# BUG FIX – CERTIFICATE DOWNLOAD INCONSISTENCY DUE TO MISSING PARAMETER AND DATABASE CONNECTION ISSUE.

## 1. Introduction

Users were unable to download their license certificates due to an intermittent issue with the API request parameters and database connectivity.

## 2. Problem Description

i. **Issue 1**: The external certificate download API call failed intermittently because the username/email parameter was sometimes missing from the URL.

ii. **Issue 2**: The username/email was fetched from the database, but it was inconsistently retrieved—sometimes the data was available, and other times it wasn't.

iii. **Issue 3**: The database was initialized in each controller, causing inconsistent connections and occasional failures to fetch data.

## 3. Root Cause Analysis

i. The missing username/email parameter was identified in the API request

ii. The database connection was being established multiple times per request, leading to inconsistent behaviours.

iii. On testing, database connection failures were observed due to poor connection handling and the need to reconnect on each request.

## 4. Solution

i. **Fixing the API Request**: Updated the API request to include the required username/email parameter consistently.

ii. **Fixing Database Connection Handling**: Refactored the app to establish a single database connection at startup, which all requests would reuse.

iii. **Testing**: Validated the fix by testing certificate downloads and database operations, confirming that the issue was resolved.

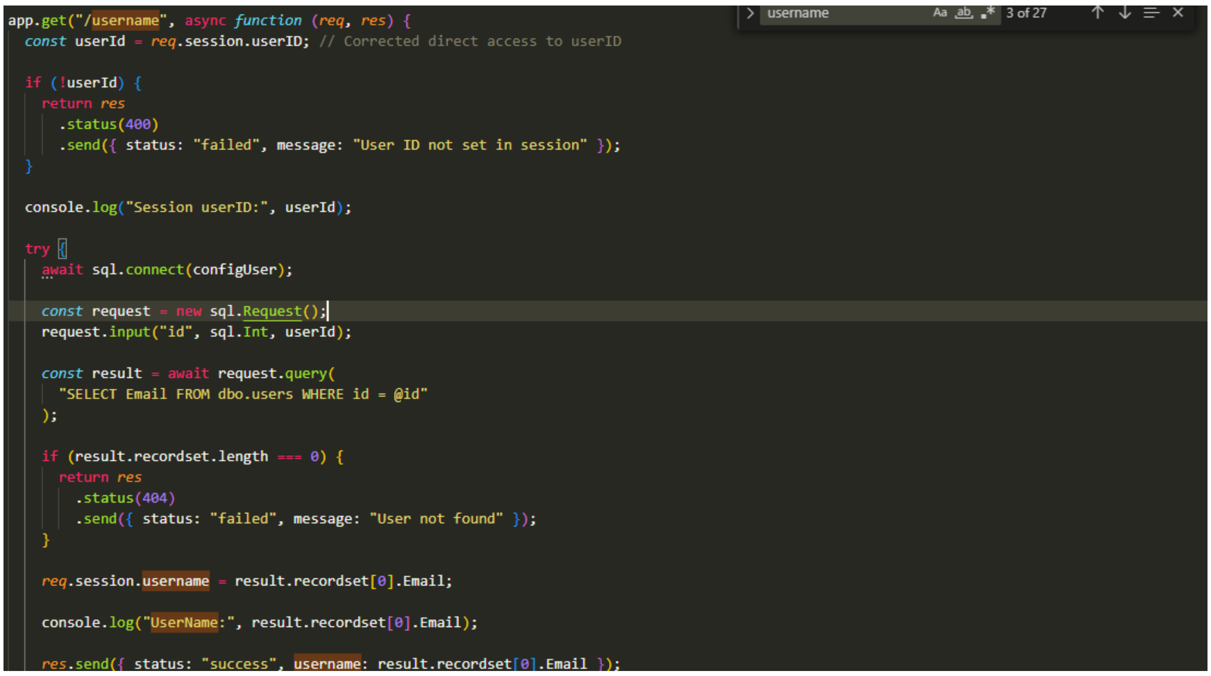
## 5. Outcome

The issue has been resolved, and users can now consistently download their certificates. The database connection issue was also fixed, and the app has been stable for 14 days without further incidents.

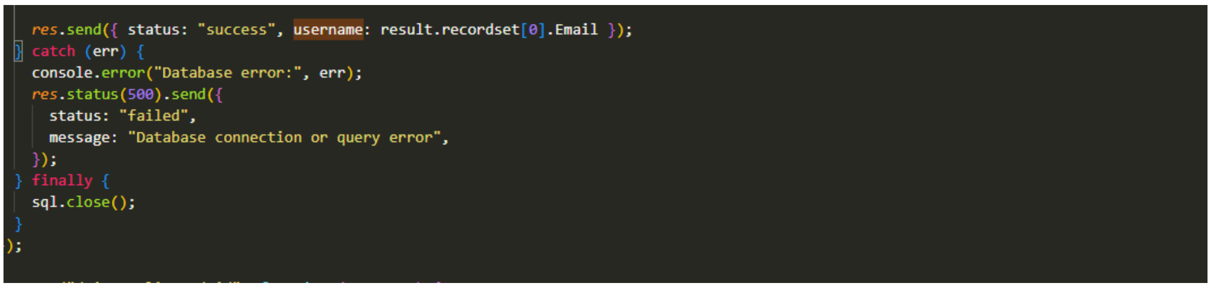
## 6. Conclusion

This fix improved both API call reliability and database stability, ensuring a smoother user experience for certificate downloads and reducing the risk of future issues.

## How it was

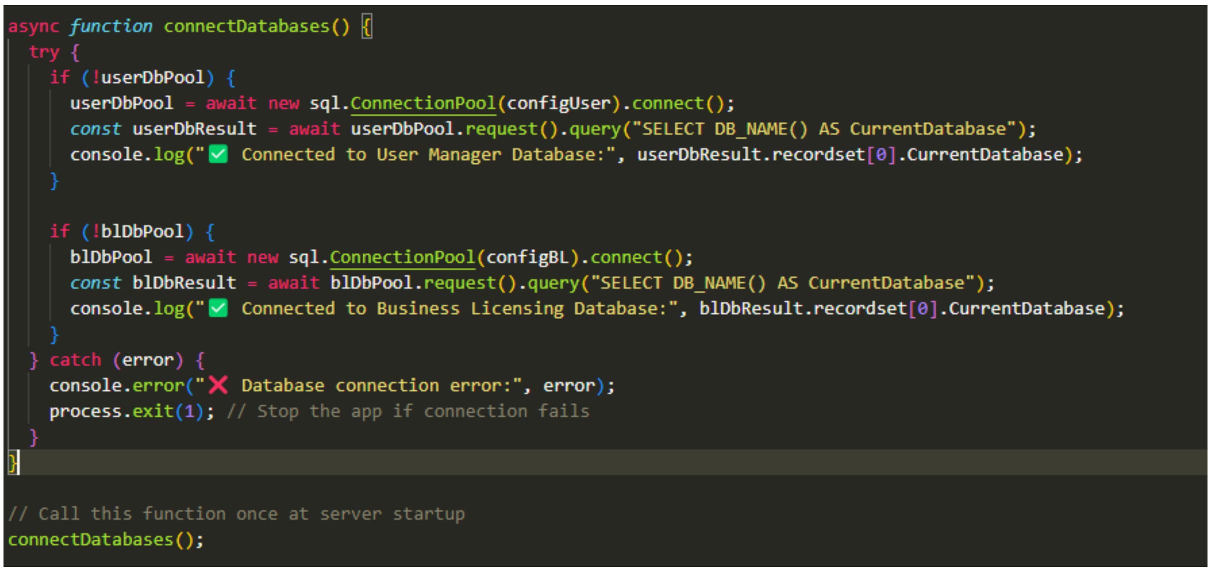


```
app.get("/username", async function (req, res) {
  const userId = req.session.userID; // Corrected direct access to userID

  if (!userId) {
    return res
      .status(400)
      .send({ status: "failed", message: "User ID not set in session" });
  }

  console.log("Session userID:", userId);

  try {
    await sql.connect(configUser);

    const request = new sql.Request();
    request.input("id", sql.Int, userId);

    const result = await request.query(
      "SELECT Email FROM dbo.users WHERE id = @id"
    );

    if (result.recordset.length === 0) {
      return res
        .status(404)
        .send({ status: "failed", message: "User not found" });
    }

    req.session.username = result.recordset[0].Email;

    console.log("UserName:", result.recordset[0].Email);

    res.send({ status: "success", username: result.recordset[0].Email });
```

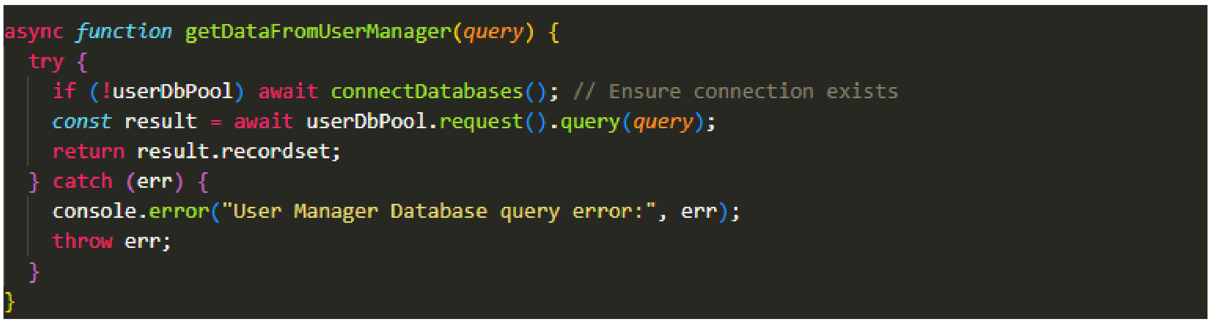


```
    res.send({ status: "success", username: result.recordset[0].Email });
  } catch (err) {
    console.error("Database error:", err);
    res.status(500).send({
      status: "failed",
      message: "Database connection or query error",
    });
  } finally {
    sql.close();
  }
});
```

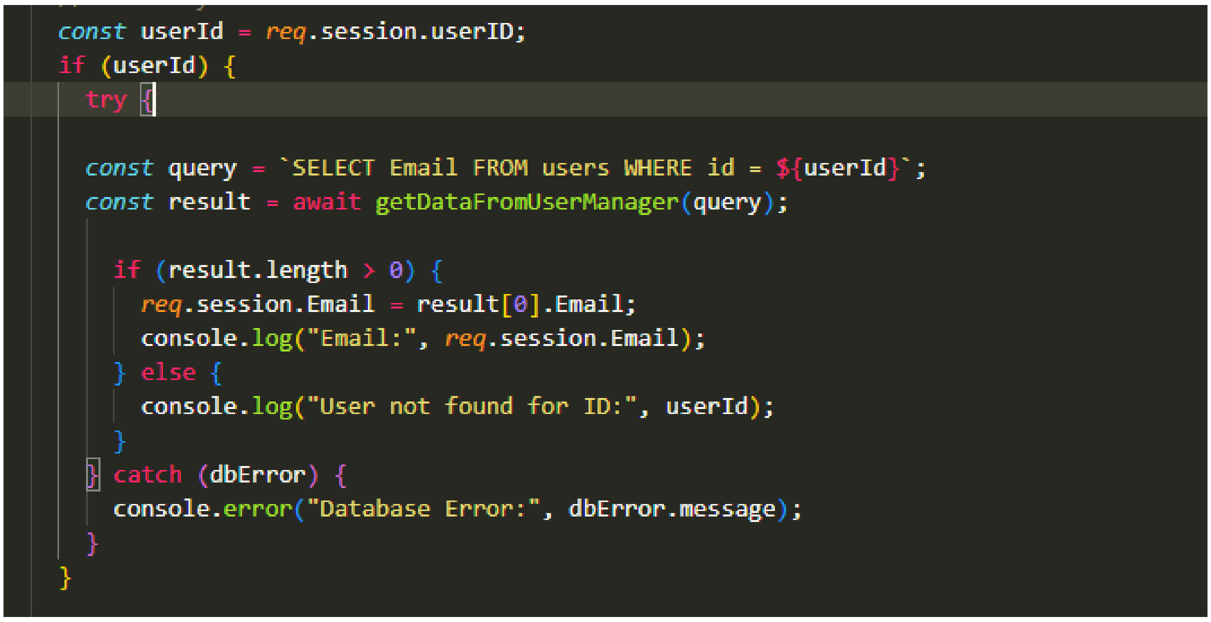
## How it works now



```
async function connectDatabases() {
  try {
    if (!userDbPool) {
      userDbPool = await new sql.ConnectionPool(configUser).connect();
      const userDbResult = await userDbPool.request().query("SELECT DB_NAME() AS CurrentDatabase");
      console.log("✅ Connected to User Manager Database:", userDbResult.recordset[0].CurrentDatabase);
    }

    if (!blDbPool) {
      blDbPool = await new sql.ConnectionPool(configBL).connect();
      const blDbResult = await blDbPool.request().query("SELECT DB_NAME() AS CurrentDatabase");
      console.log("✅ Connected to Business Licensing Database:", blDbResult.recordset[0].CurrentDatabase);
    }
  } catch (error) {
    console.error("❌ Database connection error:", error);
    process.exit(1); // Stop the app if connection fails
  }
}

// Call this function once at server startup
connectDatabases();
```



```
async function getDataFromUserManager(query) {
  try {
    if (!userDbPool) await connectDatabases(); // Ensure connection exists
    const result = await userDbPool.request().query(query);
    return result.recordset;
  } catch (err) {
    console.error("User Manager Database query error:", err);
    throw err;
  }
}
```



```
    const userId = req.session.userID;
    if (userId) {
      try {

      const query = `SELECT Email FROM users WHERE id = ${userId}`;
      const result = await getDataFromUserManager(query);

        if (result.length > 0) {
          req.session.Email = result[0].Email;
          console.log("Email:", req.session.Email);
        } else {
          console.log("User not found for ID:", userId);
        }
      } catch (dbError) {
        console.error("Database Error:", dbError.message);
      }
    }
```