**Full Stack webApp development documentation:**

1. **Creating the backend on Node.JS**
2. Install the dynamoDb locally so that database read and write can be performed without incurring any cost.

A screenshot of a computer screen

Description automatically generated

Fig 1. Database running locally (<http://localhost:8000>)

1. Create a database table ‘NodeJsProductJS’ with partition key as ‘text’ with key type HASH and attribute type String . Then insert the data from given ‘database.json’ file using Node.js project ‘dynamoDBTest’.

Please refer to this link - <https://github.com/PoojaGhosh2019/FullStack-webAPP-POC/tree/master/Backend/dynamoDBTest>

A screen shot of a computer

Description automatically generated

Fig 2 . Scanning the database table after inserting the values

1. Create backend for accessing the database, node.js project ‘dynamodb-backend’ . It provides following two API endpoints.
2. GET /ingredient/{key}
3. GET /fuzzy-search/

Please refer to this link - <https://github.com/PoojaGhosh2019/FullStack-webAPP-POC/tree/master/Backend/dynamodb-backend>

A screenshot of a computer screen

Description automatically generated

Fig 3 . Node.js server up and running

A picture containing screenshot

Description automatically generated

Fig 4 . Node.js server retrieving data from DynamoDB table using the ingredient key

1. **Creating the frontend on React.JS**

The frontend is designed in React which uses the GET APIs exposed by the backend module and displays the ingredients ‘text’ value and the ‘tags’ value on onChange event of the Input box.

Please refer to this link - <https://github.com/PoojaGhosh2019/FullStack-webApp-POC/tree/master/Frontend/healiai-test>

A screenshot of a computer

Description automatically generated

Fig 5 . Retrieving data from backend for special elements such as “abélmosk"

A screenshot of a computer screen

Description automatically generated

Fig 6 . No data updated if there is no valid request

A screenshot of a computer screen

Description automatically generated

Fig 7 . Ingredients getting updated on onChange event

A screenshot of a computer

Description automatically generated

Fig 8 . Tags getting updated on valid user’s request