Software Implementation & Integration Document

Version 1

Antonis Andreou

Gabriel Vasile

Georgios Architektonides

Jorgos Xidias

Kyriacos Andreou

Stylianos Kyprianou

Cyprus University of Technology

REVISION CHART

Version	Primary Author(s)	Description of Version	Date Completed
Final	Antonis Andreou Gabriel Vasile Georgios Architektonides Jorgos Xidias Kyriacos Andreou Stylianos Kyprianou	This is the final version of the implementation and integration document.	17/04/2025

CONTENTS

1. INTRODUCTION

1.1 Purpose

This document is a complete guide outlining the strategies, tools, and methods used during the software's integration and testing stages. It includes a thorough explanation of the chosen frameworks and technologies, highlights the difficulties faced, and details several test cases. Its primary purpose is to assist with future maintenance, whether by the current team or new developers, by providing a clear and structured understanding of the system's design and operation. This not only helps avoid repeating previous errors but also streamlines and enhances the overall development workflow.

1.2 Scope

The system caters to the specific needs of the "Mobile Garage Larnaca" company by providing a comprehensive web application. From the user's perspective, the system will provide tools to effectively manage and update the inventory of parts, including the ability to add, edit, or remove items as needed. Users will also be able to input invoice details into the system, along with the parts purchased with each invoice. In addition, the system will maintain a detailed database of client information, including essential details such as name, contact information, service address, cars, and transaction history. This ensures that all client interactions are efficiently recorded and tracked. Additionally, the system will include the details of each ongoing or finished job. The administrator will have full access to these functionalities but will also be able to manage user accounts, including creating new ones, editing existing accounts, or deleting them if necessary. Lastly, the system will offer the ability to generate detailed annual and monthly statistics to assist in analyzing business performance and planning future strategies. By providing these functionalities, the platform will support the growth and operational efficiency of the Mobile Garage company.

1.3 Definitions, Acronyms, and Abbreviations

- Administrator (Admin): Refers to the individual responsible for managing the system. The administrator has the authority to oversee all system operations, including customer management, inventory control, job card creation, and the generation of statistical reports.
- User: Refers to the registered employees of the mobile garage company who interact with the system to manage customer records, schedule appointments, track job statuses, and update inventory.
- Client: Refers to the customers of the mobile garage company who request services or purchase products. Clients may include individual customers or corporate entities.

- Database: A structured and centralized repository used to store and manage all system data, including customer records, inventory details, job information, and transaction history.

1.4 References

Software Technology course CEI_324.

1.5 Overview

This document provides a thorough breakdown of the development tools used in building the system, including brief explanations of the reasons behind their selection and how they were utilized. It also outlines the technical challenges we faced, first in general terms and then with detailed accounts of specific issues. For each problem, we describe the situation, the solution implemented, and possible alternative approaches. To support our explanations, we include code samples that illustrate key functionalities, such as database integration and API interactions. Lastly, the document presents a set of test cases covering all major system features, using both black box and glass box testing methods where appropriate.

2. DEVELOPMENT TOOLS AND FRAMEWORKS

2.1 Development Tools

2.1.1 Integrated Development Environment (IDE)

2.1.1.1 Visual Studio Code

The project's software was developed using Visual Studio Code (v1.99.2), a widely used and lightweight code editor. It was chosen for its speed and efficiency, particularly in handling large codebases. A major advantage of VS Code is its support for extensions, many of which were employed throughout the development process. Notably, the Git extension was used to seamlessly sync updates to the project's Github repository. Furthermore, the editor's collaboration features allowed team members to work together in real time, which was especially valuable when troubleshooting problems.

IDE website: https://code.visualstudio.com/

2.1.2 Version Control System (VCS)

2.1.2.1 GitHub

In building the system, every developer on the team used Git locally to manage their code. By working on individual branches, each member could develop features independently, avoiding conflicts and streamlining parallel development. GitHub acted as the central hub where all code was stored, shared, and merged. This setup made collaboration smoother, as it allowed for easy tracking, discussion, and review of changes. As a result, team coordination improved and the development process became more organized and efficient.

VCS website: https://github.com/

2.1.3 Database Management System (DBMS)

2.1.3.1 phpMyAdmin

We used PhpMyAdmin for database management, which played a key role during development. It provided a range of features that helped us manage both the structure and the content of the database. The tool allowed us to insert and modify data easily, which was essential for testing the different features we built. Moreover, its user-friendly and

flexible interface made it easier to adjust the database design whenever necessary, making it a valuable asset throughout the project.

DBMS website: https://www.phpmyadmin.net/

2.2 Application Frameworks

2.2.1 Front-end Frameworks

2.2.1.1 Bootstrap

Bootstrap was used as the front-end framework due to its flexibility and ease of use. Its comprehensive documentation simplified the design process, and the built-in components helped accelerate development. One of its key advantages is responsive design, ensuring the system looks and functions smoothly across different screen sizes, which was crucial for this project.

Front-end Framework website: https://getbootstrap.com/

2.3 Testing Tools

We didn't use any Testing Tool, because we could test our code using previously mentioned tools.

3. CHALLENGES

3.1 Overall Difficulties

- Transitioning the system from a modular to an object-oriented approach introduced delays and added complexity, as it required the team to learn and adapt to object-oriented programming concepts.
- Communication and collaboration challenges arose, as this was our first experience working together on a software development project.
- Considerable time was spent on adaptation and research, as the selected programming languages were unfamiliar to most team members.
- Frequent database modifications were necessary to support evolving and changing system functionalities.
- Delays in meetings with the client to clarify specific system features and logic also impacted our ability to stay on schedule.
- When deploying the system to the server, numerous bugs and errors surfaced, all of which needed to be resolved under tight time constraints.

3.2 Specific Technical Challenges

3.2.1 Technical Challenge 1

One of the technical challenges we faced involved the job card photo upload feature. Specifically, allowing users to upload and attach images to job cards proved more complex than anticipated. We encountered issues related to file handling, storage configuration, and validating image formats and sizes. Additionally, ensuring that the uploaded images were properly linked to the correct job card and displayed consistently across the system required careful backend and frontend integration. These obstacles added unexpected development time and required us to explore solutions around secure file storage and user feedback for upload errors.

3.2.2 Technical Challenge 2.

Another technical challenge we faced was in the accounting part of the system, where accuracy was very important. The system needed to collect numbers from different managements like jobs management, parts, invoices and additional expenses, and then use those to perform various calculations. This was more complicated than expected, as even a small mistake in where the data came from or how it was calculated could lead to incorrect totals or reports. We had to spend extra time making sure the system was always using the correct data and doing the right calculations. It also required a lot of testing and reviewing to make sure everything worked as expected and gave reliable results.

3.2.3 Technical Challenge 3.

When we started combining all the object-oriented code. Even though each part of the system worked well on its own, putting everything together caused many unexpected problems. Some features stopped working or didn't behave as expected, and we had to spend time figuring out how different classes and objects were interacting. This made debugging more complex. We faced similar issues when deploying the system from localhost to the server, certain things broke due to changes in environment, file paths, or server settings. Fixing these problems took time and added pressure near the end of the system's development.

3.2.4 Technical Challenge 4.

In the part management section, we faced difficulties with implementing supplier name suggestions. The goal was to make it easier for users to select suppliers by showing suggestions as they typed, but getting this feature to work smoothly took more time than expected. We had to make sure the suggestions were accurate, fast, and didn't slow down the system. After some trial and error, we finally found a solution that worked well. Once it was in place, we were able to reuse the same logic in the invoice management section, which made that part of the system much easier to build.

3.3 Team Coordination

3.3.1 Communication Barriers

During the project, we faced several communication challenges. One of the main issues was the lack of continuous communication between team members due to different priorities and time constraints. Some of us had completely different sleep schedules, with a few working late hours while others had daytime responsibilities. Additionally, some members were able to dedicate more time to the project, while others had other obligations. Meeting in person was incredibly difficult, which is why we relied heavily on a messaging platform to share updates about each team member's progress and to report any issues that needed to be addressed.

This way of working, although flexible, introduced some difficulties — such as inconsistencies in error messages, user interface outputs, and frontend design choices. Despite these obstacles, through regular updates and collaboration on Messenger, we managed to keep communication open and eventually succeeded in bringing everything together and making it work.

3.3.2 Task Management

Our approach to managing tasks and schedules was a mix of structured planning and flexible communication. At the beginning of the project, we used both a Gantt chart and Trello to organize tasks, assign responsibilities, and set deadlines. Trello helped us visualize what needed to be done and track progress in a simple, drag-and-drop format. However, as the project progressed, we noticed that some team members were using Trello less consistently than others. Since we already had the Gantt chart to monitor overall progress and deadlines, we eventually stopped relying on Trello, as it became less necessary than we initially thought.

For daily communication, we primarily used Messenger, which worked well given our very different schedules—some of us worked late at night, while others were only available during the day. Messenger allowed us to stay in touch quickly, share updates, report bugs, and coordinate tasks in real time.

We used GitHub for version control, which was crucial for collaborating on code and avoiding merge conflicts. OneDrive was also used to share larger files or documents that weren't easily managed through GitHub.

While we didn't stick to a strict project management system throughout the entire project, the combination of these tools allowed us to stay organized enough. However, the lack of a single, consistent platform for task tracking did create some occasional confusion—especially when coordinating overlapping tasks or keeping track of responsibilities. Despite these challenges, we managed to adapt and keep things moving forward through open communication and teamwork.

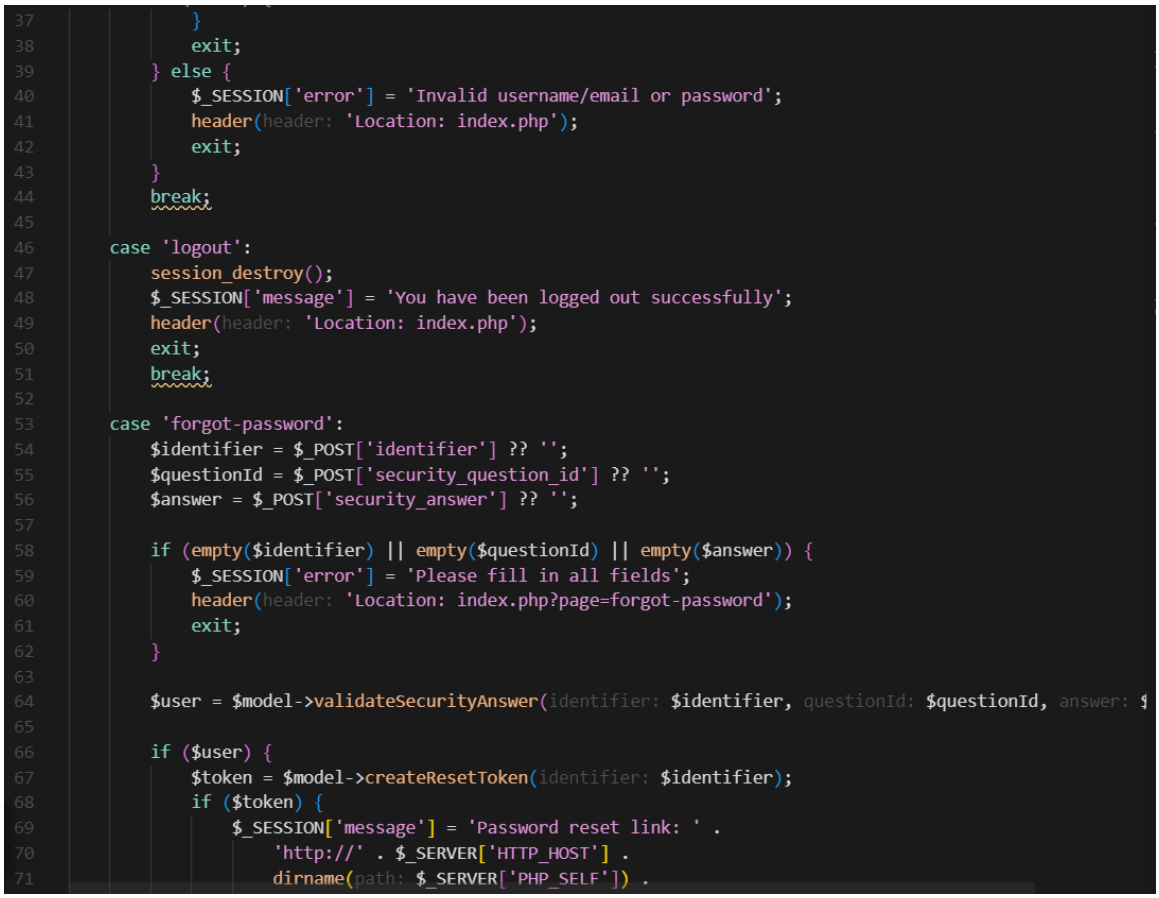
4. CODE SNIPPETS

4.1 Example Snippet 1 : Login Form

Code that creates a form where the user enters their credentials and submits the form to log in to their account.

Process.php

```php
<?php
if (session_status() === PHP_SESSION_NONE) {
    session_start();
}

require_once 'UserSystem.php';

$model = new UserModel();
$view = new UserView();

$action = $_POST['action'] ?? $_GET['action'] ?? '';

switch ($action) {
    case 'login':
        $identifier = $_POST['identifier'] ?? '';
        $password = $_POST['password'] ?? '';

        if (empty($identifier) || empty($password)) {
            $_SESSION['error'] = 'Please fill in all fields';
            header(header: 'Location: index.php');
            exit;
        }

        $user = $model->validateLogin(identifier: $identifier, password: $password);

        if ($user) {
            $_SESSION['user'] = $user;
            $_SESSION['message'] = 'Login successful!';

            // Check if there's a redirect URL stored
            if (isset($_SESSION['redirect_url'])) {
                $redirect_url = $_SESSION['redirect_url'];
                unset($_SESSION['redirect_url']); // Clear the stored URL
                header(header: 'Location: ' . $redirect_url);
            } else {
                header(header: 'Location: ../dashboard.php');
```



```
            }
            exit;
        } else {
            $_SESSION['error'] = 'Invalid username/email or password';
            header(header: 'Location: index.php');
            exit;
        }
        break;

    case 'logout':
        session_destroy();
        $_SESSION['message'] = 'You have been logged out successfully';
        header(header: 'Location: index.php');
        exit;
        break;

    case 'forgot-password':
        $identifier = $_POST['identifier'] ?? '';
        $questionId = $_POST['security_question_id'] ?? '';
        $answer = $_POST['security_answer'] ?? '';

        if (empty($identifier) || empty($questionId) || empty($answer)) {
            $_SESSION['error'] = 'Please fill in all fields';
            header(header: 'Location: index.php?page=forgot-password');
            exit;
        }

        $user = $model->validateSecurityAnswer(identifier: $identifier, questionId: $questionId, answer: $

        if ($user) {
            $token = $model->createResetToken(identifier: $identifier);
            if ($token) {
                $_SESSION['message'] = 'Password reset link: ' .
                    'http://' . $_SERVER['HTTP_HOST'] .
                    dirname(path: $_SERVER['PHP_SELF']) .
```

```
                    '/index.php?page=reset-password&token=' . $token;
            } else {
                $_SESSION['error'] = 'Failed to create reset token. Please try again.';
            }
        } else {
            $_SESSION['error'] = 'Invalid security question answer.';
        }

        header(header: 'Location: index.php?page=forgot-password');
        exit;
        break;

    case 'reset-password':
        $token = $_POST['token'] ?? '';
        $password = $_POST['password'] ?? '';
        $confirm_password = $_POST['confirm_password'] ?? '';

        if (empty($password) || empty($confirm_password)) {
            $_SESSION['error'] = 'Please fill in all fields';
            header(header: 'Location: index.php?page=reset-password&token=' . $token);
            exit;
        }

        if ($password !== $confirm_password) {
            $_SESSION['error'] = 'Passwords do not match';
            header(header: 'Location: index.php?page=reset-password&token=' . $token);
            exit;
        }

        if (strlen(string: $password) < 8) {
            $_SESSION['error'] = 'Password must be at least 8 characters long';
            header(header: 'Location: index.php?page=reset-password&token=' . $token);
            exit;
```

```php
            exit;
        }

        $tokenData = $model->validateResetToken(token: $token);

        if ($tokenData) {
            if ($model->updatePassword(identifier: $tokenData['user_id'], newPassword: $password)) {
                $model->markTokenAsUsed(token: $token);
                $_SESSION['message'] = 'Your password has been reset successfully.';
                header(header: 'Location: index.php');
                exit;
            } else {
                $_SESSION['error'] = 'Failed to update password. Please try again.';
                header(header: 'Location: index.php?page=reset-password&token=' . $token);
                exit;
            }
        } else {
            $_SESSION['error'] = 'Invalid or expired reset token.';
            header(header: 'Location: index.php?page=forgot-password');
            exit;
        }
        break;

    case 'register':
        $username = $_POST['username'] ?? '';
        $email = $_POST['email'] ?? '';
        $password = $_POST['password'] ?? '';
        $confirmPassword = $_POST['confirm_password'] ?? '';
        $questionId = $_POST['security_question_id'] ?? '';
        $answer = $_POST['security_answer'] ?? '';

        if (empty($username) || empty($email) || empty($password) || empty($confirmPassword) || empty($que
            $_SESSION['error'] = 'Please fill in all fields';
            header(header: 'Location: index.php?page=register');
            exit;
```

```
        }

        if ($password !== $confirmPassword) {
            $_SESSION['error'] = 'Passwords do not match';
            header(header: 'Location: index.php?page=register');
            exit;
        }

        if (strlen(string: $password) < 8) {
            $_SESSION['error'] = 'Password must be at least 8 characters long';
            header(header: 'Location: index.php?page=register');
            exit;
        }

        if ($model->registerUser(username: $username, email: $email, password: $password, questionId: $que
            $_SESSION['message'] = 'Registration successful! You can now login.';
            header(header: 'Location: index.php');
        } else {
            $_SESSION['error'] = 'Username or email already exists';
            header(header: 'Location: index.php?page=register');
        }
        exit;
        break;

    default:
        header(header: 'Location: index.php');
        exit;
        break;
}

```

Code to validate the credentials entered by the user in the login form.

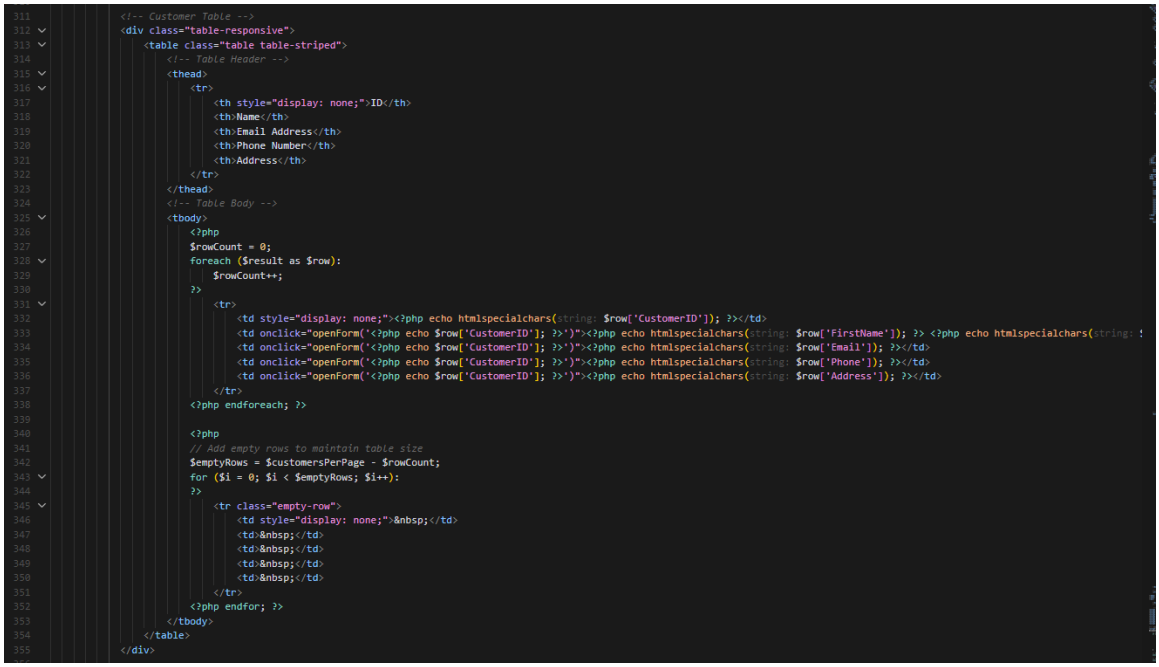
UserSystem.php

```php
<?php
if (session_status() === PHP_SESSION_NONE) {
    session_start();
}

require_once '../../config/db_connection.php';

// User Model Class
4 references | 0 implementations
class UserModel {
    12 references
    private $db;

    4 references | 0 overrides
    public function __construct() {
        global $pdo;
        $this->db = $pdo;
    }

    1 reference | 0 overrides
    public function validateLogin($identifier, $password): mixed {
        $stmt = $this->db->prepare(query: "SELECT * FROM users WHERE email = ? OR username = ?");
        $stmt->execute(params: [$identifier, $identifier]);
        $user = $stmt->fetch(mode: PDO::FETCH_ASSOC);

        if ($user && password_verify(password: $password, hash: $user['passwrd'])) {
            return $user;
        }
        return false;
    }
```

4.2 Example Snippet 2(How data is retrieved and handled in the software system)

Front-end part that displays customer data in a table (datatable).

Customer_main.php



```
<!-- Customer Table -->
<div class="table-responsive">
    <table class="table table-striped">
        <!-- Table Header -->
        <thead>
            <tr>
                <th style="display: none;">ID</th>
                <th>Name</th>
                <th>Email Address</th>
                <th>Phone Number</th>
                <th>Address</th>
            </tr>
        </thead>
        <!-- Table Body -->
        <tbody>
            <?php
            $rowCount = 0;
            foreach ($result as $row):
                $rowCount++;
            ?>
                <tr>
                    <td style="display: none;"><?php echo htmlspecialchars(string: $row['CustomerID']); ?></td>
                    <td onclick="openForm('<?php echo $row['CustomerID']; ?>')"><?php echo htmlspecialchars(string: $row['FirstName']); ?> <?php echo htmlspecialchars(string: $
                    <td onclick="openForm('<?php echo $row['CustomerID']; ?>')"><?php echo htmlspecialchars(string: $row['Email']); ?></td>
                    <td onclick="openForm('<?php echo $row['CustomerID']; ?>')"><?php echo htmlspecialchars(string: $row['Phone']); ?></td>
                    <td onclick="openForm('<?php echo $row['CustomerID']; ?>')"><?php echo htmlspecialchars(string: $row['Address']); ?></td>
                </tr>
            <?php endforeach; ?>

            <?php
            // Add empty rows to maintain table size
            $emptyRows = $customersPerPage - $rowCount;
            for ($i = 0; $i < $emptyRows; $i++):
            ?>
                <tr class="empty-row">
                    <td style="display: none;"> </td>
                    <td> </td>
                    <td> </td>
                    <td> </td>
                    <td> </td>
                </tr>
            <?php endfor; ?>
        </tbody>
    </table>
</div>
```

JavaScript part (AJAX call) used for communicating with the backend file and retrieving data in order to display it in a table (datatable).

get_print_customers.php

```php
<!-- Customer Table -->
<div class="table-responsive">
    <table class="table table-striped">
        <!-- Table Header -->
        <thead>
            <tr>
                <th style="display: none;">ID</th>
                <th>Name</th>
                <th>Email Address</th>
                <th>Phone Number</th>
                <th>Address</th>
            </tr>
        </thead>
        <!-- Table Body -->
        <tbody>
            <?php
            $rowCount = 0;
            foreach ($result as $row):
                $rowCount++;
            ?>
                <tr>
                    <td style="display: none;"><?php echo htmlspecialchars(string: $row['CustomerID']); ?></td>
                    <td onclick="openForm('<?php echo $row['CustomerID']; ?>')"><?php echo htmlspecialchars(string: $row['FirstName']); ?> <?php echo htmlspecialchars(string: $
                    <td onclick="openForm('<?php echo $row['CustomerID']; ?>')"><?php echo htmlspecialchars(string: $row['Email']); ?></td>
                    <td onclick="openForm('<?php echo $row['CustomerID']; ?>')"><?php echo htmlspecialchars(string: $row['Phone']); ?></td>
                    <td onclick="openForm('<?php echo $row['CustomerID']; ?>')"><?php echo htmlspecialchars(string: $row['Address']); ?></td>
                </tr>
            <?php endforeach; ?>

            <?php
            // Add empty rows to maintain table size
            $emptyRows = $customersPerPage - $rowCount;
            for ($i = 0; $i < $emptyRows; $i++):
            ?>
                <tr class="empty-row">
                    <td style="display: none;"> </td>
                    <td> </td>
                    <td> </td>
                    <td> </td>
                    <td> </td>
                </tr>
            <?php endfor; ?>
        </tbody>
    </table>
</div>
```

The back-end functionality that involves retrieving data from the database and returning it in JSON format.

```php
1 reference | 0 overrides
function getPaginatedCustomers($limit, $offset): array {
    global $pdo;
    $sql = "SELECT
                c.CustomerID,
                c.FirstName,
                c.LastName,
                c.Company,
                (SELECT Address FROM addresses WHERE CustomerID = c.CustomerID LIMIT 1) as Address,
                (SELECT nr FROM phonenumbers WHERE CustomerID = c.CustomerID LIMIT 1) as Phone,
                (SELECT Emails FROM emails WHERE CustomerID = c.CustomerID LIMIT 1) as Email
            FROM customers c
            ORDER BY c.FirstName ASC
            LIMIT :limit OFFSET :offset";

    $stmt = $pdo->prepare(query: $sql);
    $stmt->bindValue(param: ':limit', value: $limit, type: PDO::PARAM_INT);
    $stmt->bindValue(param: ':offset', value: $offset, type: PDO::PARAM_INT);
    $stmt->execute();
    return $stmt->fetchAll(PDO::FETCH_ASSOC);
}
```

4.3 Example Snippet 3(Examples of how to insert a new part into the database)

HTML code example for a form to add a new part.

Add_parts_form.php

```php
<?php

require_once '../config/db_connection.php';
require_once '../includes/sanitize_inputs.php';
?>

<div class="form-container">
    <div class="top-container d-flex justify-content-between align-items-center">
        <a href="javascript:void(0);" onclick="window.location.href='parts_main.php'" class="back-arrow">
            <i class="fas fa-arrow-left"></i>
        </a>
        <div class="flex-grow-1 text-center">
            <h2 class="mb-0">Add New Part</h2>
        </div>
        <div style="width: 30px;"></div>
    </div>

    <div id="successAlert" class="alert alert-success alert-dismissible fade" role="alert" style="display: none;">
        <i class="fas fa-check-circle mr-2"></i>
        <span id="successMessage"></span>
        <button type="button" class="close" data-dismiss="alert" aria-label="Close">
            <span aria-hidden="true">&times;</span>
        </button>
    </div>

    <form action="../controllers/add_parts_controller.php" method="POST" id="partsForm">
        <!-- Supplier Section -->
        <h3>Supplier Details *</h3>
        <div class="supplier-section">
            <div class="row">
                <div class="col-md-4">
                    <div class="form-group">
                        <label for="supplier">Supplier Name *</label>
                        <input type="text" id="supplier" name="supplier" class="form-control" required autocomplete="off">
                        <input type="hidden" id="supplierID" name="supplierID" required>
                        <div id="supplierSuggestions" class="suggestions-container"></div>
                    </div>
                </div>
                <div class="col-md-4">
                    <div class="form-group">
                        <label for="supplierPhone">Supplier Phone <span class="text-info">*</span></label>
                        <input type="tel" id="supplierPhone" name="supplierPhone" class="form-control">
                        <small class="form-text text-muted">Either phone or email is required</small>
                    </div>
                </div>
```

```
                    </div>
                </div>
                <div class="col-md-4">
                    <div class="form-group">
                        <label for="supplierEmail">Supplier Email <span class="text-info">*</span></label>
                        <input type="email" id="supplierEmail" name="supplierEmail" class="form-control">
                    </div>
                </div>
            </div>
        </div>

        <!-- Part Details Section -->
        <h3>Part Details *</h3>
        <div class="row">
            <div class="col-md-6">
                <div class="form-group">
                    <label for="partDesc">Part/Description *</label>
                    <input type="text" id="partDesc" name="partDesc[]" class="form-control" required autocomplete="off">
                    <div class="part-suggestions"></div>
                </div>
            </div>
            <div class="col-md-3">
                <div class="form-group">
                    <label for="piecesPurch">Pieces Purchased *</label>
                    <input type="number" id="piecesPurch" name="piecesPurch[]" class="form-control" required min="1">
                </div>
            </div>
            <div class="col-md-3">
                <div class="form-group">
                    <label for="dateCreated">Date Created *</label>
                    <input type="date" id="dateCreated" name="dateCreated" class="form-control" required>
                </div>
            </div>
        </div>
        <div class="row">
            <div class="col-md-3">
                <div class="form-group">
                    <label for="pricePerPiece">Price Per Piece *</label>
                    <input type="number" id="pricePerPiece" name="pricePerPiece[]" class="form-control" step="0.01" required min="0">
                </div>
            </div>
```

```
                </div>
            </div>
            <div class="col-md-3">
                <div class="form-group">
                    <label for="sellingPrice">Selling Price *</label>
                    <input type="number" id="sellingPrice" name="sellingPrice[]" class="form-control" step="0.01" required min="0">
                </div>
            </div>
            <div class="col-md-3">
                <div class="form-group">
                    <label for="vat">VAT (%) *</label>
                    <input type="number" id="vat" name="vat[]" class="form-control" step="0.01" required min="0" max="100" value="19">
                </div>
            </div>
            <div class="col-md-3">
                <div class="form-group">
                    <label for="priceBulk">Price Bulk (total)</label>
                    <input type="number" id="priceBulk" name="priceBulk[]" class="form-control" step="0.01" readonly>
                </div>
            </div>
        </div>

        <div class="btngroup text-center mt-4">
            <button type="submit" class="btn btn-primary">Save <i class="fas fa-save"></i></button>
        </div>
    </form>
</div>

<script>
// Calculate total price including VAT
function calculatePriceBulk() {
    const pieces = parseFloat(document.getElementById('piecesPurch').value) || 0;
    const pricePerPiece = parseFloat(document.getElementById('pricePerPiece').value) || 0;
    const vat = parseFloat(document.getElementById('vat').value) || 0;

    const subtotal = pieces * pricePerPiece;
    const vatAmount = subtotal * (vat / 100);
    const total = subtotal + vatAmount;

    document.getElementById('priceBulk').value = total.toFixed(2);
}
```

```
// Add event listeners for price calculation
document.getElementById('piecesPurch').addEventListener('input', calculatePriceBulk);
document.getElementById('pricePerPiece').addEventListener('input', calculatePriceBulk);
document.getElementById('vat').addEventListener('input', calculatePriceBulk);

// Form submission handler
document.getElementById('partsForm').addEventListener('submit', function(e) {
    e.preventDefault();

    // Validate required fields
    if (!document.getElementById('supplier').value ||
        !document.getElementById('partDesc').value ||
        !document.getElementById('piecesPurch').value ||
        !document.getElementById('pricePerPiece').value ||
        !document.getElementById('sellingPrice').value ||
        !document.getElementById('vat').value ||
        !document.getElementById('dateCreated').value) {
        alert('Please fill in all required fields');
        return;
    }

    // Validate supplier contact
    if (!document.getElementById('supplierPhone').value && !document.getElementById('supplierEmail').value) {
        alert('Please provide either a phone number or email for the supplier');
        return;
    }

    // Submit form via AJAX
    const formData = new FormData(this);

    $.ajax({
        url: '../controllers/add_parts_controller.php',
        method: 'POST',
        data: formData,
        processData: false,
        contentType: false,
        dataType: 'json',
        success: function(response) {
            if (response.status === 'success') {
                $('#successMessage').text(response.message);
                $('#successAlert').addClass('show').show();
                setTimeout(function() {
                setTimeout(function() {
                    window.location.href = 'parts_main.php';
                }, 2000);
            } else {
                alert(response.message || 'An error occurred while saving the part.');
            }
        },
        error: function(xhr, status, error) {
            console.error('AJAX Error:', {xhr, status, error});
            alert('An error occurred while saving the part. Please try again.');
        }
    });
});
</script>
```

JavaScript code example using AJAX to send form data to the back-end.

Add_parts_form.php

```
// Handle form submission
$("#partsForm").on('submit', function(e) {
    e.preventDefault();

    // Clear previous error messages
    document.querySelectorAll('.is-invalid').forEach(el => el.classList.remove('is-invalid'));
    document.querySelectorAll('.invalid-feedback').forEach(el => el.style.display = 'none');

    // Required fields validation
    const requiredFields = {
        'dateCreated': 'Date',
        'supplier': 'Supplier name',
        'partDesc': 'Part description',
        'piecesPurch': 'Pieces purchased',
        'pricePerPiece': 'Price per piece',
        'sellingPrice': 'Selling price',
        'vat': 'VAT'
    };

    let hasErrors = false;

    // Check each required field
    for (const [fieldId, fieldName] of Object.entries(requiredFields)) {
        const field = document.getElementById(fieldId);
        if (!field.value.trim()) {
            hasErrors = true;
            field.classList.add('is-invalid');
            let feedback = field.nextElementSibling;
            if (!feedback || !feedback.classList.contains('invalid-feedback')) {
                feedback = document.createElement('div');
                feedback.className = 'invalid-feedback';
                field.parentNode.insertBefore(feedback, field.nextSibling);
            }
            feedback.textContent = `${fieldName} is required`;
            feedback.style.display = 'block';

            // Focus the first invalid field
            if (!document.querySelector('.is-invalid:focus')) {
                field.focus();
                field.scrollIntoView({ behavior: 'smooth', block: 'center' });
            }
```

```
            }
        }
    }

    if (hasErrors) {
        return;
    }

    // Structure the parts data
    const parts = [];
    const partDesc = document.getElementsByName('partDesc[]');
    const piecesPurch = document.getElementsByName('piecesPurch[]');
    const pricePerPiece = document.getElementsByName('pricePerPiece[]');
    const priceBulk = document.getElementsByName('priceBulk[]');
    const sellingPrice = document.getElementsByName('sellingPrice[]');
    const vat = document.getElementsByName('vat[]');

    for (let i = 0; i < partDesc.length; i++) {
        parts.push({
            partDesc: partDesc[i].value,
            piecesPurch: piecesPurch[i].value,
            pricePerPiece: pricePerPiece[i].value,
            priceBulk: priceBulk[i].value || null,
            sellingPrice: sellingPrice[i].value,
            vat: vat[i].value
        });
    }

    // Create FormData object
    const formData = new FormData();

    // Add non-array fields
    formData.append('supplier', document.getElementById('supplier').value);
    formData.append('supplierID', document.getElementById('supplierID').value);
    formData.append('supplierPhone', document.getElementById('supplierPhone').value);
    formData.append('supplierEmail', document.getElementById('supplierEmail').value);
    formData.append('dateCreated', document.getElementById('dateCreated').value);

    // Add parts data as JSON string
    formData.append('parts', JSON.stringify(parts));
```

```
345         formData.append('parts', JSON.stringify(parts));
346
347         // Debug log
348         console.log('Sending data:', {
349             supplier: document.getElementById('supplier').value,
350             supplierID: document.getElementById('supplierID').value,
351             dateCreated: document.getElementById('dateCreated').value,
352             parts: parts
353         });
354
355         // Submit the form
356         $.ajax({
357             url: '../controllers/add_parts_controller.php',
358             method: 'POST',
359             data: formData,
360             processData: false,
361             contentType: false,
362             dataType: 'json', // Tell jQuery to expect JSON response
363             success: function(response) {
364                 console.log('Response:', response);
365                 if (response.status === 'success') {
366                     $('#successMessage').text(response.message);
367                     $('#successAlert').addClass('show').show();
368                     setTimeout(function() {
369                         window.location.href = 'parts_main.php';
370                     }, 2000);
371                 } else {
372                     alert(response.message || 'An error occurred while saving the parts.');
373                 }
374             },
375             error: function(xhr, status, error) {
376                 console.error('AJAX Error:', {xhr, status, error});
377                 console.error('Response Text:', xhr.responseText);
378                 let errorMessage = 'An error occurred while saving the parts.';
379
380                 try {
381                     const response = JSON.parse(xhr.responseText);
382                     if (response.message) {
383                         errorMessage = response.message;
```

```
383                         errorMessage = response.message;
384                     }
385                 } catch (e) {
386                     if (xhr.responseText) {
387                         errorMessage = xhr.responseText;
388                     } else {
389                         errorMessage = error;
390                     }
391                 }
392
393                 alert(errorMessage);
394             }
395         });
396     });
```

4.4 Example Snippet 4(Examples of how to insert a new part into the database)

HTML code example for a form to edit an invoice.

Edit_invoice.php

```
<!DOCTYPE html>
<html lang="en">
<head>
    <!-- Meta tags for proper character encoding and responsive design -->
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Edit Invoice</title>

    <!-- CSS and JavaScript dependencies -->
    <link rel="stylesheet" href="../assets/styles.css">
    <link href="https://getbootstrap.com/docs/4.0/dist/css/bootstrap.min.css" rel="stylesheet">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.4/css/all.min.css">
    <!-- Load jQuery first -->
    <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js"></script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>

    <!-- Styles for popup messages -->
    <style>
        .popup {
            position: fixed;
            top: 20px;
            right: 20px;
            padding: 15px;
            border-radius: 5px;
            display: none;
            z-index: 1000;
        }
        .success { background-color: #d4edda; color: #155724; }
        .error { background-color: #f8d7da; color: #721c24; }
    </style>
</head>
<body>

<!-- Main Content Container -->
<div class="form-container">
    <!-- Top Navigation Bar with Title -->
    <div class="top-container d-flex justify-content-between align-items-center">
        <!-- Back Arrow Button -->
        <a href="javascript:void(0);" onclick="openForm('<?php echo $id; ?>')" class="back-arrow">
            <i class="fas fa-arrow-left"></i>
        </a>
        <!-- Title Display -->
```

```
        <!-- Title Display -->
        <div class="flex-grow-1 text-center">
            <h2 class="mb-0">Edit Invoice</h2>
        </div>
        <!-- Empty div for spacing -->
        <div style="width: 30px;"></div>
    </div>

    <!-- Validation Error Container (hidden by default) -->
    <div id="validationErrorContainer" class="alert alert-danger" style="display: none;">
        <strong>Please fill out all required fields:</strong>
        <ul id="validationErrorList"></ul>
    </div>

    <!-- Success Alert -->
    <div id="successAlert" class="alert alert-success alert-dismissible fade" role="alert" style="display: none;">
        <i class="fas fa-check-circle mr-2"></i>
        <span id="successMessage"></span>
        <button type="button" class="close" data-dismiss="alert" aria-label="Close">
            <span aria-hidden="true">&times;</span>
        </button>
    </div>

    <!-- Invoice Edit Form -->
    <form id="invoiceForm" action="../controllers/update_invoice_controller.php?id=<?php echo htmlspecialchars(string: $id); ?>" method="POST
        <!-- Hidden input for invoice ID -->
        <input type="hidden" name="invoice_id" value="<?php echo htmlspecialchars(string: $id); ?>">
        <input type="hidden" name="InvoiceID" value="<?php echo htmlspecialchars(string: $id); ?>">

        <!-- Invoice Information Section -->
        <div class="section-header">
            <i class="fas fa-file-invoice"></i>
            <span>Invoice Information</span>
        </div>

        <!-- Invoice Number and Date Fields -->
        <div class="row">
            <div class="col-md-6">
                <div class="form-group">
                    <label for="invoiceNr">Invoice Number</label>
```

```
                    <label for="invoiceNr">Invoice Number</label>
                    <input type="text" id="invoiceNr" name="invoiceNr" class="form-control"
                        value="<?php echo htmlspecialchars(string: $invoice['InvoiceNr']); ?>">
                </div>
            </div>
            <div class="col-md-6">
                <div class="form-group">
                    <label for="dateCreated">Date Created</label>
                    <input type="date" id="dateCreated" name="dateCreated" class="form-control"
                        value="<?php echo htmlspecialchars(string: $invoice['DateCreated']); ?>">
                </div>
            </div>
        </div>

        <!-- Supplier Section -->
        <div class="section-header">
            <i class="fas fa-building"></i>
            <span>Supplier Information</span>
        </div>

        <!-- Supplier Details with Autocomplete -->
        <div class="supplier-section">
            <div class="row">
                <div class="col-md-4">
                    <div class="form-group">
                        <label for="supplier">Supplier Name</label>
                        <input type="text" id="supplier" name="supplier" class="form-control"
                            value="<?php echo htmlspecialchars(string: $invoice['SupplierName'] ?? ''); ?>">
                        <input type="hidden" id="supplierID" name="supplierID" value="<?php echo htmlspecialchars(string: $invoice['SupplierID
                        <input type="hidden" name="original_supplier_phone" value="<?php echo htmlspecialchars(string: $invoice['SupplierPhone
                        <input type="hidden" name="original_supplier_email" value="<?php echo htmlspecialchars(string: $invoice['SupplierEmail
                        <div id="supplierSuggestions" class="suggestions-container"></div>
                    </div>
                </div>
                <div class="col-md-4">
                    <div class="form-group">
                        <label for="supplierPhone">Supplier Phone <span class="text-info">*</span></label>
                        <input type="tel" id="supplierPhone" name="supplierPhone" class="form-control"
                            value="<?php echo htmlspecialchars(string: $invoice['SupplierPhone'] ?? ''); ?>">
                        <small class="form-text text-muted">Either phone or email is required</small>
```

```
                    <small class="form-text text-muted">Either phone or email is required</small>
                </div>
            </div>
            <div class="col-md-4">
                <div class="form-group">
                    <label for="supplierEmail">Supplier Email <span class="text-info">*</span></label>
                    <input type="email" id="supplierEmail" name="supplierEmail" class="form-control"
                        value="<?php echo htmlspecialchars(string: $invoice['SupplierEmail'] ?? ''); ?>"
                        oninput="validateEmailFormat(this)"
                        oninvalid="validateEmailFormat(this)">
                    <div class="invalid-feedback" style="display: none;"></div>
                    <small class="form-text text-muted">Either phone or email is required</small>
                </div>
            </div>
        </div>
    </div>

    <!-- Financial Section -->
    <div class="section-header">
        <i class="fas fa-calculator"></i>
        <span>Financial Information</span>
    </div>

    <!-- VAT and Total Price Fields -->
    <div class="row">
        <div class="col-md-6">
            <div class="form-group">
                <label for="vat">VAT (%)</label>
                <input type="number" id="vat" name="vat" class="form-control" step="0.01" min="0" max="100"
                    value="<?php echo htmlspecialchars(string: $invoice['Vat']); ?>">
            </div>
        </div>
        <div class="col-md-6">
            <div class="form-group">
                <label for="total">Invoice Total Price</label>
                <input type="number" id="total" name="total" class="form-control" step="0.01" min="0"
                    value="<?php echo htmlspecialchars(string: $invoice['Total']); ?>">
            </div>
        </div>
    </div>
```

```
    </div>

    <!-- Parts Section -->
    <div class="section-header">
        <i class="fas fa-tools"></i>
        <span>Parts Information</span>
    </div>

    <!-- Parts List and Add Part Button -->
    <div class="parts-section" >
        <div class="parts-list mb-4" style="max-height: 300px; overflow-y: auto;">
        <?php foreach ($invoice['parts'] as $part): ?>
                <div class="part-item">
                    <div class="part-header">
                        <span class="part-desc"><?php echo htmlspecialchars(string: $part['PartDesc']); ?></span>
                        <span class="pieces-badge"><?php echo $part['PiecesPurch']; ?> pieces</span>
            </div>
                    <div class="part-pricing">
                        <div class="price-item">
                            <i class="fas fa-tag"></i>
                            <span>€<?php echo number_format(num: $part['PricePerPiece'], decimals: 2); ?> per piece</span>
                        </div>
                        <?php if ($part['PriceBulk']): ?>
                        <div class="price-item">
                            <i class="fas fa-boxes"></i>
                            <span>€<?php echo number_format(num: $part['PriceBulk'], decimals: 2); ?> bulk</span>
        </div>
        <?php endif; ?>
                        <div class="price-item selling-price">
                            <i class="fas fa-shopping-cart"></i>
                            <span>€<?php echo number_format(num: $part['SellPrice'], decimals: 2); ?> selling price</span>
        </div>
        </div>
        <div class="part-actions">
                        <button type="button" class="btn btn-primary edit-part-btn"
                                data-part-id="<?php echo $part['PartID']; ?>"
                                data-part-desc="<?php echo htmlspecialchars(string: $part['PartDesc']); ?>"
                                data-pieces="<?php echo $part['PiecesPurch']; ?>"
                                data-price="<?php echo $part['PricePerPiece']; ?>"
                                data-bulk="<?php echo $part['PriceBulk']; ?>"
                                data-selling="<?php echo $part['SellPrice']; ?>">
```

```
                                <i class="fas fa-edit"></i> Edit
                            </button>
                            <button type="button" class="btn btn-danger delete-part-btn" data-part-id="<?php echo $part['PartID']; ?>">
                                <i class="fas fa-trash"></i> Delete
                    </button>
                </div>
                 </div>
                <?php endforeach; ?>
            </div>



<button type="button" class="btn btn-success" id="addPartBtn">
                    <i class="fas fa-plus"></i> Add Part
                                        </button>
            <!-- Form Submit Button -->
            <div class="btngroup text-center mt-4">
              <button type="submit" class="btn btn-primary">Save <i class="fas fa-save"></i></button>
                <button type="button" class="btn btn-secondary" onclick="openForm('<?php echo $id; ?>')">Cancel</button>
            </div>


        </form>
    </div>

    <!-- Part Edit Modal -->
    <div class="modal fade" id="partModal" tabindex="-1" aria-labelledby="partModalLabel" aria-hidden="true">
        <div class="modal-dialog">
            <div class="modal-content">
                <div class="modal-header">
                    <h5 class="modal-title" id="partModalLabel">Edit Part</h5>
                    <button type="button" class="close" data-dismiss="modal" aria-label="Close">
                        <span aria-hidden="true">&times;</span>
                    </button>
                </div>
                <div class="modal-body">
                    <form id="partForm">
                        <input type="hidden" id="partId">
                        <div class="mb-3">
```

```
                        <div class="mb-3">
                            <label for="partDesc" class="form-label">Part Description</label>
                            <input type="text" class="form-control" id="partDesc" required>
                    </div>
                        <div class="mb-3">
                            <label for="piecesPurch" class="form-label">Pieces Purchased</label>
                            <input type="number" class="form-control" id="piecesPurch" min="1" required>
                    </div>
                        <div class="mb-3">
                            <label for="pricePerPiece" class="form-label">Price Per Piece</label>
                            <input type="number" class="form-control" id="pricePerPiece" min="0" step="0.01" required>
                    </div>
                        <div class="mb-3">
                            <label for="sellingPrice" class="form-label">Selling Price</label>
                            <input type="number" class="form-control" id="sellingPrice" min="0" step="0.01" required>
                    </div>
                    </form>
                </div>
                <div class="modal-footer">
                    <button type="button" class="btn btn-secondary" data-dismiss="modal">Cancel</button>
                    <button type="button" class="btn btn-primary" id="savePartBtn">Save Part</button>
                </div>
            </div>
        </div>
    </div>

    <!-- Delete Part Confirmation Modal -->
    <div class="modal fade" id="deletePartModal" tabindex="-1" role="dialog" aria-labelledby="deletePartModalLabel" aria-hidden="true">
        <div class="modal-dialog" role="document">
            <div class="modal-content">
                <div class="modal-header">
                    <h5 class="modal-title" id="deletePartModalLabel">Delete Part</h5>
                    <button type="button" class="close" data-dismiss="modal" aria-label="Close">
                        <span aria-hidden="true">&times;</span>
                    </button>
                </div>
                <div class="modal-body">
                    <p id="deletePartModalMessage">Are you sure you want to delete this part?</p>
                </div>
                <div class="modal-footer">
```

```
            </div>
            <div class="modal-footer">
                <button type="button" class="btn btn-secondary" data-dismiss="modal">Cancel</button>
                <button type="button" class="btn btn-danger" id="confirmDeletePartBtn">Delete</button>
            </div>
        </div>
    </div>
</div>
```

JavaScript code example using AJAX to send form data to the back-end.

Edit_invoice.php

```
// Form submission handling with validation
$("#invoiceForm").submit(function(e) {
    e.preventDefault();

    // Clear previous error messages
    document.querySelectorAll('.is-invalid').forEach(el => el.classList.remove('is-invalid'));
    document.querySelectorAll('.invalid-feedback').forEach(el => el.style.display = 'none');
    $('#validationErrorContainer').hide();
    $('#successAlert').hide();

    // Required fields validation
    const requiredFields = {
        'invoiceNr': 'Invoice number',
        'dateCreated': 'Date created',
        'supplier': 'Supplier name',
        'vat': 'VAT',
        'total': 'Total price'
    };

    let hasErrors = false;
    let errorMessages = [];

    // Validate required fields
    Object.entries(requiredFields).forEach(([field, label]) => {
        const input = $(`#${field}`);
        if (!input.val().trim()) {
            input.addClass('is-invalid');
            errorMessages.push(`${label} is required`);
            hasErrors = true;
        } else {
            input.removeClass('is-invalid');
        }
    });

    // Validate that either phone or email is provided
    if (!$('#supplierPhone').val() && !$('#supplierEmail').val()) {
        $('#supplierPhone').addClass('is-invalid');
        $('#supplierEmail').addClass('is-invalid');
        errorMessages.push('Either supplier phone or email is required');
```

```
            errorMessages.push('Either supplier phone or email is required');
            hasErrors = true;
        } else {
            $('#supplierPhone').removeClass('is-invalid');
            $('#supplierEmail').removeClass('is-invalid');
        }

        // Validate numeric fields
        const numericFields = {
            'vat': 'VAT',
            'total': 'Total price'
        };

        Object.entries(numericFields).forEach(([field, label]) => {
            const input = $(`#${field}`);
            const value = input.val().trim();
            if (value && (isNaN(value) || parseFloat(value) < 0)) {
                input.addClass('is-invalid');
                errorMessages.push(`${label} must be a valid number`);
                hasErrors = true;
            }
        });

        if (hasErrors) {
            // Show error messages in the validation container
            const errorList = $('#validationErrorList');
            errorList.empty();
            errorMessages.forEach(msg => {
                errorList.append(`<li>${msg}</li>`);
            });
            $('#validationErrorContainer').show();
            return false;
        }

        // Create form data
        const formData = new FormData(this);
        formData.append('invoice_id', document.querySelector('input[name="invoice_id"]').value);
        formData.append('InvoiceID', document.querySelector('input[name="InvoiceID"]').value);

        // Collect parts data
```

```
        // Collect parts data
        const parts = [];
        $('.part-item').each(function() {
            const partId = $(this).find('.edit-part-btn').data('part-id');
            const partDesc = $(this).find('.part-desc').text().trim();
            const piecesPurch = $(this).find('.pieces-badge').text().replace(' pieces', '').trim();
            const pricePerPiece = $(this).find('.price-item:first-child span').text().replace('€', '').replace(' per piece', '').trim();
            const priceBulk = $(this).find('.price-item:nth-child(2) span').text().replace('€', '').replace(' bulk', '').trim();
            const sellingPrice = $(this).find('.selling-price span').text().replace('€', '').replace(' selling price', '').trim();

            parts.push({
                partId: partId,
                partDesc: partDesc,
                piecesPurch: piecesPurch,
                pricePerPiece: pricePerPiece,
                priceBulk: priceBulk || null,
                sellingPrice: sellingPrice
            });
        });

        // Add parts data to form data
        formData.append('parts', JSON.stringify(parts));

        // Disable submit button and show loading state
        const submitBtn = $(this).find('button[type="submit"]');
        const originalBtnText = submitBtn.html();
        submitBtn.prop('disabled', true).html('<i class="fas fa-spinner fa-spin"></i> Saving...');

        // Submit the form
        $.ajax({
            url: '../controllers/update_invoice_controller.php?id=<?php echo htmlspecialchars(string: $id); ?>',
            method: 'POST',
            data: formData,
            processData: false,
            contentType: false,
            success: function(response) {
                try {
                    // Parse response if it's a string
                    const result = typeof response === 'string' ? JSON.parse(response) : response;
```

```
                    if (result.success) {
                        $('#successMessage').text(result.message || 'Invoice updated successfully!');
                        $('#successAlert').addClass('show').show();
                        setTimeout(function() {
                            openForm('<?php echo $id; ?>');
                        }, 2000);
                    } else {
                        // Show error message
                        const errorMsg = result.message || 'An error occurred while updating the invoice.';
                        $('#validationErrorList').html(`<li>${errorMsg}</li>`);
                        $('#validationErrorContainer').show();
                    }
                } catch (e) {
                    console.error('Error parsing response:', e);
                    $('#validationErrorList').html('<li>Error processing server response</li>');
                    $('#validationErrorContainer').show();
                }
            },
            error: function(xhr, status, error) {
                console.error('AJAX Error:', {xhr, status, error});
                let errorMsg = 'An error occurred while updating the invoice.';

                try {
                    const response = JSON.parse(xhr.responseText);
                    if (response.message) {
                        errorMsg = response.message;
                    }
                } catch (e) {
                    console.error('Error parsing error response:', e);
                }

                $('#validationErrorList').html(`<li>${errorMsg}</li>`);
                $('#validationErrorContainer').show();
            },
            complete: function() {
                // Restore submit button state
                submitBtn.prop('disabled', false).html(originalBtnText);
            }
        });
    });
```

4.5 Example Snippet 5(Examples of how to insert a new part into the database)

HTML code example for a form to delete a job card.

Job_card_view.php

```php
<div class="modal fade" id="deleteModal" tabindex="-1" role="dialog" aria-labelledby="deleteModalLabel" aria-hidden="true">
    <div class="modal-dialog" role="document">
        <div class="modal-content">
            <div class="modal-header">
                <h5 class="modal-title" id="deleteModalLabel">Delete Job Card</h5>
                <button type="button" class="close" data-dismiss="modal" aria-label="Close">
                    <span aria-hidden="true">&times;</span>
                </button>
            </div>
            <div class="modal-body">
                <p>What would you like to do with the parts used in this job card?</p>

                <div class="parts-list mb-3">
                    <h6>Parts Used:</h6>
                    <div class="table-responsive">
                        <table class="table table-sm">
                            <thead>
                                <tr>
                                    <th>Part</th>
                                    <th>Quantity</th>
                                    <th>Return to Stock</th>
                                </tr>
                            </thead>
                            <tbody>
                                <?php foreach ($parts as $part): ?>
                                <tr>
                                    <td><?php echo htmlspecialchars(string: $part['PartDesc']); ?></td>
                                    <td><?php echo htmlspecialchars(string: $part['PiecesSold']); ?></td>
                                    <td>
                                        <div class="custom-control custom-checkbox">
                                            <input type="checkbox" class="custom-control-input return-part-checkbox"
                                                id="return_<?php echo $part['PartID']; ?>"
                                                data-part-id="<?php echo $part['PartID']; ?>"
                                                data-quantity="<?php echo $part['PiecesSold']; ?>">
                                            <label class="custom-control-label" for="return_<?php echo $part['PartID']; ?>"></label>
                                        </div>
                                    </td>
                                </tr>
                                <?php endforeach; ?>
                            </tbody>
                        </table>
                    </div>
                </div>
            </div>
            <div class="modal-footer">
                <button type="button" class="btn btn-secondary" data-dismiss="modal">Cancel</button>
                <button type="button" class="btn btn-danger" onclick="deleteJobCard()">Delete Job Card</button>
            </div>
        </div>
    </div>
</div>

<script>

    // Function to load the edit form
function loadEditForm(jobId) {
    $.get('../../JobCard_Management/views/edit_job_card.php', { id: jobId }, function(response) {
        $('#dynamicContent').html(response);
    });
}

    // Set modal image source when a photo is clicked
    document.querySelectorAll('.photo-link').forEach(link => {
        link.addEventListener('click', function(e) {
            e.preventDefault();
            document.getElementById('modalImage').src = this.href;
        });
    });

    // Function to show delete confirmation modal
    function showDeleteModal() {
        $('#deleteModal').modal('show');
    }

    // Sticky header logic
    window.addEventListener('scroll', function() {
        const customerNameField = document.querySelector('.form-group:first-child');
        const stickyHeader = document.getElementById('sticky-customer-header');
        const headerHeight = document.querySelector('.top-container').offsetHeight;
```

```
392         Ctrl+L to chat, Ctrl+K to generate
393         if (customerNameField) {
394             const rect = customerNameField.getBoundingClientRect();
395             // Show the sticky header when the customer name field is scrolled out of view
396             if (rect.bottom <= headerHeight + 10) {
397                 stickyHeader.classList.remove('d-none');
398 
399                 // Adjust the sticky header position to be within the form container width
400                 const formContainer = document.querySelector('.form-container');
401                 if (formContainer) {
402                     const formContainerRect = formContainer.getBoundingClientRect();
403                     stickyHeader.style.maxWidth = (formContainerRect.width * 0.8) + 'px';
404                 }
405             } else {
406                 stickyHeader.classList.add('d-none');
407             }
408         }
409     });
410 
```

JavaScript code example using AJAX to send form data to the back-end.

```
422         // Send delete request with parts information
423         fetch('../controllers/delete_job_card.php', {
424             method: 'POST',
425             headers: {
426                 'Content-Type': 'application/json'
427             },
428             body: JSON.stringify({
429                 jobId: <?php echo $jobId; ?>,
430                 partsToReturn: checkedParts
431             })
432         })
433         .then(response => response.json())
434         .then(data => {
435             if (data.success) {
436                 // Redirect to job cards list
437                 window.location.href = 'job_cards_main.php';
438             } else {
439                 alert('Error deleting job card: ' + data.message);
440             }
441         })
442         .catch(error => {
443             console.error('Error:', error);
444             alert('An error occurred while deleting the job card');
445         });
446     }
447 
```

4.6 Example Snippet 6(Examples of how to insert a new part into the database)

HTML code example for a form to print an invoice.

PrintInvoice.php

```php
<?php
require_once '../../config/db_connection.php';

$jobId = isset($_GET['id']) ? (int)$_GET['id'] : null;

if (!$jobId) {
    die('No job ID provided');
}

// Get the next invoice number
$sql = "SELECT InvoiceNr FROM invoices ORDER BY InvoiceID DESC LIMIT 1";
$stmt = $pdo->prepare(query: $sql);
$stmt->execute();
$result = $stmt->fetch();

// Determine next invoice number
if ($result && !empty($result['InvoiceNr'])) {
    // Extract only the numeric part and increment
    $currentNumber = intval(value: $result['InvoiceNr']);
    $nextNumber = $currentNumber + 1;

    // Reset to 1 if over 9999
    if ($nextNumber > 9999) {
        $nextNumber = 1;
    }
} else {
    // Start from 1 if no invoices exist
    $nextNumber = 1;
}

// Format to 4 digits with leading zeros
$invoiceNr = str_pad(string: $nextNumber, length: 4, pad_string: '0', pad_type: STR_PAD_LEFT);

// Get job card details with customer and car info
$sql = "SELECT j.JobID, j.DateStart, j.DateFinish, j.DriveCosts,
        CONCAT(c.FirstName, ' ', c.LastName) as CustomerName,
        c.Company,
        a.Address,
        car.Brand, car.Model, car.LicenseNr,
        GROUP_CONCAT(DISTINCT pn.Nr) as PhoneNumbers,
        GROUP_CONCAT(DISTINCT e.Emails) as EmailAddresses
        FROM jobcards j
        LEFT JOIN jobcar jc ON j.JobID = jc.JobID
        LEFT JOIN cars car ON jc.LicenseNr = car.LicenseNr
        LEFT JOIN carassoc ca ON car.LicenseNr = ca.LicenseNr
```

```
        LEFT JOIN carassoc ca ON car.LicenseNr = ca.LicenseNr
        LEFT JOIN customers c ON ca.CustomerID = c.CustomerID
        LEFT JOIN phonenumbers pn ON c.CustomerID = pn.CustomerID
        LEFT JOIN emails e ON c.CustomerID = e.CustomerID
        LEFT JOIN addresses a ON c.CustomerID = a.CustomerID
        WHERE j.JobID = ?
        GROUP BY j.JobID";

$stmt = $pdo->prepare(query: $sql);
$stmt->execute(params: [$jobId]);
$jobCard = $stmt->fetch(mode: PDO::FETCH_ASSOC);

// Get parts used in this job
$partsSql = "SELECT p.PartDesc, jp.PiecesSold, jp.PricePerPiece, p.Vat
             FROM jobcardparts jp
             JOIN parts p ON jp.PartID = p.PartID
             WHERE jp.JobID = ?";

$partsStmt = $pdo->prepare(query: $partsSql);
$partsStmt->execute(params: [$jobId]);
$parts = $partsStmt->fetchAll(PDO::FETCH_ASSOC);

// Calculate totals
$subtotal = 0;
$totalVat = 0;

foreach ($parts as $part) {
    $lineTotal = $part['PiecesSold'] * $part['PricePerPiece'];
    $subtotal += $lineTotal;
    $totalVat += $lineTotal * ($part['Vat'] / 100);
}

// Add drive costs to subtotal
$subtotal += $jobCard['DriveCosts'];
$total = $subtotal + $totalVat;

// Insert new invoice record
try {
    // Start transaction
    $pdo->beginTransaction();

    // Insert new invoice record
    $insertSql = "INSERT INTO invoices (InvoiceNr, DateCreated, Vat, Total)
                  VALUES (?, CURDATE(), ?, ?)";
    $insertStmt = $pdo->prepare(query: $insertSql);
```

```php
    $insertStmt = $pdo->prepare(query: $insertSql);
    $insertStmt->execute(params: [$invoiceNr, $totalVat, $total]);
    $invoiceId = $pdo->lastInsertId();

    // Link invoice to job
    $linkSql = "INSERT INTO invoicejob (JobID, InvoiceID) VALUES (?, ?)";
    $linkStmt = $pdo->prepare(query: $linkSql);
    $linkStmt->execute(params: [$jobId, $invoiceId]);

    // Commit transaction
    $pdo->commit();
} catch (Exception $e) {
    // Rollback transaction on error
    $pdo->rollBack();
    die('Error creating invoice: ' . $e->getMessage());
}

// Calculate total cost
$totalCost = $total + $jobCard['DriveCosts'];
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Invoice #<?php echo $invoiceNr; ?></title>
    <style>
        @media print {
            body {
                font-family: Arial, sans-serif;
                margin: 40px;
                color: #333;
            }
            .invoice-header {
                border-bottom: 2px solid #eee;
                padding-bottom: 20px;
                margin-bottom: 30px;
            }
            .invoice-title {
                font-size: 32px;
                color: #333;
                margin-bottom: 30px;
            }
            .company-info {
```

```
.company-info {
    float: left;
    width: 40%;
}
.invoice-info {
    float: right;
    width: 40%;
    text-align: right;
}
.clear {
    clear: both;
}
.client-info {
    margin: 10px 0;
    padding: 10px 0;
    border-bottom: 1px solid #eee;
}
table {
    width: 100%;
    border-collapse: collapse;
    margin: 20px 0;
}
th {
    background-color: #f8f9fa;
    border-bottom: 2px solid #dee2e6;
    padding: 12px;
    text-align: left;
}
td {
    padding: 12px;
    border-bottom: 1px solid #dee2e6;
}
.text-right {
    text-align: right;
}
.totals {
    width: 40%;
    float: right;
    margin-top: 20px;
}
```

```
        margin-top: 20px;
    }
    .totals table {
        width: 100%;
    }
    .totals table td {
        padding: 5px;
    }
    .total-row {
        font-weight: bold;
        font-size: 1.2em;
    }
    .footer {
        margin-top: 50px;
        padding-top: 20px;
        border-top: 1px solid #eee;
        text-align: center;
        font-size: 0.9em;
        color: #666;
    }
}
@media screen {
    body {
        font-family: Arial, sans-serif;
        margin: 40px;
        color: #333;
    }
    .invoice-header {
        border-bottom: 2px solid #eee;
        padding-bottom: 20px;
        margin-bottom: 30px;
    }
    .invoice-title {
        font-size: 32px;
        color: #333;
        margin-bottom: 30px;
    }
    .company-info {
        float: left;
```

```
            float: left;
            width: 40%;
        }
        .invoice-info {
            float: right;
            width: 40%;
            text-align: right;
        }
        .clear {
            clear: both;
        }
        .client-info {
            margin: 10px 0;
            padding: 10px 0;
            border-bottom: 1px solid #eee;
        }
        table {
            width: 100%;
            border-collapse: collapse;
            margin: 20px 0;
        }
        th {
            background-color: #f8f9fa;
            border-bottom: 2px solid #dee2e6;
            padding: 12px;
            text-align: left;
        }
        td {
            padding: 12px;
            border-bottom: 1px solid #dee2e6;
        }
        .text-right {
            text-align: right;
        }
        .totals {
            width: 40%;
            float: right;
            margin-top: 20px;
        }
        .totals table {
```

```
        .totals table {
            width: 100%;
        }
        .totals table td {
            padding: 5px;
        }
        .total-row {
            font-weight: bold;
            font-size: 1.2em;
        }
        .footer {
            margin-top: 50px;
            padding-top: 20px;
            border-top: 1px solid #eee;
            text-align: center;
            font-size: 0.9em;
            color: #666;
        }
    }
</style>
<script>
    window.onload = function() {
        window.print();
        // Close the window after print dialog closes
        window.onafterprint = function() {
            window.close();
        };
        // Also close if user cancels print dialog
        setTimeout(function() {
            window.close();
        }, 1000);
    }
</script>
</head>
<body>
    <div class="invoice-header">
        <div class="company-info">
            <img src="../../assets/logo.png" alt="Company Logo" style="max-width: 200px;">
            <p>Mobile Garage Larnaca<br>
```

```
            <p>Mobile Garage Larnaca<br>
                Phone: +35799851876<br>
                Email: mobilegaragelarnaca@outlook.com</p>
        </div>
        <div class="invoice-info">
            <h1 class="invoice-title">INVOICE</h1>
            <p>
                Invoice #: <?php echo $invoiceNr; ?><br>
                Date of Call: <?php echo date(format: 'd/m/Y'); ?><br>
                Job Start Date: <?php echo !empty($jobCard['DateStart']) ? date(format: 'd/m/Y', timestamp: strtotime(datetime: $jobCard['Date
                Job End Date: <?php echo !empty($jobCard['DateFinish']) ? date(format: 'd/m/Y', timestamp: strtotime(datetime: $jobCard['DateF
            </p>
        </div>
        <div class="clear"></div>
    </div>

    <div class="client-info">
        <div style="display: flex; flex-wrap: wrap; justify-content: space-between;">
            <div style="width: 28%;">
                <strong>Bill To:</strong><br>
                <?php echo htmlspecialchars(string: $jobCard['CustomerName']); ?><br>
                <?php if (!empty($jobCard['Company'])): ?>
                    <?php echo htmlspecialchars(string: $jobCard['Company']); ?><br>
                <?php endif; ?>
                <?php echo htmlspecialchars(string: $jobCard['Address']); ?>
            </div>
            <div style="width: 28%;">
                <strong>Contact:</strong><br>
                Phone: <?php echo htmlspecialchars(string: $jobCard['PhoneNumbers']); ?><br>
                <?php if (!empty($jobCard['EmailAddresses'])): ?>
                    Email: <?php echo htmlspecialchars(string: $jobCard['EmailAddresses']); ?>
                <?php endif; ?>
            </div>
            <div style="width: 28%;">
                <strong>Vehicle Information:</strong><br>
                <?php echo htmlspecialchars(string: $jobCard['Brand'] . ' ' . $jobCard['Model']); ?><br>
                License Plate: <?php echo htmlspecialchars(string: $jobCard['LicenseNr']); ?>
            </div>
        </div>
```

```
        </div>
    </div>

    <table>
        <thead>
            <tr>
                <th>Description</th>
                <th>Quantity</th>
                <th>Unit Price</th>
                <th>VAT %</th>
                <th class="text-right">Amount</th>
            </tr>
        </thead>
        <tbody>
            <?php foreach ($parts as $part): ?>
                <tr>
                    <td><?php echo htmlspecialchars(string: $part['PartDesc']); ?></td>
                    <td><?php echo $part['PiecesSold']; ?></td>
                    <td>€<?php echo number_format(num: $part['PricePerPiece'], decimals: 2); ?></td>
                    <td><?php echo $part['Vat']; ?>%</td>
                    <td class="text-right">€<?php echo number_format(num: $part['PiecesSold'] * $part['PricePerPiece'], decimals: 2); ?></td>
                </tr>
            <?php endforeach; ?>
            <?php if ($jobCard['DriveCosts'] > 0): ?>
                <tr>
                    <td>Drive Costs</td>
                    <td>1</td>
                    <td>€<?php echo number_format(num: $jobCard['DriveCosts'], decimals: 2); ?></td>
                    <td>0%</td>
                    <td class="text-right">€<?php echo number_format(num: $jobCard['DriveCosts'], decimals: 2); ?></td>
                </tr>
            <?php endif; ?>
        </tbody>
    </table>

    <div class="totals">
        <table>
            <tr>
```

```
            <tr>
                <td>Subtotal:</td>
                <td class="text-right">€<?php echo number_format(num: $subtotal, decimals: 2); ?></td>
            </tr>
            <tr>
                <td>VAT:</td>
                <td class="text-right">€<?php echo number_format(num: $totalVat, decimals: 2); ?></td>
            </tr>
            <tr class="total-row">
                <td>Total:</td>
                <td class="text-right">€<?php echo number_format(num: $total, decimals: 2); ?></td>
            </tr>
        </table>
    </div>

    <div class="clear"></div>

</body>
</html>
```

4.7 Example Snippet 7(Examples of how to insert a new part into the database)

Code for connecting to the database.

```
<?php
// Database configuration
$host = 'localhost';
$dbname = 'webvaria_MobileGarageLarnaca';
$username = 'webvaria_MobileGarageLarnaca';
$password = 'vn{2i1;BA}@s';

try {
    // Create a new PDO instance
    $pdo = new PDO(dsn: "mysql:host=$host;dbname=$dbname;charset=utf8mb4", username: $username, password: $password);

    // Set PDO error mode to exception for better error handling
    $pdo->setAttribute(attribute: PDO::ATTR_ERRMODE, value: PDO::ERRMODE_EXCEPTION);
} catch (PDOException $e) {
    // Handle database connection errors
    die("Database connection failed: " . $e->getMessage());
}

// Export the $pdo object for use in other files
return $pdo;
?>
```

5. TEST CASES

5.1 Test Case Selection

5.1.1 Approach Used:

- **Black Box Testing:** For UI and form validations without looking at the source code.
- **Glass Box Testing:** For internal logic validation (e.g., token expiry, password hashing, and SQL execution).

5.1.2 Selection Criteria:

- **Equivalence Partitioning** for valid vs. invalid data input.
- **Boundary Value Analysis** for edge cases (empty fields, very long inputs).
- **Code Path Coverage** in unit tests to hit all logical branches.

5.2 Test Cases

5.2.1 Feature 1: User Acess

Functionality	Input	Expected Output	Actual Output
Validate Login Credentials	Identifier: user@example.com Password: test123	Login successful	User data returned and session started
Create New User	Username: newuser Email: new@example.com Password: pass123 Question ID: 1 Answer: dog	New user registered in the database	User created successfully
Create Reset Token	Identifier: user@example.com	Reset token is generated and stored with expiry	Token returned and stored correctly

Validate Reset Token and Update Password	Token: validtoken123 New Password: newpass	Password updated in the database	Password successfully updated and token marked used
Validate Security Answer	Identifier: user@example.com Question ID: 1 Answer: dog	User is validated and allowed to reset password	Correct user returned
Verify Admin Status	Identifier: admin@example.com	Returns true if user is admin	true returned if admin

5.2.2 Feature 2: User Management

Functionality	Input	Expected Output	Actual Output
Insert a new user into the system	Username: testuser Password: test123 Email: testuser@example.com Admin: No Security Q ID: 1 Security Answer: Blue	User added successfully, success message in session	User added, redirected with session message
Handle missing input	Username: (empty) Password: test123 Email: test@example.com	Error message: Failed to add user	Redirected with “Failed to add user.” message
Delete an existing user	Username: testuser	User deleted successfully,	Deleted and redirected with success message

		redirect with session message	
Update user information	Username: testuser Email: new@example.com Admin: Yes	User updated successfully	User updated, session shows success message
Update user and change password	Username: testuser New Password: pass456	Password hashed and updated in DB	Password updated, verified with hash in DB
Prevent injection in input fields	Username: test' OR '1'='1	Sanitized input; query fails or does not return extra data	Query blocked, no injection effect

5.2.3 Feature 3: Job Card Management

Functionality	Input	Expected Output	Actual Output
Create a job card for a car that does not yet exist	Registration: ABC123 VIN: TEMP_1234 Customer: John Date: today Parts: Brake pad (Qty: 2) Photos: 2 images	New car inserted in cars table Job card saved Photos uploaded Session message: "Job card saved successfully!"	All data inserted, success message displayed
Create a job card linked to an existing car	Registration: XYZ789 (already in DB) Customer: Jane Job Report: OK	Job card created without creating new car record	Success, job card linked to existing car
Validate required fields	Registration: (empty) Customer: Bob	Error: "Registration plate is required"	Error message shown and user redirected
Add multiple parts with quantities and pricing	Parts: Brake pad (Qty: 2, Price: 20)	Job card stores part IDs, prices, and quantities	Part details stored and shown in job report

	Oil filter (Qty: 1, Price: 15)		
Update an existing job card	Job ID: 12 Change: jobEndDate and additionalCosts	Updates saved, old data replaced	Updates reflected, success message shown
Delete a job card by ID	Job Card ID: 45	Job card deleted from DB	Success message in session

5.2.4 Feature 4: Parts Management

Functionality	Input	Expected Output	Actual Output
Create a new part linked to an existing supplier	Description: "Brake Pad", PriceBulk: 10.5, SellPrice: 20.5, Supplier: "ABC Inc.	Part created successfully and linked to existing supplier	Success, part linked to existing supplier "ABC Inc."
Validate required fields for part creation	Description: (empty), PriceBulk: 10.5, SellPrice: 20.5, Supplier: "ABC Inc."	Error: "Part description is required"	Error message shown and user redirected
Ensure the part is linked to a valid supplier	Description: "Brake Pad", PriceBulk: 10.5, SellPrice: 20.5, Supplier: (non-existent)	Error: "Supplier does not exist"	Error message shown and user redirected
Update part details	PartID: 1, Description: "Brake Pad", PriceBulk: 12.0, SellPrice: 22.0, Supplier: "XYZ Ltd."	Part updated successfully	Success, part updated with new details

5.2.5 Feature 5: Customer Management

Functionality	Input	Expected Output	Actual Output
Add a new customer to the database	First Name: "John", Last Name: "Doe", Company: "ABC Corp", Address: ["123 Elm St"], Phone: ["555-1234"], Email: ["john.doe@example.com"]	Success: Customer added with ID and all details stored in the database	Customer added successfully, redirected to the customer overview page
Validate required fields for adding a customer	First Name: (empty), Last Name: "Doe", Company: "ABC Corp", Address: ["123 Elm St"], Phone: ["555-1234"], Email: ["john.doe@example.com"]	Error: "First name is required"	Error message shown: "First name is required", user redirected to the form page
Update existing customer information	Customer ID: 1, First Name: "John", Last Name: "Doe", Company: "XYZ Ltd", Address: ["456 Oak St"], Phone: ["555-6789"], Email: ["john.doe@xyz.com"]	Success: Customer information updated in the database	Customer updated successfully, redirected to the customer overview page
Delete a customer from the database and all related information	Customer ID: 1	Success: Customer deleted from the database, along with all associated records (addresses, phone numbers, emails)	Customer deleted successfully, redirected to the customer list page

5.2.6 Feature 6: Customer Management

Functionality	Input	Expected Output	Actual Output
Add a new customer to the database	First Name: "John", Last Name: "Doe", Company: "ABC Corp", Address: ["123 Elm St"], Phone:	Success: Customer added with ID and all details stored in the database	Customer added successfully, redirected to the

	["555-1234"], Email: ["john.doe@example.com"]		customer overview page
Validate required fields for adding a customer	First Name: (empty), Last Name: "Doe", Company: "ABC Corp", Address: ["123 Elm St"], Phone: ["555-1234"], Email: ["john.doe@example.com"]	Error: "First name is required"	Error message shown: "First name is required", user redirected to the form page
Update existing customer information	Customer ID: 1, First Name: "John", Last Name: "Doe", Company: "XYZ Ltd", Address: ["456 Oak St"], Phone: ["555-6789"], Email: ["john.doe@xyz.com"]	Success: Customer information updated in the database	Customer updated successfully, redirected to the customer overview page
Delete a customer from the database and all related information	Customer ID: 1	Success: Customer deleted from the database, along with all associated records (addresses, phone numbers, emails)	Customer deleted successfully, redirected to the customer list page

5.2.7 Feature 7: Accounting Management

Functionality	Input	Expected Output	Actual Output
Add a new extra expense	Description: "Towing service" Expense: 75.00 DateCreated: *(not provided, defaults to today's date)*	Success: Expense added with ID and details stored in the database	Extra expense added successfully, success message stored in session
Missing description	Description: *(empty)* Expense: 50.00	Error: "Description is required. Please provide a valid description."	Error message shown

Missing expense amount	Description: "Parts cost" Expense: *(empty)*	Error: "Expense amount is required. Please provide a valid amount."	Error message shown

5.2.8 Feature 8: Invoice Management

Functionality	Input	Expected Output	Actual Output
Add a new supplier	Name: "Acme Corp", Email: "acme@mail.com"	Supplier added to database and shown in list	Supplier successfully added and visible
Edit supplier details	Update Email: "info@acme.com" for "Acme Corp"	Supplier info updated in database	Email updated and reflected in UI
Delete a supplier	Click delete on "Acme Corp"	Supplier removed from database	Deletion successful, supplier no longer listed
Validate fields on invoice upload	Invoice Number: (empty), File: PDF	Error: "Invoice number is required"	Error shown, form not submitted

5.2.9 Feature 9: Car Management

Functionality	Input	Expected Output	Actual Output
Adding a new car record	LicenseNr (empty) Brand (empty) Model (empty) VIN (empty)	Error: "License number is required." Error: "Brand is required." Error: "Model is required." Error: "VIN is required."	Error messages displayed for each missing field when attempting to submit the form with empty fields, and user is redirected to the form page.

6. APPENDICES

Below is the updated Gantt chart of the team.

Specifications Phase

Specifications	30 Jan 2025	05 Feb 2025	6 days	All
1. Document Preparation	30 Jan 2025	30 Jan 2025	1 day	Jorgos X.
2. Gane and Sarsen Methodology	31 Jan 2025	05 Feb 2025	4.8 days	All
2.1 Data Flow Diagrams	31 Jan 2025	31 Jan 2025	1 day	All
2.2 Data Flows	31 Jan 2025	31 Jan 2025	0.9 days	All
2.3 Logic Of Processes	31 Jan 2025	01 Feb 2025	0.9 days	All
2.4 Data Stores	31 Jan 2025	31 Jan 2025	0.9 days	All
2.5 Physical Resources	03 Feb 2025	03 Feb 2025	0.7 days	All
2.6 Input/Output Specifications	03 Feb 2025	03 Feb 2025	1 day	Jorgos X.
2.7 Sizing	03 Feb 2025	03 Feb 2025	0.27 days	Gabriel
2.8 Hardware Requirements	03 Feb 2025	03 Feb 2025	0.7 days	Kyriakos
2.9 Use-case Modelling	04 Feb 2025	04 Feb 2025	1 day	Giorgos A.,Kyriakos
2.10 Class Modelling	05 Feb 2025	05 Feb 2025	0.4 days	Antonis
2.11 State Diagrams	05 Feb 2025	05 Feb 2025	0.4 days	Gabriel
3. Entity-Relationship Diagram	04 Feb 2025	05 Feb 2025	1.2 days	Gabriel
4. Software Project Management Plan	05 Feb 2025	05 Feb 2025	1 day	Jorgos X.

Design Phase

Design	06 Feb 2025	13 Feb 2025	7 days	All
1. Document Preparation	06 Feb 2025	06 Feb 2025	0.5 days	Stylianos
2. Object Oriented System Design	06 Feb 2025	13 Feb 2025	7 days	All
2.1 Detailed Design	08 Feb 2025	11 Feb 2025	3 days	All
2.2 Detailed Class Diagram	08 Feb 2025	11 Feb 2025	3 days	All
2.3 Interaction Diagrams	06 Feb 2025	08 Feb 2025	3 days	All
2.4 Clients Of Objects	11 Feb 2025	13 Feb 2025	3 days	All

Implementation Phase

Implementation	17 Feb 2025	05 Apr 2025	42 days	All
1. Database Implementation	17 Feb 2025	20 Feb 2025	4 days	Gabriel,Giorgos A.
2. Implementation of Functions	21 Feb 2025	02 Apr 2025	35 days	All
2.1 Customer Management	21 Feb 2025	27 Feb 2025	6 days	Antonis,Gabriel,Giorgos A.,Jorgos X.,Kyri…
2.1.1 Add New Customer	21 Feb 2025	27 Feb 2025	6 days	Kyriakos,Gabriel
2.1.1.1 Input	21 Feb 2025	21 Feb 2025	1 day	
2.1.1.2 Processing	22 Feb 2025	25 Feb 2025	3 days	
2.1.1.3 Output	26 Feb 2025	26 Feb 2025	1 day	
2.1.1.4 Test	27 Feb 2025	27 Feb 2025	1 day	
2.2.2 Print Customer List	21 Feb 2025	26 Feb 2025	5 days	Giorgos A.,Kyriakos,Gabriel
2.2.2.1 Input	21 Feb 2025	21 Feb 2025	1 day	
2.2.2.2 Processing	22 Feb 2025	24 Feb 2025	2 days	
2.2.2.3 Output	25 Feb 2025	25 Feb 2025	1 day	
2.2.2.4 Test	26 Feb 2025	26 Feb 2025	1 day	
2.3.3 Edit Customer	21 Feb 2025	27 Feb 2025	6 days	Gabriel,Kyriakos
2.3.3.1 Input	21 Feb 2025	21 Feb 2025	1 day	
2.3.3.2 Processing	22 Feb 2025	25 Feb 2025	3 days	
2.3.3.3 Output	26 Feb 2025	26 Feb 2025	1 day	
2.3.3.4 Test	27 Feb 2025	27 Feb 2025	1 day	
2.4.4 Delete Customer	21 Feb 2025	26 Feb 2025	5 days	Antonis,Kyriakos,Gabriel
2.4.4.1 Input	21 Feb 2025	21 Feb 2025	1 day	
2.4.4.2 Processing	22 Feb 2025	24 Feb 2025	2 days	
2.4.4.3 Output	25 Feb 2025	25 Feb 2025	1 day	
2.4.4.4 Test	26 Feb 2025	26 Feb 2025	1 day	
2.5.5 Print Customer	21 Feb 2025	25 Feb 2025	4 days	Jorgos X.,Gabriel,Kyriakos
2.5.5.1 Input	21 Feb 2025	21 Feb 2025	1 day	
2.5.5.2 Processing	22 Feb 2025	22 Feb 2025	1 day	
2.5.5.3 Output	24 Feb 2025	24 Feb 2025	1 day	
2.5.5.4 Test	25 Feb 2025	25 Feb 2025	1 day	
2.2 User Management	22 Feb 2025	03 Mar 2025	8 days	Stylianos,Gabriel
2.2.1 Add New User	22 Feb 2025	26 Feb 2025	3.5 days	Stylianos
2.2.1.1 Input	22 Feb 2025	22 Feb 2025	0.5 days	
2.2.1.2 Processing	22 Feb 2025	24 Feb 2025	2 days	
2.2.1.3 Output	25 Feb 2025	25 Feb 2025	1 day	
2.2.1.4 Test	26 Feb 2025	26 Feb 2025	0.5 days	
2.2.2 Edit User	27 Feb 2025	01 Mar 2025	3 days	Stylianos
2.2.2.1 Input	27 Feb 2025	27 Feb 2025	0.5 days	
2.2.2.2 Processing	28 Feb 2025	28 Feb 2025	1 day	
2.2.2.3 Output	01 Mar 2025	01 Mar 2025	1 day	
2.2.2.4 Test	01 Mar 2025	01 Mar 2025	0.5 days	
2.2.3 Delete User	28 Feb 2025	03 Mar 2025	3 days	Gabriel
2.2.3.1 Input	28 Feb 2025	28 Feb 2025	0.5 days	
2.2.3.2 Processing	01 Mar 2025	01 Mar 2025	1 day	
2.2.3.3 Output	03 Mar 2025	03 Mar 2025	1 day	

2.3 Parts Management	26 Feb 2025	07 Mar 2025	9 days	Antonis,Giorgos A.,Jorgos X.,Kyriakos
2.3.1 Add New Part	26 Feb 2025	03 Mar 2025	5 days	Jorgos X.
2.3.1.1 Input	26 Feb 2025	26 Feb 2025	0.5 days	
2.3.1.2 Processing	27 Feb 2025	01 Mar 2025	3 days	
2.3.1.3 Output	03 Mar 2025	03 Mar 2025	1 day	
2.3.1.4 Test	03 Mar 2025	03 Mar 2025	0.5 days	
2.3.2 Print Part List	04 Mar 2025	07 Mar 2025	4 days	Jorgos X.
2.3.2.1 Input	04 Mar 2025	04 Mar 2025	0.5 days	
2.3.2.2 Processing	05 Mar 2025	06 Mar 2025	2 days	
2.3.2.3 Output	07 Mar 2025	07 Mar 2025	1 day	
2.3.2.4 Test	07 Mar 2025	07 Mar 2025	0.5 days	
2.3.3 Edit Part	27 Feb 2025	03 Mar 2025	4 days	Giorgos A.,Jorgos X.
2.3.3.1 Input	27 Feb 2025	27 Feb 2025	0.5 days	
2.3.3.2 Processing	28 Feb 2025	01 Mar 2025	2 days	
2.3.3.3 Output	03 Mar 2025	03 Mar 2025	1 day	
2.3.3.4 Test	03 Mar 2025	03 Mar 2025	0.5 days	
2.3.4 Delete Part	27 Feb 2025	03 Mar 2025	4 days	Antonis,Jorgos X.
2.3.4.1 Input	27 Feb 2025	27 Feb 2025	0.5 days	
2.3.4.2 Processing	28 Feb 2025	01 Mar 2025	2 days	
2.3.4.3 Output	03 Mar 2025	03 Mar 2025	1 day	
2.3.4.4 Test	03 Mar 2025	03 Mar 2025	0.5 days	
2.3.5 Print Part	28 Feb 2025	03 Mar 2025	3 days	Kyriakos,Jorgos X.
2.3.5.1 Input	28 Feb 2025	28 Feb 2025	0.5 days	
2.3.5.2 Processing	01 Mar 2025	01 Mar 2025	1 day	
2.3.5.3 Output	03 Mar 2025	03 Mar 2025	1 day	
2.3.5.4 Test	03 Mar 2025	03 Mar 2025	0.5 days	
2.4 Cars Management	10 Mar 2025	29 Mar 2025	18 days	Stylianos
2.4.1 Add Car	10 Mar 2025	14 Mar 2025	5 days	
2.4.1.1 Input	10 Mar 2025	10 Mar 2025	0.5 days	
2.4.1.2 Processing	11 Mar 2025	13 Mar 2025	3 days	
2.4.1.3 Output	14 Mar 2025	14 Mar 2025	1 day	
2.4.1.4 Test	14 Mar 2025	14 Mar 2025	0.5 days	
2.4.2 Edit Car	15 Mar 2025	20 Mar 2025	5 days	
2.4.2.1 Input	15 Mar 2025	15 Mar 2025	0.5 days	
2.4.2.2 Processing	17 Mar 2025	19 Mar 2025	3 days	
2.4.2.3 Output	20 Mar 2025	20 Mar 2025	1 day	
2.4.2.4 Test	20 Mar 2025	20 Mar 2025	0.5 days	
2.4.3 Delete Car	21 Mar 2025	25 Mar 2025	4 days	
2.4.3.1 Input	21 Mar 2025	21 Mar 2025	0.5 days	
2.4.3.2 Processing	22 Mar 2025	24 Mar 2025	2 days	
2.4.3.3 Output	25 Mar 2025	25 Mar 2025	1 day	
2.4.3.4 Test	25 Mar 2025	25 Mar 2025	0.5 days	
2.4.4 Print Car	26 Mar 2025	29 Mar 2025	4 days	
2.4.4.1 Input	26 Mar 2025	26 Mar 2025	0.5 days	

2.4.4.2 Processing	27 Mar 2025	28 Mar 2025	2 days	
2.4.4.3 Output	29 Mar 2025	29 Mar 2025	1 day	
2.4.4.4 Test	29 Mar 2025	29 Mar 2025	0.5 days	
2.5 User Access	10 Mar 2025	01 Apr 2025	20 days	Kyriakos
2.5.1 Login	10 Mar 2025	15 Mar 2025	6 days	
2.5.1.1 Input	10 Mar 2025	11 Mar 2025	2 days	
2.5.1.2 Processing	12 Mar 2025	14 Mar 2025	3 days	
2.5.1.3 Output	14 Mar 2025	15 Mar 2025	2 days	
2.5.1.4 Test	15 Mar 2025	15 Mar 2025	1 day	
2.5.2 Forgot Password	17 Mar 2025	24 Mar 2025	7 days	
2.5.2.1 Input	17 Mar 2025	18 Mar 2025	2 days	
2.5.2.2 Processing	18 Mar 2025	22 Mar 2025	5 days	
2.5.2.3 Output	22 Mar 2025	24 Mar 2025	2 days	
2.5.2.4 Test	24 Mar 2025	24 Mar 2025	1 day	
2.5.3 Logout	25 Mar 2025	01 Apr 2025	7 days	
2.5.3.1 Input	25 Mar 2025	26 Mar 2025	2 days	
2.5.3.2 Processing	26 Mar 2025	28 Mar 2025	3 days	
2.5.3.3 Output	29 Mar 2025	31 Mar 2025	2 days	
2.5.3.4 Test	31 Mar 2025	01 Apr 2025	2 days	
2.6 Invoice Management	10 Mar 2025	02 Apr 2025	20.5 days	Jorgos X.
2.6.1 Add New Invoice	10 Mar 2025	17 Mar 2025	7 days	
2.6.1.1 Input	10 Mar 2025	11 Mar 2025	2 days	
2.6.1.2 Processing	12 Mar 2025	15 Mar 2025	4 days	
2.6.1.3 Output	17 Mar 2025	17 Mar 2025	1 day	
2.6.1.4 Test	17 Mar 2025	17 Mar 2025	1 day	
2.6.2 Print Invoice List	18 Mar 2025	21 Mar 2025	4 days	
2.6.2.1 Input	18 Mar 2025	18 Mar 2025	0.5 days	
2.6.2.2 Processing	19 Mar 2025	20 Mar 2025	2 days	
2.6.2.3 Output	21 Mar 2025	21 Mar 2025	1 day	
2.6.2.4 Test	21 Mar 2025	21 Mar 2025	0.5 days	
2.6.3 Edit Invoice	22 Mar 2025	25 Mar 2025	3 days	
2.6.3.1 Input	22 Mar 2025	22 Mar 2025	1 day	
2.6.3.2 Processing	24 Mar 2025	25 Mar 2025	2 days	
2.6.3.3 Output	25 Mar 2025	25 Mar 2025	1 day	
2.6.3.4 Test	25 Mar 2025	25 Mar 2025	1 day	
2.6.4 Delete Invoice	26 Mar 2025	28 Mar 2025	3 days	
2.6.4.1 Input	26 Mar 2025	26 Mar 2025	0.5 days	
2.6.4.2 Processing	27 Mar 2025	28 Mar 2025	2 days	
2.6.4.3 Output	28 Mar 2025	28 Mar 2025	1 day	
2.6.4.4 Test	28 Mar 2025	28 Mar 2025	0.5 days	
2.6.5 Print Invoice	29 Mar 2025	02 Apr 2025	3.5 days	
2.6.5.1 Input	29 Mar 2025	29 Mar 2025	0.5 days	
2.6.5.2 Processing	31 Mar 2025	01 Apr 2025	2 days	
2.6.5.3 Output	01 Apr 2025	01 Apr 2025	1 day	

2.6.5.4 Test	02 Apr 2025	02 Apr 2025	0.5 days	
2.7 Jobs Management	10 Mar 2025	02 Apr 2025	21 days	Antonis
2.7.1 Add New Job Card	10 Mar 2025	17 Mar 2025	7 days	
2.7.1.1 Input	10 Mar 2025	10 Mar 2025	1 day	
2.7.1.2 Processing	11 Mar 2025	14 Mar 2025	4 days	
2.7.1.3 Output	15 Mar 2025	15 Mar 2025	1 day	
2.7.1.4 Test	17 Mar 2025	17 Mar 2025	1 day	
2.7.2 Print Job Card List	18 Mar 2025	20 Mar 2025	3 days	
2.7.2.1 Input	18 Mar 2025	18 Mar 2025	0.5 days	
2.7.2.2 Processing	19 Mar 2025	20 Mar 2025	2 days	
2.7.2.3 Output	20 Mar 2025	20 Mar 2025	1 day	
2.7.2.4 Test	20 Mar 2025	20 Mar 2025	0.5 days	
2.7.3 Edit Job Card	21 Mar 2025	24 Mar 2025	3 days	
2.7.3.1 Input	21 Mar 2025	21 Mar 2025	0.5 days	
2.7.3.2 Processing	22 Mar 2025	24 Mar 2025	2 days	
2.7.3.3 Output	24 Mar 2025	24 Mar 2025	1 day	
2.7.3.4 Test	24 Mar 2025	24 Mar 2025	0.5 days	
2.7.4 Delete Job Card	25 Mar 2025	28 Mar 2025	4 days	
2.7.4.1 Input	25 Mar 2025	25 Mar 2025	0.5 days	
2.7.4.2 Processing	26 Mar 2025	27 Mar 2025	2 days	
2.7.4.3 Output	28 Mar 2025	28 Mar 2025	1 day	
2.7.4.4 Test	28 Mar 2025	28 Mar 2025	0.5 days	
2.7.5 Print Job Card	29 Mar 2025	02 Apr 2025	4 days	
2.7.5.1 Input	29 Mar 2025	29 Mar 2025	0.5 days	
2.7.5.2 Processing	01 Apr 2025	02 Apr 2025	2 days	
2.7.5.3 Output	02 Apr 2025	02 Apr 2025	1 day	
2.7.5.4 Test	02 Apr 2025	02 Apr 2025	0.5 days	
2.8 Acounting Management	10 Mar 2025	02 Apr 2025	21 days	Giorgos A.,Gabriel
2.8.1 View Job Cards - Details	10 Mar 2025	17 Mar 2025	7 days	Giorgos A.
2.8.1.1 Input	10 Mar 2025	10 Mar 2025	1 day	
2.8.1.2 Processing	11 Mar 2025	14 Mar 2025	4 days	
2.8.1.3 Output	15 Mar 2025	15 Mar 2025	1 day	
2.8.1.4 Test	17 Mar 2025	17 Mar 2025	1 day	
2.8.2 View Parts - Details	10 Mar 2025	15 Mar 2025	6 days	Gabriel
2.8.2.1 Input	10 Mar 2025	10 Mar 2025	1 day	
2.8.2.2 Processing	11 Mar 2025	13 Mar 2025	3 days	
2.8.2.3 Output	13 Mar 2025	14 Mar 2025	2 days	
2.8.2.4 Test	15 Mar 2025	15 Mar 2025	1 day	
2.8.3 View Invoices	17 Mar 2025	21 Mar 2025	4.5 days	Gabriel
2.8.3.1 Input	17 Mar 2025	17 Mar 2025	0.5 days	
2.8.3.2 Processing	18 Mar 2025	20 Mar 2025	3 days	
2.8.3.3 Output	20 Mar 2025	20 Mar 2025	1 day	
2.8.3.4 Test	21 Mar 2025	21 Mar 2025	0.5 days	
2.8.4 Create Report for Finances	21 Mar 2025	25 Mar 2025	4 days	Gabriel

2.8.4.1 Input	21 Mar 2025	21 Mar 2025	1 day	
2.8.4.2 Processing	22 Mar 2025	24 Mar 2025	2 days	
2.8.4.3 Output	25 Mar 2025	25 Mar 2025	1 day	
2.8.4.4 Test	25 Mar 2025	25 Mar 2025	1 day	
▾ 2.8.5 Create Report for List of Job Cards - Monthly/Yearly	26 Mar 2025	28 Mar 2025	3 days	Gabriel
2.8.5.1 Input	26 Mar 2025	26 Mar 2025	1 day	
2.8.5.2 Processing	27 Mar 2025	28 Mar 2025	2 days	
2.8.5.3 Output	28 Mar 2025	28 Mar 2025	1 day	
2.8.5.4 Test	28 Mar 2025	28 Mar 2025	1 day	
▾ 2.8.6 Create for Single Chosen Job Card	29 Mar 2025	02 Apr 2025	4 days	Gabriel
2.8.6.1 Input	29 Mar 2025	29 Mar 2025	1 day	
2.8.6.2 Processing	31 Mar 2025	01 Apr 2025	2 days	
2.8.6.3 Output	02 Apr 2025	02 Apr 2025	1 day	
2.8.6.4 Test	02 Apr 2025	02 Apr 2025	1 day	
▾ 2.8.7 Create for List of Parts - Monthly/Yearly	17 Mar 2025	21 Mar 2025	5 days	Giorgos A.
2.8.7.1 Input	17 Mar 2025	17 Mar 2025	1 day	
2.8.7.2 Processing	18 Mar 2025	19 Mar 2025	2 days	
2.8.7.3 Output	20 Mar 2025	20 Mar 2025	1 day	
2.8.7.4 Test	21 Mar 2025	21 Mar 2025	1 day	
▾ 2.8.8 Create Report for Single Chosen Part	22 Mar 2025	25 Mar 2025	3 days	Giorgos A.
2.8.8.1 Input	22 Mar 2025	22 Mar 2025	1 day	
2.8.8.2 Processing	24 Mar 2025	24 Mar 2025	1 day	
2.8.8.3 Output	25 Mar 2025	25 Mar 2025	1 day	
2.8.8.4 Test	25 Mar 2025	25 Mar 2025	1 day	
▾ 2.8.9 Create Report by Month	26 Mar 2025	28 Mar 2025	3 days	Giorgos A.
2.8.9.1 Input	26 Mar 2025	26 Mar 2025	1 day	
2.8.9.2 Processing	27 Mar 2025	27 Mar 2025	1 day	
2.8.9.3 Output	28 Mar 2025	28 Mar 2025	1 day	
2.8.9.4 Test	28 Mar 2025	28 Mar 2025	1 day	
▾ 2.8.10 Create Report by Year	29 Mar 2025	01 Apr 2025	3 days	Giorgos A.
2.8.10.1 Input	29 Mar 2025	29 Mar 2025	1 day	
2.8.10.2 Processing	31 Mar 2025	31 Mar 2025	1 day	
2.8.10.3 Output	01 Apr 2025	01 Apr 2025	1 day	
▾ 3. Search	21 Feb 2025	05 Apr 2025	38 days	Antonis,Stylianos
3.1 Input	21 Feb 2025	21 Feb 2025	1 day	Stylianos
3.2 Processing	31 Mar 2025	03 Apr 2025	4 days	Antonis
3.3 Output	03 Apr 2025	04 Apr 2025	2 days	Antonis
3.4 Test	04 Apr 2025	05 Apr 2025	2 days	Antonis
▾ 4. Dashboard	29 Mar 2025	05 Apr 2025	7 days	Stylianos
4.1 Input	29 Mar 2025	29 Mar 2025	1 day	
4.2 Processing	01 Apr 2025	03 Apr 2025	3 days	
4.3 Output	04 Apr 2025	04 Apr 2025	1 day	
4.4 Test	04 Apr 2025	05 Apr 2025	2 days	
5. Integration and Test	07 Apr 2025	12 Apr 2025	6 days	All
6. Installation and Training	14 Apr 2025	19 Apr 2025	6 days	All
7. Creation of Admin Manual	08 Apr 2025	12 Apr 2025	5 days	All
8. Helpfile	03 Apr 2025	04 Apr 2025	2 days	Kyriakos