```
ui, feature

### Estimate
M
```

🔹 RUNNING INDEX (update each run; keep ≤ 25 lines)

#1 Set up Clean Architecture Foundation

#2 Implement Multi-Account User Switching

#3 Design Unified Log Detail View

(index truncated when > 25 rows)

🕑 Issue 3 delivered – next run in ~15 min.

markdown

```
### Title
Build Time Range Filter with Presets

### Description
Add a robust time range filtering component with quick-select
presets (e.g. “Today”, “Last 7 Days”, “This Month”) and custom
range support. This will be central to enabling scoped views
across charts, tables, and analysis components.
**Tasks**
- Design UI widget for time range selection with dropdown and
custom picker.
- Create `TimeRange` model with helper methods for common
presets.
- Integrate range selector into chart and table views.
- Store selected time range in Riverpod global state.
- Add visual indication of active filter and allow reset to
default.

### Acceptance Criteria
- [ ] UI allows selecting from at least 5 presets and a custom
range.
- [ ] Charts and data tables update based on selected time
range.
```

```
- [ ] Filter state persists across navigation within the
session.
- [ ] "Reset" button clears filter and returns to full
dataset.

### Labels
ui, filters, charts

### Estimate
M
```

🔹 RUNNING INDEX (update each run; keep ≤ 25 lines)

#1 Set up Clean Architecture Foundation

#2 Implement Multi-Account User Switching

#3 Design Unified Log Detail View

#4 Build Time Range Filter with Presets

(index truncated when > 25 rows)

🕑 Issue 4 delivered – next run in ~15 min.

markdown

```
### Title
Implement Daily and Rolling Usage Charts

### Description
Develop data visualizations that show daily total usage and
rolling averages (e.g., 7-day, 30-day) to help users identify
trends over time. This is essential for insight generation and
habit tracking.
**Tasks**
- Create helper method to group log data by day and calculate
totals.
- Implement rolling average logic using sliding windows over
daily sums.
- Design line chart UI for both raw daily totals and
overlaying rolling trends.
- Add dropdown toggle to switch between view types (daily vs.
```

```
rolling).
- Ensure charts respect current time range filter.

### Acceptance Criteria
- [ ] Daily total chart renders with accurate grouping and
labels.
- [ ] Rolling average chart displays overlays for at least 7-
and 30-day options.
- [ ] Switching between chart modes does not affect selected
time range.
- [ ] Charts are responsive and performant on large datasets.

### Labels
charts, analytics, ui

### Estimate
L
```

🔹 RUNNING INDEX (update each run; keep ≤ 25 lines)

#1 Set up Clean Architecture Foundation

#2 Implement Multi-Account User Switching

#3 Design Unified Log Detail View

#4 Build Time Range Filter with Presets

#5 Implement Daily and Rolling Usage Charts

(index truncated when > 25 rows)

🕑 Issue 5 delivered – next run in ~15 min.

markdown

```
### Title
Sync Logs Between Firestore and Local Isar Cache

### Description
Enable automatic bi-directional sync of log entries between
Firestore and Isar to support offline use and consistent
behavior across devices. Ensure changes in one store propagate
correctly with minimal duplication or conflicts.
```

```
**Tasks**
- Define schema and sync state tracking fields for `LogEntry` in both sources.
- Implement background sync service triggered on startup and periodically.
- Add logic to detect and resolve conflicts (e.g., latest timestamp wins).
- Ensure deletions are properly mirrored between sources.
- Write integration tests for full sync cycle under flaky network conditions.

### Acceptance Criteria
- [ ] Log entries created/edited locally are uploaded to Firestore within 1 minute.
- [ ] Log entries from Firestore appear in local Isar within 1 minute of sync.
- [ ] No duplicate entries appear after multiple edits across sources.
- [ ] Deletions in either store remove corresponding entry in the other.

### Labels
sync, firestore, offline

### Estimate
L
```

🔹 RUNNING INDEX (update each run; keep ≤ 25 lines)

#1 Set up Clean Architecture Foundation

#2 Implement Multi-Account User Switching

#3 Design Unified Log Detail View

#4 Build Time Range Filter with Presets

#5 Implement Daily and Rolling Usage Charts

#6 Sync Logs Between Firestore and Local Isar Cache

(index truncated when > 25 rows)

🕑 Issue 6 delivered – next run in ~15 min.

markdown

```
### Title
Add Table View for Recent Logs

### Description
Provide a table-based interface for browsing recent log
entries in reverse chronological order. This will support
detailed inspection and bulk actions like deletion or export
in future updates.
**Tasks**
- Build table UI with sortable columns: timestamp, duration,
device, method.
- Implement pagination and lazy loading for performance on
large datasets.
- Sync table with active time range and profile filters.
- Integrate row tap to open Log Detail View.
- Add stub checkboxes for future bulk actions.

### Acceptance Criteria
- [ ] Table displays logs in descending timestamp order by
default.
- [ ] Columns are sortable and preserve order during
pagination.
- [ ] Tapping a row opens the log detail view for that entry.
- [ ] Performance acceptable for 10,000+ entries (≤ 200ms
render time).

### Labels
ui, logs, table

### Estimate
M
```

🔹 RUNNING INDEX (update each run; keep ≤ 25 lines)

#1 Set up Clean Architecture Foundation

#2 Implement Multi-Account User Switching

#3 Design Unified Log Detail View

#4 Build Time Range Filter with Presets

#5 Implement Daily and Rolling Usage Charts

#6 Sync Logs Between Firestore and Local Isar Cache

#7 Add Table View for Recent Logs

(index truncated when > 25 rows)

🕑 Issue 7 delivered – next run in ~15 min.

markdown

```
### Title
Implement Stats Overview Dashboard

### Description
Create a high-level stats dashboard summarizing recent usage
trends, totals, and insights. This will serve as a quick-look
entry point and motivate users by visualizing progress.
**Tasks**
- Design cards for “Total Logs This Week”, “Longest Streak”,
“Most Active Hour”, etc.
- Integrate query layer to compute metrics in real time or
from cached aggregates.
- Layout grid dashboard for mobile and tablet breakpoints.
- Add skeleton loading state and error fallback messages.
- Ensure responsiveness to time range and user profile
changes.

### Acceptance Criteria
- [ ] Dashboard shows at least 4 real-time usage metrics.
- [ ] Metrics update on time range/profile switch in under
300ms.
- [ ] Dashboard layout adapts to various screen sizes.
- [ ] Loading and error states provide helpful feedback.

### Labels
dashboard, analytics, ui

### Estimate
M
```

🔹 RUNNING INDEX (update each run; keep ≤ 25 lines)

#1 Set up Clean Architecture Foundation

#2 Implement Multi-Account User Switching

#3 Design Unified Log Detail View

#4 Build Time Range Filter with Presets

#5 Implement Daily and Rolling Usage Charts

#6 Sync Logs Between Firestore and Local Isar Cache

#7 Add Table View for Recent Logs

#8 Implement Stats Overview Dashboard

(index truncated when > 25 rows)

🕑 Issue 8 delivered – next run in ~15 min.

markdown

```
### Title
Add Log Entry Submission UI with Validation

### Description
Design and implement the user interface for submitting new log entries manually. This will allow users to backfill missed sessions or test features during development. Input validation and user feedback are critical to prevent errors.
**Tasks**
- Create form UI with inputs for duration, timestamp, method, and notes.
- Add dropdowns or chips for selecting method and device type.
- Implement validation rules: duration ≥ 1 sec, timestamp not in future.
- Integrate form with local Isar cache and queue for sync to Firestore.
- Display success/failure snackbar messages on submit.

### Acceptance Criteria
- [ ] Form prevents invalid submissions with helpful inline errors.
- [ ] Logs added via UI appear immediately in local
```

```
tables/charts.
- [ ] Successfully submitted entries show “queued” tag until
synced.
- [ ] UX and visual design match existing app style.

### Labels
ui, data-entry, validation

### Estimate
M
```

🔹 RUNNING INDEX (update each run; keep ≤ 25 lines)

#1 Set up Clean Architecture Foundation

#2 Implement Multi-Account User Switching

#3 Design Unified Log Detail View

#4 Build Time Range Filter with Presets

#5 Implement Daily and Rolling Usage Charts

#6 Sync Logs Between Firestore and Local Isar Cache

#7 Add Table View for Recent Logs

#8 Implement Stats Overview Dashboard

#9 Add Log Entry Submission UI with Validation

(index truncated when > 25 rows)

🕑 Issue 9 delivered – next run in ~15 min.

markdown

```
### Title
Implement Session Length Histogram Chart

### Description
Develop a histogram visualization to analyze the distribution
of session durations. This helps users understand their
typical log lengths and detect outliers or patterns in usage
behavior.
**Tasks**
- Define histogram binning logic (e.g., 0–30s, 31s–1m, etc.).
```