את הדברים בצורה הנכונה ביותר בחרנו שכן להעדיף ביצוע שינויים לטובת מימוש נכון.  
על אף שקיימות דרכים לעקוף שינויים אלו ראינו בכך תהליך פחות נכון שעתיד ליצור בעיות במידה וידרשו להוסיף פ'יצרים למערכת או לבצע שינויים בעתיד על ידי מפתח אחר.  
  
הסיבה המרכזית לשינויים שביצענו הינה כי במימוש השרת התייחסנו לכך שכלל ההתעסקות עם נקודות העניין נוגעת למשתמשים רשומים בלבד ואילו משתמשים שאינם רשומים אינם יכולים לצפות פרטי נקודות עניין כלל.  
בכתיבת צד הלקוח, החלטנו שדבר זה אינו נכון ופרט ליכולת להחזיק רשימת מועדפים/ לשמור נק כלל הפונקציונאליות שייכת לכולם לכן העברנו את הפונקציות הבאות ממודול POI לUsers על מנת ליצור הפרדה נכונה בין בקשות שמכילות token שיש לפענח ולבדוק את התאמרתו לנתונים לבין בקשות שלא מכילות token.  
  
כמובן שניתן היה לא להעביר את הפונקציות בין המודולים ורק למנוע את הבדיקה של הtoken עבור הפונקציות הספציפיות, אבל דבר זה דרש להכניס IF ספציפים, דבר שהיה פוגע ברעיון שעומד מאחורי החלוקה לModuls ולדעתננו גם בפיתוח עתידי של המערכת לכן בחרנו שכן להעביר את הפונקציות ה"ל:

1. /POI/PopularPOI שונה ל -> /Users/PopularPOI

2. /POI/GetAllPOI שונה ל-> /Users/GetAllPOI

3. /POI/POIName/:POI\_Name שונה ל -> /Users/POIName/:POI\_Name

פונקציות שנוספו:

1. /Users/QuestionIdByUserName/:userName
2. /Users/AllCategories