**Testing To Do**

**Code & Component Testing (in Cursor)**

**What you can do in Cursor:**

* **Linting & Static Analysis:**
* Run linters (ESLint, Flake8, etc.) to catch syntax and style issues.
* **Unit Tests:**
* Run any existing unit tests for your React components, Flask routes, or utility functions.
* Write new tests for the RefreshForecastButton, API service, and error boundaries.
* **Code Review:**
* Review code for logic errors, missing edge cases, and accessibility attributes.
* **Git Operations:**
* Commit, branch, and revert code as needed.

**What you cannot do in Cursor:**

* See live UI/UX, animations, or real API responses.
* Test actual network requests or browser-specific behaviors.

**2. UI/UX & Integration Testing (in Browser/Other Software)**

**What you need to do in a browser:**

* **Visual Inspection:**
* Open your dashboard in Chrome, Firefox, Safari, and on mobile devices.
* Check layout, spacing, and responsiveness.
* **Interaction Testing:**
* Click the refresh button and verify loading spinner, success animation, and error messages.
* Change cash balance, incomes, expenses, and important dates—ensure forecast auto-refreshes.
* Test retry functionality after a failed refresh.
* **Accessibility:**
* Use Tab/Shift+Tab to navigate.
* Use screen reader tools (VoiceOver, NVDA, etc.) to check ARIA announcements.
* **Network/Performance:**
* Use DevTools to simulate slow networks and test timeout handling.
* Check for console errors or warnings.

**What you can do with other tools:**

* **Automated UI Tests:**
* Use Cypress, Playwright, or Selenium for end-to-end tests.
* **API Testing:**
* Use Postman or curl to test your Flask /api/refresh-forecast endpoint directly.

**3. Backend Testing (in Cursor & Terminal/Other Tools)**

**In Cursor:**

* Run Python/Flask unit tests for your API endpoints.
* Review logs and error handling in your Flask app.

**In Terminal/Other Tools:**

* Use Postman or curl to send test requests to your Flask API.
* Check your database to ensure forecast data updates as expected.

**4. Suggested Testing Steps**

1. **Lint and run all unit tests in Cursor.**
2. **Start your Flask backend and React frontend.**
3. **Open the dashboard in a browser:**

* Check that meters and calendar display correctly.
* Click the refresh button and verify all states (loading, success, error, retry).
* Change input data and confirm auto-refresh works (with debounce).
* Test on different screen sizes (resize window, use device emulation).
* Use keyboard navigation and screen reader to check accessibility.
* Simulate network errors (offline mode, slow 3G in DevTools).

1. **Use Postman/curl to test /api/refresh-forecast with valid/invalid tokens and data.**
2. **Check browser and server logs for errors or warnings.**
3. **If using Git, commit your working state.**

**Summary Table**

| **Task/Check** | **Cursor (IDE)** | **Browser/Other** |
| --- | --- | --- |
| Linting, static analysis | ✔️ |  |
| Unit tests (JS/Python) | ✔️ |  |
| Code review | ✔️ |  |
| Git operations | ✔️ |  |
| Visual layout, responsiveness |  | ✔️ |
| Button/UX interaction |  | ✔️ |
| API/network error simulation |  | ✔️ |
| Accessibility (keyboard/ARIA) |  | ✔️ |
| API endpoint testing (Postman) |  | ✔️ |
| End-to-end UI tests (Cypress) |  | ✔️ |

**Let me know if you want a detailed checklist for any specific area, or help writing automated tests!**