**Library User Story Elaboration**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Number** | **Priority** | | **Name** | |
| 1 | High | | Retrieve Admin login details in the admin table | |
|  | **Narrative** | | | |
| **As a/an... (Given)** | **I want to... (When)** | | **So that... (Then)** |
| User | Get admin details | | I can make use of it. |
| **Acceptance criteria** | | | |
| **Scenario 1 – Get all entry from admin table correctly** | | | |
| **Given** | **When** | | **Then** |
| * The admin table is created and set up correctly * The admin Resource.java is created and set up correctly * The admin DAO.java is created and set up correctly * The server is set up correctly * Using the GET method on Postman | * The user enters [http://localhost:8080/PremiumLibraryTeam/rest/ admin](http://localhost:8080/PremiumLibraryTeam/rest/books)  on postman * The user clicks the send button | | * All the entries in the admin table will be retrieved in JSON format |
| **Scenario 2 – Get all entry from admin table with the wrong path** | | | |
| **Given** | **When** | | **Then** |
| * The admin table is created and set up correctly * The admin Resource.java is created and set up correctly * The admin DAO.java is created and set up correctly * The server is set up correctly * Using the GET method on Postman | * The user enters http://localhost:8080/PremiumLibraryTeam/rest/ admins instead of admin on postman * The user clicks the send button | | * The user get an error message HTTP Status 404 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Number** | **Priority** | | **Name** | |
| 2 | High | | Retrieve All customers details in the customers table | |
|  | **Narrative** | | | |
| **As a/an... (Given)** | **I want to... (When)** | | **So that... (Then)** |
| User | Get customers details | | I can see all customers |
| **Acceptance criteria** | | | |
| **Scenario 1 – Get all entry from customers table correctly** | | | |
| **Given** | **When** | | **Then** |
| * The customers table is created and set up correctly * The customers Resource.java is created and set up correctly * The customers DAO.java is created and set up correctly * The server is set up correctly * Using the GET method on Postman | * The user enters [http://localhost:8080/PremiumLibraryTeam/rest/customers](http://localhost:8080/PremiumLibraryTeam/rest/customers%20)  on postman * The user clicks the send button | | * All the entries in the customers table will be retrieved in JSON format |
| **Scenario 2 – Get all entry from customers table with the wrong path** | | | |
| **Given** | **When** | | **Then** |
| * The customers table is created and set up correctly * The customers Resource.java is created and set up correctly * The customers DAO.java is created and set up correctly * The server is set up correctly * Using the GET method on Postman | * The user enters <http://localhost:8080/PremiumLibraryTeam/rest/> customer instead of customers on postman * The user clicks the send button | | * The user get an error message HTTP Status 404 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Number** | **Priority** | | **Name** | |
| 3 | High | | Retrieve customers data by email in the customers table | |
|  | **Narrative** | | | |
| **As a/an... (Given)** | **I want to... (When)** | | **So that... (Then)** |
| User | Retrieve customers data by email | | I can view the customers |
| **Acceptance criteria** | | | |
| **Scenario 1 – Get entry from customers table using the correct email** | | | |
| **Given** | **When** | | **Then** |
| * The customers table is created and set up correctly in the database * The customers Resource.java is created and set up correctly * The customers DAO.java is created and set up correctly * The server is set up correctly * Using the GET method on Postman | * The user enters Email<http://localhost:8080/PremiumLibraryTeam/rest/customers/>search/{Email} on postman * The user clicks the send button | | * The entry with the email in the customers table will be retrieved in JSON format. |
| **Scenario 2 – Get entry from customers table using the non existing email** | | | |
| **Given** | **When** | | **Then** |
| * The customers table is created and set up correctly in the database * The customers Resource.java is created and set up correctly * The customers DAO.java is created and set up correctly * The server is set up correctly * Using the GET method on Postman | * The user enters [http://localhost:8080/PremiumLibraryTeam/rest/ customers/search/{ non](http://localhost:8080/PremiumLibraryTeam/rest/%20customers/search/%7b%20non) existing email} on postman * The user clicks the send button | | * The user gets a blank page because the email does not exist. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Number** | **Priority** | | **Name** | |
| 4 | High | | Create New customer in the customers table | |
|  | **Narrative** | | | |
| **As a/an... (Given)** | **I want to... (When)** | | **So that... (Then)** |
| User | Add new customer | | I can see all customer in the customers table |
| **Acceptance criteria** | | | |
| **Scenario 1 – Create entry into the customer table correctly** | | | |
| **Given** | **When** | | **Then** |
| * The customers table is created and set up correctly * The customers Resource.java is created and set up correctly * The customers DAO.java is created and set up correctly * The server is set up correctly * Using the POST method on Postman | * The user enters <http://localhost:8080/PremiumLibraryTeam/rest/customers> on postman * The user enters the entry in JSON format * The user clicks the send button | | * The customers table entry will be created in the customers table. |
| **Scenario 2 – Create entry into the customer table incorrectly** | | | |
| **Given** | **When** | | **Then** |
| * The customers table is created and set up correctly * The customers Resource.java is created and set up correctly * The customers DAO.java is created and set up correctly * The server is set up correctly   Using the POST method on Postman | * The user enters http://localhost:8080/PremiumLibraryTeam/rest/customers on postman * The user enters the entry in JSON format with ID * The user clicks the send button | | * The user get an error message HTTP Status 500 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Number** | **Priority** | | **Name** | |
| 5 | High | | Update customer by ID in the customers table | |
|  | **Narrative** | | | |
| **As a/an... (Given)** | **I want to... (When)** | | **So that... (Then)** |
| User | Update a customer by ID | | I can make changes to the customer in the customers table |
| **Acceptance criteria** | | | |
| **Scenario 1 – Update entry by ID from customers table correctly** | | | |
| **Given** | **When** | | **Then** |
| * The customers Resource.java is created and set up correctly * The customers DAO.java is created and set up correctly * The server is set up correctly * Using the PUT method on Postman | * The user enters <http://localhost:8080/PremiumLibraryTeam/rest/customers>/{ID}on postman * The user enter the updated entry in JSON format with the entry ID * The user clicks the send button | | * The ID of the entry in the customers table will be updated. |
| **Scenario 2 – Update entry with wrong ID from customers table** | | | |
| **Given** | **When** | | **Then** |
| * The customers Resource.java is created and set up correctly * The customers DAO.java is created and set up correctly * The server is set up correctly * Using the PUT method on Postman | * The user enters <http://localhost:8080/PremiumLibraryTeam/rest/customers>/{wrongID}on postman * The user enter the updated entry in JSON format with the entry ID * The user clicks the send button | | * No changes will be made |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Number** | **Priority** | | **Name** | |
| 6 | High | | Delete customer by ID in the customers table | |
|  | **Narrative** | | | |
| **As a/an... (Given)** | **I want to... (When)** | | **So that... (Then)** |
| User | Delete a customers by ID | | The customer does not exist in the customers table |
| **Acceptance criteria** | | | |
| **Scenario 1 – Delete entry by ID from customers table correctly** | | | |
| **Given** | **When** | | **Then** |
| * The customers table is created and set up correctly * The customers Resource.java is created and set up correctly * The customers DAO.java is created and set up correctly * The server is set up correctly * Using the DELETE method on Postman | * The user enters <http://localhost:8080/PremiumLibraryTeam/rest/customers>/{ID} on postman * The user clicks the send button | | * The ID of the entry in the customers table will be deleted. |
| **Scenario 2 – Delete entry by non existing ID from customers table correctly** | | | |
| **Given** | **When** | | **Then** |
| * The customers table is created and set up correctly * The customers Resource.java is created and set up correctly * The customers DAO.java is created and set up correctly * The server is set up correctly * Using the DELETE method on Postman | * The user enters [http://localhost:8080/PremiumLibraryTeam/rest/customers/{non](http://localhost:8080/PremiumLibraryTeam/rest/customers/%7bnon) existing ID} on postman * The user clicks the send button | | * No customers was deleted because the ID doesn’t exist |
|  | **Scenario 3 – Delete entry by ID from customer table when server is down** | | | |
| **Given** | **When** | | **Then** |
| * The customers table is created and set up correctly * The customers Resource.java is created and set up correctly * The customers DAO.java is created and set up correctly * The server is down * Using the DELETE method on Postman | * The user enters <http://localhost:8080/PremiumLibraryTeam/rest/customers>/{ID} on postman   The user clicks the send button | | * The user gets a connection error. |
| **Number** | **Priority** | | **Name** | |
| 7 | High | | Retrieve All librarians details in the librarians table | |
|  | **Narrative** | | | |
| **As a/an... (Given)** | **I want to... (When)** | | **So that... (Then)** |
| User | Get librarians details | | I can see all librarians |
| **Acceptance criteria** | | | |
| **Scenario 1 – Get all entry from librarians table correctly** | | | |
| **Given** | **When** | | **Then** |
| * The librarians table is created and set up correctly * The librarians Resource.java is created and set up correctly * The librarians DAO.java is created and set up correctly * The server is set up correctly * Using the GET method on Postman | * The user enters [http://localhost:8080/PremiumLibraryTeam/rest/librarians](http://localhost:8080/PremiumLibraryTeam/rest/customers%20)  on postman * The user clicks the send button | | * All the entries in the librarians table will be retrieved in JSON format |
| **Scenario 2 – Get all entry from customers table with the wrong path** | | | |
| **Given** | **When** | | **Then** |
| * The librarians table is created and set up correctly * The librarians Resource.java is created and set up correctly * The librarians DAO.java is created and set up correctly * The server is set up correctly * Using the GET method on Postman | * The user enters http://localhost:8080/PremiumLibraryTeam/rest/librarian instead of librarians on postman * The user clicks the send button | | * The user get an error message HTTP Status 404 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Number** | **Priority** | | **Name** | |
| 8 | High | | Create New librarians in the librarians table | |
|  | **Narrative** | | | |
| **As a/an... (Given)** | **I want to... (When)** | | **So that... (Then)** |
| User | Add new librarians | | I can see all librarians in the librarians table |
| **Acceptance criteria** | | | |
| **Scenario 1 – Create entry into the librarians table correctly** | | | |
| **Given** | **When** | | **Then** |
| * The librarians table is created and set up correctly * The librarians Resource.java is created and set up correctly * The librarians DAO.java is created and set up correctly * The server is set up correctly * Using the POST method on Postman | * The user enters [http://localhost:8080/PremiumLibraryTeam/rest/librarians](http://localhost:8080/PremiumLibraryTeam/rest/librarians%20) on postman * The user enters the entry in JSON format * The user clicks the send button | | * The librarians table entry will be created in the customers table. |
| **Scenario 2 – Create entry into the librarians table incorrectly** | | | |
| **Given** | **When** | | **Then** |
| * The librarians table is created and set up correctly * The librarians Resource.java is created and set up correctly * The librarians DAO.java is created and set up correctly * The server is set up correctly   Using the POST method on Postman | * The user enters http://localhost:8080/PremiumLibraryTeam/rest/librarians on postman * The user enters the entry in JSON format with ID * The user clicks the send button | | * The user get an error message HTTP Status 500 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Number** | **Priority** | | **Name** | |
| 9 | High | | Update librarian by ID in the librarians table | |
|  | **Narrative** | | | |
| **As a/an... (Given)** | **I want to... (When)** | | **So that... (Then)** |
| User | Update a librarians by ID | | I can make changes to the librarian in the librarians table |
| **Acceptance criteria** | | | |
| **Scenario 1 – Update entry by ID from librarians table correctly** | | | |
| **Given** | **When** | | **Then** |
| * The librarians Resource.java is created and set up correctly * The librarians DAO.java is created and set up correctly * The server is set up correctly * Using the PUT method on Postman | * The user enters <http://localhost:8080/PremiumLibraryTeam/rest/librarians>/{ID}on postman * The user enter the updated entry in JSON format with the entry ID * The user clicks the send button | | * The ID of the entry in the librarians table will be updated. |
| **Scenario 2 – Update entry with wrong ID from librarians table** | | | |
| **Given** | **When** | | **Then** |
| * The librarians Resource.java is created and set up correctly * The librarians DAO.java is created and set up correctly * The server is set up correctly   Using the PUT method on Postman | * The user enters <http://localhost:8080/PremiumLibraryTeam/rest/librarians>/{wrongID}on postman * The user enter the updated entry in JSON format with the entry ID * The user clicks the send button | | * No changes will be made |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Number** | **Priority** | | **Name** | |
| 10 | High | | Delete librarian by ID in the librarians table | |
|  | **Narrative** | | | |
| **As a/an... (Given)** | **I want to... (When)** | | **So that... (Then)** |
| User | Delete a librarians by ID | | The librarian does not exist in the librarians table |
| **Acceptance criteria** | | | |
| **Scenario 1 – Delete entry by ID from librarian table correctly** | | | |
| **Given** | **When** | | **Then** |
| * The librarians table is created and set up correctly * The librarians Resource.java is created and set up correctly * The librarians DAO.java is created and set up correctly * The server is set up correctly * Using the DELETE method on Postman | * The user enters <http://localhost:8080/PremiumLibraryTeam/rest/librarians>/{ID} on postman * The user clicks the send button | | * The ID of the entry in the librarians table will be deleted. |
| **Scenario 2 – Delete entry by non existing ID from librarians table correctly** | | | |
| **Given** | **When** | | **Then** |
| * The librarians table is created and set up correctly * The librarians Resource.java is created and set up correctly * The librarians DAO.java is created and set up correctly * The server is set up correctly * Using the DELETE method on Postman | * The user enters [http://localhost:8080/PremiumLibraryTeam/rest/Librarians/{non](http://localhost:8080/PremiumLibraryTeam/rest/Librarians/%7bnon) existing ID} on postman * The user clicks the send button | | * No librarians was deleted because the ID doesn’t exist |
|  | **Scenario 3 – Delete entry by ID from librarians table when server is down** | | | |
| **Given** | **When** | | **Then** |
| * The librarians table is created and set up correctly * The librarians Resource.java is created and set up correctly * The librarians DAO.java is created and set up correctly * The server is down * Using the DELETE method on Postman | * The user enters <http://localhost:8080/PremiumLibraryTeam/rest/librarians>/{ID} on postman   The user clicks the send button | | * The user gets a connection error. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Number** | **Priority** | | **Name** | |
| 11 | High | | Create New books in the books table | |
|  | **Narrative** | | | |
| **As a/an... (Given)** | **I want to... (When)** | | **So that... (Then)** |
| User | Add new books | | I can see all books in the books table |
| **Acceptance criteria** | | | |
| **Scenario 1 – Create entry into the books table correctly** | | | |
| **Given** | **When** | | **Then** |
| * The database books table is created and set up correctly * The booksResource.java is created and set up correctly * The booksDAO.java is created and set up correctly * The server is set up correctly * Using the POST method on Postman | * The user enters <http://localhost:8080/PremiumLibraryTeam/rest/books> on postman * The user enters the entry in JSON format * The user clicks the send button | | * The books table entry will be created in the database table. |
| **Scenario 2 – Create entry into the books table incorrectly** | | | |
| **Given** | **When** | | **Then** |
| * The database books table is created and set up correctly * The booksResource.java is created and set up correctly * The booksDAO.java is created and set up correctly * The server is set up correctly * Using the POST method on Postman | * The user enters <http://localhost:8080/PremiumLibraryTeam/rest/books> on postman * The user enters the entry in JSON format with ID * The user clicks the send button | | * The user get an error message HTTP Status 500 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Number** | **Priority** | | **Name** | |
| 12 | High | | Get all books in the books table | |
|  | **Narrative** | | | |
| **As a/an... (Given)** | **I want to... (When)** | | **So that... (Then)** |
| User | Get all books | | I can view all books in the books table |
| **Acceptance criteria** | | | |
| **Scenario 1 – Get all entry from books table correctly** | | | |
| **Given** | **When** | | **Then** |
| * The books table is created and set up correctly in the database * The booksResource.java is created and set up correctly * The booksDAO.java is created and set up correctly * The server is set up correctly * Using the GET method on Postman | * The user enters <http://localhost:8080/PremiumLibraryTeam/rest/books> on postman * The user clicks the send button | | * All the entries in the books table will be retrieved in JSON format |
| **Scenario 2 – Get all entry from books table with the wrong path** | | | |
| **Given** | **When** | | **Then** |
| * The books table is created and set up correctly in the database * The booksResource.java is created and set up correctly * The booksDAO.java is created and set up correctly * The server is set up correctly * Using the GET method on Postman | * The user enters http://localhost:8080/PremiumLibraryTeam/rest/book instead of books on postman * The user clicks the send button | | * The user get an error message HTTP Status 404 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Number** | **Priority** | | **Name** | |
| 13 | High | | Get book by ID in the books table | |
|  | **Narrative** | | | |
| **As a/an... (Given)** | **I want to... (When)** | | **So that... (Then)** |
| User | Get new books by ID | | I can view the books in the books table |
| **Acceptance criteria** | | | |
| **Scenario 1 – Get entry from book table using the correct ID** | | | |
| **Given** | **When** | | **Then** |
| * The books table is created and set up correctly in the database * The booksResource.java is created and set up correctly * The booksDAO.java is created and set up correctly * The server is set up correctly * Using the GET method on Postman | * The user enters <http://localhost:8080/PremiumLibraryTeam/rest/books>/{ID} on postman * The user clicks the send button | | * The entry with the ID in the books table will be retrieved in JSON format. |
| **Scenario 2 – Get entry from book table using the non existing ID** | | | |
| **Given** | **When** | | **Then** |
| * The books table is created and set up correctly in the database * The booksResource.java is created and set up correctly * The booksDAO.java is created and set up correctly * The server is set up correctly * Using the GET method on Postman | * The user enters [http://localhost:8080/PremiumLibraryTeam/rest/books/{ non](http://localhost:8080/PremiumLibraryTeam/rest/books/%7b%20non) existing ID} on postman * The user clicks the send button | | * The user get a blank page because the ID does not exist. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Number** | **Priority** | | **Name** | |
| 14 | High | | Get book by books Name in the books table | |
|  | **Narrative** | | | |
| **As a/an... (Given)** | **I want to... (When)** | | **So that... (Then)** |
| User | Get new books by Name | | I can view the books in the books table |
| **Acceptance criteria** | | | |
| **Scenario 1 – Get entry from book table using the correct Book Name** | | | |
| **Given** | **When** | | **Then** |
| * The books table is created and set up correctly in the database * The booksResource.java is created and set up correctly * The booksDAO.java is created and set up correctly * The server is set up correctly * Using the GET method on Postman | * The user enters <http://localhost:8080/PremiumLibraryTeam/rest/books>/search/{bookName} on postman * The user clicks the send button | | * The entry with the bookName in the books table will be retrieved in JSON format. |
| **Scenario 2 – Get entry from book table using the non existing Book Name** | | | |
| **Given** | **When** | | **Then** |
| * The books table is created and set up correctly in the database * The booksResource.java is created and set up correctly * The booksDAO.java is created and set up correctly * The server is set up correctly * Using the GET method on Postman | * The user enters [http://localhost:8080/PremiumLibraryTeam/rest/books/search/{ non](http://localhost:8080/PremiumLibraryTeam/rest/books/search/%7b%20non) existing bookName} on postman * The user clicks the send button | | * The user gets a blank page because the bookName does not exist. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Number** | **Priority** | | **Name** | |
| 15 | High | | Get book by Authors Name in the books table | |
|  | **Narrative** | | | |
| **As a/an... (Given)** | **I want to... (When)** | | **So that... (Then)** |
| User | Get new books by Authors Name | | I can view the books in the books table |
| **Acceptance criteria** | | | |
| **Scenario 1 – Get entry from book table using the correct Authors Name** | | | |
| **Given** | **When** | | **Then** |
| * The books table is created and set up correctly in the database * The booksResource.java is created and set up correctly * The booksDAO.java is created and set up correctly * The server is set up correctly * Using the GET method on Postman | * The user enters <http://localhost:8080/PremiumLibraryTeam/rest/books>/search/{authorName} on postman * The user clicks the send button | | * The entry with the authorName in the books table will be retrieved in JSON format. |
| **Scenario 2 – Get entry from book table using the non existing Authors Name** | | | |
| **Given** | **When** | | **Then** |
| * The books table is created and set up correctly in the database * The booksResource.java is created and set up correctly * The booksDAO.java is created and set up correctly * The server is set up correctly * Using the GET method on Postman | * The user enters [http://localhost:8080/PremiumLibraryTeam/rest/books/search/{ non](http://localhost:8080/PremiumLibraryTeam/rest/books/search/%7b%20non) existing authorName} on postman * The user clicks the send button | | * The user gets a blank page because the authorName does not exist. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Number** | **Priority** | | **Name** | |
| 16 | High | | Update book by ID in the books table | |
|  | **Narrative** | | | |
| **As a/an... (Given)** | **I want to... (When)** | | **So that... (Then)** |
| User | Update a book by ID | | I can make changes to the book in the books table |
| **Acceptance criteria** | | | |
| **Scenario 1 – Update entry by ID from books table correctly** | | | |
| **Given** | **When** | | **Then** |
| * The booksResource.java is created and set up correctly * The booksDAO.java is created and set up correctly * The server is set up correctly * Using the PUT method on Postman | * The user enters <http://localhost:8080/PremiumLibraryTeam/rest/books> /{ID}on postman * The user enter the updateed entry in JSON format with the entry ID * The user clicks the send button | | * The ID of the entry in the books table will be updated. |
| **Scenario 2 – Update entry with wrong ID from books table** | | | |
| **Given** | **When** | | **Then** |
| * The booksResource.java is created and set up correctly * The booksDAO.java is created and set up correctly * The server is set up correctly   Using the PUT method on Postman | * The user enters <http://localhost:8080/PremiumLibraryTeam/rest/books> /{wrongID}on postman * The user enter the updateed entry in JSON format with the entry ID * The user clicks the send button | | * No changes will be made |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Number** | **Priority** | | **Name** | |
| 17 | High | | Delete book by ID in the books table | |
|  | **Narrative** | | | |
| **As a/an... (Given)** | **I want to... (When)** | | **So that... (Then)** |
| User | Delete a book by ID | | The book does not exist in the book table |
| **Acceptance criteria** | | | |
| **Scenario 1 – Delete entry by ID from book table correctly** | | | |
| **Given** | **When** | | **Then** |
| * The database books table is created and set up correctly * The booksResource.java is created and set up correctly * The booksDAO.java is created and set up correctly * The server is set up correctly * Using the DELETE method on Postman | * The user enters <http://localhost:8080/PremiumLibraryTeam/rest/books>/{ID} on postman * The user clicks the send button | | * The ID of the entry in the books table will be deleted. |
| **Scenario 2 – Delete entry by non existing ID from book table correctly** | | | |
| **Given** | **When** | | **Then** |
| * The database books table is created and set up correctly * The booksResource.java is created and set up correctly * The booksDAO.java is created and set up correctly * The server is set up correctly * Using the DELETE method on Postman | * The user enters <http://localhost:8080/PremiumLibraryTeam/rest/books>/{ID} on postman * The user clicks the send button | | * No book was deleted because the ID doesn’t exist |
|  | **Scenario 3 – Delete entry by ID from book table when server is down** | | | |
| **Given** | **When** | | **Then** |
| * The database books table is created and set up correctly * The booksResource.java is created and set up correctly * The booksDAO.java is created and set up correctly * The server is down * Using the DELETE method on Postman | * The user enters <http://localhost:8080/PremiumLibraryTeam/rest/books>/{ID} on postman   The user clicks the send button | | * The user gets a connection error. |