

Python Programming Project

Project Title: Battery Life & Charging Time Predictor

Course: Python Programming

Student Name: Mousam Mishra

Registration No: 25BCE10773

Institute: VIT Bhopal

INTRODUCTION

This project is designed to predict remaining battery usage time, charging duration, and battery health based on real-time usage analysis. Instead of only displaying percentage, it calculates drain rate, time since last full charge, battery capacity, and charger watt rating. The system works for both mobile phones and laptops and provides practical and accurate predictions.

PROBLEM STATEMENT

Users cannot accurately estimate remaining battery life or charging time based on percentage alone. Devices show inaccurate predictions and do not consider usage patterns or charger specifications. Battery health details are also difficult to access without technical tools. A real-time prediction model is required.

FUNCTIONAL REQUIREMENTS

- Input current battery percentage
- Input hours since last full charge
- Input battery capacity (mAh)
- Input charger watt rating (W)
- Select device type: Mobile / Laptop
- Output remaining usage time (hours)
- Output charging time required (hours)
- Output battery health (%)

NON-FUNCTIONAL REQUIREMENTS

- User-friendly interface
- Accurate calculations
- Fast response time
- Compatible across devices
- Realistic and reliable output

SYSTEM ARCHITECTURE

Input Layer → Processing Layer → Output Layer

Inputs: Battery %, Time since full charge, mAh, Watt rating, Device type

Processing: Drain rate calculation, Charging time, Battery health

Output: Usage time, Charging time, Health, Status indication

DESIGN DIAGRAMS

Use Case Diagram: User → Battery

Prediction System → Output

Workflow Diagram: Start → Input values → Calculate → Display Results → End

Sequence Diagram: User → Input → Engine → Output Module → Display

Component Diagram: Input module, Calculation module, Output module

ER Diagram: Not applicable (no database usage)

DESIGN DECISIONS & RATIONALE

- Chosen Python because it is simple and powerful
- Used time-based drain rate for realistic estimation
- Added Mobile vs Laptop logic for accuracy
- Simplified UI by removing multiple option choices

IMPLEMENTATION DETAILS

- Implemented using Python with `input()` function and mathematical formulas
- Calculates drain rate from time and battery percentage
- Charging time estimated using watt-to-mA conversion
- Battery health estimated through full-cycle approximation

SCREENSHOTS / RESULTS

```
=====
      BATTERY LIFE, CHARGING & HEALTH
=====

Is your device Mobile or Laptop? mobile
Enter current battery percentage (0-100): 60
How many hours since your device was fully charged? 3
Enter battery capacity (mAh): 3870
Enter charger watt rating (W): 20

Estimated usage time remaining: 4.50 hours
Status: Good Battery Backup for Mobile

Estimated Battery Health: 100.00%
Battery Health: Excellent

Estimated charging time to full: 0.49 hours
```

```
=====
      BATTERY LIFE, CHARGING & HEALTH
=====

Is your device Mobile or Laptop? laptop
Enter current battery percentage (0-100): 46
How many hours since your device was fully charged? 11
Enter battery capacity (mAh): 7000
Enter charger watt rating (W): 65

Estimated usage time remaining: 9.37 hours
Status: Good for Laptop

Estimated Battery Health: 100.00%
Battery Health: Excellent

Estimated charging time to full: 0.37 hours
```

TESTING APPROACH

- Tested with multiple battery percentages (10–90%)
- Tested with different charger watt values (10W–120W)
- Device type comparison for accuracy
- Boundary testing for 0% and 100% cases

CHALLENGES FACED

- Predicting accurate health without historical calibration
- Setting a realistic efficiency factor
- Structuring real-time model instead of theoretical
- Output formatting and logical structuring

LEARNINGS & KEY TAKEAWAYS

- Learned real-life battery behavior principles
- Improved Python logic and programming skills
- Understood power consumption modeling
- Learnt how hardware and software interact

FUTURE ENHANCEMENTS

- Mobile app using Kivy / Flutter
- GUI interface with charts
- Historical tracking and graph-based analytics
- Automatic sensor reading without manual input

REFERENCES

- Python Documentation
- Battery degradation research articles
- Online technical resources and device testing observations