**Phase 5: Apex Programming (Developer)**

**➣Classes & Objects**

Custom Apex classes were used to separate business logic from triggers:

**• BinService.cls** → validates IoT sensor data and updates bin status.

**• RecyclingService.cls** → calculates and assigns recycling reward points.

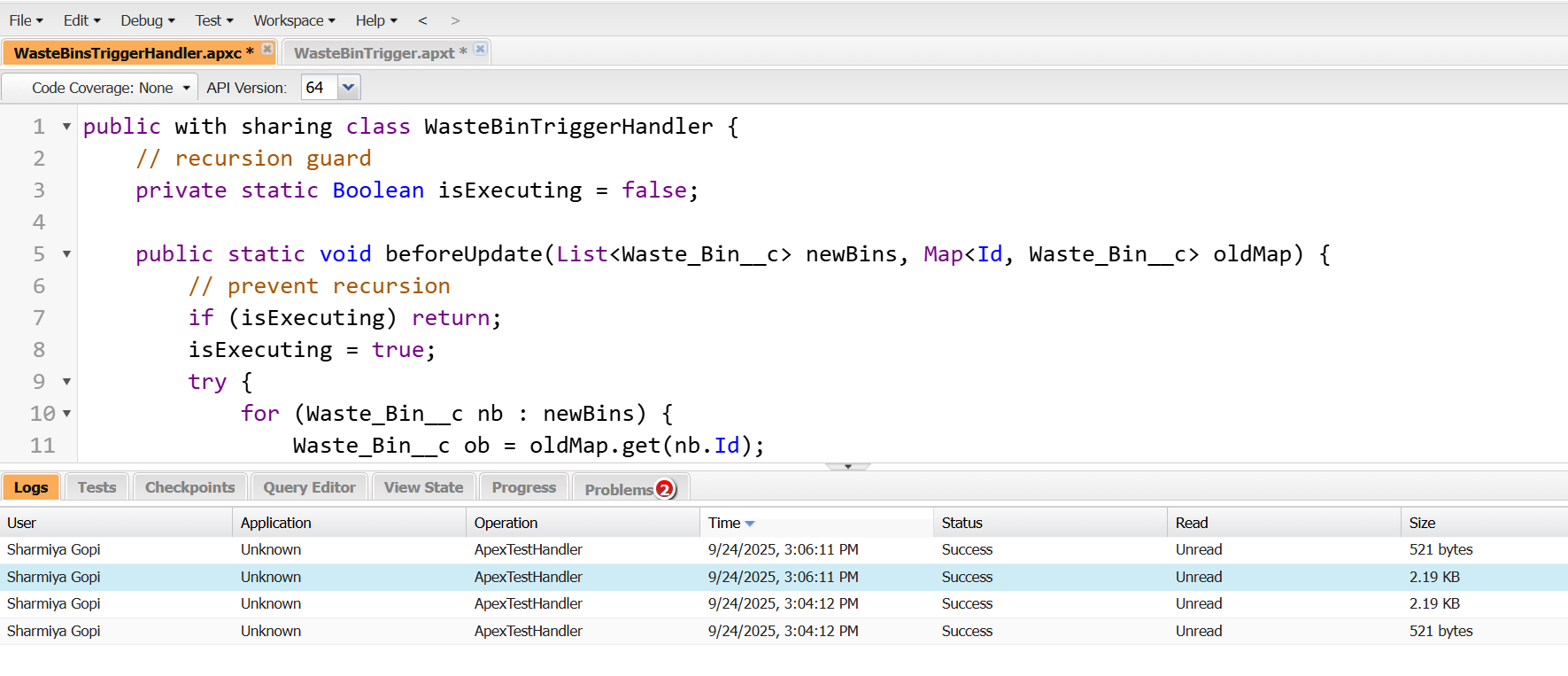
**• TruckAssignmentService.cls** → manages truck capacity and assignment.

This modular design supported reusability and easier maintenance.

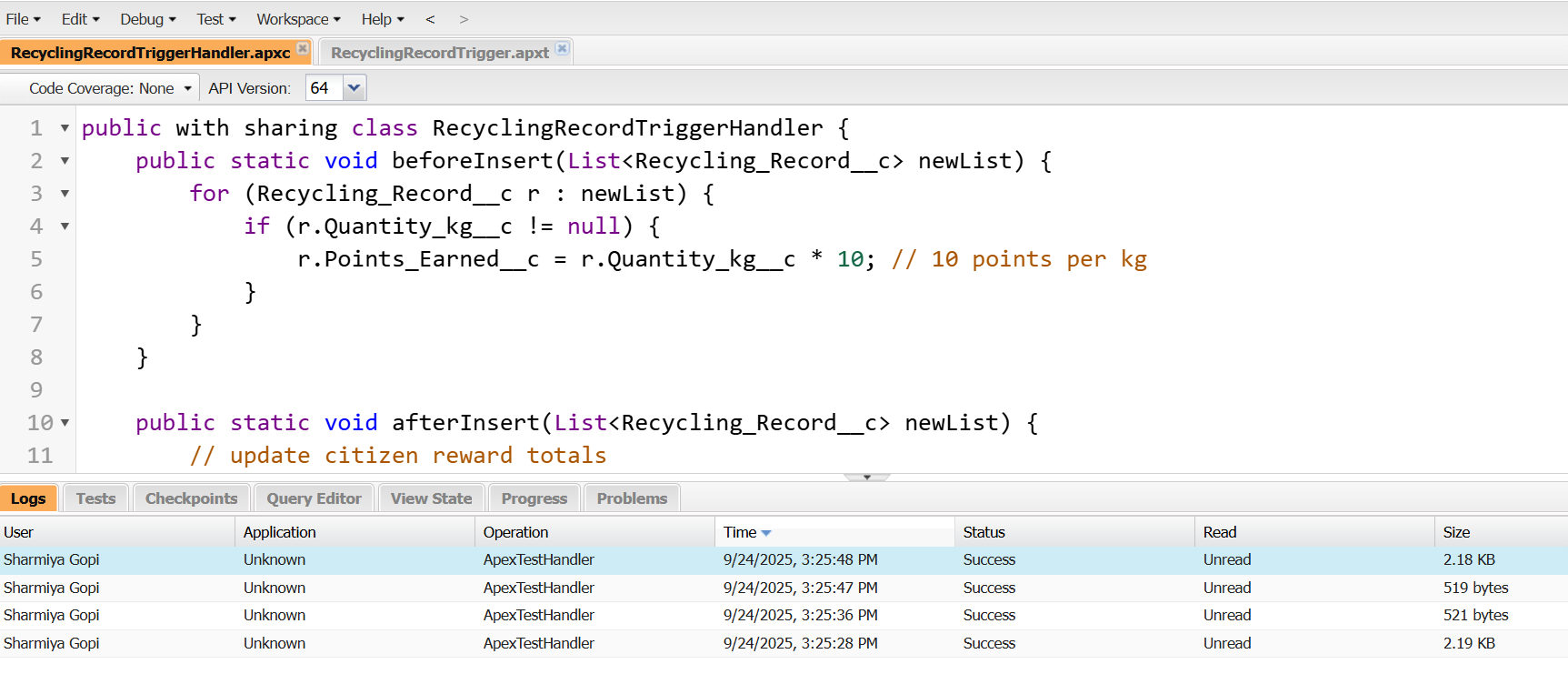
**➣Apex Triggers (before/after insert/update/delete)**

Key triggers enforced project rules:

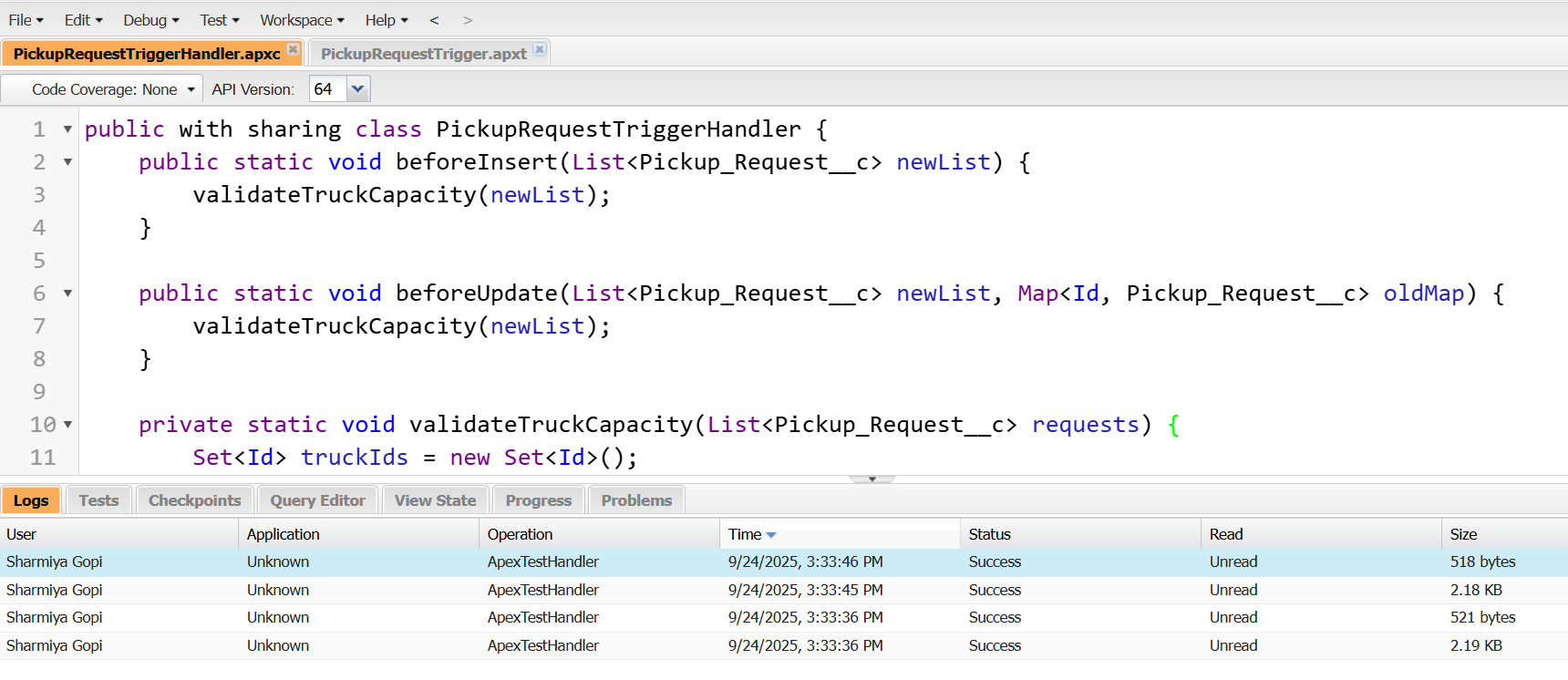
**• Bin Trigger (before update)** → when IoT fill level ≥ 80%, set bin status to Pending Collection.



**• Recycling Record Trigger (before insert/after insert)** → calculates points earned and updates citizen’s cumulative reward points.



**• Pickup Request Trigger (before insert)** → validates truck assignment, ensuring capacity isn’t exceeded.



**➣Trigger Design Pattern**

**•** A single trigger per object was maintained, delegating logic to handler classes.

**•** This avoided recursion, improved testability, and ensured bulk-safe operations.

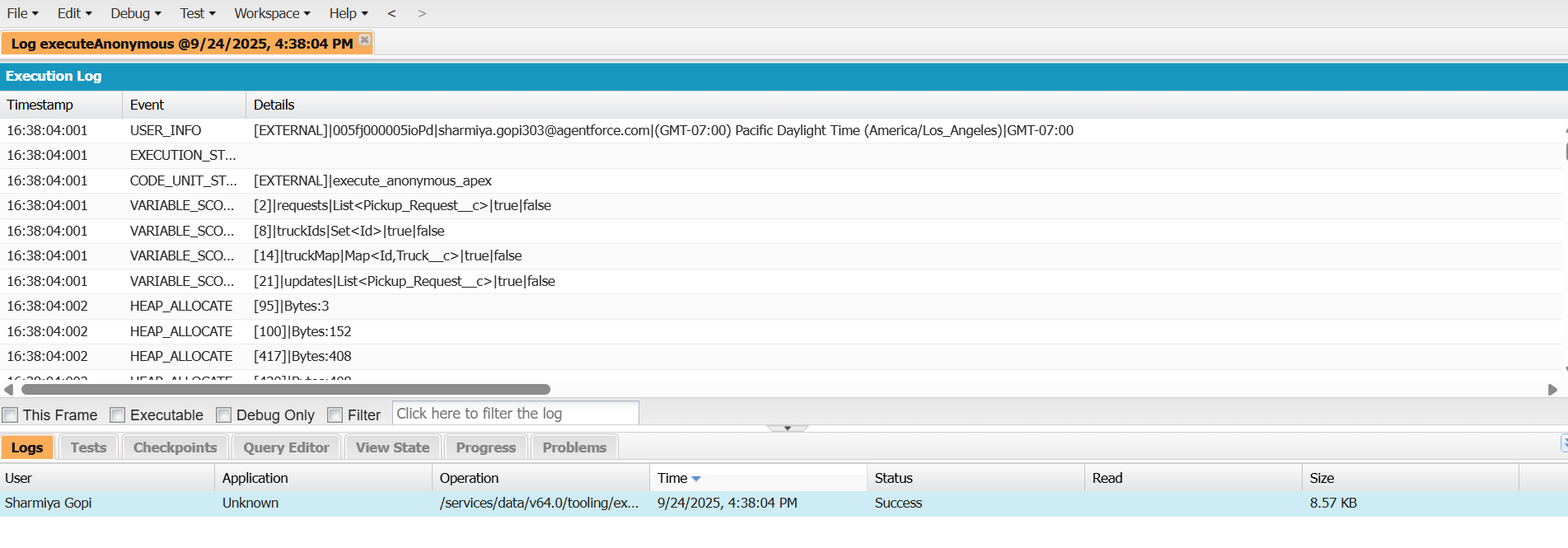
**➣SOQL & Collections (List, Set, Map)**

**• SOQL:** Used to query active trucks, bin details, and citizen reward balances.

**• List:** Managed bulk bin updates from IoT sensors.

**• Set:** Prevented duplicate truck assignments for the same pickup request.

**• Map:** Truck Id → capacity map for quick lookup during assignment.



**➣Control Statements**

**•** If/Else enforced capacity rules and bin thresholds.

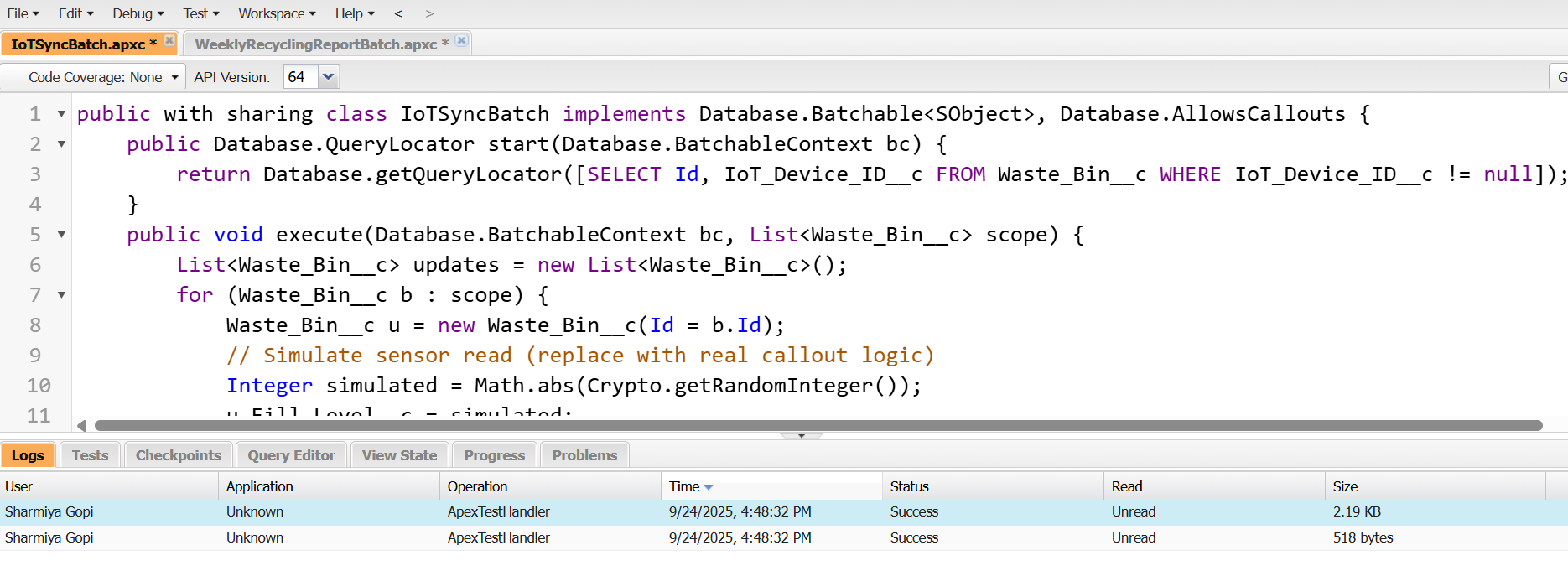
**•** For Loops iterated through IoT bins and recycling submissions.

**•** Switch (where supported) used for recycling material type handling.

**➣Batch Apex**

**• IoT Sync Batch:** Processes bin IoT data nightly and updates statuses.

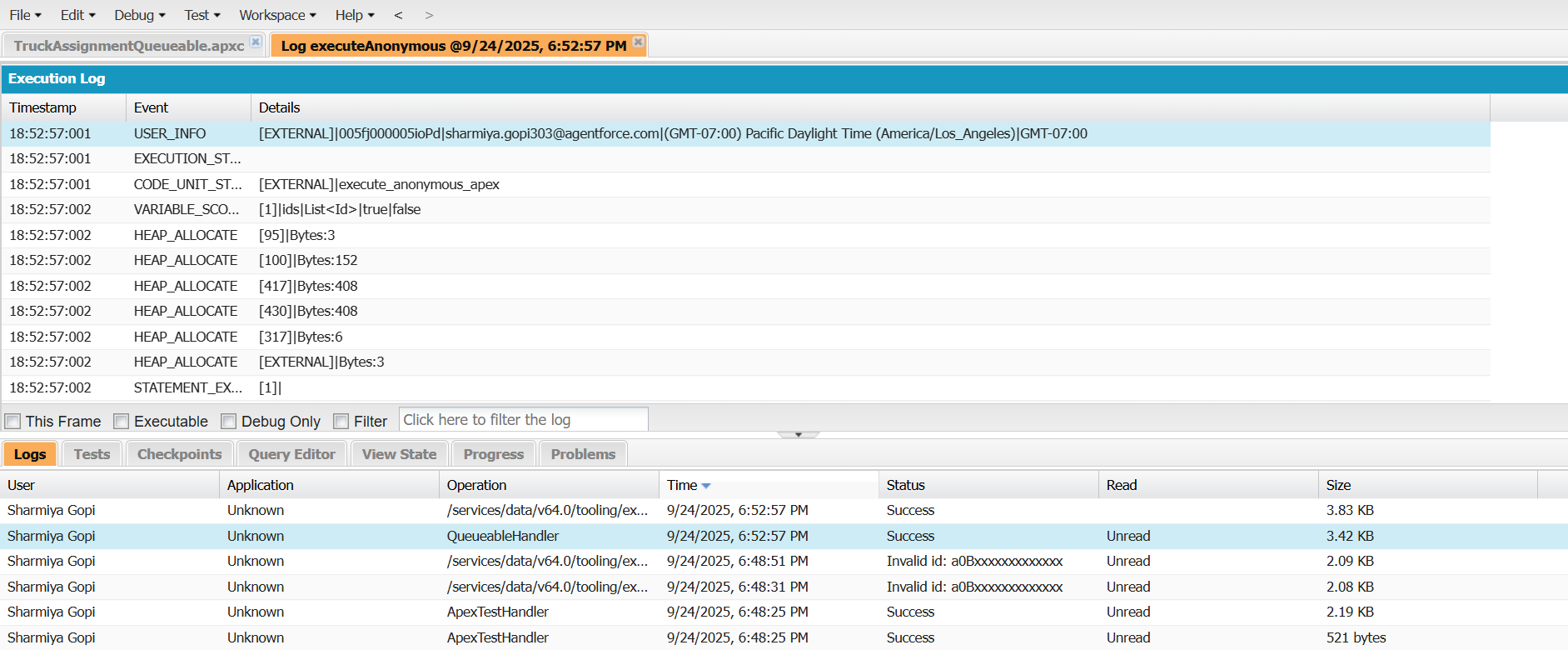
**• Weekly Recycling Report Batch:** Aggregates points, CO₂ savings, and materials recycled for dashboards.



**➣Queueable Apex**

Queueables were used for truck assignment, ensuring async execution without blocking user transactions.

Example: Assign nearest available truck after a new pickup request is logged.



**➣Future Methods**

Lightweight async logic (e.g., logging events, updating “Last Recycle Date”) was implemented using @future.

**➣Exception Handling**

**•** Custom exception class “WasteMgmtException” created for consistent error messages.

**•** Try-catch blocks prevented failures during bulk truck assignment or IoT sync.

**•** Errors logged into a custom object Error\_Log\_\_c for admin review.

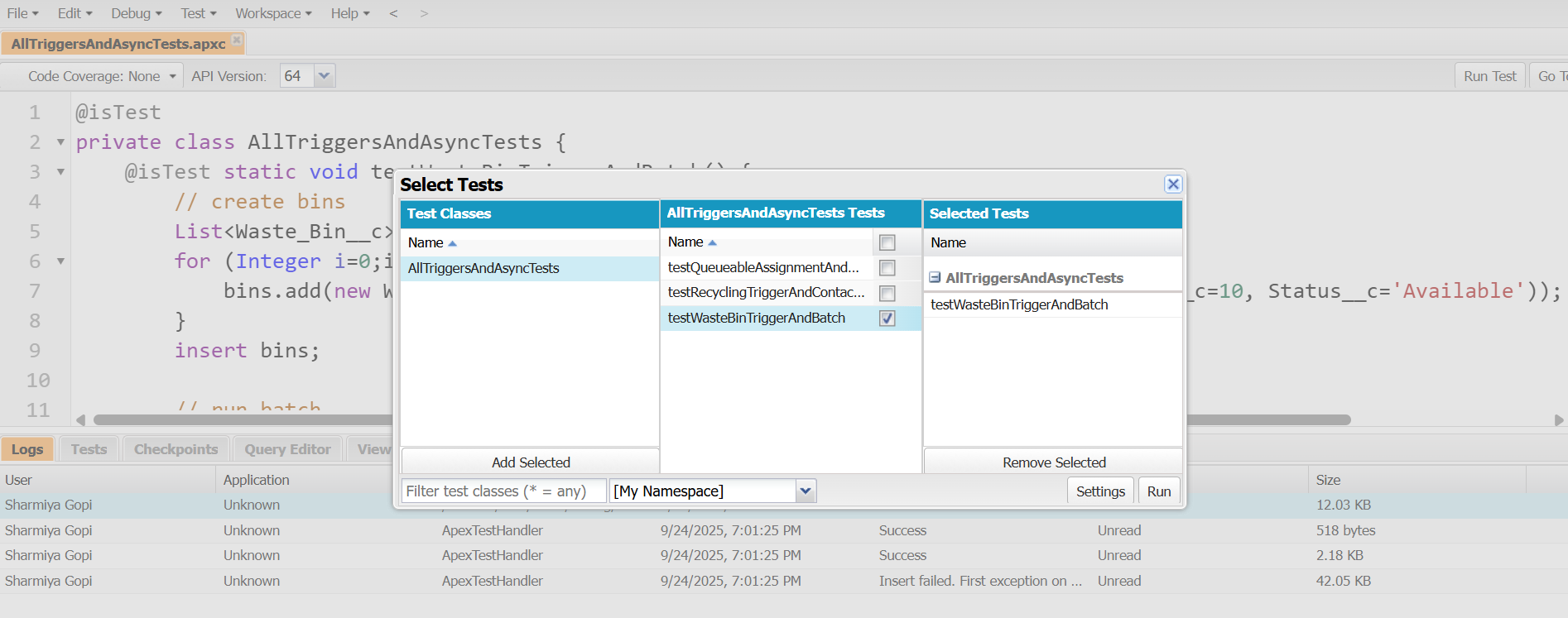
**➣Test Classes**

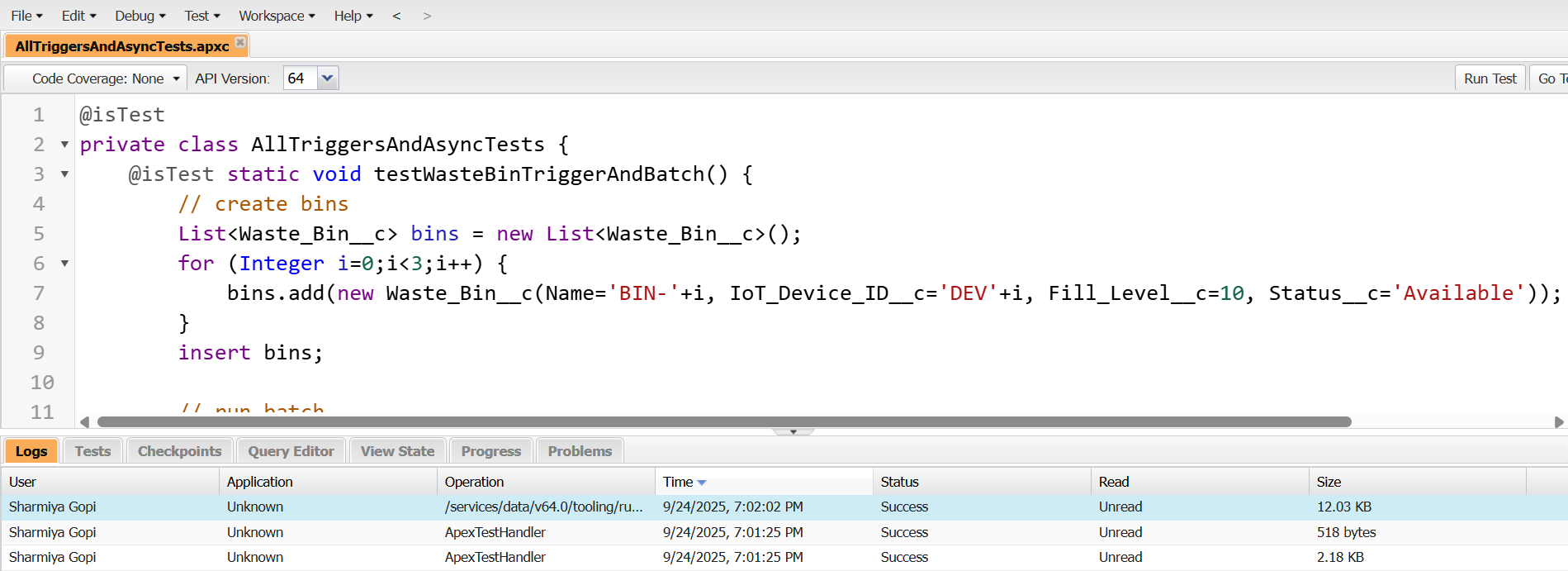
Test classes were written for each trigger and service, covering:

**•** Bin fill level → Pending Collection status.

**•** Reward point calculation and citizen updates.

**•** Truck assignment with over-capacity prevention.





**➣Asynchronous Processing**

The project used multiple async mechanisms:

**• Future** → lightweight updates.

**• Queueable** → truck assignment jobs.

**• Batch** → IoT and recycling reports.

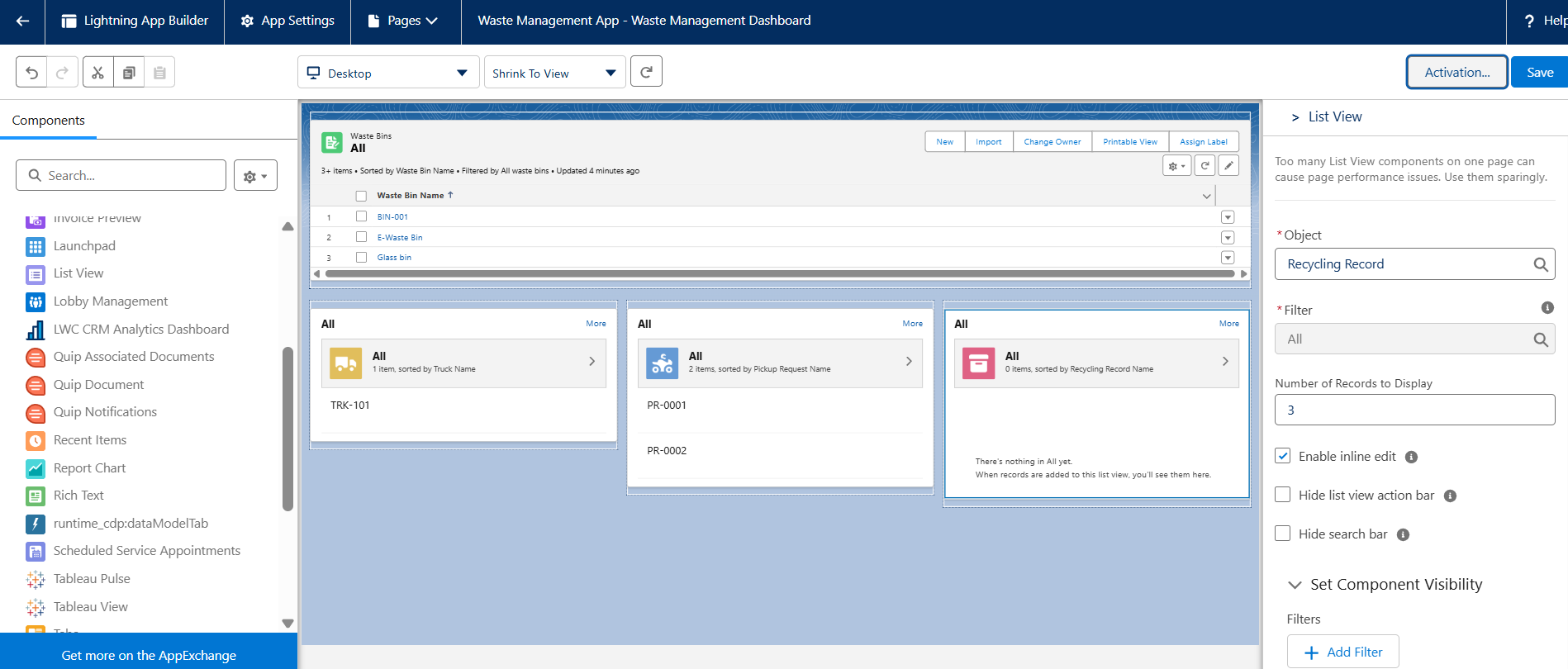
**• Scheduled** → recurring sustainability dashboards.

**Phase 6: User Interface Development**

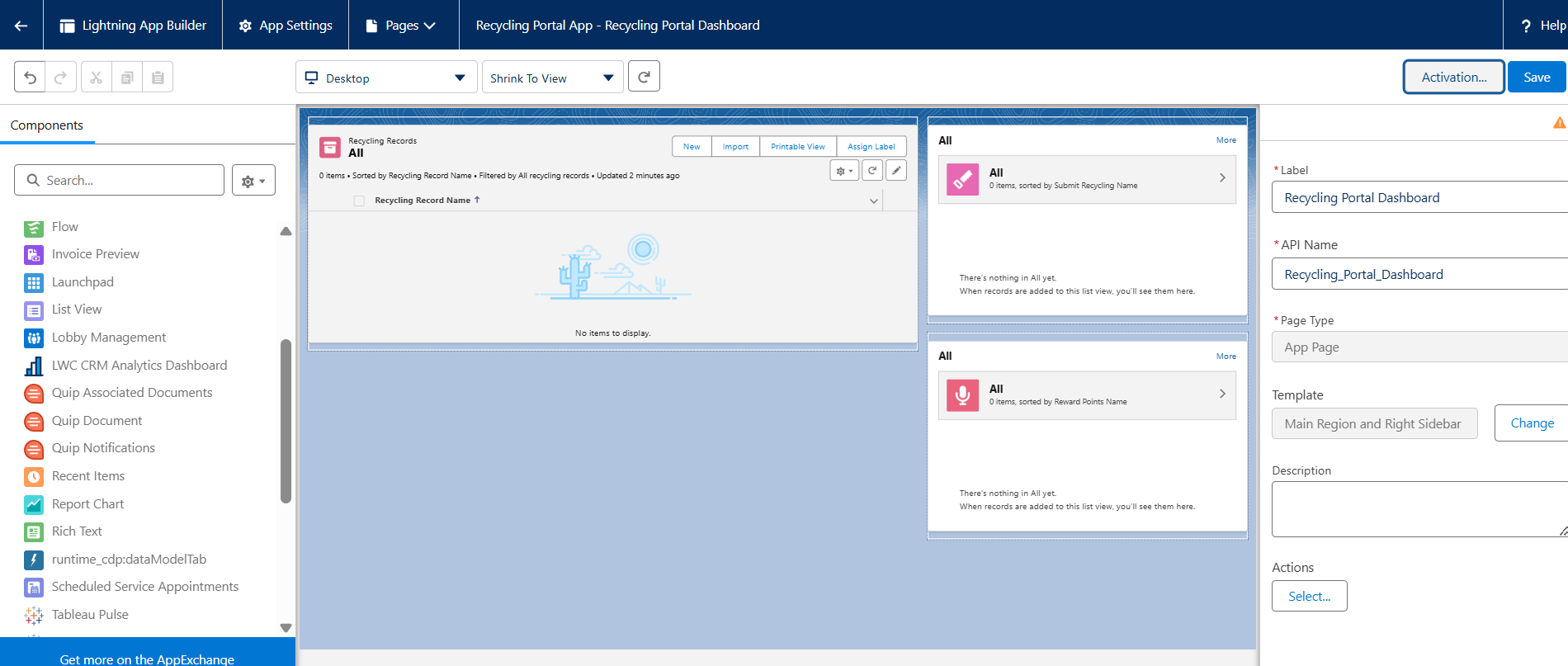
**➣Lightning App Builder**

Custom apps were built to provide role-based access:

**• Waste Management App** → for city officials and supervisors, showing bins, trucks, and pickup requests.



**• Recycling Portal App** → for citizens via Experience Cloud, allowing submission of recycling records and tracking reward points.



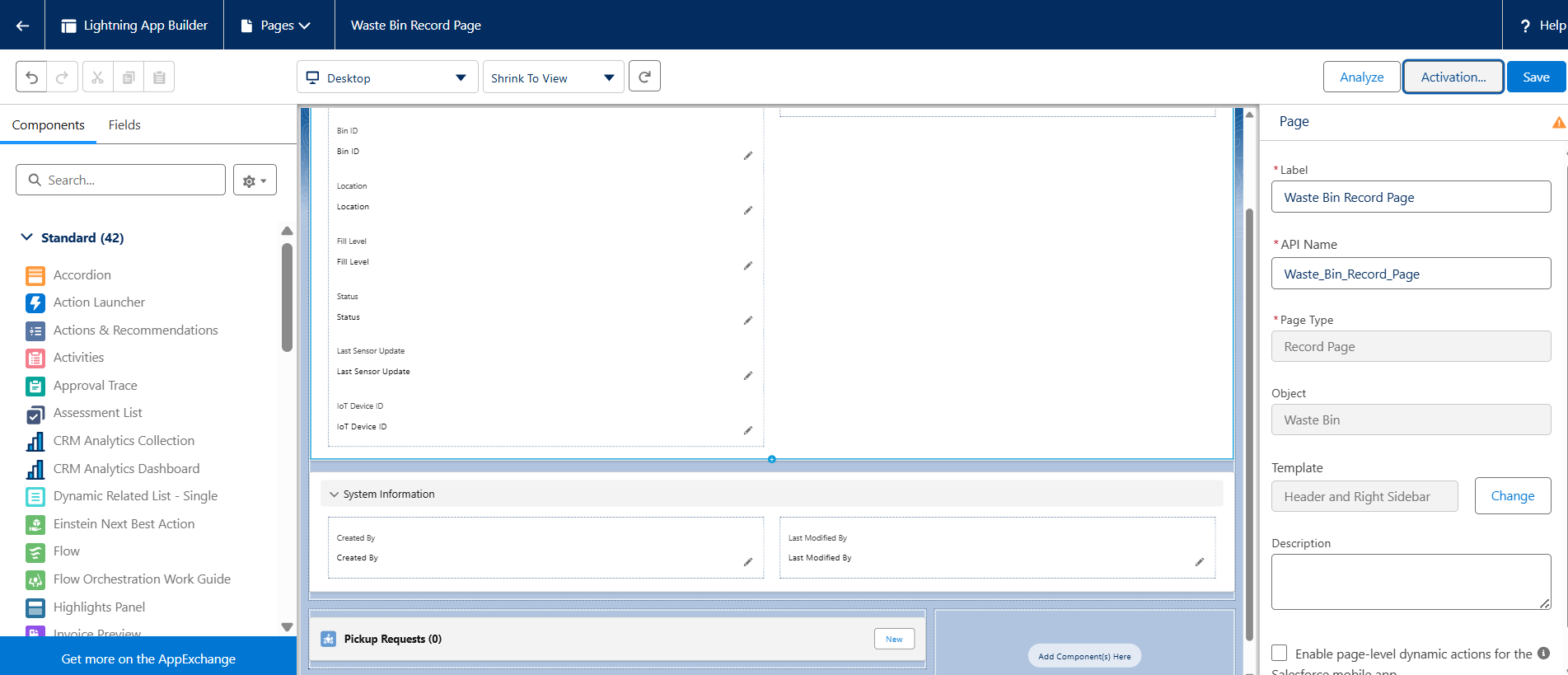
**➣Record Pages**

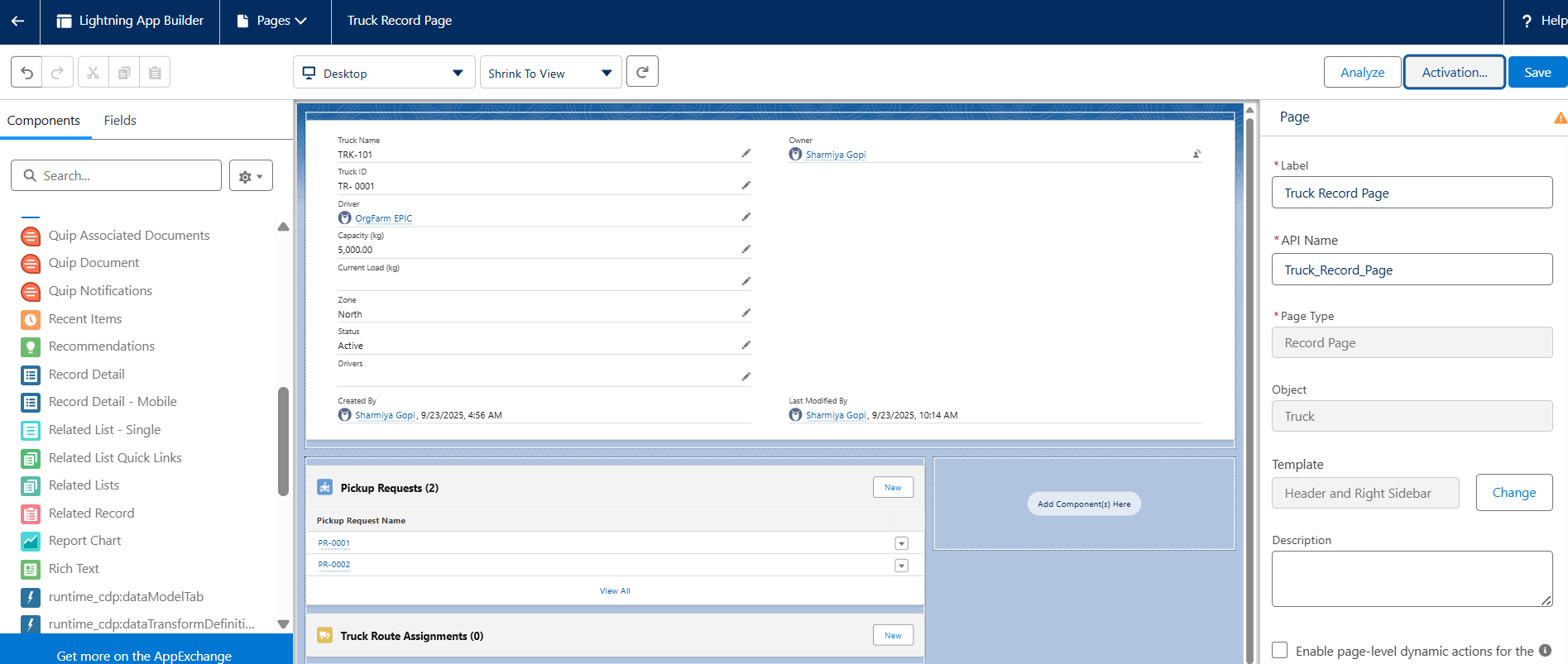
Customized record pages were designed for better usability:

**•** Waste Bin Record Page → shows IoT fill level, status, and related pickup requests.

**•** Truck Record Page → displays driver, current load, assigned routes, and related pickup requests.

**•** Citizen (Contact) Record Page → highlights reward points, recycling history, and related dashboards.





**➣Tabs**

Custom object tabs were created for quick navigation:

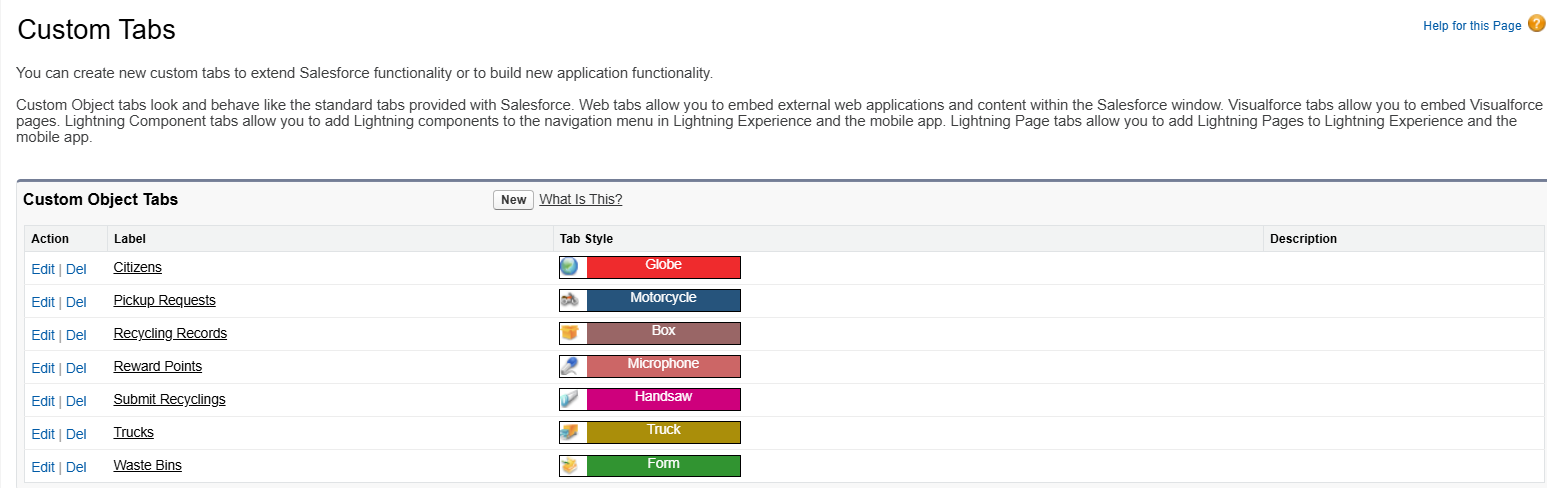
**•** Waste Bins

**•** Trucks

**•** Pickup Requests

**•** Recycling Records

These tabs allowed both admins and users to quickly access and manage records.



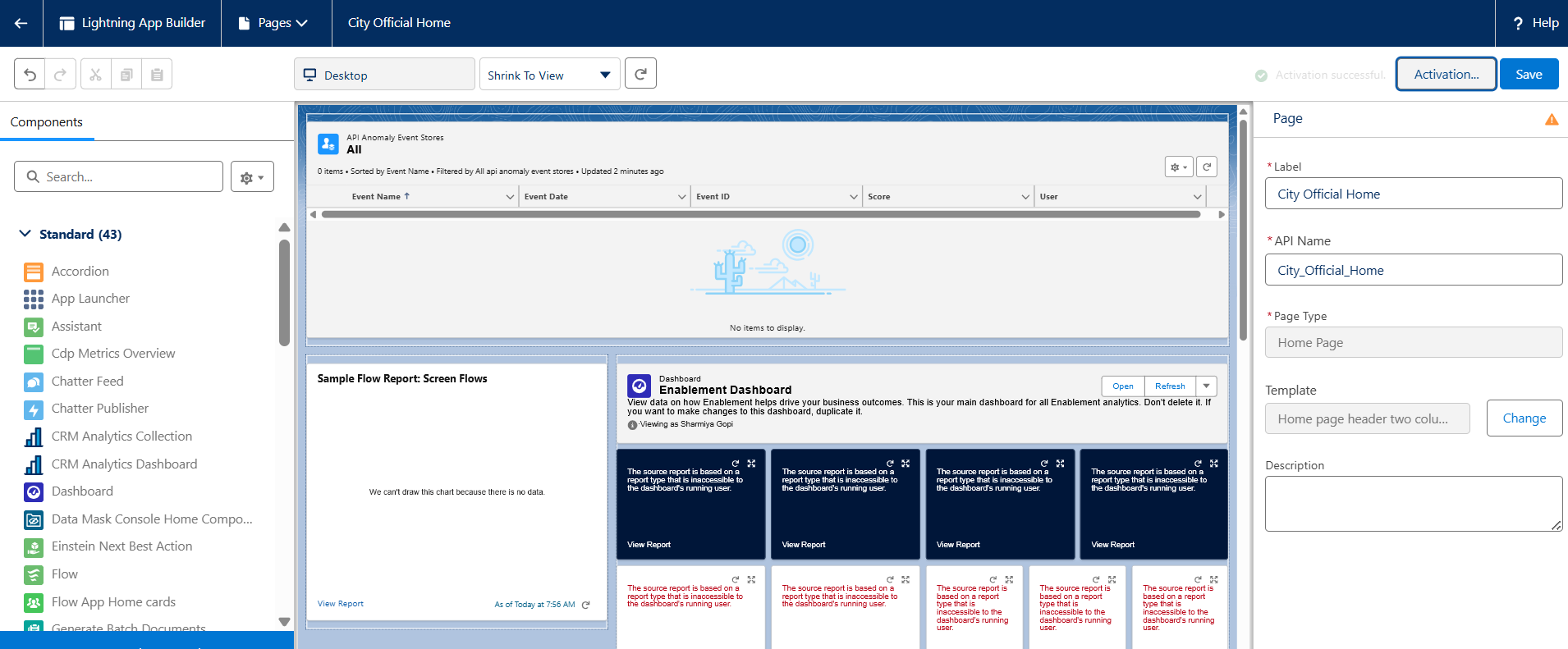
**➣Home Page Layouts**

Different home pages were created for each role:

**•** City Officials → dashboards with waste collection trends, recycling rates, and CO₂ savings.

**•** Truck Drivers → list of assigned pickup requests with navigation support.

**•** Citizens → reward summary, recycling impact, and service requests.



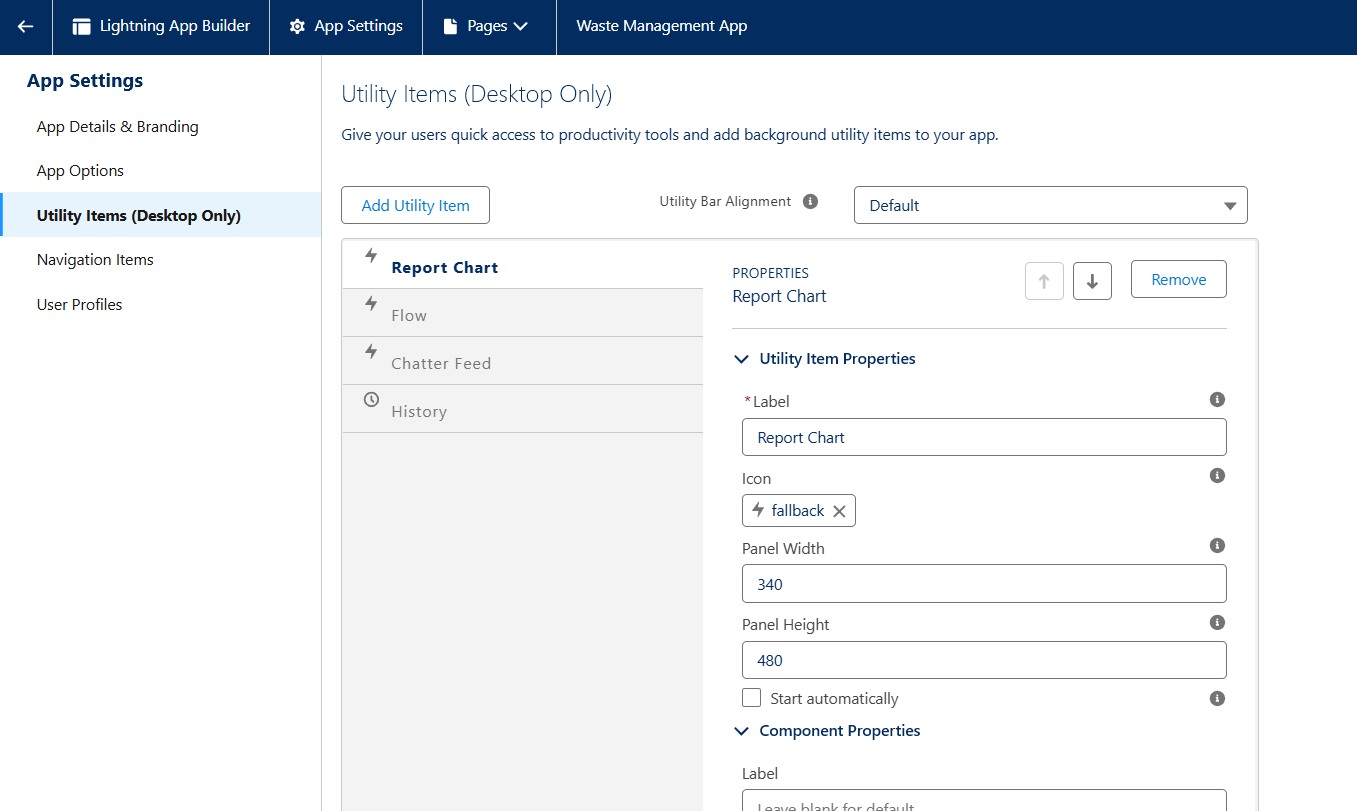
**➣Utility Bar**

The Utility Bar provided always-available tools in the console app:

**•** Notifications → real-time updates on assignments.

**•** Quick Recycling Entry → small form for logging recycling events.

**•** Reports/Charts → instant access to key metrics.



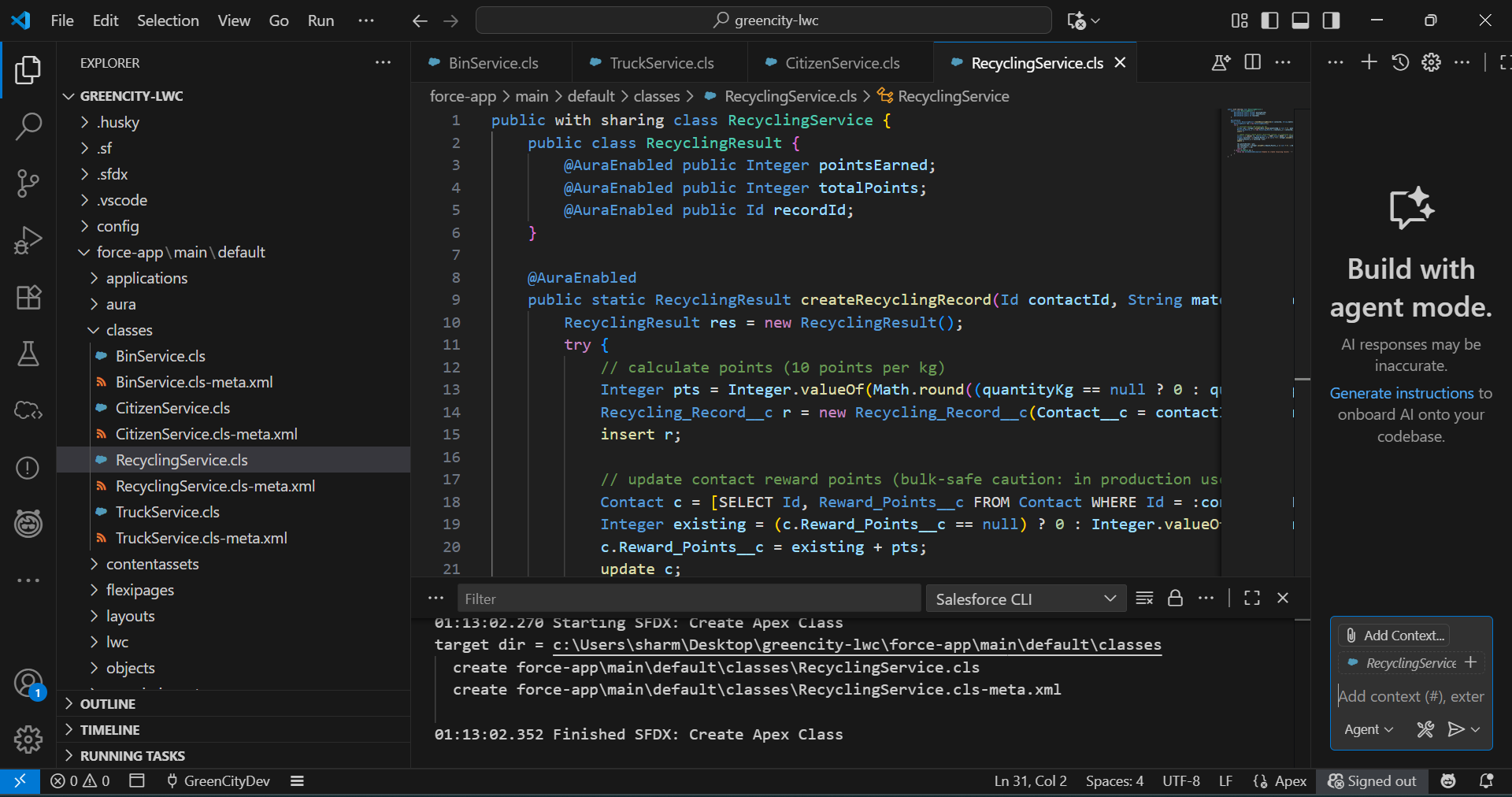
**➣Lightning Web Components (LWCs)**

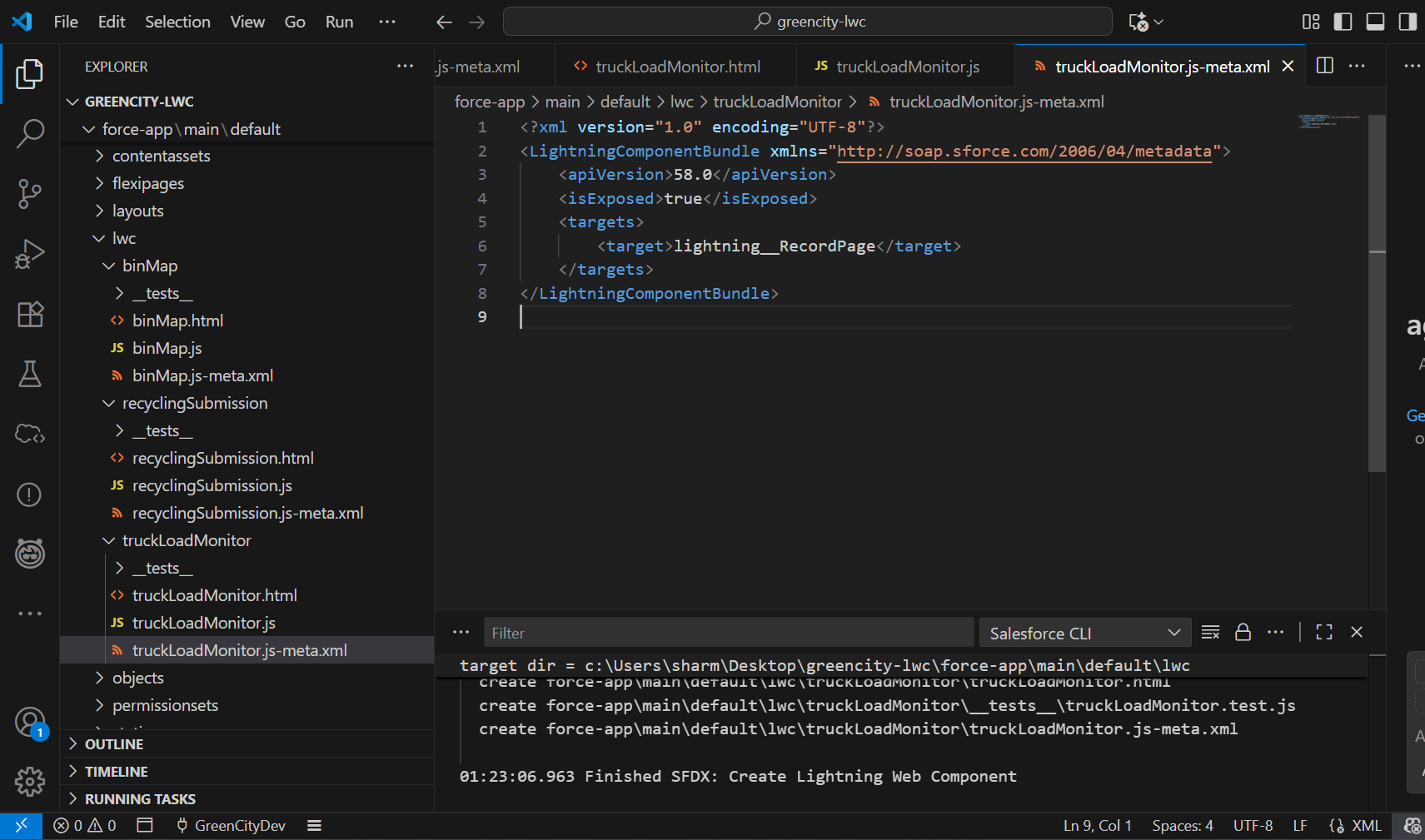
Custom LWCs were developed to extend functionality:

**•** Bin Map Component → shows bins on a map with color coding (green = empty, red = full).

**•** Recycling Submission Component → form for citizens to log material and quantity.

**•** Truck Load Monitor Component → displays truck capacity vs. current load in real time.





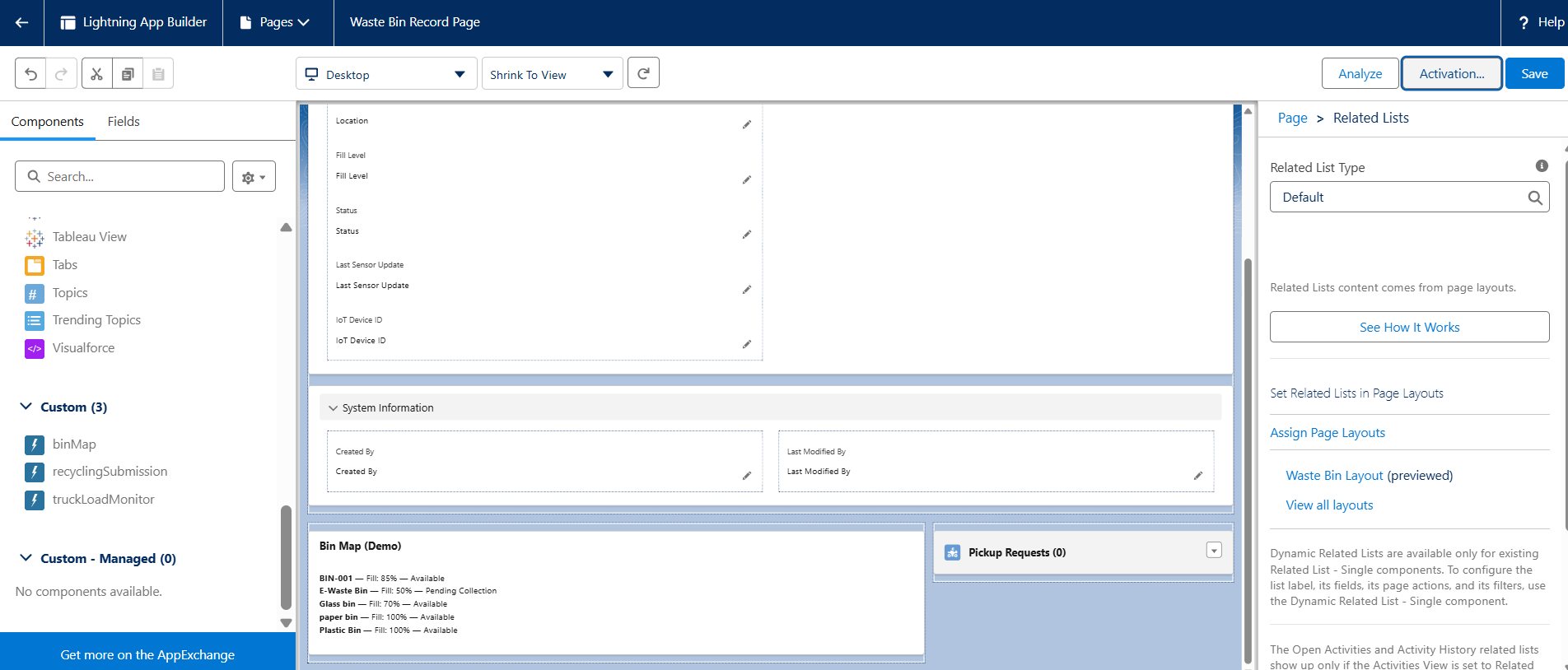
**➣Apex with LWC**

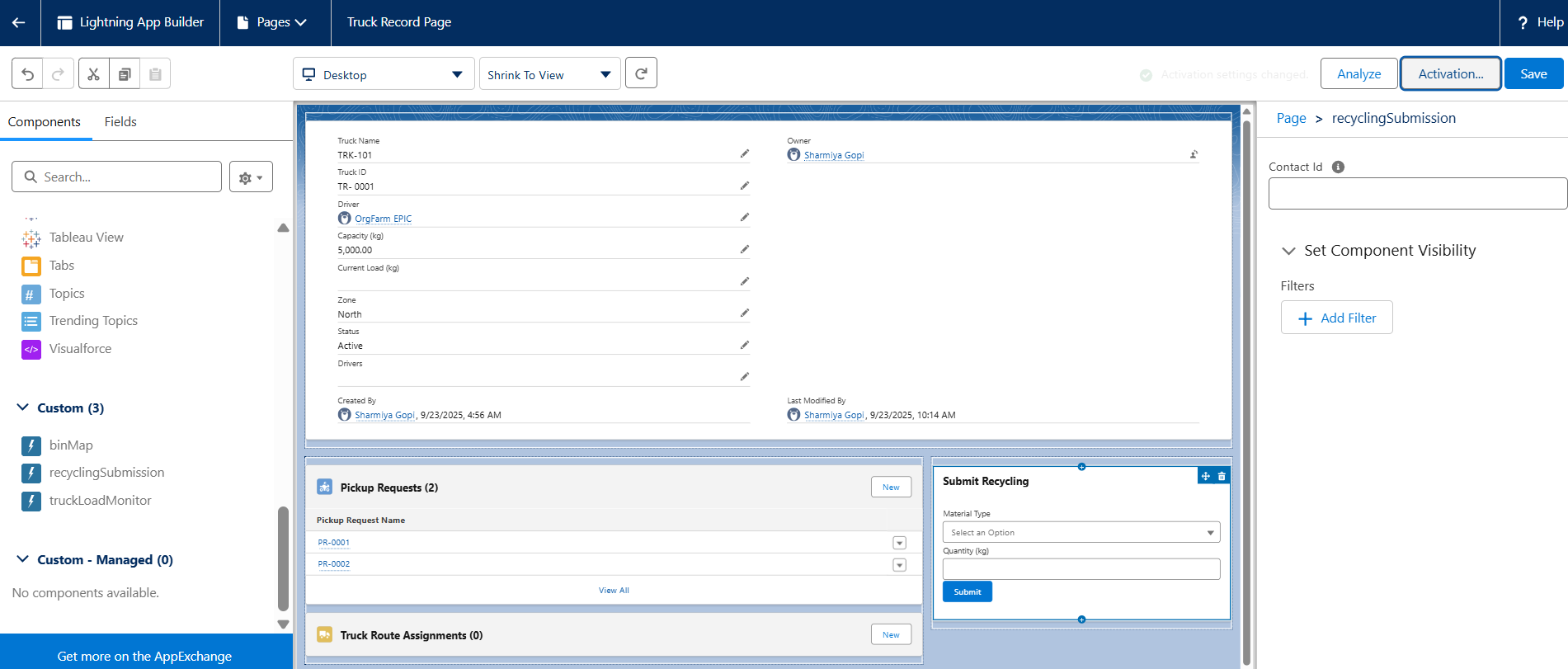
Apex controllers were used to fetch and process backend data:

**•** getActiveTrucks() → returns available trucks with remaining capacity.

**•** getCitizenPoints() → returns current reward points for logged-in citizen.

**•** assignTruckToPickup() → handles truck allocation logic.





**➣Wire Adapters**

@wire adapters simplified data access from Salesforce:

**•** @wire(getRecord, { recordId: '$recordId', fields: [...] }) → used to display bin fill levels and truck load instantly.

**•** @wire(getCitizenPoints) → displayed citizen reward points on the portal.

**➣Imperative Apex Calls**

Imperative calls were used when user actions required backend logic:

**•** Example: Clicking "Assign Truck" button calls assignTruckToPickup() Apex method to allocate a truck.

**•** Example: Submitting Recycling Form calls createRecyclingRecord() and updates points.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**