Business Requirements Specification

Mario's Pizzeria Automated Ordering Kiosk

Project	Mario's Pizzeria Automated Ordering Kiosk
Client	Mario's Pizzeria (classroom brief)
Delivery team	Shelly Software Co. (student developer, sponsor, AI pair-programmer)
Version	1.2 — adds visual enhancements (section 13) and v1.1 optional features (section 12)
Date	9 September 2026
Status	v1.0 approved for build; v1.1 optional enhancements; v1.2 visual only

Contents

1. Purpose

This document turns the classroom brief into a numbered list of requirements that can be built, tested and ticked off.

2. Background

Mario's Pizzeria wants an automated ordering kiosk. Customers give their name, pick a pizza size, answer three dietary questions, and are shown their total bill. Fast finishers may add a garlic bread side and a delivery fee. The deliverable to the teacher is a single Python script, pizza_bot.py.

Our team (Shelly Software Co.) is also producing two extras to show how requirements travel across technologies: a turtle-graphics drawing of the ordered pizza, and a static website hosting an identical kiosk in the browser.

3. Scope

3.1 In scope

- Text-based ordering flow in Python exactly as the brief describes.
- Both extension features (garlic bread, delivery).
- Input validation so wrong answers re-prompt instead of crashing.
- Turtle drawing of the finished pizza (novelty).
- Static website with showcase page and browser kiosk, hosted at pizza.locbee.ai.

3.2 Out of scope

- Real payments, saving customer records, or a database.
- Multiple pizzas per order, quantities, or a menu beyond the two sizes.
- Any server-side code on the website.

4. Stakeholders

Stakeholder	Interest
Mario (Teacher)	Receives pizza_bot.py; checks it matches the expected output.
Student Developer	Learns if/else, variables, input/print; owns the code.
Project Sponsor (Dad)	Shows how a real project is documented, built and hosted.
Customers (Family testers)	Want a kiosk that is friendly and gets the price right.

5. How the requirements were agreed

Two working practices shaped the list in section 7, and both are named here because they are the parts of this project most worth copying into the next one. Neither is a formality: they changed what got built.

5.1 MoSCoW prioritisation

Every requirement carries one of four labels. Must means the brief fails without it — the seven questions, the prices, the personalised summary. Should means it was explicitly offered as an extension and is expected of a strong submission: the garlic bread side and the delivery fee. Could means it is ours, added

because it makes the project better rather than because anyone asked: the saved order file, Shelly's drawing, the browser kiosk. Won't means agreed as out of scope for this release, which is the label people forget and the most useful of the four — writing down what you are deliberately not doing is what stops it quietly reappearing later. Section 3.2 is this project's Won't list.

The point of the labels is not decoration. If time had run short, the order in which work would have been abandoned was already agreed in writing, by the person who marks it, before any code was written. That is the whole value of prioritising early: the decision gets made calmly instead of at midnight the day before it is due.

5.2 The Three Amigos

Before a requirement was accepted it was looked at from three directions at once, which is a practice commonly called the Three Amigos. The three are not three job titles but three questions, and on this project they map onto people already named in section 4 and in the RACI Matrix.

- Business — Mario (the Teacher), who owns the brief: what problem is this solving, and how will we know it is solved? This is where the $15 worked example comes from, and it is why that single figure became the acceptance test everything else is measured against.
- Development — the Student Developer, who has to build it: is this actually buildable with plain if/else, and does the person writing it understand every line? This is the question that kept the code deliberately simple, and it is why the two helper functions are called out in section 11 rather than slipped in unexplained.
- Testing — the QA Tester (Family), whose job is to break it: what happens when someone types something silly? This is the question that produced FR-10 and test case TC-5. Nobody asked for input validation in the brief; it exists because the testing voice was in the room before the code was written, not after.

The order matters more than the ceremony. Asking the testing question after the code is finished produces a defect report; asking it before produces a requirement. FR-10 is the clearest evidence that this actually happened on this project rather than being described here for the sake of it.

6. Business rules — pricing

All prices are whole Australian dollars. The total is built up step by step, which is why the code uses total_cost = total_cost + N rather than one big formula.

Item	Price	Rule
Small pizza	$10	Base price if size is "small"
Large pizza	$15	Base price otherwise
Gluten-Free base	+$3	Only if answer is "yes"
Dairy-Free (vegan) cheese	+$2	Only if answer is "yes"
Vegetarian toppings	$0	Changes toppings, not price
Garlic Bread side	+$4	Extension — only if "yes"
Delivery	+$5	Extension — only if "delivery"
Supreme toppings (v1.1)	+$3	Optional — topping set choice
Chicken toppings (v1.1)	+$3	Optional — topping set

Item	Price	Rule
		choice
BBQ sauce (v1.1)	+$1	Optional — only if sauce is "bbq"
Extra Crispy crust (v1.1)	+$1	Optional — only if "yes"

Worked example (from the brief): Sarah, small, gluten-free yes, dairy-free yes, vegetarian yes → 10 + 3 + 2 = $15. Cheapest possible order: $10. Most expensive in v1.0: 15 + 3 + 2 + 4 + 5 = $29. With every v1.1 option: 29 + 3 + 1 + 1 = $34.

7. Functional requirements

Priority uses MoSCoW: Must (required to pass), Should (extension), Could (our extras). Optional requirements (v1.1) are listed separately in section 12 and are never part of the school submission.

ID	Requirement	Acceptance criteria	Priority
FR-01	Display a clear, friendly welcome header for Mario's Pizzeria.	Header lines with ==== and kiosk title print first.	Must
FR-02	Ask for the customer's name and store it.	Name is echoed back in the summary.	Must
FR-03	Ask for pizza size: Small ($10) or Large ($15).	"small" sets total to 10; anything else valid sets 15.	Must
FR-04	Ask if a Gluten-Free base is needed.	"yes" adds $3 and prints "Added $3 for Gluten-Free base."	Must
FR-05	Ask if Dairy-Free cheese is needed.	"yes" adds $2 and prints "Added $2 for Dairy-Free cheese."	Must
FR-06	Ask if Vegetarian toppings are wanted.	"yes" prints Mushrooms, Peppers, Olives; otherwise Classic Meat Supreme.	Must
FR-07	Print a personalised order summary with name, dietary notes and total.	Sarah/small/yes/yes/yes prints total $15 exactly as in the brief.	Must
FR-08	Ask if a Garlic Bread side is wanted.	"yes" adds $4.	Should (extension)
FR-09	Ask delivery or pickup.	"delivery" adds $5.	Should (extension)
FR-10	Reject invalid answers and re-ask instead of crashing.	Typing "medium" or "maybe" re-prompts.	Should
FR-11	Save the finished order to last_order.json.	File exists with name, size, options, total.	Could (novelty)
FR-12	Draw the ordered pizza with the turtle module (Shelly).	Large radius > small; GF crust darker; vegan cheese paler; toppings match choice.	Could (novelty)

ID	Requirement	Acceptance criteria	Priority
FR-13	Provide a browser version of the kiosk with identical pricing.	Same inputs give the same total as pizza_bot.py.	Could (showcase)

8. Non-functional requirements

ID	Quality	Requirement
NFR-01	Readability	Code uses plain if/else and descriptive variable names a Year 9 student can explain line by line.
NFR-02	Portability	Runs on standard Python 3.10+ with no third-party packages (turtle and json are built in).
NFR-03	Order of operations	Questions are asked in exactly the order the brief lists them.
NFR-04	Robustness	Upper/lower case and leading spaces in answers are accepted.
NFR-05	Web hosting	Website is fully static (HTML/CSS/JS) so it can be served from Cloudflare Pages or GitHub Pages at pizza.locbee.ai with no server code.
NFR-06	Security posture	Anything a customer types is treated as text, never as code, so it cannot run in another person's browser. The site ships security response headers, carries no secrets, and depends on no third-party code that has not been consciously accepted. Checked by TC-15 before every release.

8.1 What NFR-06 means here

A kiosk with no login, no payments and no database can still be unsafe, so "there is nothing to steal" is not the same as "there is nothing to check". The realistic risk on a site like this is cross-site scripting: the customer's name is the one piece of free text a stranger controls, and it is printed back onto the page, written into the order log, and read out of that log again on the next visit. If it were ever treated as markup rather than text, a name could carry working code into the browser of whoever opened the kiosk next.

The defence is the same in all three kiosks and it is one small function: every value that reaches the page is passed through an escape step that turns the five characters HTML cares about into harmless entities. It is applied at each place the name is printed, including the value put back into the input box and the rows of the stored order log. TC-15 exercises this with real attack strings rather than trusting the code by reading it.

Beyond that, NFR-06 covers three things that are easy to forget on a static site: no credentials or keys anywhere in the files, security response headers served with every page so the browser is told what the site is allowed to do, and a conscious decision about third-party code — the site loads one library from a public CDN, and depending on someone else's server means accepting that if it is ever tampered with, the tampering runs here. Section 5 of the Commissioning Report records what was found and what remains open.

9. Test cases

TC	Inputs (name, size, GF, DF, veg, garlic, delivery)	Expected	Covers
TC-1	Sarah, small, yes, yes, yes, no, pickup	$15	FR-01–07 (brief example)
TC-2	Tom, large, no, no, no, yes, delivery	$24	FR-08, FR-09
TC-3	Ava, small, no, no, no, no, pickup	$10	Minimum total
TC-4	Max, large, yes, yes, yes, yes, delivery	$29	Maximum total
TC-5	Type "medium" then "large" at size	Re-prompt, then $15	FR-10
TC-6	Run pizza_turtle.py after TC-1	Small, dark crust, pale cheese, veg toppings	FR-11, FR-12
TC-7	v1.1: Sarah, small, yes, yes, tomato, vegetarian, no crispy, no, pickup	$15 (regression — v1.1 must not change v1.0 prices)	FR-18–21
TC-8	v1.1: Max, large, yes, yes, bbq, supreme, crispy, yes, delivery	$34	FR-18, FR-20, FR-21
TC-9	v1.1: Ava, small, no, no, bbq, chicken, crispy, no, pickup	$15; one new line in orders.log	FR-17, FR-19
TC-10	v1.1: click through every step on enhanced.html	Shelly reaction + topping drop on each click (visual)	FR-14
TC-11	v1.1: press Email it / Text it on the receipt	mailto:/sms: link carries subject and full receipt	FR-15, FR-16
TC-12	v1.2: open deluxe.html and complete an order (TC-1 case: Sarah, small, yes, yes, tomato, vegetarian, no crispy, no, pickup)	Total is $15; canvas pizza animates through six stages (dough wobble, sauce spiral, cheese melt, toppings drop one-by-one, bake in oven, peel out); Shelly reacts per choice; receipt prints from slot; order logged	FR-22
TC-13	v1.2: complete TC-8 case (Max, large, yes, yes, bbq, supreme, crispy, yes, delivery)	Total is $34; visual rendering matches TC-8 pricing (crispy	FR-22

TC	Inputs (name, size, GF, DF, veg, garlic, delivery)	Expected	Covers
		crust charring, BBQ sauce dark brown); Shelly does delivery reaction (dash); box slides down	
TC-14	v1.2: test animations respect prefers-reduced-motion setting	Canvas and SVG animations and particle systems do not play; scene renders in static end state; flour dust particles do not animate	FR-22 (a11y)
TC-15	Security: enter four HTML/script attack strings as the customer name on enhanced.html and deluxe.html, complete the order, then reload with a hostile record planted in the stored order log	No script executes in any case; the attack string is shown as ordinary text on the receipt and in the log table; no browser dialog appears	NFR-06

10. Traceability

Where each requirement lives in the code and on the website.

Requirement	Python	Website
FR-01	pizza_bot.py — Welcome message block	kiosk.html header
FR-02 / FR-03	pizza_bot.py — name, ask_choice(size)	kiosk.html step 1
FR-04 – FR-06	pizza_bot.py — dietary if/else blocks	kiosk.html step 2
FR-07	pizza_bot.py — Final order summary	kiosk.html receipt
FR-08 / FR-09	pizza_bot.py — Extras block	kiosk.html step 3
FR-10	pizza_bot.py — ask_yes_no(), ask_choice()	kiosk.html buttons (no free text)
FR-11 / FR-12	pizza_bot.py json.dump → pizza_turtle.py	kiosk.html canvas pizza

11. Assumptions and open questions

- Note: the brief shows a straight-line script; ours adds two small helper functions (ask_yes_no, ask_choice) so a wrong answer re-asks instead of crashing. Input validation is normal Year 9 territory, so this is expected to be a plus. If the teacher prefers the plain form, the helpers can be removed and TC-1 still passes.

- Assumption: pizza.locbee.ai is the hostname; change the CNAME file if a different subdomain is chosen.
- Open: does the teacher want the extension questions before or after the summary? We ask them before so the total is correct.

12. Optional enhancements (v1.1)

These were added after v1.0 was accepted, to show how a product grows once the first version works. They live in enhanced.html and pizza_bot_enhanced.py only; kiosk.html and pizza_bot.py are unchanged so the school submission is never at risk.

ID	Requirement	Acceptance criteria	Priority
FR-14	Shelly reacts to every change: a different animation per kind of choice (hop, spin, shake, grow, dash) and a spoken line; toppings drop onto the pizza one by one.	Each button press triggers a visible reaction within 100 ms; toppings animate in over ~0.5 s; reactions are disabled when the OS asks for reduced motion.	Optional
FR-15	Send the receipt by email.	'Email it' opens the device's mail app with the receipt as the body and the total in the subject; address field is optional.	Optional
FR-16	Send the receipt by text message.	'Text it' opens the device's messages app with the receipt pre-filled; works on iOS and Android URL forms.	Optional
FR-17	Keep an order log.	Web: every finished order is added to a log table kept in the browser, with a 'Download orders.log' button. Python: every order appends one line to orders.log. Both use the same one-line format.	Optional
FR-18	Supreme topping set (+$3).	Pepperoni, Ham, Mushrooms, Peppers, Olives, Onion; total rises by $3.	Optional
FR-19	Chicken topping set (+$3).	Chicken, Red Onion, Spinach; total rises by $3.	Optional
FR-20	BBQ sauce option (+$1).	Sauce question offers Tomato ($0) or BBQ (+$1); the drawn sauce turns dark brown.	Optional
FR-21	Extra Crispy crust (+$1).	Crust darkens with char spots; receipt shows '(Extra Crispy)' after the size.	Optional
FR-22	Sponsor request: animate the site. Deliver a deluxe kiosk visual experience (v1.2) that brings the ordering flow to life without	darkened, pizzeria-at-night theme; animated pizza build on canvas with dough	Visual

ID	Requirement	Acceptance criteria	Priority
	changing what it asks or charges.	wobble, sauce spiral, cheese melt, topping drops with bounce; bake-in-oven stage with ember glow; Shelly SVG with reaction animations; receipt printer element; flour dust ambient particles; thermal paper receipt print animation.	

12.1 Design decisions and limits

- Email and text (FR-15, FR-16) use mailto: and sms: links. The site has no server, so it cannot send anything itself — the customer's own Mail or Messages app opens with the receipt filled in. Upgrade path if real sending is wanted: a Cloudflare Pages Function calling an email API (e.g. Resend) with the key stored as a Pages secret; SMS would need a paid gateway such as Twilio.
- The web order log (FR-17) is stored in the browser (localStorage), so it is per device and can be cleared by the user. The Python orders.log is a real file that grows by one line per order. Both use the identical line format so the two can be merged.
- Animations (FR-14) respect the operating system's reduced-motion setting and never block ordering.
- New prices are proposals, not from Mario's brief: Supreme +$3, Chicken +$3, BBQ +$1, Extra Crispy +$1. All v1.0 prices are unchanged, which TC-7 checks.
- pizza_turtle.py is a v1.0 script and is not updated for v1.1: a Supreme or Chicken order is drawn as Meat Supreme, BBQ sauce is drawn as tomato, and Extra Crispy is not shown. Only enhanced.html's canvas drawing reflects the v1.1 options. A v1.1 turtle is out of scope.

13. Visual enhancements (v1.2 — deluxe.html)

After v1.1 shipped, the Project Sponsor asked for the site to be animated — to make the kiosk feel alive rather than simply correct. That request is recorded as FR-22 and produced a third version, deluxe.html, which prioritises visual appeal and theatricality and does not touch ordering logic. All nine questions and all pricing remain identical to v1.1; the only changes are how they are rendered and animated.

13.1 Where this requirement came from

FR-01 to FR-13 are traceable to Mario's original classroom brief. FR-14 to FR-22 are not in the brief at all — they were requested by the Project Sponsor after v1.0 was accepted, and FR-22 is the clearest example: nobody asked for animation, the sponsor wanted it. This is normal on a real project, where a customer sees a working version and asks for more, and it is the reason optional requirements are numbered, tested and signed off in exactly the same way as the ones from the brief rather than being slipped in informally. Keeping them in a separate block is what makes it possible to prove the school submission was never altered.

13.2 v1.2 scope and design decisions

- v1.2 is a visual layer only (FR-22). No new questions, no new prices, no new toppings. Test case TC-7 (Sarah, $15) and TC-8 (Max, $34) run identically on enhanced.html and deluxe.html.
- Theme: pizzeria at night. Dark navy background (#0f151d) with warm ember glow at the top to evoke the wood-fired oven. Lighting and colour palette evoke an Italian pizzeria close to sundown.

- Canvas pizza rendering: six-stage animation sequence. (1) Dough ball with wobble, growing to disc. (2) Sauce spirals from the centre outward; BBQ sauce renders as dark brown instead of red. (3) Cheese melts outward with bubble highlights. (4) Toppings drop one by one, each with bounce and shadow; topping positions and colours match v1.1 rendering. (5) Peel lifts pizza into the oven with an ember-glow interior; crust darkens on bake with char spots if Extra Crispy (+$1) was chosen. (6) Peel comes back out and pizza is plated.
- Animated Shelly: inline SVG turtle (not imported) with per-part animation rigs. Reactions to each question: dough choice (squash), sauce/cheese (flip or nod), toppings (hop or dash), delivery (dash across the screen). Breathing, blinking, and hat-wobble run continuously.
- Receipt printer: printed paper slides out from a slot at the top of the kitchen panel, animated from the centre with a thermal-paper zigzag edge clip-path. Identical receipt format to v1.1 but styled as a printer output.
- Ambient effects: 70 flour-dust particles float down continuously. All animations respect prefers-reduced-motion and disable smoothly; the scene renders in its end state when reduced motion is active.
- Pricing and logic locked from v1.1 so that v1.0 (pizza_bot.py) and v1.1 (pizza_bot_enhanced.py, enhanced.html) can never be broken. deluxe.html lives independently.

14. Glossary

Term	Meaning
Kiosk	A self-service screen customers use to order without staff.
if / else	Code that does one thing when a condition is true and another when it is false.
Variable	A named box that stores a value, e.g. total_cost.
Turtle	Python's built-in drawing module — and our mascot, Shelly.
Static site	A website made only of files (HTML, CSS, JS) with no server program.
mailto: / sms: link	A web link that opens the device's own mail or messages app with the message already filled in.
localStorage	A small store inside the browser that keeps data on that device between visits.
RACI	Responsible / Accountable / Consulted / Informed — see the RACI Matrix document.