**Talha: Button Implementation**

**1. Buy Button**

* **Function**: Triggers a modal popup confirming purchase intent
* **Implementation**:

javascript

Copy

document.addEventListener('click', function(e) {

if (e.target.classList.contains('buy-btn')) {

document.getElementById('buyModal').style.display = 'block';

}

});

* + Uses event delegation to handle dynamic buttons
  + Toggles modal visibility via CSS display property

**2. Finance Button**

* **Function**: Redirects to finance page with pre-filled price
* **Implementation**:

javascript

Copy

onclick="window.location.href='finance.html?price=${car.price.replace(/\D/g,'')}'"

* + Strips non-numeric chars (\D) from price
  + Passes value via URL parameter

**3. Search/Reset Buttons**

* **Search**:

javascript

Copy

searchBtn.addEventListener('click', () => filterCars(searchInput.value));

* **Reset**:

javascript

Copy

resetBtn.addEventListener('click', () => {

searchInput.value = '';

showEmptyState();

});

**Malika: Finance System**

**finance.js Workflow**

1. **Price Initialization**

javascript

Copy

const urlParams = new URLSearchParams(window.location.search);

const carPrice = urlParams.get('price');

if (carPrice) document.getElementById('carPrice').value = carPrice;

1. **Payment Calculation**

javascript

Copy

const calculatePayment = () => {

const principal = price - downPayment;

const monthlyRate = interestRate / 100 / 12;

const payment = (principal \* monthlyRate) / (1 - Math.pow(1 + monthlyRate, -loanTerm));

};

* + Uses standard amortization formula
  + Handles edge cases (zero down payment, etc.)

1. **DOM Updates**
   * Dynamically updates results div with textContent

**Quinsten: Search System**

**1. Keyword Search**

* **Implementation**:

javascript

Copy

const filterCars = (term) => {

return carData.filter(car =>

Object.values(car).some(val =>

String(val).toLowerCase().includes(term.toLowerCase())

)

);

};

* + Searches all properties of car objects
  + Case-insensitive matching

**2. Empty State Design**

* **Purpose**:
  + Prevents blank page syndrome
  + Guides user interaction
* **Implementation**:

javascript

Copy

function showEmptyState() {

carGrid.innerHTML = `

<div class="empty-state">

<p>Search to view inventory</p>

</div>`;

}

* + Shows instructional UI when no search exists
  + Replaces content instead of hiding it

**Key Decision**: Chose empty state over skeleton loaders to reduce complexity while maintaining UX clarity.

**Presentation Tips**

1. **Demo Flow**:
   * Start with empty state → search → finance button → calculation
2. **Highlight**:
   * Talha: Event delegation pattern
   * Malika: Financial math accuracy
   * Quinsten: Resilient search function
3. **Time Allocation**:
   * 3 min per member + 1 min Q&A

Would you like sample dialogue for explaining these in non-technical terms?