Assignment 1: Collaborative Software Development

Deadline: 25th January 2026 EOD

Objective

The goal of this assignment is to provide students with hands-on experience in collaborative software development by implementing a software system while emphasizing software qualities (usability, reliability, performance, safety). Students will work in groups to simulate a global development environment, use collaboration tools, and practice defined team roles.

Project Options

Students will work in groups of **8 members each**, with **10 groups working on each of the following projects**:

1. Disaster Alert System (odd-numbered groups like 1, 3, 5, 7,…)

- **Description**: Design a disaster alert system that sends real-time alerts to users in affected regions.
- **Focus Areas**:
    1. **Usability**: Create a user-friendly interface for sending and receiving alerts.
    2. **Reliability**: Ensure the system can handle high traffic during disasters without downtime.
    3. **Safety**: Provide accurate alerts with no false positives/negatives.

2. Smart Parking System (even-numbered groups like 0, 2, 4, 6, …)

- **Description**: Build a system to detect available parking slots and prevent double bookings, ensuring a seamless user experience.
- **Focus Areas**:
    1. **Usability**: Provide clear parking slot availability and reservation status.
    2. **Reliability**: Prevent errors like assigning a single slot to multiple vehicles.
    3. **Performance**: Ensure the system remains fast and responsive during peak usage.

Team Roles

Each group will consist of **8 members** with the following roles:

1. **Project Manager**: Oversees the project timeline and deliverables.
2. **Requirements Engineer**: Gathers and documents requirements.
3. **Developer (2 roles)**: Implements the assigned codebase and handles bug fixes.
4. **Tester**: Creates and executes test cases for the system.
5. **Quality Assurance Engineer**: Verifies adherence to quality standards.

6. **Integration Specialist**: Handles integration of various modules.
7. **Documentation Specialist**: Prepares all project documentation.

Deliverables

Each group will submit the following:

1. **Codebase**: Complete implementation of the assigned project.
2. **Documentation**:
    - Requirements Document.
    - Testing Report.
    - User Guide.
3. **Collaboration Evidence**:
    - Screenshots/logs of Trello, GitHub, and Telegram usage.

Project 1: Disaster Alert System

Starter Code in Python

```
from datetime import datetime

# Sample database of users and locations
users = {
    "user1": {"location": "City A", "email": "user1@example.com"},
    "user2": {"location": "City B", "email": "user2@example.com"},
    "user3": {"location": "City A", "email": "user3@example.com"}
}

# Function to validate user input
def validate_location(location):
    valid_locations = {details["location"] for details in
users.values()}
    return location in valid_locations

# Function to send alerts
def send_alert(location, message):
    if not validate_location(location):
        print(f"Error: '{location}' is not a valid location.")
        return

    print(f"\nSending Alert for {location}: {message}")
```

```
    recipients = [details["email"] for user, details in users.items()
if details["location"] == location]

    if recipients:
        for email in recipients:
            print(f"Email sent to {email}")
    else:
        print("No users found in this location.")

# Main program
if __name__ == "__main__":
    print("=== Disaster Alert System ===")
    location = input("Enter affected location: ").strip()
    message = input("Enter alert message: ").strip()
    send_alert(location, message)
```

Project 2: Smart Parking System

Starter Code in Python

```
# smart_parking.py

# Sample parking lot data
parking_lot = {"A1": False, "A2": True, "B1": True, "B2": False}  #
True = Available, False = Occupied

# Function to display parking status
def display_parking_status():
    print("\nParking Lot Status:")
    for slot, status in parking_lot.items():
        print(f"Slot {slot}: {'Available' if status else 'Occupied'}")

# Function to validate slot input
def validate_slot(slot):
    return slot in parking_lot

# Function to book a slot
def book_parking_slot(slot):
    if not validate_slot(slot):
```

```
        print(f"Error: Slot '{slot}' does not exist.")
        return

    if parking_lot.get(slot):
        parking_lot[slot] = False
        print(f"Slot {slot} successfully booked!")
    else:
        print(f"Slot {slot} is already occupied.")

# Function to release a slot
def release_parking_slot(slot):
    if not validate_slot(slot):
        print(f"Error: Slot '{slot}' does not exist.")
        return

    if not parking_lot.get(slot):
        parking_lot[slot] = True
        print(f"Slot {slot} has been released.")
    else:
        print(f"Slot {slot} is already available.")

# Main program
if __name__ == "__main__":
    print("=== Smart Parking System ===")
    while True:
        display_parking_status()
        action = input("\nEnter 'book', 'release', or 'exit':
").strip().lower()
        if action == "exit":
            break
        elif action in ["book", "release"]:
            slot = input("Enter slot: ").strip()
            if action == "book":
                book_parking_slot(slot)
            elif action == "release":
                release_parking_slot(slot)
        else:
```

```
            print("Invalid action. Please enter 'book', 'release', or
'exit'.")
```

Submission Steps

1. **Prepare Files**:
    - `Assignment_1_Report.docx` (documentation).
    - Python code files (`disaster_alert.py` or `smart_parking.py`).
    - Collaboration evidence (screenshots/logs from Trello, GitHub, and Telegram).
2. **Organize into Folders**:
    - Create a folder: `Assignment_1_Group_<GroupNumber>`.
    - Inside, create subfolders:
        - `Code/` (for `.py` files).
        - `Documentation/` (for the Word report).
        - `Collaboration Evidence/` (for screenshots/logs).
3. **Compress Folder**:
    - Right-click on the parent folder (`Assignment_1_Group_<GroupNumber>`).
    - Select **Send to > Compressed (zipped) folder** (Windows) or **Compress** (Mac).
    - Name the `.zip` file: `Assignment_1_Group_<GroupNumber>.zip`.
4. **Submit**:
    - Upload the `.zip` file to the LMS (one per group).