**Objective:**

Develop a Telegram bot that automates the key operations for **Persist Ventures (PV)**, including task management, performance tracking, automated reporting, and integration with external tools (e.g., Google Sheets, Trello). The bot should simplify team communication, track assignments, manage deadlines, and automate recurring tasks.

**Part 1: Bot Development Fundamentals**

**1. Create a Telegram Bot**

* **Task**:
  + Create a simple Telegram bot using the **Telegram Bot API**.
  + The bot should handle basic interactions, including /start and /help commands.
  + The bot should reply with a greeting message and a simple description of its capabilities when the user sends /start.
* **Assessment Criteria**:
  + Correctly set up the Telegram bot.
  + Basic interaction (e.g., /start).
  + Simple, user-friendly responses.

**2. Task Assignment Feature**

* **Task**:
  + Implement a command /assign\_task that allows an admin to assign tasks to a specific user.
  + Tasks should have a description and a due date.
  + The bot should notify the assignee when they are assigned a task, including the task description and deadline.
* **Assessment Criteria**:
  + Correctly implementing command structure.
  + Handling task assignment logic.
  + Sending messages to both the admin and the assigned user.

**Part 2: Automation & Workflow**

**3. Task Completion & Progress Tracking**

* **Task**:
  + Implement the /complete\_task command that allows a user to mark a task as completed.
  + The bot should update the task status in the database and notify the admin when the task is completed.
  + Add a /progress command that allows users to check the progress of their assigned tasks (e.g., "Task 1: 50% complete").
* **Assessment Criteria**:
  + Accurate task status updates.
  + Proper progress tracking.
  + Clean user interaction for completion updates and progress checking.

**4. Automatic Reminders**

* **Task**:
  + The bot should send **automatic reminders** (e.g., 24 hours before a task deadline).
  + Implement a background task using a scheduling library (e.g., **Celery** for Python) to send reminders to users about upcoming due dates.
* **Assessment Criteria**:
  + Correct use of scheduling libraries.
  + Timely and accurate reminders sent.
  + System to track and schedule reminders for pending tasks.

**Part 3: API Integrations & External Tools**

**5. Google Sheets API Integration**

* **Task**:
  + Integrate the bot with **Google Sheets** to store and manage task data (task description, assignee, deadline, status).
  + When a task is assigned, the bot should update the Google Sheets file with task details.
  + Optionally, provide a command /task\_report that generates a simple report of tasks from the Google Sheets document and sends it as a message.
* **Assessment Criteria**:
  + Correct integration with **Google Sheets API**.
  + Reliable reading/writing data to/from Google Sheets.
  + Cleanly formatted task reports.

**6. Integration with External Task Management Tool (Trello, Asana, or Slack)**

* **Task**:
  + Integrate the bot with a task management tool such as **Trello** (or **Asana** or **Slack**).
  + When a task is completed on Telegram, automatically update the corresponding card/list on Trello (or a similar task board).
  + Optionally, integrate Slack for team notifications.
* **Assessment Criteria**:
  + Successful integration with the selected task management tool.
  + Synchronization of task updates between Telegram bot and external tool.
  + Sending notifications to the team on task completions.

**Part 4: Database Handling & Data Persistence**

**7. Database Design & Task Management**

* **Task**:
  + Implement a basic database (e.g., **SQLite**, **PostgreSQL**, or **MongoDB**) to store task data (task name, description, assigned user, deadline, status).
  + Design and implement the following:
    - A table/collection for storing tasks.
    - A table/collection for storing user information (user ID, name, role).
  + Use **SQLAlchemy** (for Python) or **Mongoose** (for Node.js) to interact with the database.
* **Assessment Criteria**:
  + Proper database schema design.
  + Efficient task data storage and retrieval.
  + Proper task status updates in the database.

**Part 5: Reporting & Promotion System**

**8. Performance Reporting & Milestone Tracking**

* **Task**:
  + Implement a reporting system to track user progress and overall team performance.
  + Include a /report command that generates a simple report of completed tasks by all users within the last week.
  + Implement a promotion milestone system: once a user completes a certain number of tasks, they receive an automatic promotion notification or equity reward.
* **Assessment Criteria**:
  + Clear and concise report generation.
  + Effective milestone tracking and promotions.
  + Handling user progress and task completion metrics.

**Part 6: Bonus Features (Optional)**

**9. Error Handling & Resilience**

* **Task**:
  + Implement basic error handling for common issues, such as network errors, failed API calls, or incorrect command formats.
  + The bot should respond with user-friendly error messages.
* **Assessment Criteria**:
  + Robust error handling for common failure scenarios.
  + User-friendly error messages.

**10. Bot Scalability & Optimization**

* **Task**:
  + Suggest or implement improvements to make the bot scalable if the user base grows (e.g., handling a higher volume of tasks or users).
  + Consider database indexing, background jobs for reminders, and caching for performance.
* **Assessment Criteria**:
  + Thoughtful approach to scalability.
  + Performance optimizations that would be suitable for larger teams.