BytePizza AI Designer Example

Generate solutions using FrameworX AI Designer

This example showcases a solution built with FrameworX AI Designer, with a focus on the visual layer.

Solution for download: BytePizza.dbsln

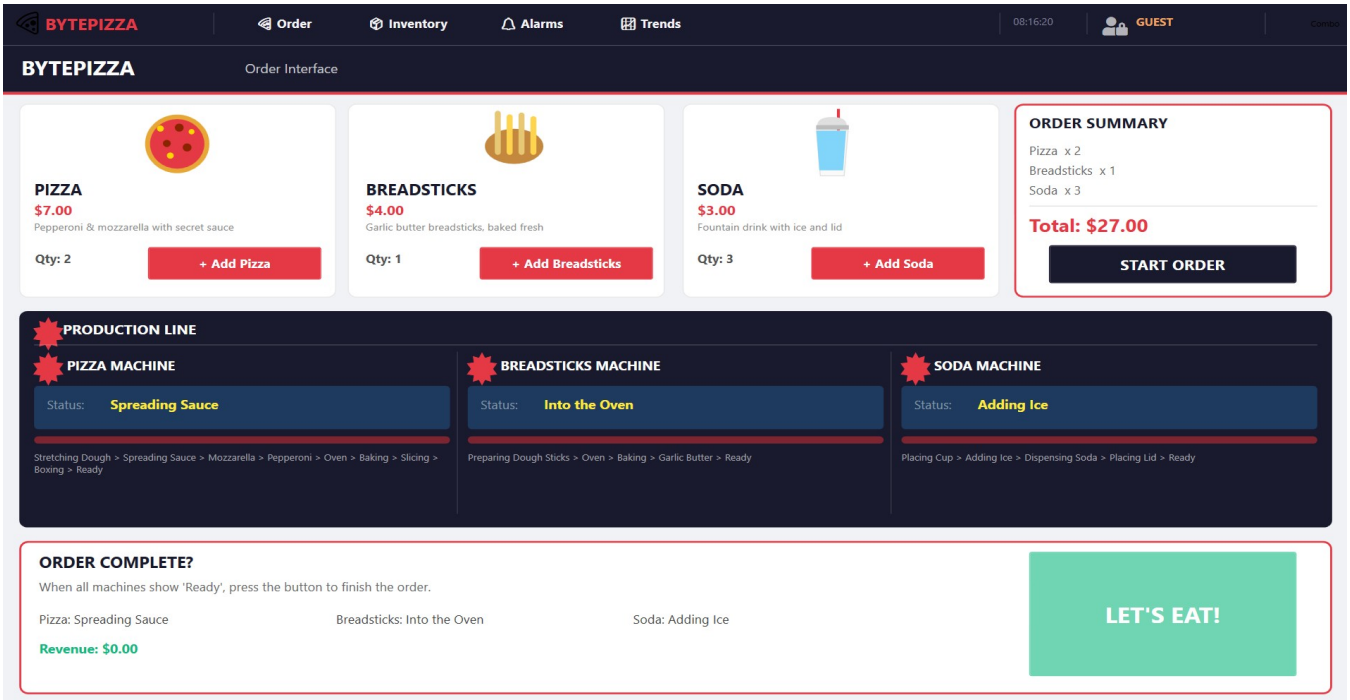
Showcases:

- Generate a complete restaurant-operations SCADA solution end-to-end from two natural-language prompts.
- Model a Unified Namespace for three production machines, six inventory items, and revenue tracking.
- Drive realistic machine state sequences from a server-side simulation script that ticks every second.
- Configure low-inventory alarms across all six inventory tags with a single Warning group.
- Historize all inventory levels and revenue continuously for trend visualization.
- Build a polished multi-page HMI with order entry, inventory management, alarms, and trends.
- Wire customer ordering, automatic inventory consumption, and revenue accumulation through expressions and scripts behind the displays.

Summary

BytePizza simulates a fast-food pizza restaurant operation. It demonstrates real-time order processing, inventory management, machine state simulation, alarm monitoring, trend visualization, and revenue tracking — all built with FrameworX and AI-assisted engineering via FrameworX AI Designer.

The solution showcases how FrameworX can model a complete operational workflow: a customer places an order, production machines process each item through realistic state sequences, inventory is consumed, revenue is accumulated, and operators can monitor all KPIs through a polished multi-page HMI.



Prompts

Prompt 1 - Backend: Tags, Historian, Alarms and Automation Script

```
You are a senior industrial automation engineer specialized in FrameworX (Tatsoft),
SCADA/HMI, and configuration of communication points (tags) for process control
and monitoring systems.

<context>
I am building a FrameworX solution that simulates a fully automated pizza restaurant
called "BytePizza". The restaurant has 3 automated machines: Pizza, Breadsticks, and
Soda. Each machine has 2 tags: Active (Boolean) and State (String).

When Active is set to True, the machine begins cycling through its preparation states
automatically, advancing every 3 seconds until reaching the final "Ready" state.
When Active is set to False, the machine resets to "Off".

Menu prices:
- Pizza — $7
- Breadsticks — $4
- Soda — $3

Machine state sequences:

Pizza: Off ? Stretching Dough ? Spreading Sauce ? Adding Mozzarella ?
Adding Pepperoni ? Into the Oven ? Baking ? Slicing ? Boxing ? Pizza Ready

Breadsticks: Off ? Preparing Dough Sticks ? Into the Oven ? Baking ?
Adding Garlic Butter ? Breadsticks Ready

Soda: Off ? Placing Cup ? Adding Ice ? Dispensing Soda ? Placing Lid ? Soda Ready
</context>

<instructions>
Complete the following tasks in order. After each task, confirm completion before
moving to the next.

1. **Create a new FrameworX solution** named "BytePizzaSBaT".

2. **Create the complete tag structure** in the Unified Namespace:
   - Restaurant/Machines/Pizza/Active (Boolean, initial: False)
   - Restaurant/Machines/Pizza/State (String, initial: "Off")
   - Restaurant/Machines/Pizza/TickCounter (Integer, initial: 0)
   - Restaurant/Machines/Breadsticks/Active (Boolean, initial: False)
   - Restaurant/Machines/Breadsticks/State (String, initial: "Off")
   - Restaurant/Machines/Breadsticks/TickCounter (Integer, initial: 0)
   - Restaurant/Machines/Soda/Active (Boolean, initial: False)
   - Restaurant/Machines/Soda/State (String, initial: "Off")
   - Restaurant/Machines/Soda/TickCounter (Integer, initial: 0)
   - Restaurant/Inventory/Pizza/DoughInventory (Integer, initial: 50)
   - Restaurant/Inventory/Pizza/SauceInventory (Integer, initial: 50)
   - Restaurant/Inventory/Pizza/BoxInventory (Integer, initial: 50)
   - Restaurant/Inventory/Breadsticks/DoughSticksInventory (Integer, initial: 50)
   - Restaurant/Inventory/Soda/CupInventory (Integer, initial: 50)
   - Restaurant/Inventory/Soda/LidInventory (Integer, initial: 50)
   - Restaurant/Revenue/TotalRevenue (Float, initial: 0.0)

3. **Configure Low alarms** on each inventory tag with setpoint < 5. Use descriptive
   messages (e.g., "Pizza Dough Inventory Low — Below 5 units").

4. **Configure Historian** on all inventory tags and TotalRevenue:
   - History Enabled: True
   - Deadband Mode: Absolute
   - Historical Deadband: 0.01
   - Sample Mode: On Change
   - Min Time Between Samples: 1 second

5. **Create a periodic C# script task** named "MachineStateCycler" running every
   1000ms. The script must:
```

```
   - Read each machine's Active and State tags
   - When Active is True and State is not "Ready", increment the machine's TickCounter
   - When TickCounter reaches 3, advance State to the next step in its sequence
     and reset TickCounter to 0
   - When Active becomes False, reset State to "Off" and TickCounter to 0
   - Digital tags return int in FrameworX C# — use (int)@Tag.Path == 1 for boolean checks

6. **Create 3 security users**:
   - admin — Administrator permissions (password: "password")
   - Operator — Operator permissions
   - Customer — Guest permissions

After all tasks are complete, provide a summary table of all tags created.
</instructions>

<examples>
<example>
Tag: Restaurant/Machines/Pizza/Active
Type: Boolean (Digital)
Initial: False
Description: Indicates whether the pizza machine is actively preparing an order
</example>
<example>
Tag: Restaurant/Inventory/Pizza/DoughInventory
Type: Integer
Initial: 50
Description: Units of pizza dough in stock — alarm triggers when below 5
</example>
<example>
Tag: Restaurant/Revenue/TotalRevenue
Type: Float
Initial: 0.0
Description: Accumulated revenue from all completed orders ($)
</example>
</examples>
```

Prompt 2 - Displays

```
You are a senior industrial automation engineer and UI designer specialized in
FrameworX (Tatsoft), building professional SCADA/HMI displays with applied dynamics.

<context>
The BytePizzaSBaT solution already has all backend tags configured:

Machines (each has Active, State, TickCounter):
- Restaurant/Machines/Pizza/
- Restaurant/Machines/Breadsticks/
- Restaurant/Machines/Soda/

Inventory:
- Restaurant/Inventory/Pizza/DoughInventory
- Restaurant/Inventory/Pizza/SauceInventory
- Restaurant/Inventory/Pizza/BoxInventory
- Restaurant/Inventory/Breadsticks/DoughSticksInventory
- Restaurant/Inventory/Soda/CupInventory
- Restaurant/Inventory/Soda/LidInventory

Revenue:
- Restaurant/Revenue/TotalRevenue

Menu prices: Pizza $7 | Breadsticks $4 | Soda $3

Machine states:
- Pizza: Off ? Stretching Dough ? Spreading Sauce ? Adding Mozzarella ?
  Adding Pepperoni ? Into the Oven ? Baking ? Slicing ? Boxing ? Pizza Ready
- Breadsticks: Off ? Preparing Dough Sticks ? Into the Oven ? Baking ?
  Adding Garlic Butter ? Breadsticks Ready
- Soda: Off ? Placing Cup ? Adding Ice ? Dispensing Soda ? Placing Lid ? Soda Ready
```

```
Users and page access:
- admin: all pages
- Operator: all pages except Inventory Management
- Customer: Order Interface only
</context>

<instructions>
Build 5 displays one at a time. After each display, STOP and wait for review before
proceeding.

Visual identity — BytePizza brand:
- Background: dark charcoal/navy (#1A1A2E)
- Accents: warm red/orange (#E53935, #FF7043)
- Cards: white/light gray, rounded corners
- Icons: simple geometric shapes only — no complex food illustrations
- Logo: "BytePizza" wordmark with a simple pizza slice icon (circle + triangle wedge)

Display 1 — Order Interface (MainPage):
- Top half: 3 item cards (Pizza, Breadsticks, Soda) with quantity selectors (+/-)
  and price. Running order total shown prominently. START ORDER button.
- Bottom half: 3 machine status panels, one per item ordered. Each shows the
  machine name, current State tag value as text, and a simple progress indicator.
  Active machines highlight in orange; machines at "Ready" state highlight in green.
- LET'S EAT button: enabled only when all active machines reach their Ready state.
  On press: decrements relevant inventory tags by quantity ordered, adds order value
  to TotalRevenue, resets all machines (Active ? False).
- Implement business logic in C# CodeBehind. Quantity controls lock during active
  orders.

Display 2 — Inventory Management:
- Grid of inventory cards, one per tag. Each card shows tag name, current value
  (bidirectional — editable), and a low-stock warning when value < 5 (red highlight).
- TotalRevenue panel with current value and a Reset button (sets to 0).
- Reset All Inventory button to restore all tags to 50.
- Access restricted to admin only.

Display 3 — Alarming:
- Full-page AlarmViewer component.
- Operators and admin can acknowledge and shelve alarms.

Display 4 — Trending:
- Full-page TrendChart component.
- On page load, pre-load pens for the two Soda inventory tags
  (CupInventory and LidInventory).
- Use TrendPenList/Children format for pens — flat arrays lose pens on save.

Display 5 — Navigation Header (docked):
- Fixed top dock visible on all pages.
- BytePizza logo on the left.
- Navigation buttons: Order | Inventory | Alarms | Trends
- Inventory button hidden for Customer role (security dynamic).
- Current user name displayed on the right.

Start with Display 1 — Order Interface.
</instructions>

<examples>
<example>
Component: START ORDER button
Dynamic: Enabled when at least 1 item has quantity > 0. On click ? sets Active = True
on each machine with qty > 0, locks quantity controls.
Color: gray (#9E9E9E) when disabled ? orange (#FF7043) when enabled.
</example>
<example>
Component: Pizza machine status panel
Dynamic: FillColorDynamic on background — #FF7043 (in progress) ? #4CAF50 (Pizza Ready).
Text binding to Restaurant/Machines/Pizza/State.
Visibility: shown only if Pizza was included in the order.
</example>
<example>
Component: DoughInventory card
```

```
Dynamic: Bidirectional binding to Restaurant/Inventory/Pizza/DoughInventory.
Background fill ? #EF5350 (red) when value < 5.
</example>
</examples>
```

Architecture Summary

Module	Objects	Description
UNS (Tags)	26 tags, 1 UserType	Full restaurant data model: Inventory, Machines, Orders, Revenue
Alarms	6 items, 3 groups	Low inventory warnings (threshold: 5 units)
Historian	7 tags logged	All inventory levels + revenue
Scripts	5 tasks, 2 classes	Machine state cycler + built-in startup/shutdown
Datasets	4 DBs, 1 query	AlarmHistorian, Retentive, RuntimeUsers, TagHistorian
Displays	10 pages, 1 layout	Full HMI with order interface, inventory, trends, alarms
Security	5 users	admin, Operator, User, Customer, Guest

Unified Namespace (UNS) - Tag Structure

All tags are organized under the `Restaurant/` root hierarchy with clear separation between server-side shared data and client-side UI state.

Inventory (Integer)

Tag Path	Description
Restaurant/Inventory/Pizza/DoughInventory	Pizza dough inventory count
Restaurant/Inventory/Pizza/SauceInventory	Pizza sauce inventory count
Restaurant/Inventory/Pizza/BoxInventory	Pizza box inventory count
Restaurant/Inventory/Breadsticks/DoughSticksInventory	Breadsticks dough sticks inventory count
Restaurant/Inventory/Soda/CupInventory	Soda cup inventory count
Restaurant/Inventory/Soda/LidInventory	Soda lid inventory count

Machines

Tag Path	Type	Description
Restaurant/Machines/Pizza/Active	Digital	Pizza machine active state
Restaurant/Machines/Pizza/State	Text	Current production step (e.g., "Stretching Dough", "Pizza Ready")
Restaurant/Machines/Pizza/TickCounter	Integer	Internal tick counter for state timing
Restaurant/Machines/Breadsticks/Active	Digital	Breadsticks machine active state
Restaurant/Machines/Breadsticks/State	Text	Current production step
Restaurant/Machines/Breadsticks/TickCounter	Integer	Internal tick counter
Restaurant/Machines/Soda/Active	Digital	Soda machine active state
Restaurant/Machines/Soda/State	Text	Current production step
Restaurant/Machines/Soda/TickCounter	Integer	Internal tick counter

Revenue

Tag Path	Type	Description
Restaurant/Revenue/TotalRevenue	Double	Accumulated revenue from all completed orders

Client Tags (Per-Session UI State)

Tag Path	Type	Description
Restaurant/Order/QtyPizza	Integer	Pizza quantity for current order
Restaurant/Order/QtyBreadsticks	Integer	Breadsticks quantity for current order
Restaurant/Order/QtySoda	Integer	Soda quantity for current order
Restaurant/Order/OrderTotal	Double	Computed order total ($7 pizza, $4 breadsticks, $3 soda)
Restaurant/Order/OrderStarted	Digital	Flag: order is being processed
Restaurant/Order/AllReady	Digital	Flag: all ordered items are ready
Restaurant/Order/PizzaReady	Digital	Flag: pizza machine reached "Pizza Ready"
Restaurant/Order/BreadsticksReady	Digital	Flag: breadsticks machine reached "Breadsticks Ready"
Restaurant/Order/SodaReady	Digital	Flag: soda machine reached "Soda Ready"

Scripts

MachineStateCycler (Server Task): The core simulation engine. Runs cyclically on the server and advances each machine through its preparation states every 3 ticks when the machine's `Active` flag is set to 1.

Pizza Machine States: Stretching Dough Spreading Sauce Mozzarella Pepperoni Oven Baking Slicing Boxing Pizza Ready

Breadsticks Machine States: Preparing Dough Sticks Oven Baking Garlic Butter Breadsticks Ready

Soda Machine States: Placing Cup Adding Ice Dispensing Soda Placing Lid Soda Ready

Order Interface display - client-side logic

Client-side logic embedded in the order interface drives the full order lifecycle:

- When the display opens, all order quantities, totals, and flags reset to zero.
- While the display is open, the order total is recalculated cyclically and machine readiness is checked.
- When the operator clicks Start Order, the corresponding machines are activated for the ordered items.
- When the operator clicks Let's Eat, the order is finalized: revenue is added, inventory is subtracted, and machines reset.

Inventory Management display - client-side logic

- A Reset All button resets all six inventory items to 50 units.
- A Reset Revenue button resets total revenue to $0.

Alarms

All alarm items are configured as **Low (Lo)** condition alarms in the **Warning** group with a threshold of **5 units**.

Monitored Tag	Condition	Limit	Message
Pizza/DoughInventory	Lo	5	Pizza Dough Inventory Low — Below 5 units
Pizza/SauceInventory	Lo	5	Pizza Sauce Inventory Low — Below 5 units
Pizza/BoxInventory	Lo	5	Pizza Box Inventory Low — Below 5 units

Breadsticks/DoughSticksInventory	Lo	5	Breadsticks Dough Sticks Inventory Low — Below 5 units
Soda/CupInventory	Lo	5	Soda Cup Inventory Low — Below 5 units
Soda/LidInventory	Lo	5	Soda Lid Inventory Low — Below 5 units

Alarm Groups: AuditTrail, Critical (requires acknowledgment), Warning (no ack required).

Historian

7 tags are logged to **Table1** for historical trending:

- All 6 inventory tags (DoughInventory, SauceInventory, BoxInventory, DoughSticksInventory, CupInventory, LidInventory)
- TotalRevenue

Displays (HMI)

All content pages follow a **unified light theme** design language:

- Light gray background (`#FFF0F2F5`)
- Dark header bar (`#FF1A1A2E`) with red accent line (`#FFE63946`) and "BYTEPIZZA" branding
- White content cards (`#FFFFFFFF`) with light borders and colored accent strips per category
- Color coding: Gold for Pizza, Brown for Breadsticks, Blue for Soda, Red for critical actions, Green for success/trends

Page List

Display	Type	Description
MainPage	Canvas	Project home page / landing
OrderInterface	Canvas	Full order workflow: menu cards (Pizza $7, Breadsticks $4, Soda $3), order summary, production line status with real-time machine states, and "Let's Eat" completion button
InventoryManagement	Canvas	6 inventory cards (3×2 grid) with live values, editable TextBoxes, color-coded status bars, total revenue display, and reset actions
TrendPage	Canvas	Full-screen TrendChart with 6 pens tracking all inventory levels (5-minute duration, auto-scale, legend panel)
AlarmPage	Canvas	Full-screen AlarmViewer with comprehensive column set (Ack, ActiveTime, TagName, Group, Value, Priority, State, Message, etc.)
Header	Canvas	Persistent navigation header with page buttons
HeaderMobile	Canvas	Mobile-optimized header
HeaderPad	Canvas	Tablet-optimized header
LogOn	Canvas	Security login/logout interface
SelectPage	Canvas	Page navigation selector

Visual Dynamics Used

- **FillColorDynamic:** Inventory status bars change color based on stock level (red/yellow/green thresholds at 0/5/10)
- **VisibilityDynamic:** Production progress bars visible only when machine is active
- **ActionDynamic:** Add-to-order buttons use RunExpressions to increment quantities; Start Order and Let's Eat buttons trigger client-side logic
- **SecurityDynamic:** "Let's Eat" button disabled until all ordered items reach Ready state

---

Security

User	Role
admin	Administrator
Operator	Operator
User	Standard User
Customer	Customer
Guest	Guest

How to Run

1. Open the solution in FrameworX Designer
2. Start Runtime
3. Navigate to **OrderInterface** to place an order
4. Add items (Pizza, Breadsticks, Soda) using the + buttons
5. Click **START ORDER** to activate production machines
6. Watch machines cycle through states in the Production Line panel
7. When all machines show "Ready", click **LET'S EAT!** to complete the order
8. Check **InventoryManagement** to see updated stock levels
9. Monitor **TrendPage** for inventory consumption over time
10. View **AlarmPage** when inventory drops below 5 units

In this section...
