Module: Programming Fundamentals

Map SDLC phases to build an Attendance Tracker. Learners outline user stories, select a model (Agile/Waterfall), and produce a coding standards checklist; verify local Python + IDE setup via a 'hello UST' script.

SDLC Phases to build Attendance Tracker

1. SDLC Mapping to Attendance Tracking System

Requirement Gathering & Analysis:

Functional Requirements:

- User login (Admin/User)

- Mark attendance using QR code

- View attendance status

- Leave request and approval

- Generate attendance reports

Non-Functional Requirements:

- Data security

- Accuracy of time and date

- Easy to use interface

Outcome:

Clear understanding of system requirements.

Design:

High-Level Design:

- Web application with frontend, backend, and database

- Modules: Login, Attendance, Leave, Admin Dashboard

Low-Level Design:

- Database tables: User, Attendance, Leave
- APIs for login, attendance, leave
- UI screens for users and admin

Development:

- Frontend: HTML, CSS, JavaScript
- Backend: Python (Flask)
- Database: MySQL/SQLite
- QR code integration

Testing:

- Unit Testing
- Integration Testing
- System Testing
- Performance Testing
- User Acceptance Testing

Deployment:

- Deploy application to production environment

Maintenance:

- Bug fixes
- Feature updates

2. **User Stories**

- As a user, I want to log in so that I can access the system.

- As a user, I want to mark my attendance using QR code so that my presence is recorded.

- As an admin, I want to view attendance reports so that I can monitor users.

- As a user, I want to apply for leave so that my absence is recorded properly.

3. **Model Selection**

Selected Model: Agile

Reason:

Agile is flexible and supports continuous improvement.

Agile Roles:

- Product Owner (PO)

- Scrum Master (SM)

- Development Team (DT)

Sprint Plan (2 weeks per sprint):

Sprint 1: Login module and basic UI

Sprint 2: Attendance and QR integration

Sprint 3: Leave management

Sprint 4: Reports and dashboard

4. **Coding Standards Checklist**

- Use meaningful variable and function names

- Follow proper indentation

- Add comments for clarity

- Handle errors properly

- Avoid hard-coded values

- Use version control (Git)

5. Python & IDE Setup Verification

To verify the Python and IDE setup, a simple script was executed.

Code:

```
print("Hello UST")
```

Output:

Hello UST