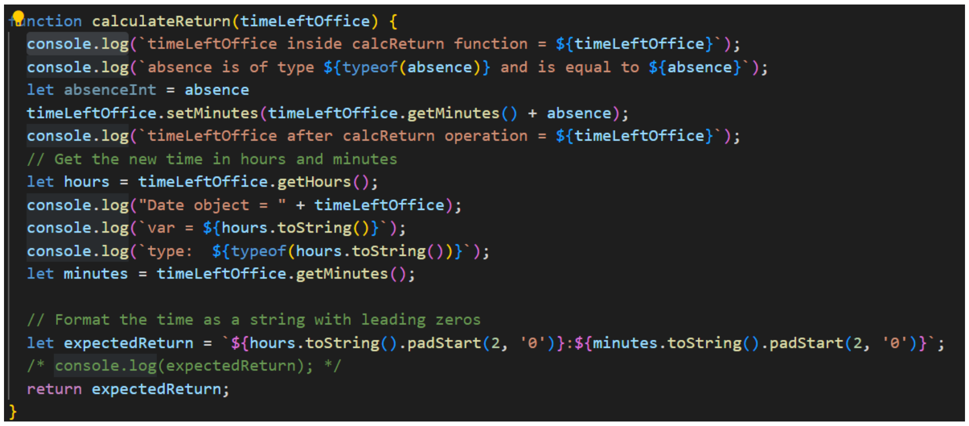
REFLECTION REPORT

The development of any project involves facing numerous challenges, and the web application in question is no exception. During the development process, we encountered several issues that required resolution to ensure the project's success.

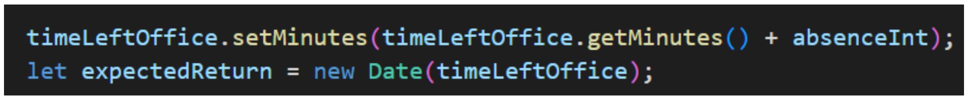
The project was divided into sprints and epics to make it easier to manage and monitor. This division helped us break down complex features into smaller, more manageable components. The use of agile methodology enabled the team to stay on top of the project's progress and ensure that it was delivered on time.



```
function calculateReturn(timeLeftOffice) {
  console.log(`timeLeftOffice inside calcReturn function = ${timeLeftOffice}`);
  console.log(`absence is of type ${typeof(absence)} and is equal to ${absence}`);
  let absenceInt = absence
  timeLeftOffice.setMinutes(timeLeftOffice.getMinutes() + absence);
  console.log(`timeLeftOffice after calcReturn operation = ${timeLeftOffice}`);
  // Get the new time in hours and minutes
  let hours = timeLeftOffice.getHours();
  console.log("Date object = " + timeLeftOffice);
  console.log(`var = ${hours.toString()}`);
  console.log(`type:  ${typeof(hours.toString())}`);
  let minutes = timeLeftOffice.getMinutes();

  // Format the time as a string with leading zeros
  let expectedReturn = `${hours.toString().padStart(2, '0')}:${minutes.toString().padStart(2, '0')}`;
  /* console.log(expectedReturn); */
  return expectedReturn;
}
```

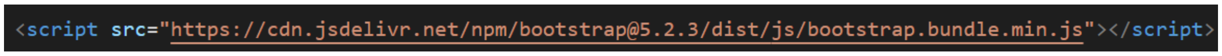
One of the significant issues we encountered was the "Invalid Date" error. The issue was related to a change in the Date object's type somewhere in the code, causing toString() not to work correctly. To resolve this issue, we identified the code section that caused the change and corrected it by passing the absence variable as a number, which was the intended type. We also renamed the absence variable to absenceInt for clarity.



```
timeLeftOffice.setMinutes(timeLeftOffice.getMinutes() + absenceInt);
let expectedReturn = new Date(timeLeftOffice);
```

Another challenge we faced was the date calculation issue. During the operation that calculated the 'returnTime,' the date object was being converted to a string object using the 'toString()' method. To address this issue, we created a new Date() object, using the string as the parameter, to ensure that we were always working with a 'Date()' object.

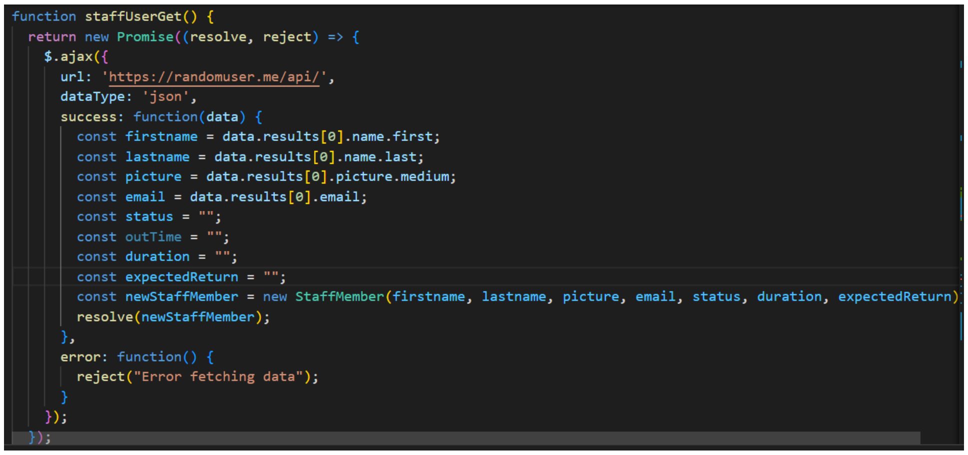
The customer requested that we implement a Bootstrap 'Toast' in the project requirements. However, we encountered issues with the implementation of toasts. Initially, the toasts were always showing, and there was no styling applied. We identified that the issue was due to the application using Bootstrap 3 instead of Bootstrap 4, which had introduced toasts. After loading Bootstrap 4, the toasts started working as expected.



```
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.min.js"></script>
```

Another issue we faced was that the toast was showing when "staffOut" was canceled. To fix this issue, we refactored the code and used if statements to check if the input from the prompt was truthy or falsey.

Finally, we encountered an issue where the Status, Out Time, Duration, and Expected Return were not using the objects' properties. According to the project's specifications, we needed to implement this feature before the project's delivery. To resolve this issue, we refactored the scripts to ensure the full use of the object properties in all instances of HTML creation by JavaScript. We also updated the property values using pure JavaScript without utilizing the DOM in these procedures, ensuring the maintainability of the codebase according to object-oriented design principles.

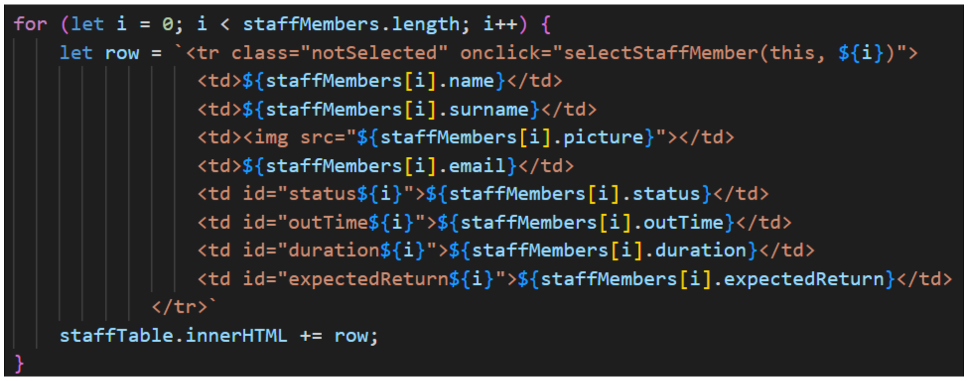


```javascript
function staffUserGet() {
  return new Promise((resolve, reject) => {
    $.ajax({
      url: 'https://randomuser.me/api/',
      dataType: 'json',
      success: function(data) {
        const firstname = data.results[0].name.first;
        const lastname = data.results[0].name.last;
        const picture = data.results[0].picture.medium;
        const email = data.results[0].email;
        const status = "";
        const outTime = "";
        const duration = "";
        const expectedReturn = "";
        const newStaffMember = new StaffMember(firstname, lastname, picture, email, status, duration, expectedReturn)
        resolve(newStaffMember);
      },
      error: function() {
        reject("Error fetching data");
      }
    });
  });
```

In conclusion, while the development of the web application presented several challenges, we were able to resolve them through collaboration and effective problem-solving techniques. The use of agile methodology helped us stay on track and deliver the project on time. We were able to provide the customer with a reliable and efficient web application that met their requirements.

By addressing these issues and implementing solutions, we were able to improve the functionality and usability of the web application. Throughout the development process, we faced several challenges, but by utilizing our problem-solving skills and collaborating effectively as a team, we were able to overcome these challenges and produce a high-quality product.

Another challenging task was ensuring that all object properties were being utilized in the HTML creation by JavaScript. This was important not only to meet project specifications but also for the maintainability of the codebase according to object-oriented design principles. By refactoring our code, we were able to ensure that all object properties were fully utilized in all instances of HTML creation, as well as updating property values using pure JavaScript without DOM utilization in these procedures.



```
for (let i = 0; i < staffMembers.length; i++) {
    let row = `<tr class="notSelected" onclick="selectStaffMember(this, ${i})">
            <td>${staffMembers[i].name}</td>
            <td>${staffMembers[i].surname}</td>
            <td><img src="${staffMembers[i].picture}"></td>
            <td>${staffMembers[i].email}</td>
            <td id="status${i}">${staffMembers[i].status}</td>
            <td id="outTime${i}">${staffMembers[i].outTime}</td>
            <td id="duration${i}">${staffMembers[i].duration}</td>
            <td id="expectedReturn${i}">${staffMembers[i].expectedReturn}</td>
        </tr>`
    staffTable.innerHTML += row;
}
```

Overall, we are proud of the final product we produced and the teamwork and problem-solving skills we demonstrated throughout the development process. We look forward to implementing any necessary updates and improvements and continuing to deliver high-quality products to our clients.