Project Title: Digital Twin for EV Battery Health & Predictive Maintenance

Abstract: This project develops a Digital Twin–based predictive maintenance framework for Electric Vehicles (EVs). Real-time IoT sensor data such as battery temperature, voltage, SoH, and SoC are captured using embedded hardware and transmitted to a cloud server via MQTT. A virtual twin mirrors the physical battery's behavior and predicts Remaining Useful Life (RUL) using AI/ML models trained on the EVIoT-PredictiveMaint dataset. The system enables failure prediction, smart maintenance scheduling, and performance optimization. A web dashboard visualizes real-time parameters, alerts, and recommendations, demonstrating how Digital Twin concepts can enhance EV fleet reliability and sustainability.

Justification & Uniqueness:

- Current EV monitoring solutions show only instantaneous metrics.
- Your project adds **predictive intelligence** (RUL + failure probability) using ML and **real-time IoT streaming** to a **cloud-based digital twin** — not just static telemetry.
- Integration with **OPC UA + MQTT** ensures interoperability between embedded nodes and cloud analytics.

Scalability Options:

- Multiple IoT sensor nodes can send data through MQTT brokers (e.g., Mosquitto) to the cloud.
- Cloud services (AWS IoT Core, ThingsBoard, or Firebase + Node-RED) visualize and store data.
- OPC UA can link the Digital Twin to industrial platforms.

AI/ML Models:

Use the following pipeline:

- **Feature extraction:** battery voltage, SoC, temperature, cycles
- **Modeling:**
  - *LSTM / GRU* → predict RUL (time-series)
  - *Random Forest / XGBoost* → classify failure probability
  - *SHAP/LIME* → explainability
- These models can continuously update via federated learning simulation.

Sensing / Actuation / Recommendation:

- **Sensing:** battery voltage, temperature, current (real or simulated)
- **Synthetic data:** can be generated from the EVIoT-PredictiveMaint dataset for cloud testing
- **Actuation:** LED / relay / buzzer triggers when battery health < threshold
- **Recommendation:** cloud dashboard suggests "Schedule Maintenance" or "Reduce Load"

[ESP32 / Simulated Sensor]
↓ MQTT
[Cloud Broker + Database]
↓
[AI Layer]
├─ Public Model (General)
└─ User Model (Personalized)
↓
[Recommendation Engine]
↓
[Unity 3D Twin + Web Dashboard]
↓
[Actuators / Alerts / OPC UA Nodes]

Phase-wise Project Roadmap:

Phase 1 – Ideation & Planning (Week 1–2)

📘 **Deliverables:**

- Finalize title and concept (✅)
- Prepare 3 PPT ideas for evaluation (you'll use this as the main one)
- Define the **functional block diagram**
  (you can use the one you wrote — with ESP32 → MQTT → Cloud → AI → Unity Twin → OPC UA)
- Identify datasets and hardware

📦 **Tasks:**

- Select dataset → EVIoT-PredictiveMaint (✅)
- Choose sensors: voltage, temperature, current, SoH, SoC
- Confirm connectivity protocols: MQTT for IoT, OPC UA for digital twin interface
- Finalize tools:
  - **Hardware:** ESP32 (sensor node), LED/Relay for actuation
  - **Software:** Node-RED / ThingsBoard / Firebase / AWS IoT Core
  - **AI/ML Models:** LSTM (for RUL), RandomForest/XGBoost (for failure probability)
  - **Digital Twin UI:** Unity 3D (for visualization) + Web Dashboard (Flask/Streamlit)

Phase 2 – Data Layer & Sensing (Week 3–5)

📘 **Deliverables:**

- Collect or simulate data using dataset (battery voltage, SoH, SoC, temperature)
- Generate synthetic data if real sensors are unavailable

- Prepare a **data pipeline** for streaming to the cloud

📦 **Tasks:**

- Write Python/ESP32 script for MQTT publishing (real/simulated data)
- Create MQTT topics (battery/voltage, battery/temp, battery/health)
- Set up **Mosquitto Broker (Local)** or **AWS IoT Core / ThingsBoard**
- Store sensor logs in database (Firebase / MongoDB)

📊 **Output:**
→ JSON data packets transmitted to the cloud in real time.

Phase 3 – Cloud + IoT Integration (Week 6–8)

📘 **Deliverables:**

- Functional MQTT data stream connected to cloud
- Real-time data visualization on Node-RED / ThingsBoard dashboard
- Setup OPC UA endpoint for interoperability

📦 **Tasks:**

- Create MQTT → Cloud → Database flow
- Implement simple visual plots (SoH vs time, temperature vs cycles)
- Enable OPC UA server connection to allow Unity to fetch real-time updates
- Implement alert triggers (LED, buzzer, notification) when thresholds cross

📊 **Output:**
→ End-to-end IoT connectivity and visualization.

Phase 4 – AI/ML Layer (Week 9–11)

📘 **Deliverables:**

- Train ML models for:
  - **RUL prediction (LSTM/GRU)**
  - **Failure classification (RandomForest/XGBoost)**
- Create **public model** (general battery health) and **user-specific model** (fine-tuned)
- Implement **Explainability layer (SHAP/LIME)** for interpretability

📦 **Tasks:**

- Preprocess dataset (normalize SoH, SoC, etc.)
- Train & evaluate models using sklearn/Keras
- Save trained models (model_public.pkl, model_user.pkl)

- Create a Python Flask/Node.js API to serve predictions via REST/MQTT

📊 **Output:**
→ AI-powered predictive maintenance engine.

Phase 5 – Unity 3D Digital Twin + Visualization (Week 12–14)

📘 **Deliverables:**

- Develop Unity 3D model of EV battery pack
- Map real-time sensor data → 3D model (battery color = health)
- Create virtual indicators (charging, temperature rise, etc.)
- Sync Unity dashboard with cloud via MQTT or REST API

📦 **Tasks:**

- Import 3D EV battery model in Unity
- Connect Unity to MQTT broker
- Create animation triggers for events (e.g., red flash = overheating)
- Add "Maintenance Recommendation" pop-ups inside dashboard

📊 **Output:**
→ Interactive digital twin visualization.

Phase 6 – Web Dashboard + User Personalization (Week 15–17)

📘 **Deliverables:**

- Create a responsive web app (Flask/Streamlit/React)
- Display:
  - Battery parameters (temp, SoC, SoH)
  - RUL predictions
  - Recommendations (reduce load, schedule service)
- Integrate **public model vs personalized model results**
- Enable user login (Firebase Auth)

📦 **Tasks:**

- Fetch predictions from cloud API
- Show graphs (Matplotlib / Plotly)
- Add "User Profile" → adjust predictions using personal driving history
- Add SHAP visualizations for transparency

📊 **Output:**
→ Fully functional dashboard + recommendation system.

Phase 7 – Final Integration & Testing (Week 18–20)

📘 **Deliverables:**

- Combine all layers:
    - Sensor Node → MQTT → Cloud → AI → Unity + Web UI → Actuators
- Test with multiple simulated users
- Document performance metrics (accuracy, latency, RUL error)
- Prepare project report and demo video

📦 **Tasks:**

- Optimize communication between Unity and Cloud
- Finalize alerts, graphs, and predictive accuracy
- Conduct team-wise knowledge sharing (everyone understands full system)

📊 **Output:**
→ Working digital twin system for EV battery health.

Phase 8 – Final Evaluation (Nov–Dec 2025)

📘 **Deliverables:**

- Live demo (hardware prototype + Unity Twin + Cloud Dashboard)
- Complete report + PPT + Viva
- Include:
    - Screenshots of dashboards
    - Block diagrams

- Model results (confusion matrix, RUL graph)
- Source code references

📊 **Output:**
→ End-to-end Digital Twin-based Predictive Maintenance prototype.

🧱 **Tools & Technologies Summary**

Category	Tool / Framework
Hardware	ESP32 / simulated sensor node
IoT Protocols	MQTT, OPC UA
Cloud	AWS IoT Core / ThingsBoard / Firebase
Database	MongoDB / Firebase Realtime DB
AI/ML	LSTM, GRU, RandomForest, XGBoost, SHAP, LIME
Frontend	Unity 3D (3D Twin), Streamlit / Flask (Dashboard)
Networking	Wi-Fi, MQTT Broker (Mosquitto)
Programming	Python, C++ (ESP32), JavaScript, Unity C#