Assignment Pathlock 2025

Every Student needs to develop any one out of 2 programming Exercise. Follow the steps to perform the below tasks.

Programming Exercise:

1. Development of a "refrigerator" app

The refrigerator will have the following functionalities to manage products.

Stage 1:

Insert - inserting a product into the refrigerator while entering quantity

Consume - enter how much was used, for example half a liter milk, 250gms vegetables etc.

Status - the refrigerator will keep a current quantity for each item.

In addition, the system will keep a history of purchase / consumption.

Stage 2:

Adding support for expired products.

Each item will have an expiration date.

The system will alert users when there are items in the refrigerator that are about to expire**.**

When the expiration date on the item passes, the system will deduct its quantity from stock and show the user a message on the subject - to remove the item from the refrigerator

Stage 3:

Create a shopping list based on past shopping and actual consumption.

For example, if I buy milk and another is about to run out, the app will offer me more milk

Evaluation Criteria:

- Correctness: The program should calculate the item's quantity correctly.
- Code Structure: Clean, modular, and readable code.
- Efficiency: Handles a reasonable number of items in the refrigerator efficiently.
- Documentation: Proper comments explaining the logic.
- Edge Cases: Consider cases like no items, single item, and large quantities.

1. Evaluate Expression

Assignment Pathlock 2025

You have to write a program which takes Mathematical expression as an input, evaluate the expression and provide the result in output. You can write the solution in any programming language. The program should consider the associativity and precision of the operators.

Constraint:
You must write your own logic of the calculation.
You can use any data structure.

Example

- **Input:** 25+5-(4*5-5)

Output: 15

- **Input:** 24-8+9*2-10/5

Output: 32

API Assignments

Every Students needs to perform any 1 out of 2 API Assignments. Follow the steps mentioned in the Tasks below.

- **API Assignment 1: User Provisioning System**
- **API Assignment 2: Project Management System**

API Assignment 1: User Provisioning System

Objective:

Develop a RESTful API for a **User Provisioning System** that manages users, roles, and their access permissions in an organization.

Requirements:

Assignment Pathlock 2025

Entities:

- **User**

ID (unique, auto generated)
Name
Email (unique)
Status (Active/Inactive)
Created Date

- **Role**

ID (unique, auto generated)
Name (e.g., Admin, Manager, Employee)
Description

- **User Role Assignment**

ID (unique, auto generated)
User ID
Role ID
Assigned Date

API Endpoints:

- **User Management APIs**

POST /users – Create a new user.
GET /users – Get all users (filter by status).
GET /users/{id} – Get details of a specific user.
PUT /users/{id} – Update user details (e.g., change status).
DELETE /users/{id} – Soft delete a user (mark status as "Inactive").

- **Role Management APIs**

POST /roles – Add a new role.
GET /roles – List all roles.
GET /roles/{id} – Get details of a specific role.
PUT /roles/{id} – Update role details.

Assignment Pathlock 2025

DELETE /roles/{id} – Delete a role.

- **User Role Assignment APIs**

POST /user-roles – Assign a role to a user.
GET /user-roles – List all user-role assignments (filter by user or role).
DELETE /user-roles/{id} – Remove a role assignment from a user.

Functional Requirements:

- A user can have multiple roles, but duplicate assignments should not be allowed.
- Validate all inputs (e.g., unique email for users, non-empty fields).
- Use meaningful HTTP status codes for API responses.
- Ensure proper error handling (e.g., user not found, role already assigned).

Technologies to Use:

- **Backend Language**: Any (C#, Java, Python etc.).
- **Database**: Any (SQLite, MySQL, File based etc.).
- **API Documentation**: Postman.

Additional Features:

- **Search and Filtering:** Implement **search functionality** for users by name or email.
- **Pagination:** Add **pagination** for listing users and roles.

Evaluation Criteria:

- **Functionality:** All required endpoints must work correctly.
- **Validation:** Proper error handling and input validation.
- **Code Quality:** Clean, modular, and maintainable code.
- **Documentation:** Clear setup instructions and well-documented APIs.
- **Scalability:** Efficient database queries and API design.

Assignment Pathlock 2025

Submission Instructions:

- Share code zip file.
- Include a README.md file with setup instructions.
- Attach API documentation (Postman collection).
- Provide an API testing screenshot

API Assignment 2: Project Management System

Objective:

Develop a RESTful API for managing projects and tasks within an organization. The system should allow basic project and task management and user assignment.

Requirements:

Entities:

- **Project**

ID (unique, auto generated)
Name
Description
Start Date
End Date
Status (e.g., Active, Completed, On Hold)

- **Task**

ID (unique, auto generated)
Project ID (foreign key)
Name
Description
Assigned To (User ID)
Due Date
Status (e.g., Pending, In Progress, Completed)

- **User**

Assignment Pathlock 2025

ID (unique, auto generated)
Name
Email (unique)

API Endpoints:

Project APIs

POST /projects – Create a new project.
GET /projects – List all projects.
GET /projects/{id} – Get details of a specific project.
PUT /projects/{id} – Update project details.
DELETE /projects/{id} – Delete a project.

Task APIs

POST /tasks – Create a task for a project.
GET /tasks – List all tasks (filter by project or user).
GET /tasks/{id} – Get details of a specific task.
PUT /tasks/{id} – Update task details (e.g., change status).
DELETE /tasks/{id} – Delete a task.

User APIs

POST /users – Create a new user.
GET /users – List all users.
GET /users/{id} – Get details of a specific user.
PUT /users/{id} – Update user details.
DELETE /users/{id} – Delete a user.

Functional Requirements:

- Each task must belong to a project.
- A task cannot be created without assigning it to a user.
- A user cannot be assigned more than 5 active tasks at a time.
- Automatically mark a project as "Completed" when all its tasks are marked "Completed."
- Validate input fields (e.g., unique email for users, non-empty fields).

Assignment Pathlock 2025

Technologies to Use:

- **Backend Language**: Any (C#, Java, Python etc.).
- **Database**: Any (SQLite, MySQL, File based etc.).
- **API Documentation**: Postman.

Additional Features:

- **Search and Filtering:** Implement search for projects and tasks by name or status.
- **Pagination:** Add pagination for listing projects and tasks.

Evaluation Criteria:

- **Functionality:** All required endpoints must work correctly.
- **Validation:** Proper error handling and input validation.
- **Code Quality:** Clean, modular, and maintainable code.
- **Documentation:** Clear setup instructions and well-documented APIs.
- **Scalability:** Efficient database queries and API design.

Submission Instructions:

- Share code zip file.
- Include a README.md file with setup and usage instructions.
- Attach API documentation (Postman collection)
- Provide an API testing screenshot

QA Testing Assignment

Every student needs to perform any 1 out of 2 Testing Assignments. Follow the steps in tasks given below.

Assignment Pathlock 2025

1. Ride-Sharing App Feature Testing

Feature Specification:

A ride-sharing app introduces a feature for "Scheduled Rides", allowing users to book a ride up to 7 days in advance.

- Users can set the pickup location, destination, and time.
- Users receive notifications 30 minutes and 5 minutes before the ride.
- Drivers can accept scheduled rides in advance and receive reminders.
- A cancellation policy applies: users cancelling less than 30 minutes before the ride incur a penalty fee.

Tasks:

1. Identify the key testing areas for this feature.
2. What are the attractive features that can be added to any taxi application?

2. Mobile Banking App Feature Testing

Feature Specification:

You are testing a mobile banking app that has introduced a new feature: "Quick Fund Transfer". The feature allows users to send money instantly to another bank account. Key aspects of the feature include:

1. The user can transfer amounts ranging from ₹1 to ₹2,00,000 per transaction.
2. Supports NEFT, UPI, and RTGS payment modes.
3. Requires a 2-step authentication process: Login with credentials and OTP verification.

4. Allows saving beneficiary details for quick future transfers.

5. Displays transaction history and real-time status updates.

Tasks:

1. Identify Key Testing Areas:

List the main areas you would focus on to ensure the feature works correctly.

2. Create a Test Strategy Considering:

o Different User Personas: Address various types of users (e.g., frequent users, new users, users with limited technical knowledge).

o Various Payment Methods: How would you test NEFT, UPI, and RTGS functionality?

o Edge Cases

o Security Considerations.

o Performance Scenarios.