# Lab 2

**Question 1: Explain the logic of the code above (Hint: think about the difference between the variables, value and dbValue).**  
The code checks for restaurant data in Redis first (value). If found, it sends the cached data ("cached": true). If not, it queries MongoDB (dbValue). If found, it caches the result in Redis and sends it ("cached": false). If neither has the data, it responds with an error.

**Question 2: Open the web browser and go to the URL, http://localhost:8000/restaurants/40356649 ; take a screenshot of the output and place it inside the DOCX file.**

A screen shot of a computer program

Description automatically generated

**Question 3: Open the web browser and go to the URL, http://localhost:8000/restaurants/40356649, again; what has changed from the answer of Question 2? Take a screenshot of the output and place it inside the DOCX file.**

The second time, the call fetches the data from the Redis cache rather than from the database. Key "cached" is true for the second call.

**Question 4: Explain what the curl command above intends to achieve.**

The curl command sends a POST request to create a new restaurant entry with the specified data (restaurant\_id, name, borough, cuisine) in the database.

**Question 5: Open the web browser and go to the URL, http://localhost:8000/restaurants/99999998; take a screenshot of the output and place it inside the DOCX file.**

A screenshot of a computer

Description automatically generated

**Question 6: Explain what the curl command above intends to achieve.**

This command sends a DELETE request to remove the restaurant with restaurant\_id: 99999998 from the database.

**Question 7: Open the web browser and go to the URL, http://localhost:8000/restaurants/99999998 again; what is the issue with the current caching implementation? Explain. Also, take a screenshot of the output and place it inside the DOCX file.**

A screenshot of a computer

Description automatically generated

Even after the restaurant was deleted from the database, it still exists in Redis cache. This results in stale data being returned for future requests to the same ID, causing inconsistency between the cache and the database.

**Question 8: Copy and paste the code updated into the DOCX file; follow the steps below to test and update the code as needed**

app.delete('/restaurants/:id', async (req, res) => {

  const restaurant\_id = req.params["id"];

  await redisClient.del(restaurant\_id);

  db.collection("restaurants").deleteOne({

    restaurant\_id: restaurant\_id,

  }).then(result => result.acknowledged && result.deletedCount >= 1 ?

    res.send("Success") :

    res.status(500).send("Failed")

  ).catch(() => res.status(500).send("Not Found"));

});

**Question 9: Open the web browser and go to the URL, http://localhost:8000/restaurants/99999997 ; take a screenshot of the output and place it inside the DOCX file.A screenshot of a computer

Description automatically generated**

**Question 10: Open the web browser and go to the URL, http://localhost:8000/restaurants/99999997 again; if the Step 1 of Exercise 3 is implemented correctly, the web browser must show “Not Found”. Take a screenshot and place it inside the DOCX file.**

A screenshot of a computer

Description automatically generated