Certainly! Let's break down the requirements for Stage 3 and let me guide you through building the live working project for a Library Management System:

Stage 3: Implementation and Documentation

1. Develop the System

1. Define Database Tables:
   - Based on our earlier design, create tables for the Library Management System. You'll likely have tables for `Book`, `User`, `Transaction`, etc.
After installing PostgreSQL, you can interact with the database using the command-line tool called `psql` or by using a graphical user interface (GUI) tool like pgAdmin. I'll guide you through using `psql` for this example.

Using pgAdmin :

1. Open pgAdmin:
2. - If you prefer a graphical interface, open pgAdmin.

2. Connect to PostgreSQL Server:
   - Connect to your PostgreSQL server using the provided credentials.

3. Create Database and Tables:
   - In the pgAdmin interface, you can create a new database and tables using the GUI.

4. Execute SQL Queries:
   - You can also execute SQL queries directly through the Query Tool in pgAdmin.

Example: Creating a Database and Tables

Assuming you have PostgreSQL installed, here's a simple example to create a database and a table:

1. Create a Database:

```
CREATE DATABASE library_db;
```

2. Connect to the Database:

```
psql -U your_username -d library_db
```

3. Create a Table:

1. “User”
2. Author
3. Genre
4. Book
5. Transaction
6. Review
7. Fine
8. Wishlist

Remember to replace `your_username` with your PostgreSQL username. Adjust the SQL commands based on your specific requirements.

INSERT statements

```
-- Insert data into "User" table
INSERT INTO "User" (UserID, FirstName, LastName, Email, PhoneNumber, Address)
VALUES
(1, 'John', 'Doe', 'john.doe@email.com', '123-456-7890', '123 Main St'),
(2, 'Alice', 'Smith', 'alice.smith@email.com', '987-654-3210', '456 Oak St'),
(3, 'Robert', 'Johnson', 'robert.j@email.com', '555-123-7890', '789 Pine St'),
(4, 'Sophia', 'Williams', 'sophia.w@email.com', '111-222-3333', '321 Elm St'),
(5, 'Michael', 'Brown', 'michael.b@email.com', '444-555-6666', '567 Birch St'),
(6, 'Emily', 'Jones', 'emily.j@email.com', '777-888-9999', '890 Cedar St'),
(7, 'Daniel', 'Davis', 'daniel.d@email.com', '333-222-1111', '234 Maple St'),
(8, 'Emma', 'Taylor', 'emma.t@email.com', '999-888-7777', '432 Willow St'),
(9, 'Matthew', 'Miller', 'matthew.m@email.com', '666-555-4444', '678 Pine St'),
(10, 'Olivia', 'Anderson', 'olivia.a@email.com', '000-111-2222', '876 Oak St'),
(11, 'Ethan', 'Thomas', 'ethan.t@email.com', '555-666-7777', '987 Birch St'),
(12, 'Ava', 'White', 'ava.w@email.com', '111-222-3333', '765 Maple St'),
(13, 'Noah', 'Harris', 'noah.h@email.com', '444-555-6666', '543 Cedar St'),
(14, 'Isabella', 'Smith', 'isabella.s@email.com', '777-888-9999', '234 Elm St'),
(15, 'Liam', 'Johnson', 'liam.j@email.com', '999-888-7777', '876 Willow St'),
(16, 'Mia', 'Taylor', 'mia.t@email.com', '333-222-1111', '321 Pine St'),
(17, 'Jackson', 'Williams', 'jackson.w@email.com', '111-222-3333', '567 Cedar St'),
(18, 'Sophie', 'Brown', 'sophie.b@email.com', '555-666-7777', '890 Birch St'),
(19, 'Lucas', 'Thomas', 'lucas.t@email.com', '777-888-9999', '432 Maple St'),
(20, 'Chloe', 'Anderson', 'chloe.a@email.com', '999-888-7777', '765 Willow St');
```

```
-- Insert data into "Author" table
INSERT INTO Author (AuthorID, FirstName, LastName)
VALUES
(1, 'F. Scott', 'Fitzgerald'),
(2, 'Harper', 'Lee'),
(3, 'George', 'Orwell'),
(4, 'Jane', 'Austen'),
(5, 'J.D.', 'Salinger'),
(6, 'J.K.', 'Rowling'),
(7, 'J.R.R.', 'Tolkien'),
(8, 'Aldous', 'Huxley'),
(9, 'Fyodor', 'Dostoevsky'),
(10, 'Gabriel', 'García Márquez'),
(11, 'Charles', 'Dickens'),
(12, 'Herman', 'Melville'),
(13, 'Homer', ''),
(14, 'Dante', 'Alighieri'),
(15, 'Oscar', 'Wilde'),
(16, 'Emily', 'Brontë'),
(17, 'Leo', 'Tolstoy'),
(18, 'Oscar', 'Wilde'),
(19, 'Leo', 'Tolstoy'),
(20, 'C.S.', 'Lewis');
```

```
-- Insert data into "Genre" table
INSERT INTO Genre (GenreID, GenreName)
VALUES
(1, 'Fiction'),
(2, 'Classics'),
```

```
(3, 'Dystopian'),
(4, 'Romance'),
(5, 'Coming of Age'),
(6, 'ComicFantasy'),
(7, 'Adventure'),
(8, 'Science Fiction'),
(9, 'Philosophy'),
(10, 'Magical Realism'),
(11, 'Mystery'),
(12, 'Gothic'),
(13, 'Epic Poetry'),
(14, 'Allegory'),
(15, 'Satire1'),
(16, 'Tragedy'),
(17, 'Historical Fiction'),
(18, 'Satire2'),
(19, 'Philosophical'),
(20, 'Fantasy');
```

```
-- Insert data into "Book" table
INSERT INTO Book (BookID, Title, AuthorID, GenreID, ISBN, PublishedYear, Available)
VALUES
(1, 'The Great Gatsby', 1, 1, '9780142437260', 1925, true),
(2, 'To Kill a Mockingbird', 2, 2, '9780061120084', 1960, true),
(3, '1984', 3, 3, '9780451524935', 1949, true),
(4, 'Pride and Prejudice', 4, 4, '9780141439518', 1813, true),
(5, 'The Catcher in the Rye', 5, 1, '9780241950432', 1951, true),
(6, 'Harry Potter and the Sorcerer''s Stone', 6, 2, '9780590353427', 1997, true),
(7, 'The Hobbit', 7, 3, '9780261102217', 1937, true),
(8, 'Brave New World', 8, 4, '9780060850524', 1932, true),
(9, 'Crime and Punishment', 9, 1, '9780143058145', 1866, true),
(10, 'The Lord of the Rings', 7, 3, '9780261103252', 1954, true),
(11, 'One Hundred Years of Solitude', 10, 4, '9780061120084', 1967, true),
(12, 'The Great Expectations', 11, 1, '9780141439563', 1861, true),
(13, 'Moby-Dick', 12, 2, '9780142437246', 1851, true),
(14, 'The Odyssey', 13, 3, '9780451474339', -800, true),
(15, 'The Divine Comedy', 14, 4, '9780142437222', 1320, true),
(16, 'The Brothers Karamazov', 9, 1, '9780143058145', 1880, true),
(17, 'The Picture of Dorian Gray', 15, 2, '9780141439570', 1890, true),
(18, 'Wuthering Heights', 16, 4, '9780141439556', 1847, true),
(19, 'The War and Peace', 9, 1, '9780141439570', 1869, true),
(20, 'The Chronicles of Narnia', 17, 3, '9780066238500', 1950, true)
ON CONFLICT (ISBN) DO NOTHING;
```

```
-- Insert data into "Transaction" table
INSERT INTO Transaction (TransactionID, UserID, BookID, BorrowDate, ReturnDate)
VALUES
(1, 1, 1, '2023-01-01', '2023-01-15'),
(2, 2, 2, '2023-02-01', '2023-02-15'),
(3, 3, 3, '2023-03-01', '2023-03-15'),
-- Add more transactions as needed

-- Continue with other previously inserted User and Book combinations
(4, 4, 4, '2023-04-01', '2023-04-15'),
```

```
(5, 5, 5, '2023-05-01', '2023-05-15'),
(6, 6, 6, '2023-06-01', '2023-06-15');
-- Add more transactions as needed

-- Ensure that UserIDs and BookIDs exist in the corresponding tables
```

```
-- Insert data into "Review" table
INSERT INTO Review (ReviewID, UserID, BookID, Rating, Comment)
VALUES
(1, 1, 1, 5, 'Great book! Highly recommended.'),
(2, 2, 2, 4, 'Enjoyed the plot twists.'),
(3, 3, 3, 3, 'Average read, could be better.'),
(4, 4, 4, 5, 'Captivating storyline.'),
(5, 5, 5, 2, 'Not my favorite, but okay.'),
(6, 6, 6, 4, 'Well-written and engaging.'),
-- Add more reviews as needed

-- Continue with other previously inserted User and Book combinations
(7, 1, 2, 4, 'Liked the character development.'),
(8, 2, 3, 5, 'Couldnot put it down!'),
(9, 3, 4, 3, 'Expected more from the ending.'),
(10, 4, 5, 4, 'Excellent choice for book clubs.'),
(11, 5, 6, 2, 'Found it a bit slow-paced.'),
(12, 6, 1, 5, 'One of my all-time favorites.');
-- Add more reviews as needed

-- Ensure that UserIDs and BookIDs exist in the corresponding tables
```

```
-- Insert data into "Fine" table with valid TransactionID values
INSERT INTO Fine (FineID, TransactionID, Amount, Paid)
VALUES
(1, 1, 5.00, true),
(2, 2, 8.50, false),
(3, 3, 3.75, true),
(4, 4, 6.20, false),
(5, 5, 10.00, true),
(6, 6, 2.50, false),
-- Ensure that the following TransactionID values exist in the Transaction table
(7, 1, 7.80, true),
(8, 2, 4.25, false),
(9, 3, 6.50, true),
(10, 4, 9.00, false);
```

```
-- Insert data into "Wishlist" table
INSERT INTO Wishlist (WishlistID, UserID, BookID)
VALUES
(1, 1, 2),
(2, 1, 5),
(3, 2, 1),
(4, 3, 3),
(5, 3, 6),
(6, 4, 4),
(7, 5, 7),
(8, 5, 10),
```

(9, 6, 9),
(10, 7, 8);

-- Add more wishlist entries as needed

---

2. Implement Functionalities:
  - Develop features for inserting, deleting, updating data, generating analytical reports, and handling transactions. For example: (Refer stage3 Document for queries)
    - Insert: Add a new book to the system.
    - Delete: Remove a user from the database.
    - Update: Change the availability status of a book.
    - Analytical Report: Generate a report on the most borrowed books.
    - Transaction: Borrow and return books with rollback functionality.

3. Error Trapping and Recovery:
  - Implement error handling mechanisms. For example, if a user tries to borrow a book that is not available, catch the error and provide feedback without crashing the system.

4. User Interface:
  - Choose a user interface (CLI, website, or desktop application) and implement it. Ensure that users can interact with the system seamlessly.

2. Database Backend

1. Choose an RDBMS:
  - Select an open-source RDBMS like PostgreSQL. Install and set it up to host your database.

2. Data Model in BCNF:
  - Ensure that your database tables follow Boyce-Codd Normal Form (BCNF).

3. Define Keys and Indexes:
  - Define primary keys and indexes to optimize data retrieval, considering scalability.

3. Query Demonstrations

A. Join Queries:
  - Create a query that joins at least 5 tables in a meaningful way. For example, generate a report that lists all books, their authors, genres, borrowers, and return dates.
2 queries with solution provided.
B. An analytical report with sorting option

```
-- Analytical Report: Average Ratings of Books, Sorted by Rating in Descending Order
SELECT
   B.Title AS BookTitle,
   AVG(R.Rating) AS AverageRating
FROM
   Book B
LEFT JOIN
   Review R ON B.BookID = R.BookID
GROUP BY
   B.BookID, B.Title
ORDER BY
   AverageRating DESC;
```

In this query:

We're selecting the book title and calculating the average rating using the AVG() function.
The LEFT JOIN ensures that even books without reviews are included in the report.

The GROUP BY clause groups the results by book, and the ORDER BY clause sorts the results by average rating in descending order.
You can customize the query based on the specific fields or criteria you want to include in your analytical report. Adjust the sorting options, fields, or additional conditions to meet your project requirements.

C. A transaction that involves more than a single SQL statement with at least one justifiable rollback.
Certainly! Let's consider a scenario where a user is borrowing a book, and the transaction involves multiple SQL statements with a justifiable rollback in case of an error:

```
-- Begin Transaction
BEGIN;

-- Step 1: Check Book Availability
DECLARE available BOOLEAN;
SELECT Availability INTO available FROM Book WHERE BookID = 1 FOR UPDATE;

-- If the book is not available, rollback the transaction
IF NOT available THEN
   ROLLBACK;
   RAISE EXCEPTION 'The selected book is not available for borrowing.';
END IF;

-- Step 2: Record the Borrowing Transaction
INSERT INTO "Transaction" (UserID, BookID, BorrowDate)
VALUES (1, 1, CURRENT_DATE);

-- Step 3: Update Book Availability
UPDATE Book SET Availability = FALSE WHERE BookID = 1;

-- Commit the transaction
COMMIT;
```

Explanation:

We start a transaction using BEGIN.
We use a DECLARE statement to check the availability of the book and lock the row for update to avoid race conditions.
If the book is not available, we rollback the transaction with an exception.
If the book is available, we proceed to step 2 by inserting a new record into the "Transaction" table.
In step 3, we update the "Availability" column of the "Book" table to mark the book as borrowed.
Finally, we commit the transaction using COMMIT.
If any step fails, the transaction is rolled back, ensuring data consistency. This example demonstrates a transaction involving multiple SQL statements with a justifiable rollback in case of an error.

4. User Manual

1. Essential System Environments:
  - Detail the system requirements and the environments needed for deploying the project.

2. Deployment Instructions:
  - Provide step-by-step instructions on deploying your project on a given platform. Include any configuration steps.

3. Description of Functionalities:
  - Concisely describe how users can utilize each functionality in your system.

5. Final Team Report

1. Project Work Execution:
  - Detail how the team executed tasks during the semester and how it aligns with the initial team contract.

2. Reflection on Project Development:
  - Reflect on the challenges faced, successes achieved, and lessons learned during the project development process.

6. Submission

1. Final Software Architect:
  - Host your final software architect in a Git repository. Include the repository address in your final team report.

2. Submission Zip File:
  - Create a zip file containing the user manual (PDF) and the final team report (PDF).

Additional Considerations

- Data Set:
  - Include a realistic subset of data for demonstration purposes. Ensure proper usage rights and disclose its origin.

- Testing:
  - Test the system thoroughly with both small and large datasets to ensure functionality and performance.

- User Demonstration:
  - Prepare a short demonstration showcasing key functionalities for the course staff during the final submission.

This breakdown should guide you through the implementation of Stage 3. Adjust the steps based on your project specifics and requirements. If you have specific questions or need more details on any step, feel free to ask!