EV Charging Usage Tracker App – Product Requirements Document

Overview

Product Summary: This iOS mobile app helps electric vehicle (EV) owners track their electricity usage and costs for personal charging sessions at home (specifically in a shared residential building). The user charges their EV using the building's main electricity supply and needs a simple way to log each charging session's meter readings, calculate how many kWh were used, and determine the cost to reimburse the building. The app focuses on **usability, simplicity, and automation** to minimize user effort and errors.

Context & Problem: In a shared housing scenario, an EV owner's charger is connected to the building's main meter. The building's electricity bill covers the EV charging, so the EV owner must track their usage and pay back the cost. Without a dedicated sub-meter or smart charger app, the user must manually read the main meter before and after charging, do math, and keep records. This process can be tedious and error-prone. The EV owner needs a **quick, reliable, and transparent** way to record each charge and calculate the cost, with proof (e.g. a photo of the meter) for accountability.

Solution: The EV Charging Usage Tracker app streamlines this process by providing:

- **Quick Session Logging:** Automatically recalling the last meter reading, letting the user input the new reading, and calculating the difference in kWh and cost with one tap.
- **Record Keeping:** Storing each session's details (timestamp, readings, energy used, cost, and an optional photo proof) in a history log.
- **Payment Tracking:** Allowing the user to mark sessions as paid/unpaid to keep track of what they have reimbursed.
- **Analytics:** Visualizing electricity usage and costs over time (daily, weekly, monthly) so the user can spot trends and manage their consumption.
- **Configurable Rate:** Using a customizable electricity price (default **0.6402 ILS/kWh**) to calculate costs, so the user can update the rate if the utility price changes.

Target User: A single EV owner in a multi-unit residence (e.g. apartment or condo) with a personal charger on the building's power. **Scope is limited to a single user and single**

vehicle, with all data stored locally on the device (no login or multi-user support is required). This keeps the app focused and simple.

Goals & Objectives:

- **Accuracy:** Ensure each session's kWh usage and cost are calculated correctly, reducing billing disputes.

- **Ease of Use:** Minimize manual data entry through automation (prefill last reading, etc.) and provide a clean, intuitive interface. The user should be able to log a session in seconds.

- **Accountability:** Provide features like photo proof and timestamps to build trust with building management or roommates when sharing the data.

- **Insight:** Give the user insight into their charging habits and expenses over time with simple graphs and summaries.

- **No Hassle:** Avoid complex setups – no need for internet connectivity, accounts, or external devices. The app should work offline and save data on the device reliably.

Features

The app includes the following core features and functionalities:

- **Quick Session Entry & Calculation:** When starting a new charging session entry, the app automatically **prefills the previous meter reading** (the last recorded value) to save time and reduce errors. The user inputs the **new meter reading** after charging. By tapping a **"Calculate"** button, the app computes the **energy used (kWh)** for that session (new reading minus previous reading) and then calculates the **total cost** by multiplying the kWh by the electricity rate. For example, if the difference is 20 kWh and the rate is 0.6402 ILS/kWh, the cost will be 12.804 ILS. The calculated kWh and cost are displayed immediately so the user can review them before saving. *(Input validation:* the app will prevent or warn if the new reading is lower than the previous reading or if the input is non-numeric, catching common errors).

- **Photo Proof Attachment:** The user can attach a **photo of the meter** reading as proof. On the New Session screen, an **"Add Photo"** button lets the user snap a picture of the meter (using the camera) or choose an existing photo. This photo serves as evidence of the meter value and is saved with the session. Attaching a photo is optional but encouraged for transparency. The app will store this image along with the session data, and the image can be viewed later in the session details.

- **Save Session Record:** After entering the new reading (and optional photo) and calculating the results, the user saves the session. Saving will **record a timestamp** (date and time of entry) and store all relevant details: previous reading, new reading, **kWh used**, **total cost**, and the photo and/or any notes. The previous reading field for the *next* session will automatically update to this session's new reading. This automation ensures the next time the user opens the app, the last known reading is ready to go, making sequential logging seamless. (All session data is stored locally on the device; it persists between app launches. No network connection is needed for core functionality.)

- **Session History & Payment Tracking (History Tab):** The app provides a **History tab** showing a list of all past charging sessions in chronological order (most recent first). Each entry in the list shows a brief summary: date/time, kWh used, cost, and an indicator for payment status (paid or unpaid). The user can manage entries with intuitive swipe gestures:

  - **Mark as Paid:** Swiping left on a session reveals an option to mark it as "Paid." This action flags the session (e.g. with a green check mark or changes its status icon) indicating the user has settled the cost for that session with the building. Paid sessions could appear visually distinct (grayed out or labeled as paid) to differentiate from unpaid.

  - **Mark as Unpaid:** Swiping right on a session can mark it as "Unpaid" (for example, if the user needs to revert a paid status or indicate a session still needs payment). This allows correction if a session was marked paid by mistake or if the user wants to mark it as outstanding again. Newly saved sessions default to "unpaid" until the user marks them otherwise.

  - These swipe actions use standard iOS patterns for list management to ensure the interactions feel familiar.

- **Session Detail View:** Tapping on any session in the History list opens a **detailed view** of that session. The detail screen displays all information for the selected charging session:

  - **Meter Readings:** previous and new meter readings (e.g. "Previous: 10500, New: 10560 kWh"), and the calculated **difference (60 kWh)** and **cost** for that session.

  - **Timestamp:** the date and time when the session was recorded.

  - **Payment Status:** an indicator or toggle showing if the session is marked paid or unpaid (with the ability to change it here as well, e.g. a toggle switch or button).

  - **Photo Proof:** the photo taken for that session is displayed (thumbnail that can be tapped to view full-screen). This lets the user or another viewer verify the meter reading.

- **Comments/Notes:** a text field or section for **user comments**. The user can add or edit a note (for example, “Charged from 20% to 80%” or “Paid John on 5/10”). The comments are saved with the session for future reference. This is useful for adding context to sessions.

- **Share Option:** On this detail screen, a **Share** button allows the user to share the session details via WhatsApp or other apps. Tapping share will open the iOS share sheet with a pre-composed message including the session date, kWh, cost, and perhaps the photo. For example, the user can quickly send a WhatsApp message to a building manager: “Charging session on Jan 10: 60 kWh used, 38.41 ILS total. See attached meter photo.” This makes it easy to report and settle charges. (The share feature leverages the device’s share functionality, so it could also be used for email, etc., but WhatsApp is a primary use-case).

- **Analytics Dashboard (Analytics Tab):** The app includes an **Analytics tab** that provides visualizations of the user’s charging history data. This section helps the user understand their usage patterns and costs over time:

  - **Energy Usage Graphs:** Charts display **electricity usage (kWh)** over time – for example, a bar graph per day or line graph over weeks/months. The user can toggle between time scales such as **daily**, **weekly**, and **monthly** views. For instance, a weekly view might show total kWh each week for the last few months, while a daily view could show the last 30 days of usage.

  - **Cost Graphs:** Similarly, graphs show **total cost (ILS)** over time for the same intervals. This could be layered on the same chart or as a separate toggle (e.g. a segmented control to switch between “Usage” and “Cost” graphs).

  - **Trends & Summaries:** Key summary statistics are displayed alongside the graphs. For example: “Total kWh used this month” vs last month, “Total cost this month” vs last, or average cost per session. Visual cues (like up/down arrows or percentage changes) can indicate trends (e.g. “Your usage this week is 10% higher than last week”). This gives the user insight into whether their charging habits are increasing or if any savings occur over timeapps.apple.com. The focus is on clear, easy-to-read charts – perhaps using simple colors and labels since this is a personal tracker, not an overly complex analytics tool.

  - The analytics should update automatically as new sessions are added. If the user hasn’t logged many sessions yet, the Analytics tab can show a friendly message or sample graphic to demonstrate what will be shown once data is available.

- **Settings (Configurable Rate & Preferences):** A settings section allows the user to customize a few key parameters:
    - **Electricity Rate per kWh:** The user can set or edit the price of electricity (defaulted to **0.6402 ILS per kWh** on first install). This value is used for all cost calculations. The user can update it if the electricity tariff changes or if they move somewhere with a different rate. The UI for this might be a simple number input (with numeric keypad) or a slider with fine adjustment. The app should store the rate and use the latest value for new sessions. (Each session's cost is calculated at the time of logging and stored, so historical session costs remain at the rate that was in effect when they were recorded).
    - **Currency and Units:** By default, currency is ILS (Israeli Shekel) and unit is kWh. Since the app is initially for a specific use case, we assume these are fixed. However, it could use the device's locale to set currency symbol automatically, or allow changing currency in settings if needed (e.g. for users in other regions in the future). For now, this might not be exposed if out of scope.
    - **Other Settings:** Minimal additional settings to keep the app simple. Possibly an "About" section or help link, and toggle for light/dark mode (or simply follow iOS system theme by default). No login or user profile settings needed (single-user scope). Data backup settings might be considered if needed (see Optional Enhancements for ideas like iCloud backup).

All the above features are designed to work together to provide a smooth experience. The app uses standard iOS design patterns (tab navigation, lists, forms, share sheet, etc.) to ensure familiarity. **Automation and simplicity are emphasized** – for example, auto-prefilling the last reading and auto-calculating the cost to minimize manual math. Similar energy-tracking apps have shown that providing graphs and timely reminders can help users stay on top of usageapps.apple.com, so our app includes an Analytics view and focuses on easy logging to encourage consistent use.

User Flow

Below is a breakdown of the typical user flows for key tasks in the app, illustrating how a user will interact with the features step by step.

1. Recording a New Charging Session

(This flow occurs after the user finishes an EV charging session and wants to log it.)
Step 1: The user opens the app, which by default opens to the **New Session screen** (or a prominent "New Session" button). The app automatically displays the **Last Recorded Meter**

Reading (e.g. “Previous reading: 10,500 kWh”) in a read-only field, so the user knows the baseline.

Step 2: The user reads the current value on the physical electricity meter (connected to the charger) and enters this number into the **“New Meter Reading”** field. A numeric keypad is used for input to make entering large numbers easy and to prevent invalid characters.

Step 3: (Optional) The user taps **“Add Photo”** and uses their iPhone camera to take a clear photo of the meter showing the reading. The photo thumbnail appears on the entry form as an attachment. If the user prefers not to add a photo, they can skip this step.

Step 4: The user taps the **“Calculate”** button. The app instantly calculates the **kWh used** for this session by subtracting the previous reading from the new reading. It also multiplies by the set rate to compute the **cost**. For example, if previous was 10500 and new is 10560, it shows “Energy Used: 60 kWh” and “Cost: 38.41 ILS” (assuming 0.6402 rate). These results are displayed on screen, giving the user immediate feedback. (If the user made a mistake like entering a number smaller than the previous reading, an error message is shown instead of a result, prompting them to correct the input).

Step 5: The user taps **“Save Session”** (or if the UI is streamlined, tapping Calculate might also act to save automatically, but typically a confirmation is good). Upon saving, the session is stored in history. A brief confirmation (toast or alert) might inform “Session saved!” The app updates the internal “last reading” value to the new reading just entered. The user’s inputs are then cleared or the form resets, ready for the next time. At this point, the user can exit the app or navigate to other sections. The new session will now appear at the top of the History list.

2. Viewing and Managing Session History

(This flow happens when the user wants to review past charges or mark payments.)
Step 1: The user navigates to the **History tab** from the bottom navigation bar. They see a **list of past sessions**, each showing key info (date, kWh, cost, paid/unpaid status). For example, an entry might read: “Jan 10, 2025 – 60 kWh – 38.41 ILS – [Unpaid].” Unpaid sessions might have a red dot indicator, while paid ones have a green check, for quick scanning.

Step 2: To mark a session as paid after reimbursing the building, the user **swipes left** on that session’s cell. A **“Mark as Paid”** action appears (for instance, a button with a checkmark icon). The user taps it, and the session’s status instantly updates to paid (the entry could now show [Paid] and perhaps turn green or strike-through the amount, etc., design to indicate it’s settled). There’s no need to navigate away; the change is done in-line. If the user accidentally marks the wrong item, they can swipe right to mark it unpaid again (the **swipe right** gesture could reveal an “Mark as Unpaid” action for paid items). This two-way swipe functionality ensures flexibility in updating status.

Step 3: To view full details of a particular session (e.g., to see the photo or add a note), the user **taps** on that session entry. The app navigates to the **Session Detail View** for that record. Here the user can see everything: “Jan 10, 2025 at 8:15 PM”, previous reading 10500, new reading 10560, usage 60 kWh, cost 38.41 ILS, status Paid/Unpaid, the photo of the meter (which they can tap to enlarge and verify the reading), and any existing comments.

Step 4: On the detail screen, the user can perform a few actions:

- **Add/Edit Comment:** The user taps a text field or an “Edit” button in the comments section to add a note. For instance, they type “Paid cash on Jan 11 to building fund” or “Charged vehicle from 20% to 80%.” When they finish, they hit Save (or it auto-saves the note). The comment is now associated with that session.

- **Toggle Paid Status:** If they prefer, the user could also mark as paid/unpaid from here (for example, a toggle switch “Mark this session as Paid” that they can flip). This would reflect back in the History list.

- **Share Session:** The user taps the **Share** icon on the detail view. The iOS share sheet opens with options (WhatsApp, iMessage, Email, etc.). The app prepares a message like: *“EV Charging Session – Jan 10, 2025: Used 60 kWh costing 38.41 ILS. Meter readings 10500→10560. (Photo attached.)”* The user selects WhatsApp (for instance), chooses the contact (e.g., building manager), and sends the message. This allows quick sharing of proof and cost for reimbursement. After sharing or reviewing, the user closes the detail and goes back to History or another tab.

Step 5: The user can scroll through History to view other sessions. The list may be long over time; if needed, we might allow filters (e.g., show only unpaid), but by default it’s a full chronological log. The user now has an easy way to see which sessions are unpaid (outstanding to pay) and the total of those (e.g., they might manually sum or in a future enhancement, see a total unpaid amount at top of history).

3. Analyzing Usage and Cost Trends

(This flow is about the user checking their charging statistics.)
Step 1: The user taps the **Analytics tab** in the app. The screen opens with a default view, say “Monthly Usage.” Here the user sees a chart – for example, a bar chart of the *past 6 months* of EV charging consumption. Each bar might show total kWh per month. Above or below the chart, the app might summarize: “Total in March: 180 kWh, Cost: 115.24 ILS; Total in April: 150 kWh, Cost: 96.03 ILS,” etc., giving a quick comparison.

Step 2: The user wants to drill down, so they switch the time frame. Perhaps there’s a toggle for **Daily / Weekly / Monthly**. The user selects **Weekly**. The chart transitions (or the user swipes) to show each week’s usage for the last couple of months. They notice, for example, one week spiked higher, indicating maybe a long trip was charged that week. The app might highlight “Week of Jan 8–14: 140 kWh, highest in 3 months” to draw attention to peak usage.

Step 3: The user then switches to **Cost** view (maybe tapping a “Cost” button or legend). Now the chart shows cost in ILS on the Y-axis instead of kWh. The shape/trend is similar (just scaled by the rate), but this directly shows how much money was spent on charging over time. The user sees, for instance, that in the last 3 months they spent ~300 ILS on EV charging.

Step 4: The Analytics screen also provides **summaries** like: “Average session: 50 kWh, 32.01 ILS” or “Current month so far: 120 kWh, 76.82 ILS.” These help the user gauge their usage at a glance. Perhaps a small text might note “Compared to last month, you’ve used 10% more energy.” This gentle insight can motivate the user to be mindful of usage or simply inform them. If the user has a goal to stay under a certain budget, these stats are immediately useful.

Step 5: If the user has relatively few sessions logged, the analytics might be sparse. In such cases, the app can show a friendly message like “Log more sessions to see detailed analytics” or just show whatever data is available (e.g., if only one month of data, just show that). Over time, as the history grows, the analytics becomes more valuable. The user can always come back to this tab to check on their charging habits, especially at the end of the month or year when tallying costs.

4. Updating Settings (Changing the Rate)

(This flow covers adjusting the app settings, notably the electricity rate.)
Step 1: The user accesses **Settings** – either via a dedicated tab or a gear icon (perhaps at the top of the main screen or in a “More” tab). The Settings screen shows the configurable options.

Step 2: The user sees the **Electricity Price per kWh** field displaying “0.6402 ILS” (the default or currently set value). If the local electricity tariff changed (e.g., the utility raised the price or a new rate was agreed with the building), the user taps this field to edit it. A numeric keypad appears. The user updates the value, for example, to 0.6700 ILS/kWh.

Step 3: The user saves or the value auto-saves on change (with perhaps a small confirmation like the new rate being displayed). Going forward, any **new charging session** will use this updated rate for cost calculations. (Historical sessions remain calculated at the old rate; the app does not retroactively change past records – this could be noted in a help tip to avoid confusion).

Step 4: The user might also toggle other preferences in settings if available. For instance, if there’s a setting for “Require photo each session” or notification reminders (in case we add that), those would be managed here. In this basic flow, after changing the rate, the user exits Settings (back to whichever tab they want).

This covers the primary user flows. Throughout all flows, the app prioritizes minimal input and clear feedback. For example, forms are short and pre-filled when possible, calculations happen instantly, and any errors (like invalid meter readings) are explained in plain language. The UI uses familiar patterns (tabs, lists, buttons) so that even a non-technical user can navigate easily.

By following these flows, the user can **consistently log their EV charging usage, monitor their expenses, and ensure they pay the correct amount** to the building on time.

Wireframe Suggestions

Below is an outline of the key screens in the app and suggestions for their layout and UI components. (These are not actual designs, but conceptual wireframes described in words to guide development and design.)

- **New Session Screen:** This is the main screen the user interacts with to log a charge. At the top, show a label like “Previous Reading: **<last_reading>** kWh” so the user knows the starting point. Below it, a **numeric input field** labeled “New Meter Reading” for the user to type the latest meter value. Use a large, clear font for these readings since accuracy is crucial. Next to the input or below it, include the **“Add Photo”** button (a camera icon) – tapping it opens the camera or photo library picker. After the input field, include a prominent **“Calculate”** button (perhaps a large primary button at the bottom of the form). When pressed, the result (kWh used and cost) can appear just below the button or in a popup: e.g., show a summary text “You used X kWh, costing Y ILS.” There could then be a **“Save Session”** button if not auto-saved. The screen’s design should be clean and focused – likely a simple form on a scrollable view, with maybe an app header that says “New Session” and a settings gear icon in a corner. The use of color can highlight the calculate button and result (for example, using the accent color to display the cost). If possible, disabling the Calculate button until a new reading is entered (to enforce input) would be good. This screen benefits from large tap targets and clear labels, as it’s used often possibly in a garage with one hand.

- **History Screen:** A list view showing all past sessions. Use a standard **table view (list)** with each row showing a summary: perhaps **Date** (as the title, e.g., “Jan 10, 2025”), and a subtitle or secondary text with “X kWh – Y ILS – Paid/Unpaid”. Alternatively, the row could be split into multiple columns or sections: left side date, center kWh & cost, right side an icon for status (a checkmark for paid or a pending icon for unpaid). Tapping the row opens details. Implement **swipe actions** on each row: swipe left to reveal a red “Mark Paid” (or green, depending on design preference; typically iOS swipe left is for delete or important actions, we can use it for paid which might be considered a positive action – we could use swipe left for “Delete session” if ever needed, but here we mainly need paid/unpaid toggling). To avoid confusion, maybe use **swipe left for “Delete”** (if we allow deleting a logged session in case of mistakes) and **swipe right for “Mark as Paid/Unpaid”** with a checkmark icon. This mimics how Mail app might use swipe actions. If delete is not needed or is hidden behind an edit mode, then swipe left can be for mark paid. In any case, ensure the swipe gestures are clearly indicated with colored backgrounds and icons. The history list can be grouped by year or month if there are many entries (e.g., headers for January 2025, February 2025, etc.) – an optional enhancement for organization. At the top of the History screen, a small **total summary** could be shown (optional): e.g., “Unpaid sessions: 3 (Total 120 ILS)” so the user immediately sees how much they owe. This screen likely has a title "History" at the top. Scrolling behavior is standard; if the list is empty (no sessions yet), show a placeholder message like "No sessions logged yet" and maybe an arrow pointing to the New Session tab.

- **Session Detail Screen:** This screen shows the full details for one session record. Layout suggestion: at the top, display the **date and time** prominently (e.g., "January 10, 2025 – 8:15 PM"). Below that, perhaps in a content section, list the readings: "Previous: 10500 kWh, New: 10560 kWh" and then **"Energy Used: 60 kWh"** and **"Cost: 38.41 ILS"** in bold so they stand out. Next, display the **Photo** if one was saved: show it as a medium-sized image thumbnail. If the user taps it, it can open full screen using the standard image viewer. Under the photo or alongside it, indicate the **Payment status** – e.g., a label or colored pill saying "UNPAID" (in red) or "PAID" (in green). Possibly include a small button or switch to toggle this status right there (e.g., an edit icon next to it or tapping the label toggles status with a confirmation). Then, a **Comments** area: if a comment exists, show it in a text box (read-only mode until an edit button is pressed). If no comment yet, show a faint "Add a note…" prompt that the user can tap to start typing. There should be an **Edit** or **Add Comment** button that brings up the keyboard for that field. Finally, at the bottom or top navigation, include a **Share** button (an action/export icon). Tapping share triggers the iOS share sheet with the prepared content. The screen might be scrollable if content is long (especially with a photo and possibly long comments). Navigation will have a back button to return to History. This detail view should prioritize clarity – key numbers (kWh, cost) are easily readable at a glance, and the photo is visible. Use icons or labels for paid/unpaid to reduce text clutter (e.g., a checkmark icon with "Paid" label). All interactive elements like share or edit comment should be clearly visible.

- **Analytics Screen:** The Analytics tab content can be arranged with a top control for selecting the timeframe and metric, and a main area for the chart. For example, at the top, have a segmented control or two tabs: one for **Usage** and one for **Cost** (so the user can switch the metric graphed). Just below, another segmented control or dropdown for **Daily / Weekly / Monthly** range view. Alternatively, a single graph could have a toggle or interactive pinch zoom to switch granularity, but simpler to start with explicit controls. Default might be monthly usage. So the UI might have something like: `[Usage | Cost]` toggle and `[Day | Week | Month]` toggle. The main panel shows a chart – possibly use a simple line graph if showing a trend over time, or bar chart if showing discrete periods. Under the chart (or overlaying it) put labels for key values (e.g., each bar or point might have a small label of value). Underneath, have a **summary section**: this could be a few statistic cards or lines of text. For example:
    - "Total this month: 150 kWh, 96.03 ILS"
    - "Last month: 180 kWh, 115.24 ILS"

    - “Difference: -16% usage, -16% cost” (in this example usage went down).

    - Or “Average per session: 50 kWh (32.01 ILS)”. These give context to the chart. The design should use colors consistently (e.g., maybe the app’s theme color for the graph line or bars). Each axis on the chart should be labeled (Time on X, kWh or ILS on Y). Ensure the chart is responsive to data; if there are only a few data points, they are spaced appropriately. Possibly include a legend or title on the chart like “Monthly Energy Consumption”. The Analytics screen should scroll if content exceeds one screen (e.g., if showing both usage and cost charts one above the other, though likely we just show one at a time to keep it simple). If using one chart at a time, maybe below the chart after the stats, show a smaller secondary graph or table – but to maintain simplicity, better to have just one main graph view toggled by user input. The UI should encourage exploration without overwhelming; using the native iOS charts (or a library) that allows simple interaction (maybe tapping a bar to see exact values) could be nice.

- **Settings Screen:** This screen can be fairly minimal. A simple list style (table view grouped settings) is appropriate:

    - One cell or section for **Electricity Rate**: tapping it opens a number pad or a detail screen where the user can input a new rate. We can also display the current rate in the cell (e.g., “Electricity Price per kWh – 0.6402 ILS”). If we want inline editing, we could have the cell itself contain a text field showing 0.6402 that the user can tap and edit in place.

    - Possibly another section for **Preferences**: e.g., a toggle “Use Dark Mode” (or just rely on system theme), or “Enable Reminders” (if we add notifications to remind meter readings – see enhancements).

    - An **About** section: a cell that when tapped shows app version, developer info, maybe a link to a help/FAQ or support email.

    - Since there’s no login, we don’t need profile or account settings. Since data is local, maybe a “Backup Data / Export Data” option could appear here (again, see Optional Enhancements – if implemented, the settings is where an export to CSV or iCloud backup trigger might live).

    - Use standard toggles and input fields here. This screen is likely accessed infrequently, but it should still be simple. A navigation bar title “Settings” and groupings for different types of settings would organize it.

Design considerations: Across all screens, maintain a consistent color scheme and typography. Important figures (like kWh and cost) should use a larger or bold font. Buttons like

“Calculate” or “Save” should use the system default or an accent color to stand out. Use of icons: a camera icon for adding photo, share icon, edit icon for comments, checkmark/x for paid status, etc., will make the interface more intuitive (less text heavy). Animations can be minimal – maybe a brief highlight when a session is saved or a nice transition when flipping between analytics views – but nothing too distracting.

By following these wireframe suggestions, designers can create a prototype that aligns with standard iOS patterns and focuses on the key tasks. The layout is optimized for quick one-handed use (important if the user is in a parking garage holding a phone and perhaps other items). Each screen’s content is kept to what’s necessary, to uphold **simplicity and clarity**. This will ensure the app is not only functional but also pleasant and straightforward to use.

Optional Enhancements

While the core features above cover the primary needs, there are several additional ideas inspired by similar apps in the energy/EV space that could further enhance the user experience or be added in future versions. These are **optional enhancements** beyond the initial scope:

- **OCR for Meter Reading (Automated Data Entry):** Implement Optical Character Recognition to read the meter value from the photo the user takes. This would automate the input of the new meter reading – the user could simply snap a photo of the meter and the app would detect the digits and fill in the reading for them. This feature would save time and reduce input errors. (Many modern apps use image recognition for convenience; here it would leverage iOS’s Vision framework or similar). The user could still adjust or confirm the detected number, but it adds a layer of automation.

- **Reminders & Notifications:** Provide an option for the app to send a **local notification reminder** to the user to log a session or check the meter. For example, if the user typically charges nightly or on a schedule, the app could remind them at a certain time (“Don’t forget to record your EV meter reading!”). Or if a long time (e.g., two weeks) passes without any new session logged, a reminder could ask if they charged their car and forgot to log it. This helps maintain consistent recordsapps.apple.com. The reminders should be configurable (the user can turn them on/off in Settings, and perhaps set the preferred time).

- **Multiple Vehicle or User Profiles:** While the current scope is single-user/single-EV, a future enhancement could allow **multiple profiles** – for instance, if a household has two EVs using the same meter or the user moves between two locations. Each profile could have its own meter readings and history. Similarly, support for **multiple meters** (if the user wanted to track another utility like home electricity usage separately) could be added, akin to how some energy apps let you track several metersapps.apple.comapps.apple.com. This would broaden the app’s utility beyond the single scenario.

- **Data Export and Sharing:** Expand the sharing capabilities by allowing the user to **export all history data** to a file. For example, generate a **CSV or PDF report** of all sessions (or for a selected date range). This would be useful for maintaining personal records or sending a summary to building management on a monthly basis. A one-click "Export to CSV" could compile all sessions with their dates, kWh, costs, paid status, etc., and allow the user to email it to themselves or a property managerev.energyev.energy. A PDF report could even include the photos (as a mini report of charges). This kind of feature is found in some EV charging apps and is great for expense tracking or auditsev.energy.

- **Cloud Sync/Backup:** Implement iCloud sync or backup so that the user's data is safely stored and can be accessed across devices. For example, if the user has an iPhone and an iPad, they could use the app on both and data (history, settings) would stay in sync. Even without multiple devices, an automatic iCloud backup ensures that if the user gets a new phone, their charging history isn't lost. This might require a minimal user sign-in with Apple ID (or just use iCloud by default since the user is on iOS). Security and privacy would have to be considered, but since it's personal data not highly sensitive, this is manageable.

- **Advanced Analytics & Insights:** Enrich the Analytics tab with more insights:

  - **Trends and Forecasting:** The app could project the month's total based on current usage or show a trendline. For example, "You are on track to use ~200 kWh this month, costing ~128 ILS, based on your current rate of usage."

  - **Comparison with Averages:** If data becomes available, compare the user's consumption with averages (perhaps national EV charging averages or typical EV consumption) to provide context. E.g., "Your average daily charging is 5 kWh, compared to an average EV owner at 7 kWh" – this could motivate or inform the user.

  - **Cost Savings vs. Gasoline:** For an EV owner, it's often interesting to know how much they saved by using electricity instead of fuel. An optional setting could allow the user to input their car's efficiency or a comparable gas price, and the app could display "You saved X ILS this month compared to fueling a gasoline car," or "CO_2 saved: Y kg" for environmental impact. This is inspired by features in some EV apps that highlight environmental benefits and cost savings of driving electric.

- **Integration with Smart Hardware:** In the future, integrate with hardware or APIs for more automation:

  - If the building installs a **smart sub-meter or IoT device** for the charger, the app could connect via Bluetooth/WiFi or API to fetch readings automatically, eliminating manual entry entirely. This would turn the app into a real-time monitor.

    - **Integration with vehicle telematics:** Some EVs provide data on how much energy was taken in during a charge. If the user's car or charger has an API, the app might import session data from there (however, since this app's primary method is via meter reading, this is a parallel approach).

    - **Home automation integration (HomeKit):** Possibly log sessions or trigger automations via Apple HomeKit or Siri Shortcuts. For example, a Siri voice command like "Log my charging session" could prompt for the new reading and take a photo hands-free.

- **Improved Payment Tracking:** Expand the paid/unpaid tracking to include more payment-related functionality:

    - Perhaps allow the user to input **when/how they paid** (date or method), not just mark paid. For instance, mark a session paid and attach a note "paid via bank transfer on 5th". Or allow grouping sessions and marking all as paid at once when settling a monthly bill.

    - A **summary of outstanding balance** (total unpaid amount) could be displayed in the History or in a dedicated "Payments" section. The app could then reset or mark a whole month as paid once the user settles with one payment.

    - Possibly generate a simple **invoice** or receipt for the unpaid sessions for the building management, which could be part of the export feature.

- **UI Customization and Themes:** Offer a choice of app theme or color scheme (some users might want a dark theme in the garage at night – though this can be covered by supporting iOS dark mode). The existing "Meter Readings" app, for instance, offers multiple color themesapps.apple.com. We could allow the user to pick an accent color or light/dark mode preference in settings for personalization.

- **Localization:** While the initial use-case is an Israeli user (ILS currency, etc.), planning for localization can be a future enhancement. This means easy translation of the app to other languages and adapting to other currencies/decimal formats. This would broaden the app's appeal to EV owners in other countries who have similar needs (e.g., someone in a condo in the US who wants to track in USD).

- **Tutorial or Onboarding:** An optional onboarding flow for first-time users that teaches them how to use the app (for example, a quick walkthrough illustrating how to read the meter and input the data, how to mark sessions paid, etc.). This can improve initial usability, especially for users unfamiliar with meter reading. This could be a set of screens or tooltips highlighting features on first launch.

Each of these enhancements should be carefully weighed against complexity. The core app is intentionally simple; adding too many features could complicate the user experience. However, features like OCR, reminders, and data export directly add convenience without significantly burdening the user, so they could be high on a future roadmap. Other ideas like cloud sync or advanced analytics would require more development effort but could greatly increase the app's power and appeal to power users.

Prioritization of Enhancements: If we rank these ideas, automations like OCR and reminders might come first (to further minimize user effort). Data export and backup come next for peace of mind and flexibility. More niche features (like fuel savings comparisons or HomeKit integration) might be later as "delighters" for those who want extra insights. By continuously improving with such features, the app can stay very user-friendly for the basic needs, while offering more depth for users who want it.

Conclusion: This Product Requirements Document outlines a comprehensive plan for an iOS app dedicated to tracking EV charging sessions in a home/shared building context. By focusing on core functionalities that ensure accuracy, simplicity, and useful insights, the app will solve a real pain point for EV owners who need to manually track their electricity usage. The proposed features prioritize ease-of-use (with automation like prefilled data and one-tap calculations) and transparency (saving details and photos for each session). The user flows and wireframes ensure that using the app will be intuitive and quick, fitting naturally into the routine of charging an EV. Additionally, the optional enhancements section provides a vision for how the app can evolve, drawing inspiration from industry best practices and user feedback. Overall, this PRD serves as a blueprint to guide the development team in building an app that makes personal EV charging management **effortless and efficient**.