A red text on a white background

Description automatically generated

**School of Informatics & IT**

**Application Development Project (40%)**

**Database Application Development (30%)**

**(System Code Implementation)**

**AY 24/25 Oct Semester**

**Project 2 Report**

Contents

[**Declaration of Originality** 3](#_Toc188610378)

[ADEV Part 2: System Code Implementation (40%) 6](#_Toc188610379)

[Frontend Implementation (25%) 6](#_Toc188610380)

[User Interface (10%) 8](#_Toc188610381)

[Additional Features (5%) 14](#_Toc188610382)

[DBAV Part 2: System Code Implementation (30%) 16](#_Toc188610383)

[API Implementation (16%) 16](#_Toc188610384)

[Additional Features (4%) 20](#_Toc188610385)

[Error Handling (10%) 21](#_Toc188610386)

A red text on a white background

Description automatically generated

**SCHOOL OF INFORMATICS & IT**

**Diploma in Cybersecurity & Digital Forensics**

**Project 2 Report**

Student Name (Matric Number) : 2402075B

Tutorial Group : TC07

Jenny LING (TP)

Tutor : Venkat Ramanan Renganathan

Submission Date : 24/1/2025

**Declaration of Originality**

I am the originator of this work and I have appropriately acknowledged all other original sources used as my references for this work.

I understand that Plagiarism is the act of taking and using the whole or any part of another person’s work, including work generated by AI, and presenting it as my own.

I understand that Plagiarism is an academic offence

and if I am found to have committed or abetted the offence of plagiarism in relation to this submitted work, disciplinary action will be enforced.

**Declaration on the use of Generative AI tools for assignments**

|  |
| --- |
| Describe how you have used Generative AI tools such as ChatGPT or Dall.E- in your assignment.  Share the link to the conversations you had with the AI tool (i.e., the prompts you used and the responses you get from the AI tool).  **Please refer to this PDF on “How to share the conversations made with ChatGPT?”** |
|  |
| How do you indicate the reference?  The content generated by AI tools are not retrievable except by the user who generated them, so they are considered non-recoverable sources. Although non-recoverable data or quotations in APA Style papers are usually cited as personal communications, with ChatGPT-generated text there is no person communicating. Quoting text from ChatGPT chat is therefore more like sharing the output of an algorithm, with a reference list entry and the corresponding in-text citation.  According to the official APA Style site, ChatGPT references should be cited as:  E.g. OpenAI. (2024). *ChatGPT* (Feb 13 version) [Large language model].  <https://chat.openai.com/chat>  Note: The information in parentheses refers to the update or revision date of the model used. Refer to the release notes in the ChatGPT application. |

**Important Note:**

* Do not copy answers produced by the AI tool in totality as it is considered as plagiarism.
* Do not rely on any information produced by the AI tool blindly. You should always verify the answer with other sources. Do not assume that these answers provided by the AI tool are correct.
* To achieve quality outputs from the AI tool, you should provide good prompt that is clear and specific. Be precise and provide context. Avoid asking open-ended questions.

# ADEV Part 2: System Code Implementation (40%)

## Frontend Implementation (25%)

List down all the URL to the different pages of the website.   
Provide the sample input values if any.

|  |  |  |
| --- | --- | --- |
| **URL** | **Purpose** | **Inputs** |
| <http://127.0.0.1:8080/display_vending_machines.html> | Main page displaying information of each vending machines such as floor, block and payment methods. |  |
| <http://127.0.0.1:8080/display_vending_machine1_items.html> | Item page for vending machine 1, displaying all the items and navigation to add item page and item restock page. |  |
| <http://127.0.0.1:8080/display_vending_machine2_items.html> | Item page for vending machine 2, displaying all the items and navigation to add item page and item restock page. |  |
| <http://127.0.0.1:8080/display_item_restock.html> | Item restock page for vending machine 1, displaying the list of items that has 0 quantity. |  |
| <http://127.0.0.1:8080/display_item_restock2.html> | Item restock page for vending machine 2, displaying the list of items that has 0 quantity. |  |
| <http://127.0.0.1:8080/add_vending_machine1_items.html> | Add item page for vending machine 1, displays a form that asks for Item Name, Item Cost, Item Image(URL), Item Availability and Quantity to fill in for adding an item. | Item Name: Pokka Ice Lemon Tea  Item Cost: 1.50  Item Image(URL): /images/pokkaicelemontea.png  Item Availability: 1  Quantity: 5 |
| <http://127.0.0.1:8080/add_vending_machine2_items.html> | Add item page for vending machine 2, displays a form that asks for Item Name, Item Cost, Item Image(URL), Item Availability and Quantity to fill in for adding an item. | Item Name: Pokka Ice Blueberry Tea  Item Cost: 1.50  Item Image(URL):/images/pokkaiceblueryberrytea.png  Item Availability: 1  Quantity: 10 |

|  |  |  |
| --- | --- | --- |
| <http://127.0.0.1:8080/update_vending_machine1_items.html> | Update item page for vending machine 1, displays the detail of an item such as Item ID, Item Name, Item Cost, Item Image(URL), Item Availability and Quantity. The details can be changed after clicking the update button if there are any changes in the details. | ID:1  Item Name: Pokka Ice Peach Tea  Item Image(URL): /images/pokkaicepeachtea.png  Item Availability: Available(1)  Quantity: 7 |
| <http://127.0.0.1:8080/update_vending_machine2_items.html> | Update item page for vending machine 2, displays the detail of an item such as Item ID, Item Name, Item Cost, Item Image(URL), Item Availability and Quantity. The details can be changed after clicking the update button if there are any changes in the details. | ID:17  Item Name: Ribena Sparkling Juice Can Drink  Item Image(URL): /images/ribenacan.png  Item Availability: Available(1)  Quantity: 10 |

## User Interface (10%)

Provide screenshot(s) of the user interface for each of the core features.

Add more tables if required.

|  |
| --- |
| Feature: Main Page |
| Screenshot: |
| Feature: Vending Machine 1 Item Page |
| Screenshot: |

|  |
| --- |
| Feature: Vending Machine 2 Item Page |
| Screenshot: |

|  |
| --- |
|  |

|  |
| --- |
| Feature: Vending Machine 1 Add Item Page |
| Screenshot: |

|  |
| --- |
| Feature: Vending Machine 2 Add Item Page |
| Screenshot: |

|  |
| --- |
| Feature: Vending Machine 1 Update Item Page |
| Screenshot: |

|  |
| --- |
| Feature: Vending Machine 2 Update Item Page |
| Screenshot: |

## Additional Features (5%)

Provide screenshot(s) of the additional features.  
Explain how each additional feature enhances the website. Write a report on no more than **150 words** PER mechanism:

|  |
| --- |
| Additional Feature: Item Restock Page for Vending Machine 1 and Vending Machine 2 |
| Screenshots: |
| Explanation:  The item restock page is a feature designed to automatically identify and display items that requires replenishment where the quantity is zero. It lists down items with zero quantity and provides comprehensive item details such as Item ID, Item Image, Item Name, Item Cost, Item Quantity, and Restock Quantity. This eliminates the need for the admin to manually scroll through all the items to identify which items needs to be restock, saving time and effort for the admin. Additionally, the page helps to facilitate accurate communication with vendors by providing all necessary item details, making it easy to place orders or provide vendors with the required information. This ensures a more efficient and error-free restocking process. |

# DBAV Part 2: System Code Implementation (30%)

## API Implementation (16%)

Write out the related SQL statement, route pattern, HTTP method and payload (if any) for every functionality that your web application has. You will demonstrate to your Lecturer if the functionalities work using Postman based on the routes you created in the route file, and the functions that you wrote in the controller and DB files.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Functionalities of Web Application** | **Related SQL code**  **(Full and complete SQL Statements, not partial)** | **URL Route Pattern** | **Payload** | **HTTP Method** | **Result From Postman**  **(Pass/Fail)**  (To be filled by Lecturer) |
| Display all vending machines | SELECT vending\_machine.vending\_machine\_id,vending\_machine.location.block,vending\_machine.location.floor,  GROUP\_CONCAT(vending\_machine.payment\_method.payment\_name SEPARATOR '/') AS payment\_methods  FROM vending\_machine.vending\_machine  JOIN vending\_machine.location ON vending\_machine.location\_id = vending\_machine.location.location\_id  JOIN vending\_machine.vending\_payment ON vending\_machine.vending\_machine\_id = vending\_payment.vending\_id  JOIN vending\_machine.payment\_method ON vending\_payment.payment\_id = vending\_machine.payment\_method.payment\_id  GROUP BY vending\_machine.vending\_machine\_id, vending\_machine.location.block, vending\_machine.location.floor; | /vending\_machines |  | GET |  |
| Display all items in vending machine 1 | SELECT vending\_machine.vending\_item.vending\_machine\_id,item.item\_id,item.item\_name,item.item\_cost,item.item\_image,item.availability,item.item\_quantity FROM vending\_machine.vending\_item  JOIN vending\_machine.item ON vending\_machine.item.item\_id = vending\_machine.vending\_item.item\_id WHERE VENDING\_MACHINE\_ID=1; | /vending\_machine1\_items |  | GET |  |
| Retrieve vending machine 1 item using id | SELECT vending\_machine.vending\_item.vending\_machine\_id, vending\_machine.item.item\_id, vending\_machine.item.item\_name, vending\_machine.item.item\_cost,vending\_machine.item.item\_image,vending\_machine.item.availability, vending\_machine.item.item\_quantity FROM vending\_machine.vending\_item  JOIN vending\_machine.item ON vending\_machine.item.item\_id = vending\_machine.vending\_item.item\_id WHERE vending\_machine.vending\_item.vending\_machine\_id=1 AND vending\_machine.item.item\_id = ?; | /vending\_machine\_1\_items/:id  Example:  /vending/machine\_1\_items/1 |  | GET |  |
| Add an item in vending machine 1 | INSERT INTO VENDING\_MACHINE.ITEM (item\_name, item\_cost, item\_image, availability, item\_quantity) VALUES (?, ?, ?, ?, ?);  INSERT INTO VENDING\_MACHINE.VENDING\_ITEM(vending\_machine\_id, item\_id) VALUES (1, ?); | /add/vending\_machine1\_item | { "item\_name": "Pokka Ice Peach Tea",  "item\_cost": 1.50,  "item\_image":"/images/pokkaicepeachtea.png",  "availability": 1 ,  "item\_quantity":5  } | POST |  |
| Update an item in vending machine 1 | UPDATE VENDING\_MACHINE.ITEM SET item\_name = ?, item\_cost = ?, item\_image = ?, availability = ?, item\_quantity = ? WHERE item\_id = ?; | /update/vending\_machine1\_item/:id  Example:  /update/vending\_machine1\_item/2 | { "item\_name": "Pokka Ice Peach Tea",  "item\_cost": 1.50,  "item\_image":"/images/pokkaicepeachtea.png",  "availability": 1 ,  "item\_quantity":10  } | PUT |  |
| Delete an item in vending machine 1 | DELETE FROM VENDING\_MACHINE.ITEM WHERE item\_id = ?; | /delete/vending\_machine1\_item/:id  Example:  /delete/vending\_machine1\_item/3 |  | DELETE |  |
| Display all items in vending machine 2 | SELECT vending\_machine.vending\_item.vending\_machine\_id,item.item\_id,item.item\_name,item.item\_cost,item.item\_image,item.availability,item.item\_quantity FROM vending\_machine.vending\_item JOIN vending\_machine.item ON vending\_machine.item.item\_id = vending\_machine.vending\_item.item\_id  WHERE VENDING\_MACHINE\_ID=2; | /vending\_machine2\_items |  | GET |  |
| Retrieve vending machine 2 item using id | SELECT vending\_machine.vending\_item.vending\_machine\_id,item.item\_id,item.item\_name,item.item\_cost,  item.item\_image,item.availability,item.item\_quantity FROM vending\_machine.vending\_item  JOIN vending\_machine.item ON vending\_machine.item.item\_id = vending\_machine.vending\_item.item\_id  WHERE vending\_machine.vending\_item.vending\_machine\_id= 2 AND vending\_machine.item.item\_id = ?; | /vending\_machine2\_items/:id  Example:  /vending\_machine2\_items/17 |  | GET |  |
| Add an item in vending machine 2 | INSERT INTO VENDING\_MACHINE.ITEM (item\_name, item\_cost, item\_image, availability, item\_quantity) VALUES (?, ?, ?, ?, ?);  INSERT INTO VENDING\_MACHINE.VENDING\_ITEM(vending\_machine\_id, item\_id) VALUES (2, ?); | /add/vending\_machine2\_item | {  "item\_name": "Suntory Boss Coffee Black No Sugar",  "item\_cost": "3.00",  "item\_image": "/images/suntorycoffee.png",  "availability": 1,  "item\_quantity": 9  } | POST |  |
| Update an item in vending machine 2 | UPDATE VENDING\_MACHINE.ITEM SET item\_name = ?, item\_cost = ?, item\_image = ?, availability = ?, item\_quantity = ? WHERE item\_id = ?; | /update/vending\_machine2\_item/:id  Example:  /update/vending\_machine2\_item/18 | {  "item\_name": "Suntory Boss Coffee Black No Sugar",  "item\_cost": "3.00",  "item\_image": "/images/suntorycoffee.png",  "availability": 1,  "item\_quantity": 19  } | PUT |  |
| Delete an item in vending machine 2 | DELETE FROM VENDING\_MACHINE.ITEM WHERE item\_id = ?; | /delete/vending\_machine2\_item/:id  Example:  /delete/vending\_machine2\_item/19 |  | DELETE |  |

## Additional Features (4%)

Write out the related SQL statement, route pattern, HTTP method and payload (if any) for the additional features in ADEV Part 2:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Functionalities of Web Application** | **Related SQL code**  **(Full and complete SQL Statements, not partial)** | **URL Route Pattern** | **Payload** | **HTTP Method** | **Result From Postman**  **(Pass/Fail)**  (To be filled by Lecturer) |
| Display item restock list of vending machine 1 | SELECT vending\_machine.item\_restock.item\_restock\_id,vending\_machine.vending\_item.vending\_machine\_id,vending\_machine.item.item\_id,vending\_machine.item.item\_name,vending\_machine.item.item\_cost,  vending\_machine.item.item\_image,vending\_machine.item.availability,  vending\_machine.item.item\_quantity,vending\_machine.item\_restock.restock\_quantity FROM vending\_machine.item\_restock  JOIN vending\_machine.item ON vending\_machine.item\_restock.item\_id = vending\_machine.item.item\_id  JOIN vending\_machine.vending\_item ON vending\_machine.item.item\_id =vending\_machine. vending\_item.item\_id WHERE vending\_machine.vending\_item.vending\_machine\_id = 1; | /item\_restock\_1 |  | GET |  |
| Display item restock list of vending machine 2 | SELECT vending\_machine.item\_restock.item\_restock\_id,vending\_machine.vending\_item.vending\_machine\_id,vending\_machine.item.item\_id,vending\_machine.item.item\_name,vending\_machine.item.item\_cost,  vending\_machine.item.item\_image,vending\_machine.item.availability,  vending\_machine.item.item\_quantity,vending\_machine.item\_restock.restock\_quantity FROM vending\_machine.item\_restock  JOIN vending\_machine.item ON vending\_machine.item\_restock.item\_id = vending\_machine.item.item\_id  JOIN vending\_machine.vending\_item ON vending\_machine.item.item\_id =vending\_machine. vending\_item.item\_id  WHERE vending\_machine.vending\_item.vending\_machine\_id = 2; | /item\_restock\_2 |  | GET |  |

## Error Handling (10%)

Explain how you implemented the 2 (TWO) error-handling mechanisms on your server-side system, and why they are important. Write a report on no more than **200 words** PER mechanism:

|  |
| --- |
| Error handling 1: Using throw error in database query |
| Implementation in code:  Explaination:  I implemented a db.query method with two parameters: sql (the query string) and a callback function(error, result). The callback function also has two parameters: error (for storing error details) and result (for storing successful query results). If an error occurs during the database query, the error parameter stores the details of the error. The throw error statement raises the error and terminates the current request’s execution. If there are no errors, the query result is stored in the result parameter and sent as a JSON response via res.json(result). |
| Importance:  It is important to have this error-handling mechanism because it prevents the application from continuing with corrupted or incomplete data due to unhandled query errors. If an error occurs during the query, continuing the execution of the code may result in additional complications. Throwing the error allows the server to detect and highlight the error immediately, making debugging more efficient and effective. This contributes to the overall reliability and maintainability of the application. |

|  |
| --- |
| Error handling 2: Transaction Management for adding item |
| Implementation in code:    Explanation:  db.beginTransaction starts the transaction. If an error occurs, the error message “Transaction initiation failed” is returned and the process is halted. The first query(sql1) inserts the item into item table using parameters in req.body. If the first query fails, the transaction is rolled back, and the error message “Failed to insert item into VENDING\_ITEM table” is returned. Upon success, the newly inserted item’s itemId is used in the second query(sql2) to insert it into vending item table. If the second query fails, the transaction is rolled back to avoid partial updates. If both queries succeed, the db.commit commits the transaction. If an error occurs during the commit process, the transaction is rolled back, and a response “Transaction commit failed” is sent. If everything succeeds, the server sends a success message “Item added successfully!”. |
| Importance:  It is important to have transaction management because it is used for operations involving multiple related database queries. It maintains database consistency, especially involving multiple related queries by preventing partial updates. For example, in my code, I inserted an item into two related tables, item and vending item. Also, rolling back in the event of errors guarantees that the database remains reliable and in a valid state. |