

STRONGMIND

UX CHALLENGE

Darshankumar Rabadiya

Table of Contents

1. Introduction.....	4
1.1 Objective	4
1.2 Scope	5
1.2.1 Topping Inventory Management:.....	5
1.2.2 Pizza Creation and Management:	5
1.2.3 Accessibility and Responsiveness:	5
1.3 Target Users	6
1.3.1 Pizza Store Owner:	6
1.3.2 Pizza Chef	6
2. Research.....	7
2.1 User Personas	7
2.1.1 John's User Persona:.....	7
2.1.2 Antonio's User Persona:.....	8
2.2 User Stories	9
2.2.1 User Stories for John Mitchell (Pizza Store Owner):	9
2.2.2 User Stories for Antonio Rossi (Pizza Chef):	9
2.3 Assumptions	10
2.3.1 Technology Proficiency:	10
2.3.2 Device Usage:	10
2.3.3 Environment:	10
2.3.4 Operational Needs:	11
2.3.5 User Behavior:.....	11
3. User Flows & Design Decisions.....	12
3.1 Design Decisions for Topping Inventory Management UI:.....	12
Flow 1: Viewing Topping Dashboard	12
Flow 2: Viewing and Managing Toppings	12
Flow 3: Adding a New Topping	12
Flow 4: Editing a Topping	13
Flow 5: Deleting a Topping	13
Flow 6: Handling Errors	14
3.1.1 Dashboard (Owner)	15
3.1.2 Topping Inventory List	16
3.1.3 Add/Edit Topping	17
3.1.4 Error Handling	18

3.1.5 Notifications	19
3.2. Design Decisions for Chef's Pizza Creation and Management UI:	20
Flow 1: Viewing Pizza Dashboard	20
Flow 2: Creating a New Pizza (Multi-Step).....	20
Flow 3: Editing an Existing Pizza	21
Flow 4: Deleting a Pizza	21
Flow 5: Handling Errors	22
3.2.1 Pizza Dashboard.....	23
3.2.2 Multi-Step Pizza Creation Flow	24
3.2.3 Confirm Pizza	25
3.2.4 Error Handling	26
3.2.5 Notifications	27
4. Accessibility Considerations	28
4.1 Error Message Component	28
4.2 Form Fields Component	29
4.3 Success Pop up Component.....	30
4.4 Pizza Card Component	31
4.5 Step Indicator Component.....	32

1. Introduction

1.1 Objective

The goal of this project is to design an intuitive and user-centered pizza management system that caters to the specific needs of a pizza store. The system is aimed at streamlining two core functionalities:

- Topping Inventory Management for the pizza store owner, ensuring efficient tracking and updating of ingredients.
- Pizza Creation Process for the chef, enabling a seamless step-by-step workflow for creating custom pizzas.

The overarching objective is to reduce inefficiencies, minimize errors, and provide a visually appealing and accessible solution for both user groups.

1.2 Scope

The system encompasses the following key features and functionalities:

1.2.1 Topping Inventory Management:

- Dashboard to monitor topping inventory, including stock status and recent updates.
- Add, edit, and delete functionalities for managing toppings.
- Alerts and notifications for low-stock or out-of-stock items.
- Visual and text-based categorization of toppings for better organization.

1.2.2 Pizza Creation and Management:

- Step-by-step process for building pizzas, from selecting crust to adding toppings.
- Ability to save, edit, and delete pizzas.
- Visual representation of pizza composition, including selected toppings and quantities.
- Error handling for duplicate pizza names or missing elements.

1.2.3 Accessibility and Responsiveness:

- Design adheres to accessibility standards for visually impaired users.
- Simplified navigation and clear visual cues to ensure ease of use.

1.3 Target Users

This system is designed to meet the needs of two primary user groups:

1.3.1 Pizza Store Owner:

- Responsible for managing toppings inventory, including adding, updating, and deleting items.
- Requires real-time insights into stock levels and clear alerts for low-stock or duplicate entries.

1.3.2 Pizza Chef

- Focused on creating and managing pizzas in a fast-paced environment.
- Needs a responsive, step-by-step workflow for pizza creation, with clear guidance and easy access to topping availability.

2. Research

2.1 User Personas


2.1.1 John’s User Persona:



“ I don’t have time for complicated tools. I need a system that just works so I can focus on growing my business. ”

AGE 32
LOCATION Chandler, Arizona
ROLE Owner and Manager
TECH PROFICIENCY Intermediate

DEVICES PREFERENCES

 Desktop  Tablet

JOHN MITCHELL

PIZZA STORE OWNER

John is a 32-year-old small business owner who runs a bustling pizza shop in Chandler, Arizona. With years of experience managing operations, he relies on intuitive tools to handle inventory and keep his store running smoothly. Balancing his time between the office and the floor, he needs systems that work seamlessly across desktop and tablet.

GOALS

- Manage toppings and inventory efficiently using a desktop during work hours.
- Monitor and update the system on a tablet when walking around the store.
- Ensure topping data is accurate and prevents duplicate entries.
- Use clear visuals to identify inventory issues quickly.

FRUSTRATIONS

- Difficulty switching between desktop and tablet if the system isn’t optimized for both.
- Lack of actionable error messages when issues (like duplicate toppings) occur.
- Overly complicated interfaces that require extra training or time to use effectively.

NEEDS

- An intuitive interface that works seamlessly on desktops and tablets.
- Visual clarity, such as a dashboard with topping images/icons and easy-to-read labels.
- Quick and simple CRUD (Create, Read, Update, Delete) actions to manage inventory.

BEHAVIOR PATTERNS

- Uses a desktop in the back office to manage inventory and administrative tasks.
- Switches to a tablet during store walkthroughs or meetings with staff for quick updates.
- Regularly checks and updates inventory at the end of the day to plan for orders.

2.1.2 Antonio’s User Persona:



“In a busy kitchen, I need quick, simple tools that let me focus on making great pizzas, not fixing mistakes.”

AGE	45
LOCATION	Chandler, Arizona
ROLE	Head Chef
TECH PROFICIENCY	Basic

DEVICES PREFERENCES



ANTONIO ROSSI

PIZZA STORE OWNER

Antonio Rossi is a 45-year-old head chef in John's pizza shop in Chandler, Arizona. Passionate about crafting delicious pizzas, Antonio thrives in a fast-paced kitchen, juggling custom orders and new creations. He relies heavily on a responsive tablet interface to keep up with the demands of busy hours, prioritizing speed and accuracy in every pizza he makes.

GOALS

- Quickly prepare and customize pizzas using a tablet during busy kitchen hours.
- Access topping data and customization options on a single, responsive interface.
- Visualize pizzas and their toppings in real time to ensure accuracy.

NEEDS

- A tablet-friendly interface that is simple, responsive, and visual.
- Real-time feedback for pizza preparation, such as topping previews and validation messages.
- Quick access to topping lists with search/filter options for better efficiency.

FRUSTRATIONS

- Non-responsive systems that are hard to navigate on tablets.
- Difficulty managing toppings during peak hours if the UI is cluttered.
- Lack of immediate visual feedback when customizing pizzas.

BEHAVIOR PATTERNS

- Relies entirely on a tablet while working in the kitchen, keeping both hands free for food preparation.
- Prefers touch-based interactions, like tap-to-add or visual selection over complex navigation.

2.2 User Stories

2.2.1 User Stories for John Mitchell (Pizza Store Owner):

- **Topping Management:**
As a store owner, I want to add, edit, or remove pizza toppings so that I can keep the menu up-to-date and cater to customer preferences.
- **Inventory Tracking:**
As a store owner, I want to monitor topping inventory levels in real-time so that I can manage stock efficiently and prevent shortages.
- **Error Prevention:**
As a store owner, I want the system to alert me when a duplicate topping is being added so that I can maintain a clean and accurate menu.
- **User-Friendly Interface:**
As a store owner, I want an intuitive and clutter-free dashboard so that I can manage toppings without extensive training.

2.2.2 User Stories for Antonio Rossi (Pizza Chef):

- **Pizza Customization:**
As a chef, I want to drag and drop toppings onto a virtual pizza so that I can visualize and customize orders accurately.
- **Real-Time Feedback:**
As a chef, I want immediate visual confirmation when a topping is added or removed so that I can ensure the order matches customer requests.
- **Tablet Optimization:**

As a chef, I want the system to be optimized for tablet use so that I can operate it efficiently in the kitchen environment.

- Error Alerts:

As a chef, I want the system to notify me if a selected topping is unavailable so that I can inform the customer and suggest alternatives.

These user stories provide a clear framework for developing features that address the specific needs of both the store owner and the chef, ensuring a user-centered design approach.

2.3 Assumptions

2.3.1 Technology Proficiency:

- The pizza store owner has intermediate technology skills and is comfortable with desktop and tablet systems.
- The pizza chef has basic technology skills and prefers simple, intuitive interfaces on tablets.

2.3.2 Device Usage:

- The pizza store owner primarily uses a desktop for detailed inventory management but switches to a tablet for quick updates during walkthroughs.
- The pizza chef primarily uses a tablet for pizza creation while working in the kitchen.

2.3.3 Environment:

- The system will be used in a busy and fast-paced environment, where speed and clarity are essential.

- Network connectivity is stable in most scenarios, ensuring the system can rely on real-time updates.

2.3.4 Operational Needs:

- Topping inventory updates occur frequently, requiring real-time tracking and alerts.
- Pizza creation workflows should minimize the time required for selection and input.

2.3.5 User Behavior:

- Both users prefer visual guidance and clear feedback for actions and errors.
- Duplication (e.g., duplicate topping names, pizza names) is a common challenge that needs proactive prevention.

3 User Flows & Design Decisions

3.1 Design Decisions for Topping Inventory Management UI:

Flow 1: Viewing Topping Dashboard

Objective: Provide the owner with an overview of the current inventory status.

1. Start at Dashboard:
 - Open the Topping Dashboard.
 - View key metrics: Total Toppings, Low Stock, Out of Stock, and Newly Added items in card format.

Flow 2: Viewing and Managing Toppings

Objective: Allow the owner to view and interact with the full inventory of toppings.

1. Start at Inventory Page:
 - Open the Topping Inventory List screen.
 - View a grid of cards, each displaying details like topping name, category, quantity, and restock date.
2. Access Actions:
 - Click on the three-dot menu on a card to view options like "Edit" and "Delete."
3. End Result:
 - The owner can interact with individual toppings as needed.

Flow 3: Adding a New Topping

Objective: Enable the owner to add a new topping to the inventory.

1. Start at Add Topping Page:
 - Click the "Add Topping" button from the Inventory List or Dashboard.
1. Fill Form:
 - Enter details: Name, Category (via dropdown), Stock Quantity, Restock Date.
 - Upload an image of the topping.
2. Save Topping:
 - Click "Save" to finalize.
3. End Result:

- A success notification confirms the addition.
- The owner is redirected to the Topping Inventory List with the new topping displayed.

Flow 4: Editing a Topping

Objective: Allow the owner to update details of an existing topping.

1. Start at Inventory Page:
 - Open the Topping Inventory List screen.
2. Access Edit Form
 - Click “Edit” from the card's three-dot menu or the topping detail view.
 - The edit form preloads with the existing topping details.
3. Make Changes:
 - Modify fields as needed (e.g., update quantity, change restock date).
4. Save Changes:
 - Click “Save” to finalize the updates.
5. End Result:
 - A success notification confirms the update.
 - The topping card reflects the updated information.

Flow 5: Deleting a Topping

Objective: Remove an existing topping from the inventory.

1. Start at Inventory Page:
 - Open the Topping Inventory List screen.
2. Initiate Deletion:
 - Click “Delete” from the card's three-dot menu or the topping detail view.
 - A confirmation dialog appears.
3. Confirm Deletion:
 - Click “Confirm” to proceed with the deletion.
4. End Result:
 - A success notification confirms the deletion.

Flow 6: Handling Errors

Objective: Provide feedback and resolution for topping-related errors

1. Trigger Error:
 - Enter a duplicate topping name or invalid field data (e.g., empty required fields) during Add or Edit operations.
2. Show Error Message:
 - Inline feedback appears near the relevant field with a clear error message.
3. Resolve Error:
 - Correct the error by entering valid data.
4. End Result:
 - The error clears, and the owner successfully completes the operation.

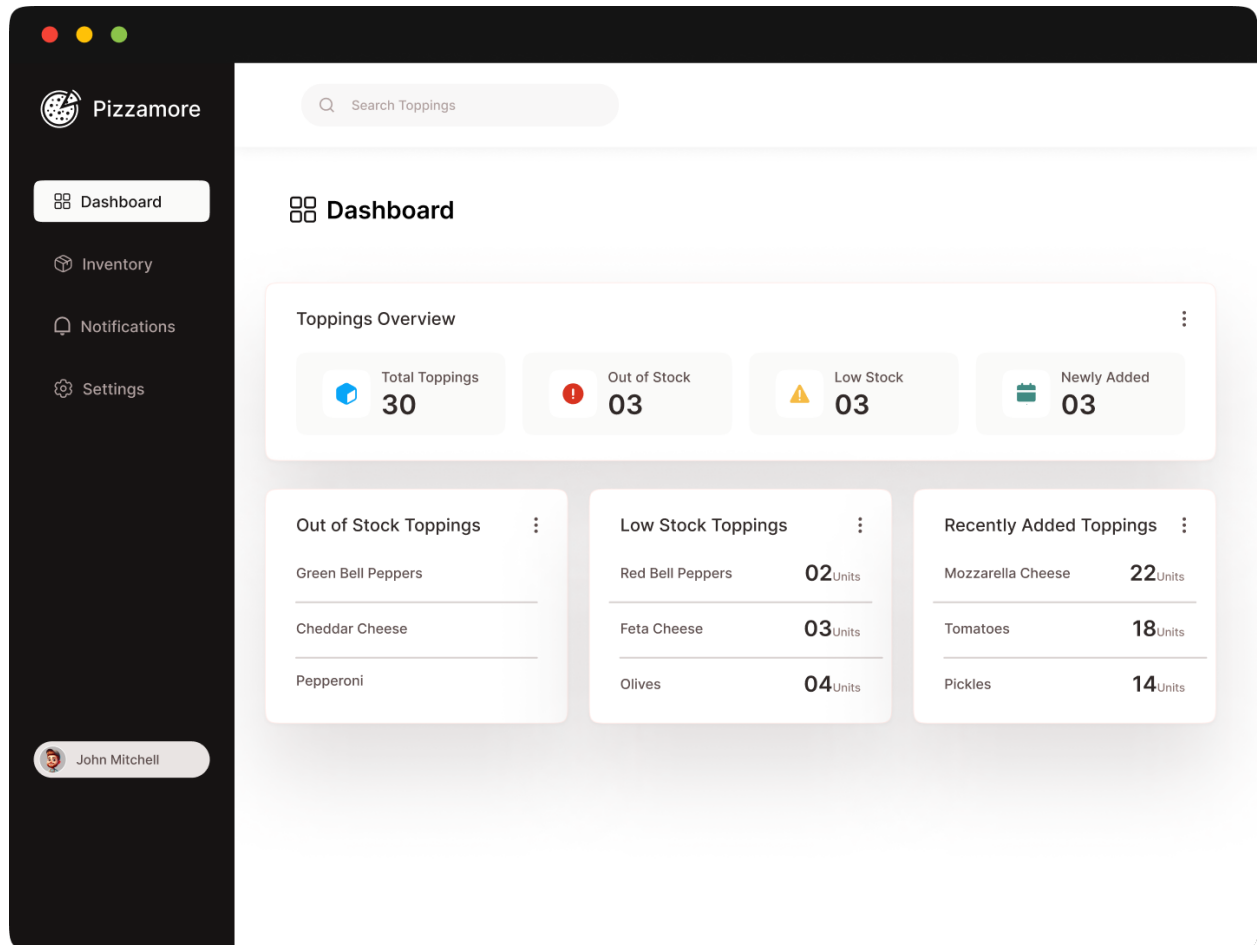
3.1.1 Dashboard (Owner)

Decision:

The dashboard provides an at-a-glance overview of topping inventory metrics using card-based visualizations. Key metrics include total toppings, low stock, out-of-stock, and newly added items.

Rationale:

- Ensures the owner can quickly assess inventory status.
- Card-based layout organizes data visually for easy scanning.
- Alerts and notifications highlight critical items for immediate action.



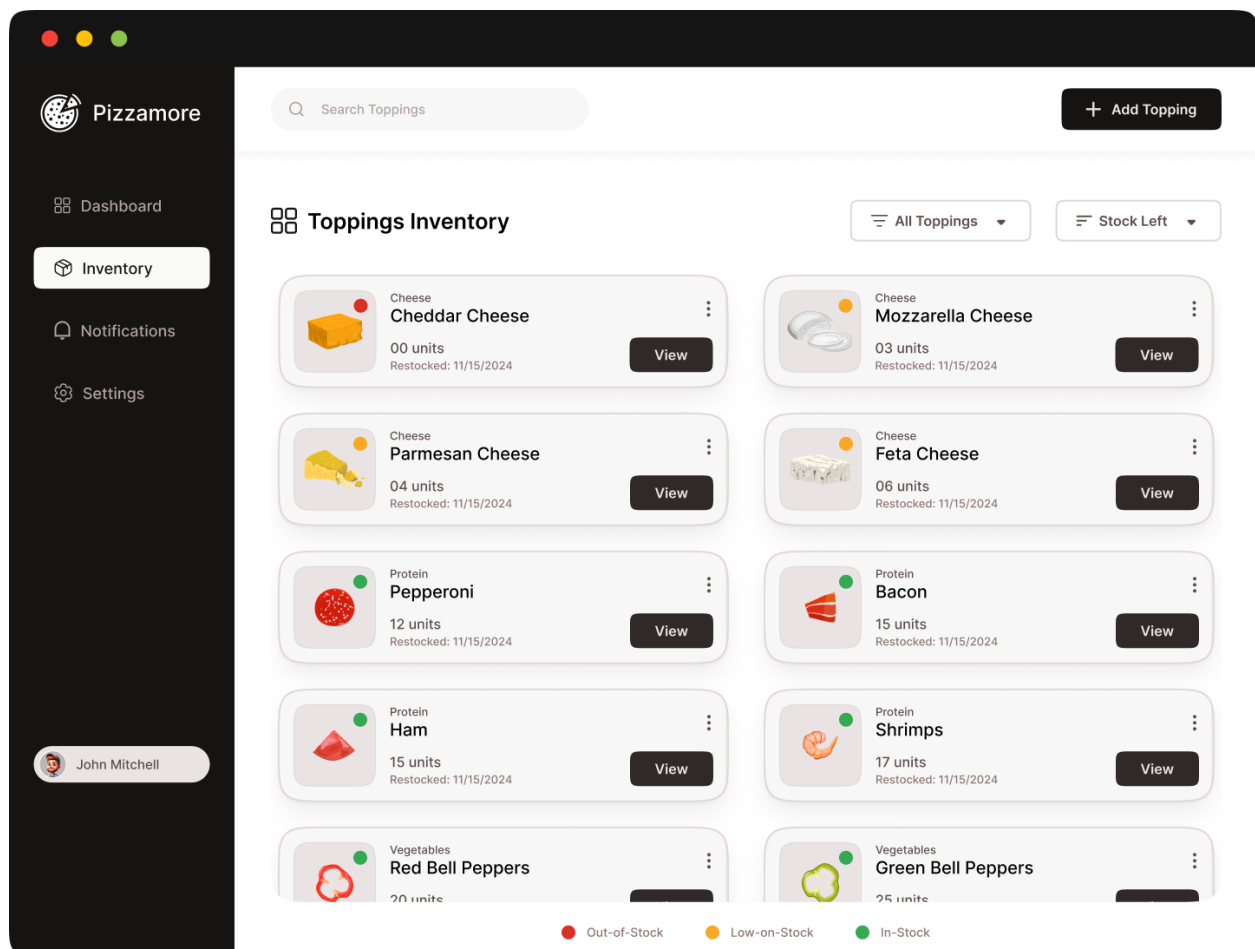
3.1.2 Topping Inventory List

Decision:

A card-based layout is used to display toppings with key details like name, category, stock quantity, and restock date. Each card includes options for "Edit," "Delete," and a quick menu for additional actions.

Rationale:

- Cards provide clear organization and readability for toppings.
- Quick-access actions (edit/delete) streamline inventory management.
- Color-coded stock indicators (red for out-of-stock, yellow for low stock) enhance usability.



3.1.3 Add/Edit Topping

Decision:

The add/edit topping form uses a minimal, clean layout with dropdowns for category selection, file upload for images, and validation to prevent duplicate entries.

Rationale:

- Simplified input fields make data entry faster and error-free.
- Dropdowns ensure consistent category selection.
- Validation prevents errors, such as adding duplicate topping names.

The screenshot displays the Pizzamore web application interface. On the left is a dark sidebar with the Pizzamore logo and navigation links: Dashboard, Inventory (highlighted), Notifications, and Settings. At the bottom of the sidebar is a user profile for John Mitchell. The main content area has a search bar labeled 'Search Toppings'. Centered on the screen is a light gray modal window titled 'Add Topping' with a close button (X). Inside the modal, there is an 'Upload image' section with a camera icon. Below this are four input fields: 'Topping Name' (text input), 'Topping Category' (dropdown menu), 'Toppings Stock' (text input), and 'Stock Date' (text input). At the bottom of the modal are two buttons: 'Cancel' and 'Save'.

3.1.4 Error Handling

Decision:

For errors like duplicate topping names, inline feedback appears next to the relevant input field with a concise error message and guidance to resolve the issue.

Rationale:

- Inline feedback ensures clarity by highlighting the problem area.
- Actionable error messages guide users in correcting mistakes.
- Reducing frustration enhances user confidence and system reliability.

The screenshot displays the Pizzamore application interface. On the left is a dark sidebar with the Pizzamore logo and navigation links: Dashboard, Inventory, Notifications, and Settings. At the bottom of the sidebar is a user profile for John Mitchell. The main content area has a search bar labeled 'Search Toppings' and an '+ Add Topping' button. A modal titled 'Add Topping' is open, featuring a close button (X) in the top right. Inside the modal, there is a placeholder image for a topping. Below the image are four input fields: 'Topping Name' (containing 'Mozzarella Cheese' with a red exclamation mark icon), 'Topping Category' (a dropdown menu showing 'Select Topping Category'), 'Toppings Stock' (containing 'Enter Stock Quantity'), and 'Stock Date' (containing 'Enter Stock Date'). At the bottom of the modal are 'Cancel' and 'Save' buttons. To the right of the modal, an error message box is displayed with the title 'Name already exists' and a close button (X). The message text reads: 'A topping named 'Mozzarella' already exists. Use a unique name to avoid confusion in your inventory.'

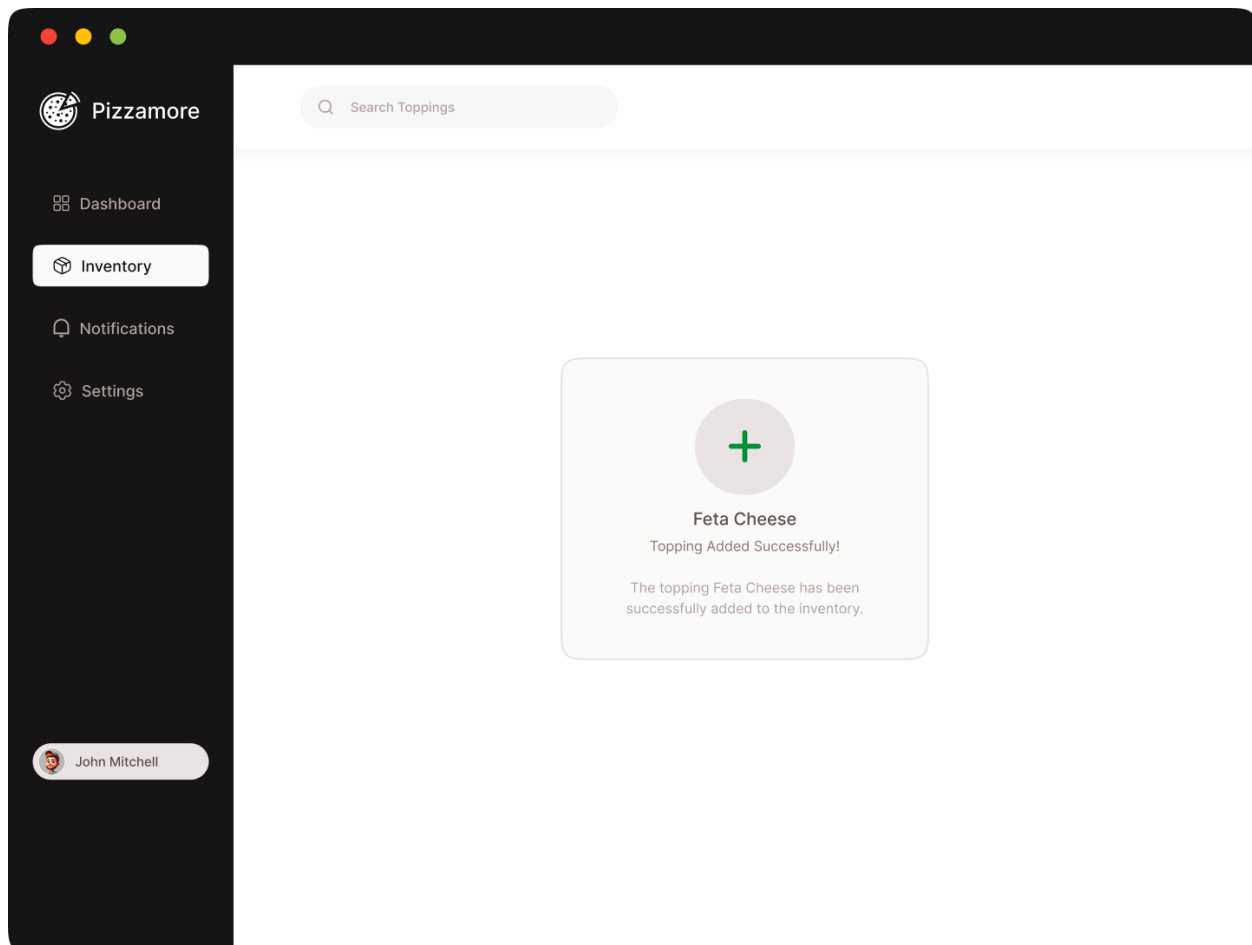
3.1.5 Notifications

Decision:

Notifications appear for key inventory events, such as "Topping Added Successfully" or "Low Stock Alert." They are designed to be prominent yet non-intrusive.

Rationale:

- Keeps users informed about inventory updates without overwhelming them.
- Alerts provide actionable insights, enabling timely decisions.
- Consistent visual styles (icons, colors) improve user comprehension.



3.2 Design Decisions for Chef's Pizza Creation and Management UI:

Flow 1: Viewing Pizza Dashboard

Objective: View all pizzas currently available and access individual pizza details.

1. Start at Dashboard:
 - Open the Pizza Dashboard screen.
 - View a grid of pizza cards with images, names, crust types, and toppings.
2. Select Pizza Card:
 - Tap a pizza card to open the detailed view.
3. End Result:
 - The chef sees full details of the selected pizza and available actions (Edit/Delete).

Flow 2: Creating a New Pizza (Multi-Step)

Objective: Guide the chef through a step-by-step pizza creation process.

1. Start Creation:
 - Tap "Create Pizza" on the dashboard to open the pizza creation flow.
2. Step 1: Choose Base:
 - Select a crust type (e.g., thin or thick crust).
 - Tap "Next" to proceed.
3. Step 2: Choose Sauce:
 - Select a sauce type (e.g., marinara, alfredo).
 - Tap "Next" to proceed.
4. Step 3: Add Cheese:
 - Select one or more cheese options (e.g., mozzarella, cheddar).
 - Tap "Next" to proceed.
5. Step 4: Add Protein:
 - Select proteins (e.g., pepperoni, bacon).
 - Tap "Next" to proceed.
6. Step 5: Add Vegetables:
 - Choose vegetable toppings (e.g., onions, peppers).
 - Tap "Next" to proceed.
7. Step 6: Confirm Pizza:

- Review the summary, including an image of the pizza, selected ingredients, and pizza name.
 - Tap “Save” to complete the creation process.
8. End Result:
- A success notification confirms the pizza was added.
 - The chef returns to the Pizza Dashboard.

Flow 3: Editing an Existing Pizza

Objective: Modify details of an already created pizza.

1. Start at Dashboard:
 - Tap a pizza card to view its details.
2. Open Edit Screen:
 - Tap "Edit" to open the pizza creation flow.
 - The flow preloads with the current pizza details.
3. Make Changes:
 - Modify any step (e.g., change sauce, add toppings).
4. Save Changes:
 - Tap “Save” on the confirmation screen to finalize edits.
5. End Result:
 - A success notification confirms the updates.
 - The chef returns to the Pizza Dashboard with updated details.

Flow 4: Deleting a Pizza

Objective: Remove an existing pizza from the system.

1. Start at Dashboard:
 - Tap a pizza card to view its details.
2. Initiate Deletion:
 - Tap "Delete" to open a confirmation dialog.
3. Confirm Deletion:
 - Tap "Confirm" in the dialog to proceed.
4. End Result:
 - A success notification confirms the deletion.
 - The pizza is removed from the dashboard.

Flow 5: Handling Errors

Objective: Provide feedback and resolution for duplicate pizza names.

1. Start Creation:
 - Enter a duplicate pizza name during the creation or editing flow.
2. Trigger Error:
 - The system detects the duplication and displays an inline error near the name input field.
3. Resolve Error:
 - Update the pizza name to a unique value.
4. End Result:
 - The error clears, and the chef continues the creation process.

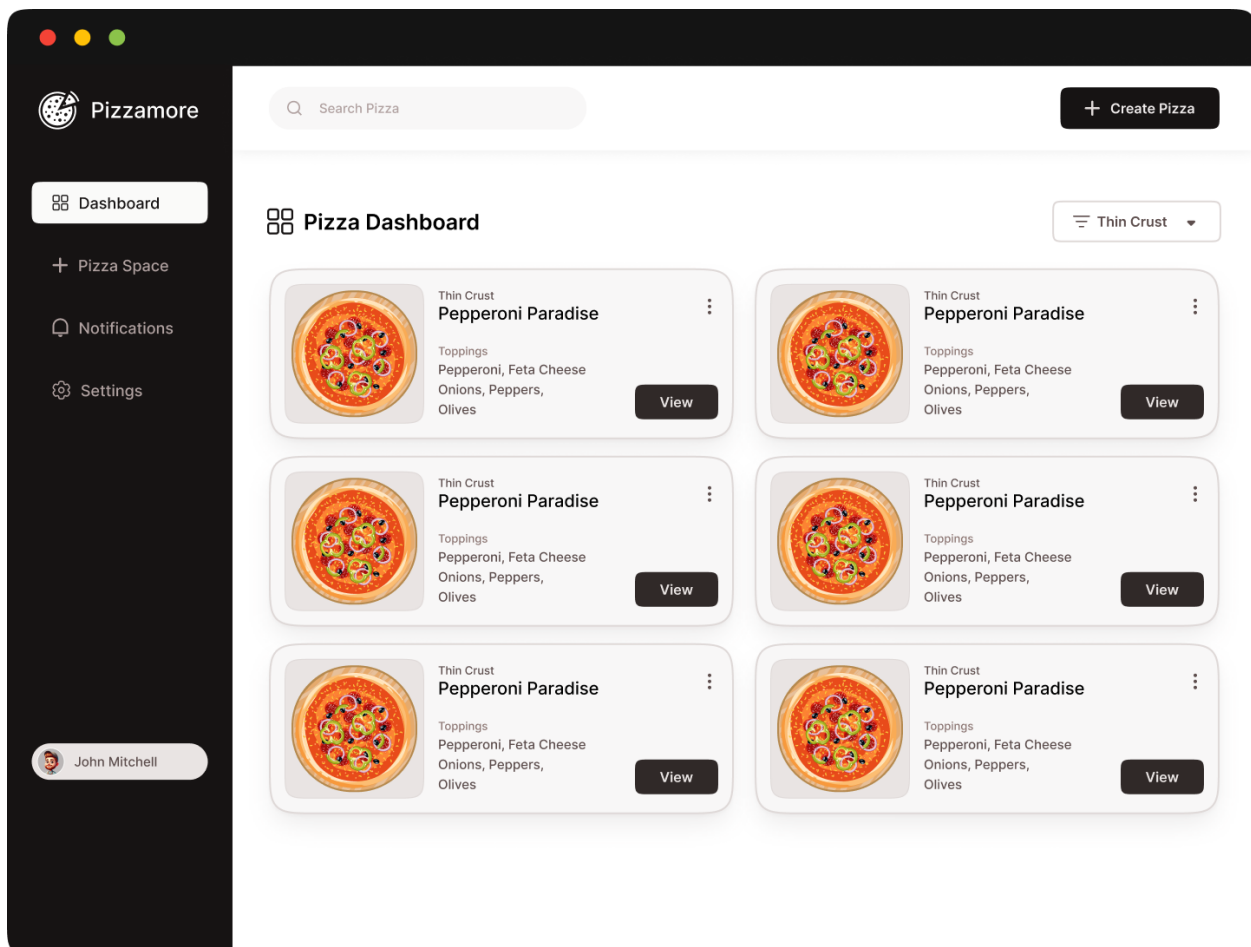
3.2.1 Pizza Dashboard

Decision:

The pizza dashboard uses a card-based design to display pizzas with details like name, crust type, and toppings. Each card includes options for "View," "Edit," and a quick menu for additional actions.

Rationale:

- Card layout provides a visually organized view of pizzas, allowing quick scanning.
- Action buttons (View/Edit/Delete) ensure streamlined access to key functions.
- Visual cues like pizza images enhance engagement and usability.



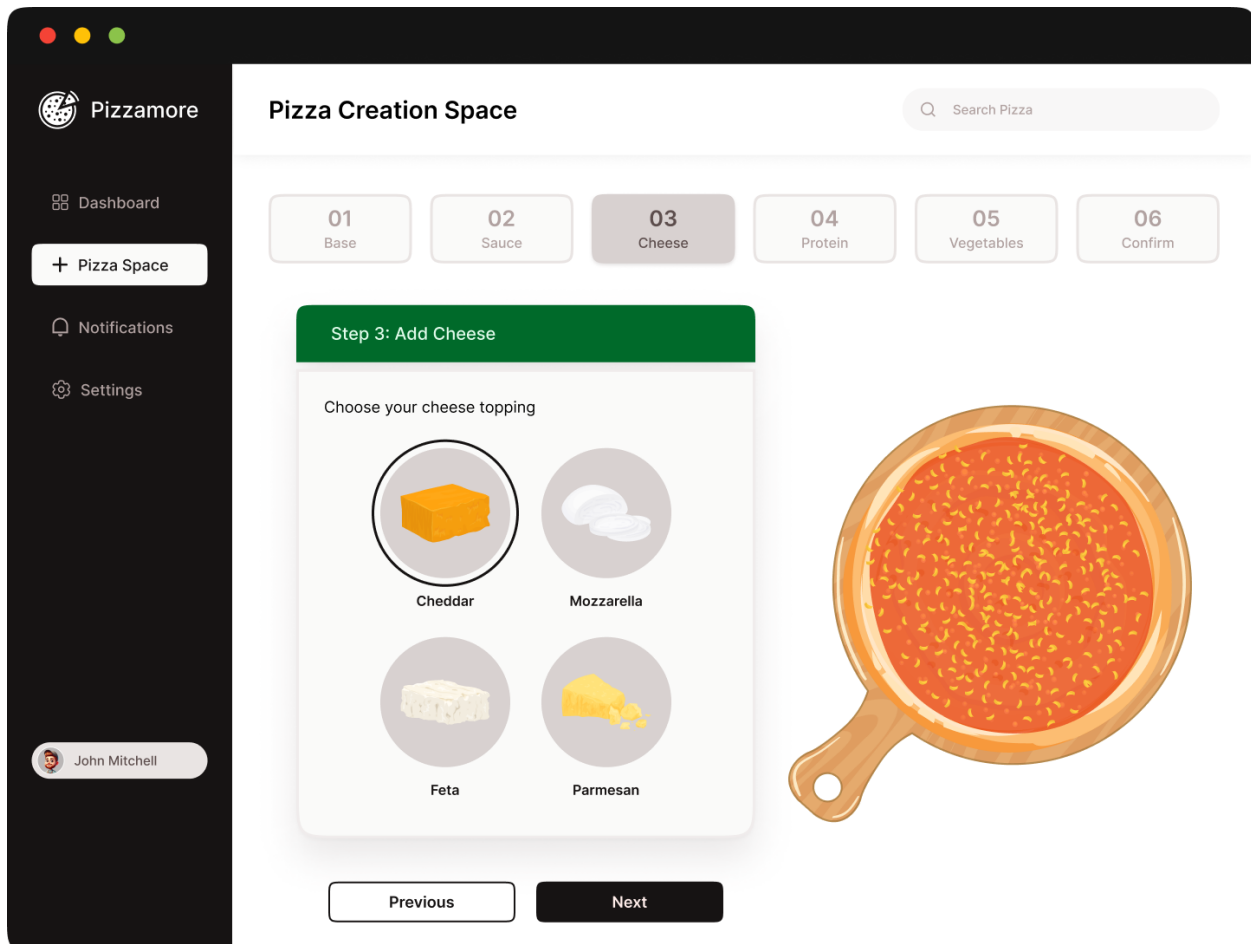
3.2.2 Multi-Step Pizza Creation Flow

Decision:

The pizza creation flow is divided into multiple steps (e.g., base, sauce, cheese, protein, vegetables), with visual previews for each selection. A progress indicator tracks the chef's position in the flow.

Rationale:

- Step-by-step navigation simplifies complex tasks and ensures accuracy.
- Visual previews improve clarity and help chefs verify their selections in real-time.
- Progress indicators provide clear guidance on task completion.



3.2.3 Confirm Pizza

Decision:

The confirmation screen displays a visual summary of the pizza, including name, selected ingredients, and an image. Clear action buttons for "Save" and "Cancel" allow chefs to finalize the pizza.

Rationale:

- Visual feedback ensures the chef can review and verify all details before saving.
- Final checkpoint reduces the chance of errors, like missing or incorrect ingredients.
- Call-to-action buttons provide clear navigation options

The screenshot displays the Pizzamore Pizza Creation Space interface. On the left is a dark sidebar with the Pizzamore logo and navigation links: Dashboard, Pizza Space (highlighted with a plus icon), Notifications, and Settings. At the bottom of the sidebar is a user profile for John Mitchell. The main area is titled 'Pizza Creation Space' and features a search bar. Below the title is a horizontal sequence of six steps: 01 Base, 02 Sauce, 03 Cheese, 04 Protein, 05 Vegetables, and 06 Confirm (which is highlighted in grey). The 'Step 5: Confirm Pizza' modal is active, showing a green header and the instruction 'Review your creation and give it a name'. It includes a text input field for the 'Pizza Name' and a 'Pizza Overview' table. To the right of the modal is a large, appetizing illustration of a pizza on a wooden peel. At the bottom of the modal are 'Previous' and 'Save' buttons.

Pizza Overview	
Base	Thick Crust
Sauce	Tomato Sauce
Cheese	Cheddar Cheese
Protein	Pepperoni
Vegetables	Tomatoes, Mushrooms, Olives

3.2.4 Error Handling

Decision:

Error states, such as duplicate pizza names, are handled with inline feedback near the relevant input field. A clear error message provides actionable guidance.

Rationale:

- Inline feedback minimizes confusion by highlighting the problem area directly.
- User-friendly error messages enhance clarity and prevent frustration.
- Validation during input prevents errors like duplicate names in the first place.

The screenshot displays the 'Pizzamore' application interface. On the left is a dark sidebar with navigation links: 'Dashboard', '+ Pizza Space' (highlighted), 'Notifications', and 'Settings'. At the bottom of the sidebar is a user profile for 'John Mitchell'. The main content area is titled 'Pizza Creation Space' and features a search bar. Below the title is a horizontal sequence of six steps: '01 Base', '02 Sauce', '03 Cheese', '04 Protein', '05 Vegetables', and '06 Confirm' (which is the active step). The 'Step 5: Confirm Pizza' modal is open, prompting the user to 'Review your creation and give it a name'. The 'Pizza Name' input field contains 'Pepperoni Spices' and has a red error icon. Below the input is a 'Pizza Overview' table:

Base	Thick Crust
Sauce	Tomato Sauce
Cheese	Cheddar Cheese
Protein	Pepperoni
Vegetables	Tomatoes, Mushrooms, Olives

At the bottom of the modal are 'Previous' and 'Save' buttons. An error message box is displayed on the right, stating: 'Name already exists. A Pizza named 'Pepperoni Spices' already exists. Use a unique name to avoid confusion.' A blurred image of a pizza is visible in the background.

3.2.5 Notifications

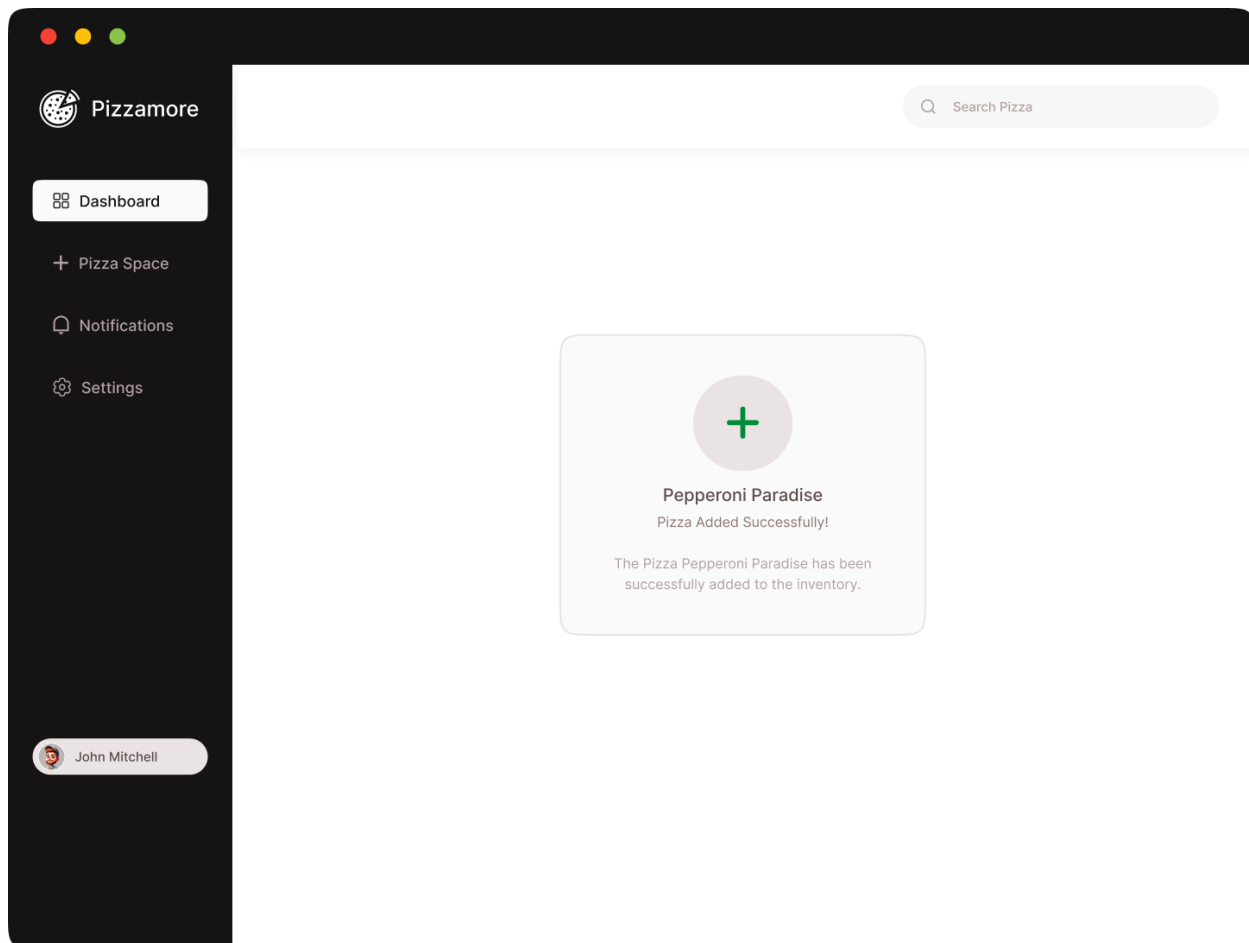
Decision:

Notifications provide feedback for successful actions "Pizza Added Successfully".

Notifications appear prominently but fade after a few seconds to avoid distraction.

Rationale:

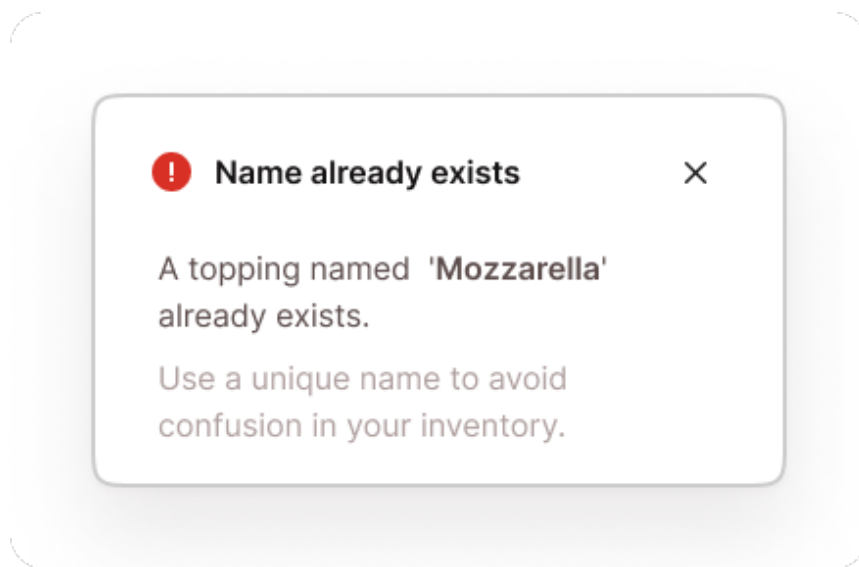
- Keeps chefs informed about critical actions, like successful pizza creation.
- Non-intrusive design ensures notifications do not disrupt workflow.
- Clear messaging improves user confidence and task progression.



4. Accessibility Considerations

4.1 Error Message Component

- **Clear and Concise Message:** The error message is straightforward and easy to understand, providing users with the necessary information to resolve issues.
- **Visual Indicator:** The exclamation mark icon provides a clear visual cue for the error, ensuring that users can easily identify the type of message being presented.
- **Accessibility Considerations:**
 - Error messages are accompanied by easily recognizable icons and concise text, ensuring that users with cognitive impairments or those using screen readers can quickly understand the issue.
 - High contrast for text and icons to improve visibility for users with visual impairments.



4.2 Form Fields Component

- **Clear and Concise Labels:** The labels for the form fields are descriptive and concise, helping users understand the purpose of each field (e.g., "Topping Name").
- **Visual Clarity:** The form fields are visually distinct, and it's easy to identify each input area.
- **Default, Hover, and Active States:** Visual feedback is provided with distinct states for the form fields, such as changes in color, border, or shadow, indicating the user's interaction (default, hover, active states).
- **Accessibility Considerations:**
 - Labels are paired with the correct form fields, ensuring screen reader compatibility.
 - The active state (e.g., when a field is selected) is clearly highlighted, providing visual feedback for users with visual impairments.
 - High contrast text to improve readability for users with low vision.

The diagram illustrates four form fields, each with a label and an input area, enclosed in a dashed purple border. The fields are arranged vertically and represent different states:

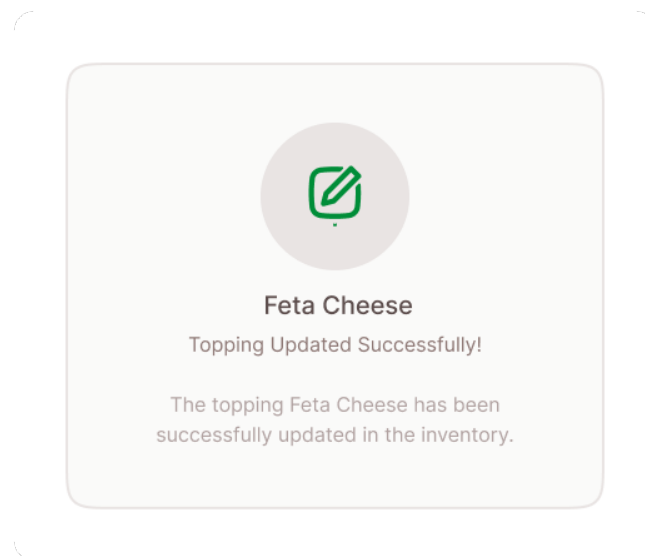
- Field 1 (Default):** The label is "Label" and the input area contains the text "Placeholder".
- Field 2 (Hover):** The label is "Label" and the input area contains the text "Placeholder". The input area has a thin grey border.
- Field 3 (Active):** The label is "Label" and the input area contains the text "Typed Value". The input area has a thick black border.
- Field 4 (Disabled):** The label is "Label" and the input area contains the text "Typed Value". The input area has a thin grey border.

4.3 Success Pop up Component

- **Clear and Concise Message:** The icon and text on the button provide a straightforward action, indicating to the user that they can edit the component.
- **Visual Confirmation:** The green Edit icon provides a clear visual cue for success, ensuring users can easily identify when an action has been successfully completed.
- **Accessibility Considerations:**

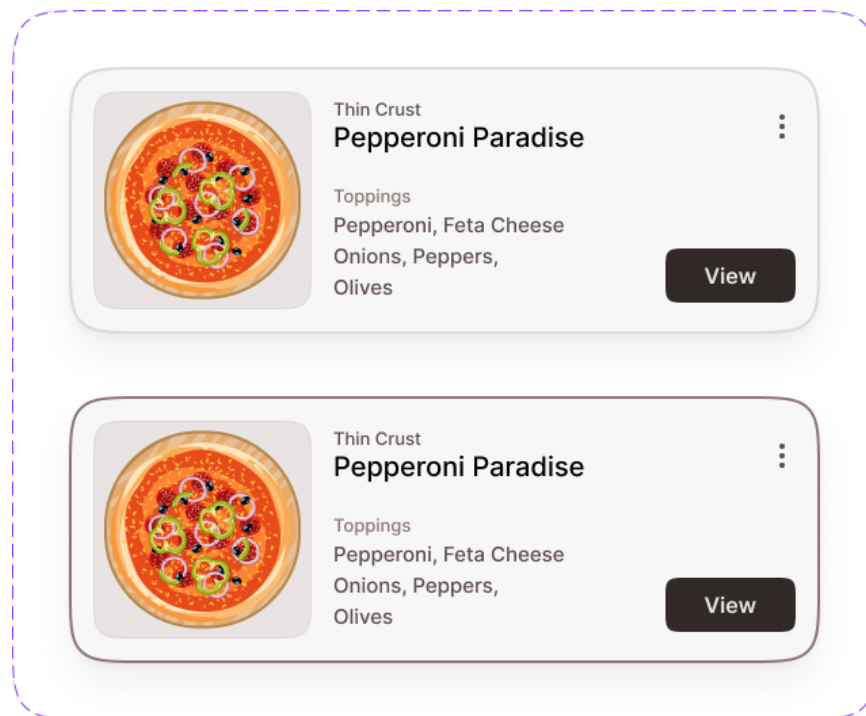
A clear visual indicator (green icon) supports users with cognitive impairments in understanding the success of an action.

The button's text and icon are large enough to ensure readability, and it adheres to color contrast standards for users with visual impairments.



4.4 Pizza Card Component

- **Clear and Concise Information:** The card displays essential details about the pizza, including its name, crust type, and toppings.
- **Visual Clarity:** The image of the pizza is used to visually represent the dish, making it easier for users to recognize the type of pizza.
- **Default, Hover, and Active States:** The card has distinct visual states for default, hover, and active states, with changes in color, border, or shadow to indicate interaction.
- **Accessibility Considerations:**
 - Clear labeling of pizza details supports screen readers, with alternative text for images to describe the pizza.
 - High contrast between text and background for better visibility.
 - Visual feedback for active states ensures users can easily interact with the card.



4.5 Step Indicator Component

- **Clear and Concise Labels:** The steps are clearly labeled (e.g., "01 Base," "02 Sauce"), guiding users through a defined sequence of actions.
- **Visual Differentiation:** The active step is visually differentiated from inactive steps through changes in color or design.
- **Accessibility Considerations:**
 - The steps are easy to follow, with distinct visual cues indicating the current active step.
 - The differentiation between active and inactive steps ensures that users with cognitive impairments can easily track their progress.
 - High contrast text and background color enhance readability, especially for users with visual impairments.

