COMP1100 — Full Course Notes

Table of contents

Week 1 **2**
Introduction to Innovation . . . 2

Week 2 **3**
Business Model Canvas . . . 3

Week 3 **6**
Experimentation and Customer Relationships . . . 6

Week 4 **8**
Minimum Viable Products and Prototyping . . . 8

Week 5 **9**
Prompt Engineering . . . 9
The History of Computing . . . 11

Reference material **13**
The Business Model Canvas . . . 13
COMP1100 Diagram Notation . . . 15
Customer Discovery . . . 27
Customer Relationships and Channels . . . 30
Experimentation . . . 33
The History of Computing and AI . . . 35
Innovation and Value . . . 41
Large Language Models . . . 43
The Lean Startup Methodology . . . 45
Minimum Viable Products (MVPs) . . . 47
Prompt Engineering . . . 49
Prototyping . . . 52
Test Cards and Learning Cards . . . 54
The Value Proposition Canvas . . . 56

Week 1

Introduction to Innovation

Week 1 seminar. Course logistics are in comp1100[1].

Outline

1. Course overview and expectations
2. What innovation is, and why we need a process
3. The Lean Startup and customer development
4. Customer discovery, step by step
5. Divergent thinking; the project challenges

What is innovation?

See innovation[2] — several definitions, Drucker on value being what the customer gets out (not what the supplier puts in), the product-failure table that makes "no market fit" a 35% risk, and the four shifts from development to innovation.

The four shifts, briefly: from a client to a **market segment**; from a system to a **value proposition**; from development to **experimentation**; from a requirements specification to a **Minimum Viable Product**.

The Lean Startup

See lean-startup[3] — the customer development process (discovery → validation → creation → building), the search/execution split, the pivot, and Paul Graham on the mistake you are going to make.

Customer discovery

See customer-discovery[4] for the full four-step process, the user/customer distinction, the medical alert system worked example, early adopters, and the seven customer development heuristics.

The single sentence to carry out of week 1 is heuristic #1: **there are no facts inside your building, so get outside.**

[1] courses/comp1100/comp1100.pdf
[2] courses/comp1100/innovation.pdf
[3] courses/comp1100/lean-startup.pdf
[4] courses/comp1100/customer-discovery.pdf

Divergent thinking

When generating ideas: don't challenge or criticise; quantity, not quality; don't analyse or organise (but do record and draw). *Then* identify significant themes and refine. See innovation[5].

The challenge

> Improving the student experience at UQ (and beyond?)

Teams pick one or two from the project challenges.

Key takeaways

- Innovation = creating value for someone.
- Failure is often because we do not understand our market.
- Success is more likely if we get out of the building.
- Innovation process = generate and test ideas early fail early.
- Failure = opportunity to learn.

Homework

- **Decide this week** whether you want to do this course — and if not, unenrol.
- Read the course overview.
- Read Chapter 1, Introduction to innovation.
- Read the chapter on Customer discovery.

Next week

Seminar: 2025-08-04-business-model-canvas[6]. Studio: team allocation and team dynamics.

Week 2

Business Model Canvas

Week 2 seminar. See comp1100[7] for course logistics.

[5] courses/comp1100/innovation.pdf
[6] courses/comp1100/2025-08-04-business-model-canvas.pdf
[7] courses/comp1100/comp1100.pdf

Outline

1. The business model canvas — the nine blocks
2. Front stage vs back stage
3. The canvas as a record of customer discovery
4. The value proposition canvas
5. Ground rules for interviewing

The business model canvas

See business-model-canvas[8] for all nine blocks and the question each one answers, the front/back stage split, and — the part that makes it more than a template — how each cell starts as a *hypothesis* and becomes a *fact* as the customer-discovery[9] loop turns.

COMP1100 focuses on the front stage: customer segments, value propositions, channels, customer relationships, revenue streams.

The value proposition canvas

See value-proposition-canvas[10] — the value map and customer profile, the intersection of what your customer wants / what you do well / what your competitors do well, and the six ways to innovate from a customer profile.

Interviewing

See value-proposition-canvas[11] for the seven ground rules (section 4.5 of the course notes). The two that change an interview most:

- **Get facts, not opinions** — don't ask "Would you…?", ask "When was the last time you…?"
- **Always open doors at the end** — "Who else should I talk to?" Always.

Exercise: the front stage of a successful product

As a team, select a successful software product (**not** designed by Apple) and write down the front stage of its business model canvas:

[8] courses/comp1100/business-model-canvas.pdf
[9] courses/comp1100/customer-discovery.pdf
[10] courses/comp1100/value-proposition-canvas.pdf
[11] courses/comp1100/value-proposition-canvas.pdf

Customer segments (pick one)	Value propositions	Channels	Customer relationships	Revenue streams
Who is it for? What are their characteristics? What problems do they have?	What are the features that solve the customers' problems?	How is it delivered to you? Web? Mobile app? Post? Physical stores? Via partners or direct?	How do they build relationships? Support? Updates? Community engagement? Training?	How do they generate revenue? Subscriptions? One-off transactions? Advertising? Selling support?

Key takeaways

- Business model canvas = a way to understand and visualise business models.
- Value proposition canvas = a way to understand and visualise the value proposition and market fit.
- These guide our thinking **and** represent our learnings/lessons.
- When we interview customers to construct profiles, we need to **listen**.

Homework

Team: meet your team in the studio this week (your legend allocates you); hold a meeting outside class to get to know each other; set up a weekly team meeting separate from studio time; select one or two challenges to work on for the semester.

Individually: set up a one-on-one meeting with your supervisor when they contact you.

Next week

Seminar: 2025-08-11-experimentation-and-customer-relationships[12]. Studio: none — public holiday.

[12] courses/comp1100/2025-08-11-experimentation-and-customer-relationships.pdf

Week 3

Experimentation and Customer Relationships

Week 3 seminar, in two topics. See comp1100[13] for course logistics.

Outline

1. Business model canvas (revision) — hypotheses turning into facts
2. Experimentation: why, what makes a good hypothesis, test and learning cards
3. Customer relationships: get, keep, grow
4. Channels

Experimentation

See [experimentation] — the two types (generative/exploratory and evaluative) and two reasons, the scientific method, Steve Blank's seven-step lean startup loop, the six data collection methods, and what experiments should look like.

Falsifiability is the whole game. An exercise: design a *non*-falsifiable hypothesis and say why it isn't falsifiable. For example —

> "This solution will help this segment of customers improve their health over the long term."

— then convert it into one that can come back false.

Test cards and learning cards

See test-cards-and-learning-cards[14] for both formats and four worked examples from the medical alert system, including the rejected hypothesis that produced a more valuable side-finding than the accepted one.

[13] courses/comp1100/comp1100.pdf
[14] courses/comp1100/test-cards-and-learning-cards.pdf

Customer relationships

See customer-relationships-and-channels[15] — the get/keep/grow lifecycle with its viral loop, the five foundational questions, the awareness→interest→activation funnel and its arithmetic, paid vs earned demand creation, and the churn table.

The churn numbers are the ones worth remembering: at 1% monthly churn you keep 69.8% of customers after three years; at 10% you keep 3.14%.

Key takeaways

- Experiments = gain insights about customers and products.
- Hypotheses must be **falsifiable**.
- Test cards structure our hypotheses; learning cards structure our learnings.
- Refuted hypothesis ask **why**.

Housekeeping

- All non-personal communication with your legend goes through **MS Teams**; general questions to 'General'.
- **One-on-one meetings: you must sign up and attend.** This is a participation requirement — it's a meeting with a supervisor.

Homework

As a team: agree on which project challenges to tackle (can be more than one). Derive a set of hypotheses and **5–10 interview questions** to understand the problems UQ students have related to your challenges — one set for the whole team.

Individually: interview **at least 2 customers each** (not family members or close friends), giving 8–10 interviews across the team. It's fine to go out in pairs, but everyone does their own two. For each: (a) **ask for consent**, (b) **record** the interview, (c) **transcribe** it to text with the software from this week's activity.

As a team, before next week's studio: analyse the interview data. What are the common themes? The common pains? Are there particular segments of the UQ student population that have these pains? Then revise your interview questions based on what you learnt — did the interviews suggest new hypotheses?

[15] courses/comp1100/customer-relationships-and-channels.pdf

Next week

Seminar: 2025-08-18-minimum-viable-products-and-prototyping[16].

Week 4

Minimum Viable Products and Prototyping

Week 4 seminar. See comp1100[17] for course logistics.

Outline

1. Submissions and feedback
2. Minimum viable products — what they're for, the six principles, the types
3. Prototyping — the four axes, paper prototypes, wireframes, Wizard of Oz
4. The economics of design

Minimum viable products

See minimum-viable-products[18] — what an MVP is for (it exists **to test the ability of the product to meet minimal customer needs**, not to earn revenue and not to find bugs), the six principles, the five types, and the prototype-vs-MVP table.

The distinction to hold onto: an MVP is **not a full product, but it is a product, not a prototype**. A prototype is shown to people; an MVP is released to them.

> "By far the dominant reason for not releasing sooner was a reluctance to trade the dream of success for the reality of feedback."
>
> — Kent Beck, *Approaching a Minimum Viable Product* (2009)

Chapter 6 of the course notes covers this.

[16] courses/comp1100/2025-08-18-minimum-viable-products-and-prototyping.pdf
[17] courses/comp1100/comp1100.pdf
[18] courses/comp1100/minimum-viable-products.pdf

Prototyping

See [prototyping] — prototyping as a form of experimentation inside the design cycle, the four axes (representation, precision, interactivity, evolution), paper prototyping, wireframes, Wizard of Oz, and the economics-of-design cost ladder.

The ladder is the practical takeaway: low fidelity prototypes are *super cheap* and focus on flow and interactivity; high fidelity ones are *quite cheap* and focus on look, feel and usability; code prototypes are *more expensive* and focus on code feasibility; the MVP is *most expensive* and focuses on getting to market. Spend the cheapest amount that answers your question.

Have code generation tools killed wireframing?

An open question, with arguments on both sides — see [prototyping]. The strongest point for "no": wireframes are *intentionally* low fidelity, to get feedback on the user **experience** rather than the user **interface**.

Key takeaways

- MVP = the most important features.
- MVP purpose: **experiment!**
- Simple and easy to build.
- Different MVPs allow us to run different experiments.

Next week

Seminar: 2025-08-25-the-history-of-computing[19] and 2025-08-25-prompt-engineering[20] — the last seminars of the semester.

Week 5

Prompt Engineering

Week 5b seminar: the second half of the week 5 session and the **last seminar of the semester**. Everything from week 6 on is studio and project work. See comp1100[21] for course logistics.

[19] courses/comp1100/2025-08-25-the-history-of-computing.pdf
[20] courses/comp1100/2025-08-25-prompt-engineering.pdf
[21] courses/comp1100/comp1100.pdf

The AI policy

> Under what circumstances can you use generative AI or other related tools to complete your work in COMP1100?
>
> a) None. It is prohibited.
> b) For coding only — no use for writing hypotheses, test cards, etc. **c) For anything except generated interview or other data.**
> c) For any task.

The exception is the important part, and large-language-models[22] explains why: a model will fluently generate plausible interview data, and fabricated data defeats the entire purpose of customer-discovery[23].

Outline

1. What is machine learning?
2. Large language models — how they work, and where they break
3. Prompt engineering: seven tactics
4. LLMs as tools for thought

Machine learning and LLMs

See large-language-models[24] — curve fitting as the intuition for machine learning, next-token prediction with the "The woman ran up the _____" probability tables, and the apples example where the model subtracts 300 from both the red and the green count and reports −99 green apples without noticing.

Prompt engineering

See prompt-engineering[25] for all seven tactics with worked prompts:

1. Use delimiters
2. Ask for structured output
3. Ask it to check whether conditions are satisfied
4. Specify the steps required
5. Iterate
6. Be as specific as possible

[22] courses/comp1100/large-language-models.pdf
[23] courses/comp1100/customer-discovery.pdf
[24] courses/comp1100/large-language-models.pdf
[25] courses/comp1100/prompt-engineering.pdf

7. Build code incrementally

LLMs as tools for thought

The most transferable idea is to ask the model to *interrogate you* rather than answer you:

> I am doing a project using the Lean Startup methodology. I have a hypothesis about my solution that I want to test. I would like to construct a test card for it. Can you please interact with me to help me construct this? **Do not give me answers, just ask a series of questions** until I have the test card. Provide feedback on my answers.

The resulting test card[26] contains your team's hypotheses, not generated ones — which is what keeps it on the right side of the AI policy above.

Homework

Teams: by the submission next Thursday 4:00pm you should have a clear idea of your customer segments and their problems, have those hypotheses formalised as test cards, have tested them, and recorded results as learning cards. You should be starting to formulate what solutions you could provide — some teams may have tested solutions already.

So in week 5 you should be:

- Running final tests for hypotheses related to customer segments and their problems.
- Analysing interview data for common problems and pain points.
- Starting to brainstorm potential solutions leading to a unique value proposition.
- Preparing your repository to submit iteration 1 — see the submission requirements under 'Assessment' and in the week 5 studio handout.

Individuals: complete your interviews about your problem hypotheses, record them (with consent), have them transcribed, then commit and push the transcripts to your GitHub repository so the team can use them. **Also participate in the data analysis** — that's expected of everyone in the team.

The History of Computing

Week 5a seminar. The second half of the same session is 2025-08-25-prompt-engineering[27]. See comp1100[28] for course logistics.

[26] courses/comp1100/test-cards-and-learning-cards.pdf
[27] courses/comp1100/2025-08-25-prompt-engineering.pdf
[28] courses/comp1100/comp1100.pdf

Outline

1. In what year was the first programmable computer built?
2. The eras of computing and AI — the golden age, two winters, the revival
3. Another AI winter?
4. History and representation

The history

See history-of-computing-and-ai[29] for the full reference: the six defensible answers to when the first programmable computer was built (87 BCE through 1951), the era-by-era timeline, the Turing Test, Dartmouth 1956, both AI winters and what went wrong in each, the revival, and six people the field's history remembers.

Why this is in an innovation course

This is more than history. The argument is the pattern:

- **1956:** "Machines will be capable, within twenty years, of doing any work a man can do." — Herbert Simon
- **1958:** the Perceptron is "the embryo of an electronic computer that will be able to walk, talk, see, write, reproduce itself and be conscious of its existence." — *New York Times*
- **1970:** "In from three to eight years we will have a machine with the general intelligence of an average human being." — Marvin Minsky
- **1999:** "Artificial intelligence will reach human levels by around 2029." — Ray Kurzweil
- **2014:** "We will have fully self-driving cars on the road by 2017." — Elon Musk
- **2024:** "We are now confident we know how to build AGI as we have traditionally understood it." — Sam Altman

Twice, that pattern of overpromising ended in a funding collapse. **Amara's Law** names the shape of it:

> We tend to overestimate the effect of a technology in the short run and underestimate the effect in the long run.

For a course about testing your hypotheses[30] rather than believing them, this is the field's own cautionary tale — the hype cycle is what happens when nobody makes a falsifiable claim.

> "By far the greatest danger of Artificial Intelligence is that people conclude too early that they understand it." — Eliezer Yudkowsky

[29] courses/comp1100/history-of-computing-and-ai.pdf
[30] courses/comp1100/experimentation.pdf

Representation

Six people the field's history remembers: Ada Lovelace, Alan Turing, Margaret Hamilton (who coined the term *software engineering*), Grace Hopper, Fei-Fei Li and Timnit Gebru. See history-of-computing-and-ai[31].

Next

2025-08-25-prompt-engineering[32] — the second half of the same session.

Reference material

The Business Model Canvas

Strategyzer's **Business Model Canvas** (Alexander Osterwalder) is a one-page way to understand and visualise a business model, split into nine blocks. In COMP1100 it doubles as the artefact the team submits — it holds both the current thinking *and* the record of what has been learned.

The nine blocks

Block	The question it answers
Customer segments	Which customers and users are we serving? Which are their problems/pains?
Value propositions	What are we offering them? What is getting done for them — do they care?
Channels	How does each customer segment want to be reached? Through which interaction points, and how do we deliver this?
Customer relationships	What relationships are we establishing with each segment? Personal? Automated?
Revenue streams	What are customers really willing to pay for? Are we generating transactional or recurring revenues?
Key resources	Which resources do we need to deliver our business model? Which assets are essential?

[31] courses/comp1100/history-of-computing-and-ai.pdf
[32] courses/comp1100/2025-08-25-prompt-engineering.pdf

Block	The question it answers
Key activities	Which activities do we need to perform to deliver our business model? Which are crucial?
Key partners	Which other organisations do we rely on to deliver our business model?
Cost structure	What are the expenses we incur delivering our business model? Salaries, rent, services, technology subscriptions?

Front stage and back stage

The canvas divides down the middle:

- **Front stage** — customer segments, value propositions, channels, customer relationships, revenue streams. Everything that faces the customer.
- **Back stage** — key partners, key activities, key resources, cost structure. Everything that makes delivery possible.

COMP1100 focuses on the front stage. The five questions that matter here: who are our customers, what are we offering them, what relationships are we establishing with them, how do we deliver our service to them, and will they pay for this — and how?

The canvas as a record of learning

This is the part that makes the canvas more than a planning template. Every cell starts life as a **hypothesis** and becomes a **fact** only once it has been tested — so the canvas visibly tracks progress through the customer-discovery[33] loop:

Stage	State of the canvas
1. Define hypotheses	Every filled cell says *Hypothesis*
2. Test the problem	Customer-segment and relationship cells start turning into *Fact*
3. Test the solution	Value-proposition cells turn into *Fact*
4. Verify, pivot, or refine	A mix — and the remaining hypotheses are the next round's work

A canvas that is still all hypotheses after several weeks is itself the finding.

[33] courses/comp1100/customer-discovery.pdf

A template is available from the course notes: business_model_and_testing_templates.pptx.

Key takeaways

- The business model canvas is a way to understand and visualise business models.
- The value-proposition-canvas[34] is a way to understand and visualise the value proposition and market fit.
- These **guide our thinking AND represent our learnings**.

COMP1100 Diagram Notation

The authority for how diagrams are drawn in COMP1100. Every convention below was inferred from the lecture decks themselves — **this course ships no written notation guide, style guide or drawing standard.** Nothing here is quoted from a document; it is all read off the drawings.

How to read this note

Citations. `L02 p7` means *lecture deck 02, page 7 of the PDF*. Printed slide numbers drift from PDF page numbers in several decks (L03 PDF p40 is printed "41"), so the PDF page is the anchor.

Deck keys:

Key	Deck
L01	Lecture 01 — Introduction and customer discovery (57 pp)
L02	Lecture 02 — Business model canvas (39 pp)
L03	Lecture 03 — Experimentation and customer relationships (61 pp)
L04	Lecture 04 — Prototyping and MVPs (27 pp)
L05a	Lecture 05a — History of computing (46 pp)
L05b	Lecture 05b — Prompt engineering (28 pp)

Consistency labels. Each convention says how well it holds: *throughout* (seen in three or more decks), *repeated* (two decks, or many slides in one), *single deck*, or *once only* — the last meaning I am extrapolating from one instance and you should treat it as weak.

Colours are sampled from the rendered PDF pages, not from a style sheet, so they are the values the deck actually paints, not values anyone documented.

[34] courses/comp1100/value-proposition-canvas.pdf

What I sampled

I read **133 of the 258 pages** as rendered images, chosen to cover every page that carries a drawing, spread across all six decks rather than concentrated in one:

Deck	Pages read as images	Of
L01	24–33, 41, 47–52	57
L02	4–39	39
L03	10–24, 29–35, 40, 45–57	61
L04	6–14, 17–27	27
L05a	13–21, 26, 32–35	46
L05b	5–7, 15–21	28

I also extracted the text layer of **all 258 pages** to locate figure-bearing pages and captions, so no attribution line in the semester went unseen even where I did not render the page. Pages I did not render are overwhelmingly title cards, Acknowledgement of Country, bulleted-text slides, quiz slides, photographs and housekeeping.

Source lineage

The course names its sources. This matters because most of the notation is inherited rather than invented.

Strategyzer (Alexander Osterwalder) is the source for every canvas and card. The business model canvas figure is captioned `Source: https://www.strategyzer.com/` (L02 p7); the test card is captioned "The structure of a test card. Adapted from: https://www.strategyzer.com/" (L03 p29) and the learning card identically (L03 p33). *Throughout* — these are the only attributions given for the canvas and card shapes, in two different decks.

Steve Blank is the source for the process models. The experimentation pipeline carries "The process model for experimentation, according to Steve Blank. Adapted from: https://steveblank.com/2014/10/24/17577/" (L03 p17). *Once only* as an explicit caption, but the customer-development chain of L01 p28 is Blank's four-stage model unattributed.

L01 p27 is the closest thing this course has to a bibliography, an "Inspiration and Sources" block naming four authors and their books: Tina Seelig (*inGenius*), Steve Blank (*The Four Steps to the Epiphany*, *The Startup Owner's Manual*), Alexander Osterwalder (*Business Model Generation*), Eric Ries (*The Lean Startup*). If a figure needs a textbook to conform to, these are the four.

The course's own notes site is the working authority for canvas layout. L02 p19 and L02 p32 point at `comp1100.github.io/innovation/4_value_proposition_canvas.html`, and L02 p20 carries a download link to `comp1100.github.io/_downloads/.../business_model_and_testing_`

The greyscale template slides (below) are pages of that PowerPoint file. *Repeated* (L02, L03). **Anything drawn for assessment should match the template file, not the hand-drawn Strategyzer illustration.**

Third-party figures are credited inline where reused: `Source: https://clevertap.com/blog/minimum-viable-product/` (L04 p6), "All figures by Unknown Authors: Licensed under CC BY-SA" (L04 p14), "This Photo by Unknown Author is licensed under CC BY-SA-NC" (L03 pp14, 37–39), `Image source: <url>` (L01 p29, L01 p48, L05a p34).

Two visual registers

The decks switch deliberately between two looks, and which one you are in determines colour, typography and whether the UQ logo appears.

1. Teaching slides — UQ purple. Either a purple gradient ground (roughly `#241242` at the left through `#4C2273` to pale mauve at the right) with white type, or a light-grey ground `#F2F2F2` with purple type `#51247A`. UQ logo top right, slide number bottom right. Slide titles are **ALL CAPS**. *Throughout.*

2. Working-artefact slides — greyscale. Ground `#E6E6E6`, no purple, black rules, title in **bold black sentence case at the top left** ("Business model canvas", "Value proposition canvas", "Test card", "Learning card"). These are the downloadable template pages. L02 pp20, 23–26, 30, 31; L03 pp29, 33, 57. *Repeated.* Note: L03 pp11–12 render the same BMC template *with* the UQ logo and with dark-indigo rather than pure-black rules — see Inconsistencies.

L05a and L05b add a third look on some slides: light-grey ground with **navy** `#094183` bold ALL-CAPS titles and navy figure captions (L05a pp15, 26, 32–35; L05b pp5–7, 15–17, 21). *Repeated*, but see Inconsistencies — the same two decks also use purple titles.

The colour system

One ordered palette carries almost all the semantic colour in the course.

Role	Value	Where
Aubergine (step 1 / most fundamental / widest)	`#241242`, `#1F0E3C`	L01 p24, p31, p50; L03 p16, p55; L04 p17, p25
Blue (step 2)	`#0552A2`, `#004C97`	L01 p24, p31, p50; L03 p16; L04 p17, p25
Teal (step 3)	`#1A688C`, `#156082`	L01 p24, p31, p50; L03 p16, p55; L04 p17, p25
Green-teal (step 4 / end state)	`#259191`	L01 p24, p31; L03 p16; L04 p17, p25

Role	Value	Where
Pink	`#FD7CEC`, `#FF73E0`	L01 p50; L03 pp47–55; L04 p17
Magenta	`#A0007C`	L01 p50; L03 pp51–55
Red (refuted only)	`#FD0000`	L02 pp24–26; L03 p12
Purple accent for text on light grounds	`#51247A`	*throughout*

Ordered sequences run aubergine → blue → teal → green-teal. Four-item sequences use exactly these four, in this order, whether laid out as a chain (L01 p24 Concept / Product Development / Test / Launch; L01 p28 Customer Discovery / Validation / Creation / Company Building) or as a cycle (L01 p31 and L02 p22, the customer discovery process). *Throughout* — three decks. **Green-teal reliably marks the end state**: it is "Launch" (L01 p24), "Company Building" (L01 p28), "4. Verify, pivot, or refine" (L01 p31), and "MVP" in both L04 p17 and L04 p25.

Pink and magenta extend the ramp when more than four bands are needed — the Rogers adoption curve (L01 p50: aubergine, blue, teal, pink, magenta) and the get/keep/grow funnel (L03 p55). *Repeated.*

Red plus a struck-through X means "refuted". Nothing else in the course is red. *Repeated* (L02 pp24–26, L03 p12).

Front stage / back stage is a fixed two-colour split: back stage is mint `#91C9C8` with its label in teal `#249090`; front stage is lavender `#A98FBC` with its label in purple. Labels sit centred beneath each half in small caps. L02 p17 and L03 p10 — *repeated*, two decks, identical treatment.

Within the experimentation pipeline a separate two-colour code applies (L03 pp17–24): **lime `#BCD652`** for hypotheses, test cards and the run-tests loop; **cyan `#15B6DC`** for canvases (the BMC, value map and customer profile icons) and learning cards. So lime = things you are still guessing at, cyan = things that hold knowledge. *Single deck*, but eight consecutive slides.

Shapes

Shape	Means	Citation	Consistency
Rounded rectangle, 3-D bevelled, white text	one step of a named process	L01 p24, p31, p32, p47, p49; L02 p22–26; L03 p16	*throughout*
Circle / sphere, white text	one stage in a chain or cycle	L01 p28; L04 p17, p25	*repeated*

Shape	Means	Citation	Consistency
Circle with a ring of arrowheads around it	an *iterative* stage	L01 p28, p30	single deck
Pyramid, three horizontal bands	a hierarchy, most fundamental at the base (Vision / Strategy / Product)	L01 p49; L03 p40	*repeated*, identical both times
Three overlapping circles	a Venn intersection; the intersection is the thing being defined	L02 p28, L04 p11 (identical figure: "Your value proposition")	*repeated*
Trapezoid narrowing left→right	a funnel stage losing volume ("Get")	L03 pp45, 47–52, 55	*single deck*, many slides
Rectangle of constant width	a funnel stage holding volume ("Keep")	L03 pp50–52, 55	single deck
Trapezoid widening left→right	a funnel stage gaining volume ("Grow")	L03 pp53–55	single deck
Bell curve split into vertical bands	a population split by proportion	L01 p50	once only
Flat coloured square/rectangle, no border	a bare category card, no relation implied	L01 p26, p27	single deck
Large grey rounded panel with a darker grey header bar, white text	one of two things being contrasted; the header bar names it in caps, the body describes it in sentence case	L01 p41 (USER / CUSTOMER), p47 (CUSTOMER / PRODUCT)	single deck, twice
Filled black circle	start and end node of a flow (UML-style)	L05a p17	once only
Dashed rectangle enclosing nodes	a subsystem grouping	L05a p17	once only
Rounded rectangle, white outline, enclosing other nodes	a named phase grouping ("Search", "Execution")	L01 p28	once only
Red octagonal STOP sign on an edge	a gate that must be passed before the next stage	L01 p28, p30	single deck

The sticky-note card is the course's signature symbol. A rectangle with the bottom-right corner turned up, and a black pushpin at the top right, placed *inside* a business model canvas

block. Its fill states what is known:

Fill	Text	Means	Citation
White, dark outline	`Hypothesis`	untested	L02 p23, L03 p11
Blue `#0552A2`	`Hypothesis`, plus a grey outline chevron at its right	currently under test	L02 p24, L03 p12
Blue `#0552A2` / teal `#1A688C`	`Fact`	confirmed	L02 pp24–26, L03 p12
Red `#FD0000` with a black X drawn corner to corner	`Hypothesis`	refuted	L02 pp24–26, L03 p12
Green-teal `#259191`	`Hypothesis`	newly raised after a pivot	L02 p26

Repeated — two decks, six slides, with the fill of a resolved card matching the colour of the customer-discovery stage the slide is currently in.

Stage marker. A slide elaborating one step of the customer discovery cycle repeats that step's node, in that step's colour, off to one side: as a large rounded rectangle in the right third on the purple teaching slides (L01 p32, p41, p47, p49) and as a small badge in the top-right corner on the greyscale template slides (L02 p17, p23, p24, p25, p26). *Repeated*, two decks, nine slides.

Lines and arrows

Flow direction is left to right for chains, clockwise for cycles. Chains: L01 p24, p28; L03 p17–24; L04 p25. Cycles: L01 p31, L02 p22 and L03 p16 all run clockwise. *Throughout.* (Only L01 p31 and L02 p22 add an entry arrow from the left and an exit arrow to the right; L03 p16 draws the cycle closed, with neither.)

Line	Means	Citation	Consistency
Thin black straight arrow, solid triangular head	"then" — the next stage	L03 pp17–24; L04 p25; L05a p35	*repeated*
Grey 3-D block arrow	"then", on the purple teaching slides	L01 p24, p28, p31; L01 p49 and L03 p40 (pointing out to a label)	*repeated*

Line	Means	Citation	Consistency
Thick curved arrow returning right→left	a feedback loop, labelled on the curve ("Viral loop")	L03 pp45, 47–52, 55	single deck, many slides
Two arrows pointing at each other, → ←	*fit* between two things; the word "FIT" sits above them	L02 pp30, 31; L03 p57	*repeated*
One double-headed arrow	also *fit*, labelled "Product/market fit"	L01 p47	see Inconsistencies
Thin diagonal leader line from a caption to its target	an annotation pointing at part of a figure	L01 p28 ("Our focus", "And a bit of this.")	once only
Dashed thin line, no arrowhead	attaches a sticky note to the canvas block it belongs to	L03 pp17–19	single deck
Dashed vertical line, no arrowhead	maps a canvas block to the sub-canvas that expands it	L02 p31	once only
Dashed arrow	backtracking in a search tree, as against solid arrows for descent	L05a p19	once only
Plain straight line, no arrowhead, parent above children	a taxonomy / is-a hierarchy	L05a p26	once only

Green double-headed vertical arrow with + at the top and − at the bottom means "prioritise these along this axis" — items are placed along the shaft. L03 pp17, 19, 20, 21. *Single deck*, four slides, entirely consistent.

Named artefacts and their fixed layouts

Business model canvas (L02 p20, the template; also pp23–26, L03 pp11–12, L03 p57). Nine blocks in a fixed grid, black rules on `#E6E6E6`: left to right — Key partners (full height) | Key activities over Key resources | Value propositions (full height) | Customer relationships over Channels | Customer segments (full height); then a bottom band of Cost structure (left half) and Revenue streams (right half). Each block carries a pale grey watermark pictogram in its lower-right corner (handshake, clipboard, books, diamond, headset, megaphone, people, cheque, coins). Block names are **sentence case**, centred at the top of the block. *Throughout* — this exact layout appears in three decks and never varies.

Value proposition canvas (L02 p30). Two shapes side by side. **Left, the value map**: a square with diagonals from all four corners to a small central square; the regions read "Products/Services" (left), "Gain creators" (upper right), "Pain relievers" (lower right). **Right, the customer profile**: a circle divided into three sectors by lines radiating from a small central circle; "Customer jobs" (right), "Gains" (upper left), "Pains" (lower left). A smiley watermark sits in the gains sector and a frowny in the pains sector. The two shapes face each other with → ← between them. *Repeated* (L02 pp29, 30, 31).

Test card (L03 p29). Greyscale, two blank cards side by side separated by a vertical black rule. Field labels in **small grey caps**, each followed by a sentence stem in black on the line below, fields separated by horizontal black rules:

- `HYPOTHESIS` — "We believe that" — with `Critical:` at the right, rated by three warning triangles of increasing size
- `TEST` — "To verify that, we will"
- `METRIC` — "And measure" — with `Time required:` rated by three clocks of increasing size
- `CRITERIA` — "We are right if"
- `INTERVIEW QUESTIONS` — "Therefore, we will ask"

Increasing icon size means increasing magnitude on every rating scale in the course (triangles, clocks, thumbs, tick boxes). *Repeated*, four scales across two card templates.

Learning card (L03 p33). Same idiom. `HYPOTHESIS` — "We believed that" (past tense, deliberately, against the test card's present); `OBSERVATION` — "We observed"; `LEARNING AND INSIGHTS` — "From that we learnt that", with `Data reliability:` rated by three thumbs-up; `DECISIONS AND ACTIONS` — "Therefore, we will", with `Action required:` rated by three tick boxes.

Get / keep / grow funnel (L03 p55, built up over pp45–55). One horizontal band, left to right, mirror-symmetric about the "Keep" rectangle. Band colours are fixed by position, not by name: aubergine at the wide outer ends (Awareness, Referrals), teal next in (Interest, Cross-sell), pink at the narrow neck (Activation, Up-sell), magenta for the Keep rectangle (Loyalty programs). Stage- group names — "Get", "Keep", "Grow" — sit above the funnel in large purple; stage names sit inside the bands in white. Viral loops arc back underneath in thick teal and are labelled along the curve. *Single deck*, but nine consecutive slides.

Four-step cycles (L01 p31, L02 p22, L03 p16). Four rounded rectangles at 12, 3, 6 and 9 o'clock, joined by a thick grey 3-D ring broken between the boxes, an entry block arrow from the left, and where there is an exit, a block arrow to the right into a small ring-arrow circle. Steps are numbered `1.`–`4.` inside the box. *Throughout.*

Labels, capitalisation, naming

- **Slide titles are ALL CAPS.** *Throughout*, every deck.

- **Block, node and stage names inside diagrams are sentence case** — "Key partners", "Value propositions", "Design value propositions", "Loyalty programs", "Test the problem". *Throughout.*
- **Numbered steps carry their number in the label**: `1. Define hypotheses`, `2. Test the problem`, `3. Test the solution`, `4. Verify, pivot, or refine` (L01 p31, L02 p22). The same wording is reused verbatim on the stage markers and badges. *Repeated.*
- **`#n` is the enumerator for unordered lists of items** — `#1 #2 #3 #4` on L01 p10, L01 p27 and L03 p26. Not `1.`, which is reserved for ordered process steps. *Repeated*, two decks.
- **Australian/British spelling**: "prioritise", "visualise", "organise", "analyse", "recognises". *Throughout.*
- **Attribution captions sit below the figure**, centred or left-aligned, in the slide's accent colour (purple on teaching slides, navy in L05a/L05b): `Source: <url>` (L02 p7, L04 p6), `Adapted from: <url>` (L03 pp17, 29, 33), `Image source: <url>` (L01 p29, L05a p34). *Throughout.*
- **In L05a, technical figures are captioned `<Technique> - <Authors> (<Year>)`** centred beneath the figure in navy: "Single-layer perceptron – Rosenblatt (1958)" (p21), "Prolog – Colmerauer and Roussel (1972)" (p26), "Bayesian Networks – Pearl (1988)" (p32), "Backpropagation in deep neural networks – Rumelhart, Hinton, and Williams (1986)" (p35). *Single deck*, five figures.
- **Two-way comparisons are drawn as two equal panels side by side**, heading centred at the top of each panel in caps, items left-aligned beneath: "YES? / NO?" (L04 p26), "PROTOTYPE / MVP" (L04 p27), "PAID / EARNED" (L03 p49). *Single deck plus one*, three slides.

Tables

The course uses tables constantly but **does not have one table style** — see Inconsistencies. What *is* stable:

- **The header row is always bold and always distinguished by fill** from the body rows. L01 p25, L02 p32, p33, p35, L03 p52. *Throughout.*
- **Two-column explanatory tables are headed `<Thing> | Explanation`** — "Strategy | Explanation" (L02 p32), "Ground rule | Explanation" (L02 p33). *Single deck.*
- **Percentages are written with a `%` sign and no decimal unless needed** (38%, 3.14%, 13.5%). *Repeated* (L01 p25, p50; L03 p52).
- **Proportions attached to a category name go in parentheses on the line below**: "Early Adopters (13.5%)" (L01 p50). *Once only.*

Mathematical and code notation

- **Variables are italic; subscripts are used for indices**: $i \;\; i \;\; i \;\rightarrow o$ (L05a p21); $x \ldots x$, $w \ldots w$, *net* , , o (L05a p35). *Repeated.*
- **Model equations are set in bold italic and used as the chart title**: `y = ax + b`, `y = ax²  + b`, `y = ax + by + c` (L05b pp5–7). Exponents are true superscripts. *Single deck*, three consecutive slides.
- **Arithmetic is shown inline with the working, not just the answer**: `(1−0.01)³ = 69.8%`, `(1/0.01) = 100 months` (L03 p52). *Once only.*
- **Conditional probability uses `P(X|Y)` and `~` for negation**: `P(C|J,~S) = 0.9` (L05a p32). *Once only* (an imported figure).
- **Charts**: axis titles in bold letter-spaced caps in the accent colour, chart title small and centred above, data as small filled round markers, the fitted line thicker than the data (L05b pp5–7, L05a p33). A fitted *surface* is a translucent teal plane (L05b p7). *Repeated.*
- **Prompt and template placeholders are `[SQUARE BRACKETS IN CAPS]`**: `[DATE]`, `[NAME]`, `[NUMBER]`, `[PARTS]`, `[SUMMARY]` (L05b pp19–24). In running prose the same idea is bold plus brackets: `[target]`, `[undertake behaviour]` (L03 p30). *Repeated.*
- **Delimiters shown to students are `"""triple quotes"""` and `$$$`** (L05b p19). *Single deck.*
- **Bold marks the part being added or chosen**: the clause a tactic adds to a prompt (L05b p21), the highest-probability next token (L05b pp15–16), the correct quiz answer on the reveal slide (L04 p8). *Repeated*, three decks.
- **`_____` (a blank underline) marks a slot to be predicted or filled** (L05b p17). *Once only.*
- **Code is monospaced in a bordered box** with syntax colouring — blue predicates, red constants (L05a p26); plain monospace with `//` comments (L05a p20). *Single deck.*

Inconsistencies

Places where COMP1100 does the same thing more than one way. **These need a decision before a new figure is drawn; do not silently pick one.**

1. **Does the four-colour ramp encode step number, or position on the circle?** L01 p31 and L02 p22 give step 1 aubergine, 2 blue, 3 teal, 4 green-teal — but they also place step 1 at 9 o'clock. L03 p16 (the scientific method) gives step 1 blue, 2 teal, 3 green-teal, 4 aubergine, and places step 1 at 12 o'clock. The colours are bound to clock position in both, so the *step numbers* get different colours in the two figures. Decide which the rule is.

2. **Aubergine-then-blue, or blue-then-aubergine, in a linear chain?** L01 p24 runs Concept (aubergine) → Product Development (blue) → Test (teal) → Launch (green-teal). L04 p25 runs Low fidelity prototypes (blue) → High fidelity prototypes (aubergine) → Code prototypes (teal) → MVP (green-teal). Same four colours, first two swapped.

3. **A "Fact" card's colour is not stable across the build.** On L02 p24 the two confirmed facts in Customer segments are bright blue `#0552A2`; on L02 p25 and p26 the same two cards are teal `#1A688C`. So it is unclear whether a card takes the colour of the stage at which it was resolved, or of the stage the slide is currently showing.

4. **Two different arrow idioms for "fit".** L02 pp30/31 and L03 p57 use two separate arrows pointing at each other, → ←, with the word "FIT" above. L01 p47 uses a single grey double-headed arrow labelled "Product/market fit". Same relation, different glyph.

5. **A chain of stages is drawn with rounded rectangles in one deck and circles in another.** L01 p24 and L01 p31 use rounded rectangles; L01 p28, L04 p17 and L04 p25 use circles. Nothing distinguishes the cases.

6. **Dashed lines mean three different things.** Leader line attaching a sticky to a canvas block (L03 pp17–19); mapping between two canvases (L02 p31); backtracking in a search, opposed to solid descent (L05a p19). A reader cannot tell which from the line alone.

7. **The value proposition canvas's left half has two names.** "Value proposition map" on L02 p29 and in the thumbnail on L02 p31; "Value map" on the full-size template L02 p30.

8. **The learning card's third stem changes between template and worked example.** The template says "From that we learnt that" (L03 p33); both worked examples say "From this we learnt that" (L03 pp34, 35).

9. **At least four table styles, with no rule for choosing.** Dark-purple header with white bold caps and pale lavender body (L02 p35); teal-green header with dark bold sentence case and alternating pale-teal body (L02 pp32, 33); dark-purple header with mid-purple body and thin cyan cell rules (L01 p25); white caps on purple with horizontal white rules only and no fill (L03 p49); purple bold sentence-case header on light grey with alternating pink and teal cell rules (L03 p52).

10. **Slide titles use two colours in the same deck.** L05b has navy `#094183` bold letter-spaced caps (pp5–7, 15–17, 21) and purple `#51247A` non-bold caps (pp19, 20). L05a likewise mixes navy (pp15, 26, 32–35) and purple (pp13, 17, 19).

11. **The BMC template is rendered two ways.** L02 p20 and pp23–26: no UQ logo, pure black rules, black card outlines. L03 pp11–12: UQ logo present, dark-indigo rules and card outlines, larger type.

12. **The pushpin on a sticky card is sometimes omitted** — the third Customer segments hypothesis has no pin on L02 p23 or L03 p11, while every other card on those same slides has one.

13. **En dash and double hyphen are both used in figure captions.** "Single-layer perceptron – Rosenblatt (1958)" (L05a p21) against "Shakey the robot and A* – Hart, Nillson, and Raphael (1968)" (L05a p20) and "Stanford Research Institute Problem Solver (STRIPS) – Fikes and Nilsson (1971)" (L05a p20).

14. **Imported figures bring their own visual language and are not normalised.** L01 p52 (divergent thinking) uses light blue for diverge, green for converge and orange for constraints — a palette used nowhere else in the course, and it carries no attribution. L05a p13 (the timeline) uses a nine-colour warm/cool stock palette outside the UQ scheme. L04 p6 is blue-and-orange CleverTap branding. L05a p32 is a red-and-blue Bayesian network.

Not established by the source

Things a new COMP1100 diagram will need that the materials do not settle. **This is where invention risk lives.**

- **No named typeface anywhere.** The decks render in a humanist sans (the UQ template face) with a serif only in the UQ logo, but no deck names a font, sizes or a type scale. Any new figure's typography is a guess.
- **No stroke weights, corner radii or grid.** Nothing states rule thickness, arrow-head size, the corner radius of the rounded rectangles, node spacing, or a layout grid. All of these were measured by eye here and none is documented.
- **No hex values are published.** Every colour in this note was sampled off a rendered page. There is no palette document, so no way to know which of the two blues (`#0552A2` / `#004C97`) or the two teals (`#1A688C` / `#156082`) is canonical, or whether the difference is meaningful at all.
- **Nothing says what to do with a fifth, sixth or seventh step.** The ordered ramp is defined for four; L01 p50 and L03 p55 extend it with pink and magenta, but only for bands in a single figure, never for process steps.
- **No rule for a hypothesis that is partly confirmed.** The card vocabulary has exactly three resolved states — Fact, refuted, and newly raised. L03 p35 records an outcome that is neither ("reject this hypothesis, and test a new hypothesis about…"), and no card colour is shown for it.
- **No convention for the value proposition canvas when there are several customer segments.** L02 p31 stacks additional canvases behind the front one to suggest more exist, but never draws a second one or says how many, how offset, or how labelled.
- **Nothing distinguishes a *user* from a *paying customer* graphically.** The distinction is central (L01 p41, L03 pp31–32, pp34–35) and always carried in prose, never by a symbol, colour or icon.
- **No accessibility rule.** Several distinctions rest on colour alone — the front/back stage split, the Fact/Hypothesis card states, the funnel bands. The refuted card is the one case that adds a redundant non-colour cue (the X). No deck states whether that redundancy is required.
- **No convention for the working canvases students actually produce.** Every filled-in canvas shown is a lecturer's illustration. Nothing says whether student canvases

should use the colour states at all, whether handwritten sticky notes on a physical canvas are acceptable, or how a canvas should be captioned when submitted.
- **No sizing or aspect-ratio guidance.** Every figure is drawn to fill a 16:9 slide. Nothing indicates how these figures should be scaled for a report, a poster, or print.
- **No arrow-head style is fixed.** Solid filled triangles (L03 p17, L05a p17), open thin heads (L05a p21), grey 3-D block arrows (L01 p24), and thick tapered hand-drawn arrows (L03 p55) all appear, with no stated correspondence between head style and meaning.

See also

- business-model-canvas[35] — what the nine blocks mean
- value-proposition-canvas[36] — the value map and customer profile
- test-cards-and-learning-cards[37] — the card fields in full
- customer-discovery[38] — the four-step cycle these diagrams colour-code
- customer-relationships-and-channels[39] — the get/keep/grow funnel
- comp1100[40] — course overview

Customer Discovery

> **Customer discovery** is the process of turning an initial idea or vision about a market or customers into a set of *facts*. The process tests whether the business model for a company solves real customer problems, and whether the product features fit those customer problems.

It is the first stage of the customer development process in lean-startup[41], and the stage COMP1100 spends nearly all its time in — all three assessment submissions are iterations of it.

The four-step process

```
1. Define hypotheses
        ↓
2. Test the problem
        ↓
3. Test the solution
```

[35] courses/comp1100/business-model-canvas.pdf
[36] courses/comp1100/value-proposition-canvas.pdf
[37] courses/comp1100/test-cards-and-learning-cards.pdf
[38] courses/comp1100/customer-discovery.pdf
[39] courses/comp1100/customer-relationships-and-channels.pdf
[40] courses/comp1100/comp1100.pdf
[41] courses/comp1100/lean-startup.pdf

```
                ↓
4. Verify, pivot, or refine  →  (customer validation)
```

The loop repeats. Each pass turns more of the business-model-canvas[42] from hypothesis into fact.

Step 1 — define our hypotheses

The initial hypotheses *are* our vision. Write answers to:

1. What are our customers' top problems?
2. How does our idea help to solve these problems?
3. What would a day in the life of a customer look like before and after our product?

These are **guesses or assumptions at best** — which is exactly why they get tested rather than built on.

The step also includes sketching a vision. This means a rough sketch, not a polished rendering.

Step 2 — test the problem

Gather data to **refute or confirm** our hypotheses about the problem. This is where the distinction that runs through the whole course appears:

	Who they are
User	People who *use* a product
Customer	People who *pay for or approve* a product

They are frequently not the same people, and a hypothesis needs to say which it's about.

Step 3 — test the solution

Gather data to refute or confirm hypotheses about our *solution*. The thing being sought is **product/market fit** — the match between the customer's problems and needs on one side, and the product's features and value proposition on the other.

> "Ideas are like babies because everyone thinks theirs is cute, therefore be objective when judging your own ideas."
>
> — Tina Seelig

[42] courses/comp1100/business-model-canvas.pdf

Step 4 — verify, pivot, or refine

Refine the product, pivot the strategy, or (rarely) rethink the vision. See lean-startup[43].

A worked example: the medical alert system

The running example is an emergency alarm for older people living alone, with a speaker, microphone, emergency button, well-being check button, status light and a wireless transmitter.

Initial hypotheses:

1. Older people want to be safe.
2. Families of older people want to know, on a daily basis, that their older family member is safe.
3. Older people want to be able to initiate a call for help and cancel any call for help.
4. Older people want to maintain their independence/autonomy.
5. Older people want to be cared for.
6. Older people do not want to have to press the green button each day to let the care company know that they are alive.
7. Older people do not want to wear a visible (and ugly!) emergency alarm around their neck or wrist.

Hypothesis 1 is about both users and customers, while hypothesis 2 is mainly about *customers* — the family, not the user.

After testing the problem, hypothesis 5 was refuted outright and 7 was partly refuted, producing **new hypotheses**:

- 5. Older people want to be cared **about**. *(the original said "cared for" — a small wording change carrying a completely different product)*
- 8. Some older people wouldn't mind wearing a wireless transmitter if it was not visible to others.
- 9. Older people wanted to be in touch with **their family members**, rather than a service provider.
- 10. Older people would prefer a medical alert that allowed them to be mobile — the existing alarms were attached to a phone line and the transmitters had short range.

This is the whole method in miniature: the product that survives contact with real users is not the one that was sketched.

[43] courses/comp1100/lean-startup.pdf

Early adopters

Not all of the market matters at the start. The adoption curve:

Segment	Share
Innovators	2.5%
Early adopters	13.5%
Early majority	34%
Late majority	34%
Laggards	16%

The first release targets early adopters, not the majority.

Customer development heuristics

1. **There are no facts inside your building, so get outside.**
2. Develop for the few, not the many.
3. Early adopters make your product.
4. And they know more than you.
5. Focus groups are for big companies, not startups.
6. Surveys don't give insight compared to interviews — follow-up questions are difficult.
7. The goal for release 1 is the **minimum feature set for early adopters**.

Heuristic 6 is the reason the course's assessment requires recorded, transcribed interviews rather than a survey: you cannot ask "why?" to a form.

See value-proposition-canvas[44] for the ground rules on actually conducting those interviews, and test-cards-and-learning-cards[45] for the format the hypotheses and results are written in.

Customer Relationships and Channels

Two adjacent blocks of the business-model-canvas[46]: how you reach customers, and what relationship you build once you have.

[44] courses/comp1100/value-proposition-canvas.pdf
[45] courses/comp1100/test-cards-and-learning-cards.pdf
[46] courses/comp1100/business-model-canvas.pdf

Get, keep, grow

The relationship lifecycle has three phases, and a **viral loop** running back from *grow* to *get* when customers bring in other customers. Effort and cost rise across the three.

```
Get  →  Keep  →  Grow
  ----------------
      viral loop
```

Foundations for relationships

Five questions to answer before designing any of it:

1. Who are our customers?
2. What is their motivation for using our product? What matters to them?
3. Where do they get their information about products?
4. What (or who) influences them?
5. How do they buy things?

Get

For a web or mobile product, *get* decomposes into a funnel:

```
Awareness → Interest → Activation
```

And the funnel has brutal arithmetic. The seminar's worked example:

> 50,000 clicks at 10c per click → **10% interest** → **2% activation**

50,000 clicks costs $5,000 and yields 5,000 interested people and 100 activations — $50 per activated customer. Running the numbers early is what stops a channel strategy that can never pay for itself.

Paid vs earned demand creation

Paid	Earned
Public relations agency	Peer-reviewed publications
Trade shows	Conference appearances
Email campaigns	Blogs, guest articles
Social media campaigns	Awards

Keep

Ways to keep the customers you already have:

- Loyalty programs
- Product updates that keep their interest and continue adding value
- Contests and events to keep engagement
- Horizontal and vertical offerings
- Customer engagement and satisfaction programs — blogs, community support, email
- Social media

Why keeping beats getting: churn

The seminar answers "why focus on keeping existing customers instead of gaining new ones that bring new income?" with arithmetic:

Monthly churn	Remaining after 3 years	Average customer duration
1%	$(1-0.01)^{36} = 69.8\%$	$1/0.01 = 100$ months
5%	$(1-0.05)^{36} = 16.6\%$	$1/0.05 = 20$ months
10%	$(1-0.10)^{36} = 3.14\%$	$1/0.10 = 10$ months

The gap between 1% and 10% monthly churn is the difference between keeping 70% of your customers after three years and keeping 3%. A small recurring leak dominates almost any acquisition effort — which is why *keep* is usually the cheapest growth lever available.

Grow

Three moves:

- **Up-sell** — a bigger version of what they already buy
- **Cross-sell** — something adjacent
- **Referrals** — the customer brings the next customer, closing the viral loop

Product-channel fit

Channels are not neutral: a product has to fit the channel that reaches its segment, the same way it has to fit the market. A product whose customers are reached only through a channel you can't afford, or can't operate, has a channel problem rather than a product problem.

See value-proposition-canvas[47] for building the customer profile these decisions rest on.

[47] courses/comp1100/value-proposition-canvas.pdf

Experimentation

Why experiment?

To gain insights about customers, their problems, and our solutions — before spending the effort to build.

Two types, two reasons

Two types of experiment:

1. **Generative / exploratory** — explore a space.
2. **Evaluative** — find data to support or refute a hypothesis.

Two reasons to experiment:

1. Evaluate our understanding of our customers and market fit.
2. Evaluate products.

The scientific method

```
1. State hypothesis → 2. Design experiment → 3. Test with data → 4. Gather insights
```

The lean startup version of this loop (Steve Blank) expands into seven steps:

1. Extract hypotheses
2. Prioritise hypotheses
3. Design tests
4. Prioritise tests
5. Run tests
6. Capture learnings
7. Make progress — and repeat

Both prioritisation steps are easy to skip and shouldn't be: a team has more hypotheses than it has time, and testing the wrong one first wastes a week.

What makes a good hypothesis?

The full list: **testable, reliable** (not fake), **concise, replicable, specific, ethical**. Condensed to four:

	Property
#1	**Measurable/testable falsifiable**
#2	**Specific**
#3	**Ethical**
#4	**Simple and clear**

Falsifiability is the load-bearing one. An example of a *non*-falsifiable hypothesis:

> "This solution will help this segment of customers improve their health over the long term."

Nothing about it can come back false — "improve", "health" and "long term" are all unmeasurable as stated, so no result could ever refute it. The exercise is to rewrite it until a specific observation could prove it wrong.

Structuring a hypothesis

The test-cards-and-learning-cards[48] format gives hypotheses a fixed shape:

> We believe that [**target**] are [**showing behaviour / display interest in**] [**for reason Y**]. To verify that, we will [**undertake behaviour**] to [**identify behaviour**]. And measure [**particular variable**]. We are right if [**some measurable statement**]. Therefore, we will ask [**specific questions**].

Types of data collection

Six ways to gather evidence, roughly in increasing order of cost:

Method	What it is
Interviews	Talk with potential customers
Observations	Observe people while doing their "jobs"
Data collected by others	Find existing data that tests your hypothesis
Immersion	Work in your customer's environment
Low fidelity prototyping	Paper prototypes or low-fidelity digital prototypes
High-fidelity solutions	Implement new functionality in an existing product

See [prototyping] for the last two.

[48] courses/comp1100/test-cards-and-learning-cards.pdf

Experiments should

- Be as **small and comprehensible** as possible.
- Have **clear expected outcomes**.
- Have a clear definition of **success/failure that everyone agrees to** — agreed *before* the results come in, or the team will argue about interpretation afterwards.
- Be as brief as possible when interviewing — **preferably under 30 minutes**.

More tips

Do: listen; observe people's behaviour, tone and reactions, not just what they say; ask "why" and "why not"; ask who else you should talk to; ask if you can follow up later.

Don't: "correct" people (except to clarify a misunderstood question); convince or sell to them.

Learning from experiments

When results come back, interrogate them rather than just recording them:

- If people would not pay for this, what *would* they pay for?
- If these are not our customers, then who is?
- Did we see any **outliers** that could open up new ideas?
- **Pay attention to why our hypotheses are rejected.**

A refuted hypothesis is more informative than a confirmed one — it tells you something you didn't already believe.

Key takeaways

- Experiments = gain insights about customers and products.
- Hypotheses must be **falsifiable**.
- **Test cards** structure our hypotheses; **learning cards** structure our learnings.
- Refuted hypothesis ask *why*.

The History of Computing and AI

In what year was the first programmable computer built?

The question is a trick. There are six defensible answers:

Year	Machine
87 BCE	**Antikythera machine**
1642	**Pascaline**
1805	**Jacquard machine**
1837	**Analytical Engine**
1943	**Colossus**
1951	**CSIRAC**

Which one counts depends entirely on what you mean by "programmable" and "computer" — which is the point.

Timeline

Period	Era	What happened
1837		Charles Babbage invents the Analytical Engine. **Ada Lovelace** recognises it is a general purpose calculating machine.
1941–1943		British mathematicians and engineers build the **Bombe** and **Colossus** for cracking codes during World War II.
1950		**Alan Turing** proposes the Turing Test as a goal for artificial intelligence, rather than defining "intelligence".
1956		The **Dartmouth Conference** is held. The proposal coins the term *artificial intelligence*.
1956–1974	**The Golden Era**	Foundational search algorithms and artificial neural networks are invented.
1974–1980	**The first AI winter**	The hype of the golden era did not eventuate. Funding dried up and excitement dwindled.

Period	Era	What happened
1980–1987	**The Knowledge Era**	Knowledge-based systems, such as Prolog and expert systems, invented.
1987–1993	**The second AI winter**	The hype of the knowledge era did not eventuate. Funding dried up and excitement dwindled.
1994–now	**The Revival Era**	Intelligent agents, computational power, the Internet, and machine learning.

The Turing Test (1950)

> "The new form of the problem can be described in terms of a game which we call the 'imitation game' "
>
> — Alan Turing, *Computing Machinery and Intelligence*, Mind, LIX(236):433–460, 1950

Turing sidesteps defining intelligence and substitutes a *test* for it. That move — replacing an unanswerable definition with an operational check — is the same instinct as making a hypothesis falsifiable in [experimentation].

Dartmouth (1956)

The founders present: John McCarthy, Marvin Minsky, Claude Shannon, Ray Solomonoff, Alan Newell, Herbert Simon, Arthur Samuel, Oliver Selfridge, Nathaniel Rochester, Trenchard Moore.

> "We propose that a 2-month, 10-man study of artificial intelligence be carried out during the summer of 1956 at Dartmouth College… The study is to proceed on the basis of the conjecture that every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it."

Two months, ten men. The proposal's confidence is the first data point in the pattern that follows.

Its lasting output was the **divide and conquer model of AI**, splitting the problem into perception, knowledge representation, learning, reasoning and planning.

The Golden Age (1956–1974)

- **Reasoning as search** — Shakey the robot and A*, Hart, Nilsson and Raphael (1968); STRIPS (Stanford Research Institute Problem Solver), Fikes and Nilsson (1971), with its preconditions/postconditions action model.
- **Perceptrons and neural networks** — the single-layer perceptron, Rosenblatt (1958).

The first AI winter (1974–1980)

What went wrong — outcomes failed to live up to the hype:

- **Scalability**
- **Commonsense knowledge**
- **Perceptron limitations**
- **Moravec's paradox**

The result: funding dried up, global interest died down, and criticism came from philosophers and cognitive scientists.

The Knowledge Era (1980–1987)

- **Knowledge-based systems** — Prolog (Colmerauer and Roussel, 1972), and formal ontologies.
- **Expert systems** — MYCIN, for diagnosis of blood diseases (Shortliffe, Buchanan and Cohen, 1970s).

The second AI winter (1987–1993)

Same headline, different specifics — outcomes again failed to live up to the hype:

- **Scalability**
- **Maintenance**
- **The qualification problem**
- **Moravec's paradox** *(again)*

The result: funding dried up (DARPA declared AI was "not the next wave"), global interest died down, AI companies went bankrupt.

The revival (1994–present)

- **Intelligent agents and decision theory** — Bayesian networks, Pearl (1988)
- **Computational power** — Moore's law
- **The Internet and big data**
- **Machine learning** — backpropagation in deep neural networks, Rumelhart, Hinton and Williams (1986)

Another AI winter?

This is the real argument of the history. We have been here before:

> "In from three to eight years we will have a machine with the general intelligence of an average human being." — **Marvin Minsky (1970)**

> "Machines will be capable, within twenty years, of doing any work a man can do." — **Herbert Simon (1956)**

> "…the embryo of an electronic computer that [the Navy] expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence." — **New York Times on the Perceptron (1958)**

> "Artificial intelligence will reach human levels by around 2029." — **Ray Kurzweil (1999)**

> "We will have fully self-driving cars on the road by 2017" — **Elon Musk (2014)**

> "We are now confident we know how to build AGI as we have traditionally understood it" — **Sam Altman (Jan 2024)**

Amara's Law

> We tend to **overestimate** the effect of a technology in the **short run** and **underestimate** the effect in the **long run**.

Which is why the pattern above isn't simply "the predictions were wrong" — the winters came from the short-run overestimate, and the revival from the long-run underestimate.

What are some of the risks?

> "By far the greatest danger of Artificial Intelligence is that people conclude too early that they understand it." — Eliezer Yudkowsky

> "People worry that computers will get too smart and take over the world, but the real problem is that they're too stupid and they've already taken over the world." — Pedro Domingos

> "The beauty of AI and what we can do with it is currently being overshadowed by reckless hype, monotheistic techniques, and absurd deification. Yet, it's not the first time in 65+ years nor it'll be the last one. We haven't learnt a thing." — Dagmar Monett

Representation in computer science

Six people the field's history remembers:

Person	For
Ada Lovelace	The first "computer programmer"; the first person to consider general purpose computers doing more than numerical calculation
Alan Turing	Turing machines, code cracking in World War II, the Turing test, so many other things
Margaret Hamilton	Software engineer at the MIT Instrumentation Lab; developed on-board flight software for NASA's Apollo program; **coined the term "software engineering"**
Grace Hopper	Identified that computer programs can be written in a high-level language and compiled into machine code
Fei-Fei Li	Machine learning researcher; founder of AI4ALL
Timnit Gebru	Research on bias in machine learning models; founder of Black in AI

> CSSE1001 covers overlapping ground from a different angle — see CSSE1001's history of computing[49] for the hardware timeline and the machine-code-to-Python language progression.

[49] courses/csse1001/history-of-computing.pdf

Innovation and Value

What is innovation?

The week 1 seminar asked the room, laptops closed, and collected these answers:

- Solving someone's problems or trying to improve their situation
- Creating value for a client or customer
- Improving something that exists with a different mindset
- Creating a new solution to a problem that changes how things are done
- Creating meaningful items that the customer wants to pay for
- Improving upon an existing solution

The common thread — and the course's one-line definition — is **innovation = creating value for someone**.

Value is defined by the customer, not the maker

> "Quality in a product or service is not what the supplier puts in. It is what the customer gets out and is willing to pay for. A product is not quality because it is hard to make and costs a lot of money, as manufacturers typically believe. This is incompetence. Customers pay only for what is of use to them and gives them value. Nothing else constitutes quality."
>
> — Peter Drucker

This is the assumption the whole course rests on. Effort, cleverness and cost do not create value; only what the customer gets out does. It's also why the process starts with customer-discovery[50] rather than with building.

Why we need an innovation process

Because products and businesses fail, and mostly for reasons that have nothing to do with the code:

Failure reason	Percentage
Cash flow	38%
No market fit	35%
Competition	20%
Business model	19%
Regulatory challenges	18%

[50] courses/comp1100/customer-discovery.pdf

Failure reason	Percentage
Pricing issues	15%
Team	14%
Timing	10%

(The percentages exceed 100% because failures usually have several causes.) The single most addressable item on that list for a student team is **no market fit** — 35% of failures come from building something nobody wanted.

The solution the course proposes: **focus on the customer**, and answer three questions in order.

1. What are their problems?
2. Do we have a solution? Can we make one?
3. Will they pay for our solution?

From development to innovation

Four shifts distinguish an innovation process from ordinary product development:

	From	To
1	A client	A market segment (customers)
2	A system	A value proposition
3	Development	Experimentation
4	A requirements specification	A Minimum Viable Product

Each shift has its own machinery later in the course: market segments and value propositions in business-model-canvas[51] and value-proposition-canvas[52], experimentation in [experimentation], and the MVP in minimum-viable-products[53].

Divergent thinking for creative design

When generating ideas rather than evaluating them:

- **Don't challenge or criticise.**
- **Quantity, not quality.**
- **Do not analyse or organise** — but do record and draw.

[51] courses/comp1100/business-model-canvas.pdf
[52] courses/comp1100/value-proposition-canvas.pdf
[53] courses/comp1100/minimum-viable-products.pdf

- Then **identify significant themes**, and **refine**.

The separation matters: the generating phase and the judging phase are different activities, and mixing them kills the first one.

Key takeaways

- Innovation = creating value for someone.
- Failure is often because we do not understand our market.
- Success is more likely if we **get out of the building**.
- An innovation process means generating and testing ideas early, so we **fail early**.
- Failure is an opportunity to learn.

Large Language Models

Machine learning, briefly

The seminar builds up from curve fitting. Given data on sleep hours against health, machine learning is the business of finding the function that fits:

- $y = ax + b$ — a straight line
- $y = ax^2 + b$ — a curve
- $y = ax + by + c$ — a plane, once there are two inputs (sleep *and* exercise)

The model is the function; learning is finding its parameters.

How LLMs actually work

An LLM predicts the **next token**, one at a time, from a probability distribution over the whole vocabulary.

Given the prefix *“The woman ran up the ____“*:

Candidate	Probability
“hill”	1.2%
“stairs”	1.9%
“the”	1.9%
“to”	1.2%
“baking”	0.005%
“package”	0.0001%
“myriad”	0.000004%

Candidate	Probability
“gazpacho”	0.00001%

Add one more word to the context — *“The woman ran up the”* — and the distribution shifts sharply: “hill” jumps to 10.8% and “stairs” to 6.9%, while “the” collapses to 0.00001%.

That’s the whole mechanism. And it explains why *“The woman ran up the _____“* and *“What did the woman run up? _____“* are, to the model, the same kind of problem: continue the text plausibly.

Where it breaks: the apples

The seminar’s worked failure, from ChatGPT:

> **Q:** I have 419 red apples and 201 green apples. I eat 300 of the apples. How many apples do I have left?
>
> **A:** 620 (initial count) − 300 (eaten) = **320 apples left.**

Correct. Then:

> **Q:** How many red apples do I have left?
>
> **A:** 419 − 300 = **119 red apples left.**
>
> **Q:** How many green apples do I have left?
>
> **A:** 201 − 300 = **−99 green apples.** It appears that you’ve eaten more green apples than you initially had…

The model subtracts the *whole* 300 from each colour independently, producing a negative count — and then narrates the absurdity rather than noticing it. The underlying problem was never specified (which apples were eaten?), and the model doesn’t flag the ambiguity; it generates a plausible continuation for each question in isolation.

The lesson for the course: an LLM produces text that *reads* like reasoning. Verify anything that has to be true.

Why this matters for COMP1100

The course's generative AI policy permits AI for anything **except generating interview or other data** (see comp1100[54]). The apples example is the argument for that rule — a model will happily produce fluent, confident, wrong content, and interview data that was generated rather than gathered defeats the entire point of customer-discovery[55].

See prompt-engineering[56] for getting useful work out of them anyway.

The Lean Startup Methodology

The methodology the course runs on, from Eric Ries's *The Lean Startup* and Steve Blank's customer development work.

The customer development process

Blank splits building a company into two halves — **search**, then **execution**:

Phase	Stage	What happens
Search	Customer discovery	Turn the vision into facts — do these customers have this problem?
Search	Customer validation	Confirm there's a repeatable, scalable sales model
Execution	Customer creation	Create end-user demand and drive it into the channel
Execution	Company building	Transition from a search organisation to one that executes

A **pivot** loops back from validation to discovery when the facts don't support the model.

COMP1100 lives almost entirely in the **search** half, and mostly in customer discovery — the design end rather than the entrepreneurship end. See customer-discovery[57].

[54] courses/comp1100/comp1100.pdf
[55] courses/comp1100/customer-discovery.pdf
[56] courses/comp1100/prompt-engineering.pdf
[57] courses/comp1100/customer-discovery.pdf

Why the search half is the hard part

> "What you will get wrong is that you will not pay enough attention to your users.
>
> You will make up some idea in your own head that you will call your 'vision', and you will spend a lot of time thinking about your vision. In a cafe. By yourself. And build some elaborate thing without going and talking to users, because that's doing sales, which is a pain in the ass, and they might say no.
>
> You will not ship fast enough because you're embarrassed to ship something unfinished, and you don't want to face the likely feedback that you will get from shipping. You will shrink from contact with the real world, contact with your users. That's the mistake you will make."
>
> — Paul Graham, Y Combinator

Every mechanism in the course is designed against this failure mode: test cards force a hypothesis to be falsifiable before you build, MVPs force shipping before it's finished, and studio deadlines force contact with real people.

Verify, pivot, or refine

After each round of experiments there are three moves, at escalating cost:

Move	What changes
Refine	The **product** — adjust features within the same strategy
Pivot	The **strategy** — same vision, different route to it
(rethink)	The **vision** — the deepest and rarest change

The ordering is the point: refining the product is cheap, changing strategy is expensive, and abandoning the vision is a last resort. See [experimentation] for how the decision gets made from evidence rather than mood.

Inspiration and sources

The seminar names its sources explicitly:

- **Tina Seelig** — *inGenius: A Crash Course on Creativity*
- **Steve Blank** — *The Four Steps to the Epiphany*, *The Startup Owner's Manual*

- **Alexander Osterwalder** — *Business Model Generation* (the origin of business-model-canvas[58] and value-proposition-canvas[59])
- **Eric Ries** — *The Lean Startup: How today's entrepreneurs use continuous innovation to create radically successful businesses*

Minimum Viable Products (MVPs)

What an MVP is for

The seminar opens with a quiz: what is the *primary* purpose of a minimum viable product?

- **(a) To test the ability of the product to meet minimal customer needs.**
- (b) Get as much revenue as we can as early as we can.
- (c) Identify bugs in a product with a fixed feature set that meets a set of requirements.
- (d) All of the above.

The answer is (a). An MVP is an **experiment**, not a revenue play and not a QA exercise.

It is not a full product — but it *is* a product

The distinction the course insists on: an MVP is not a full product, but it is a **product, not a prototype**. It has to actually work.

> "By far the dominant reason for not releasing sooner was a reluctance to trade the dream of success for the reality of feedback."
>
> — Kent Beck, *Approaching a Minimum Viable Product* (August 2009)

That quote is the same failure mode Paul Graham describes in lean-startup[60], from the other direction: shipping is avoided because shipping produces information, and information can be disappointing.

[58] courses/comp1100/business-model-canvas.pdf
[59] courses/comp1100/value-proposition-canvas.pdf
[60] courses/comp1100/lean-startup.pdf

Prototype	MVP

Prototype vs MVP

Prototype	MVP
Use for visualising and testing product ideas	Functional product with core features
Often not actually functional	Must be functional (even if minimal)
Super fast to build and test	Requires working code, deployment, etc.
Aims to test and refine ideas	Aims to test **market fit**
Ideas tested in a "controlled" environment	**Released to the market**

The dividing line is the last row: a prototype is shown to people, an MVP is released to them.

Six principles for an MVP

1. **KISS** — keep it simple, stupid.
2. **Fast to build.**
3. **Positive user experience.**
4. **Early adopters** — build for them, not the majority (see customer-discovery[61]).
5. **Iterate and refine.**
6. **Set success metrics.**

Principle 6 is the one that connects back to test-cards-and-learning-cards[62]: without a success metric agreed in advance, the MVP produces impressions rather than evidence.

Types of MVP

Not all MVPs are software:

Type	What it is
Pre-order	Take orders before the thing exists — tests willingness to pay directly
Show and tell	Demonstrate the concept and measure the reaction
Audience building	Build the audience first, product second

[61] courses/comp1100/customer-discovery.pdf
[62] courses/comp1100/test-cards-and-learning-cards.pdf

Type	What it is
"Wizard of Oz" product	The customer-facing experience is real; the machinery behind it is a human
Digital product	An actual, minimal, working piece of software

Different MVP types let you run **different experiments** — choose the one that tests the hypothesis you actually have. If the open question is "will anyone pay?", a pre-order tests it far faster than building the product would.

Competitors

The MVP has to be minimum *relative to what already exists*. The value proposition diagram from value-proposition-canvas[63] returns here — what your customer wants, what you do well, and **what your competitors do well**. Features your competitors already deliver well may be table stakes rather than optional.

Key takeaways

- MVP = the **most important features**.
- MVP purpose: **experiment!**
- Simple and easy to build.
- Different MVPs allow us to run different experiments.

See [prototyping] for the cheaper things to build first.

Prompt Engineering

Seven tactics, aimed at programming tasks. See large-language-models[64] for why they work.

Tactic 1 — use delimiters

Mark where the input starts and stops, and mark the shape of the output:

```
Here is some code in Python: $$$ … $$$

… format the output as follows: """On [DATE], [NAME] … """
```

[63] courses/comp1100/value-proposition-canvas.pdf
[64] courses/comp1100/large-language-models.pdf

Tactic 2 — ask for structured output

```
… Put these into an HTML table. The table should have three columns.

… format it as a JSON object: """On [DATE], [NAME] … """
```

Example:

> Write a python program that can generate 5 sample emails, including email header. Each email should contain a body that has a greeting, then the main part, and then a sign-off. The main part should contain an order for a part, of the format:
>
> `""" I would like to order [NUMBER] [PARTS]. """`

Tactic 3 — ask it to check whether conditions are satisfied

Tell it what to do when the input doesn't match the expected shape, rather than leaving it to improvise:

> Below is an email from someone. Extract the following into a structured format of: `"""On [DATE], [PERSON] ([SENDER EMAIL ADDRESS]) from [COUNTRY] sent an email with a message about [SUMMARY]"""`. If the email contains a phone number, determine the country number, and insert the country name. **Otherwise, write "unknown country" in place of [COUNTRY].**

Tactic 4 — specify the steps required

Tactic 5 — iterate

> The following is an email from a customer outlining their opinion on a project. Create an email that responds to them, thanks them for their review, and addresses any items in the email.

Tactic 6 — be as specific as possible

A before-and-after:

This:

> Create a Python program that downloads emails from the email address "uqt-miller@gmail.com" and gets an image from them.

Versus this:

> Create a Python program that: 1. Downloads any email in the account "uqtmiller@gmail.com" 2. Saves any image attachments (JPG or PNG) in the directory called "images" 3. Deletes the email from the email inbox.

A longer example, showing how far the specificity goes — including the error case and the runtime environment:

> Create a simple html page that: 1. Use javascript to select a random image from the directory 'images/' 2. When the image is selected, display a message that says "a new image has been received" 3. When the user clicks the message, the randomly selected image is displayed. 4. If no image is available, print "There is no image in the directory <DIRECTORY>", where DIRECTORY is the complete path. 5. A message "An email has been sent to your_family@fake.com" should be displayed. 6. Make sure this runs locally.

Tactic 7 — build code incrementally

Don't ask for the whole system at once. Ask for the page, then ask for the specific piece:

> **Prompt 1:** Create a simple html page: 1. Use javascript to select a random image from the directory 'images/' …
>
> **Prompt 2:** Create javascript code that selects the file with the most recent timestamp from a list called 'images'.

LLMs as tools for thought

The most interesting use is not asking for an answer, but asking to be interrogated:

> I am doing a project using the Lean Startup methodology. I have a hypothesis about my solution that I want to test. I would like to construct a test card for it.
>
> Can you please interact with me to help me construct this?
>
> **Do not give me answers, just ask a series of questions until I have the test card.**
>
> Provide feedback on my answers.

This inverts the usual pattern: the model runs the process and you supply the content, so the test card[65] ends up containing your team's actual hypotheses rather than a plausible-sounding generated one. It stays on the right side of the course's AI policy for exactly that reason — see comp1100[66].

Prototyping

Prototyping as experimentation

Prototyping forms a key part of the design cycle, can be **exploratory or evaluatory**, and — the point most often missed — **does not have to start with writing code**.

It sits inside the design loop:

```
Establish customer problems → Design value propositions
        → Prototype value propositions → Evaluate value propositions → MVP
```

So a prototype is a way of running an experiment ([experimentation]) about a value proposition, before committing to the MVP[67].

Four axes

Prototyping can be thought of along four axes:

Axis	Range
Representation	Online vs offline; high vs low fidelity
Precision	Level of detail
Interactivity	Fixed or open
Evolution	Rapid to iterative

Placing a prototype on these four axes is a more useful question than "is it good enough yet?" — it says what the prototype is *for*.

[65] courses/comp1100/test-cards-and-learning-cards.pdf
[66] courses/comp1100/comp1100.pdf
[67] courses/comp1100/minimum-viable-products.pdf

Paper prototyping

> An offline, interactive, rapid method of exploring aspects such as layout, information architecture, interactive design and more.

Low cost, easy to do, interactive, and fun. It's the default first move, and it deliberately scores low on representation and precision while scoring high on interactivity and speed.

Wireframes and Wizard of Oz

- **Digital prototyping using wireframes** — online, still low fidelity, focused on structure rather than looks.
- **"Wizard of Oz" prototyping** — the user experiences a working system; a human is doing the work behind the curtain. Lets you test a value proposition that you have no idea how to build yet.

The economics of design

The reason to start cheap:

	Low fidelity prototypes	High fidelity prototypes	Code prototypes	MVP
Cost	Super cheap	Quite cheap	More expensive	Most expensive
Focus	Flow & interactivity	Look & feel, usability	Code feasibility and usability	Getting to market

Each step right costs more and answers a narrower question. Spending code-prototype money to learn something a paper prototype would have told you is the mistake this table exists to prevent.

Have code generation tools killed wireframing?

A genuinely open question the seminar puts both sides of:

Yes?	No?
Quicker to generate interfaces than to draw boxes	Wireframes help focus on user experience: "what are they trying to build?"
The gap to code prototypes is smaller	LLM designs are more mediocre

Yes?	No?
Real interactions instead of simulated ones	Wireframes are *intentionally* low fidelity — to get feedback on the user **experience**, not the user **interface**

The strongest argument in the right column is the last one: low fidelity isn't a limitation to be overcome, it's what stops your interviewee commenting on the colour scheme instead of the flow. See prompt-engineering[68] for the tooling this question is about.

Test Cards and Learning Cards

Two Strategyzer card formats that give experiments a fixed shape. A **test card** is written *before* an experiment; a **learning card** is written *after* it. Together they're the unit of work in [experimentation], and the artefacts the COMP1100 submissions are built from.

Test card

Five lines, filled in before running anything:

```
We believe that          _____________________
To verify that, we will  _____________________
And measure              _____________________
We are right if          _____________________
Therefore, we will ask   _____________________
```

The "we are right if" line is what forces falsifiability — you commit to a threshold before you see the data, so you can't move the goalposts afterwards.

Worked example — users

> **We believe that** our end users are people over 65 who live on their own who would use technology to keep in touch with their family more often because they do not see them much.
>
> **To verify that, we will** interview 50 people over 65 who live on their own.
>
> **And measure** how many would like more regular digital interaction with their family.

[68] courses/comp1100/prompt-engineering.pdf

We are right if 70% of older people say that they would like more regular digital interaction with their family.

Therefore, we will ask "how often would you like to interact with your family members?"

Worked example — paying customer

Same product, different party — note that the *user* and the *customer* need separate cards:

We believe that our paying customers are close family members of our end users who would use technology to keep in touch with older family members using a photo-sharing app rather than delegate to a service, because they would like assurance that their family members are doing well in a more personal manner.

To verify that, we will interview 50 people who have family members over 65 that live on their own and have an existing emergency alarm system.

And measure how many would prefer regular (every day) digital interaction with their family member rather than delegating to a service.

We are right if 70% of family members say that they would like more regular digital interaction with their family member rather than delegating to a service.

Therefore, we will ask "how often would you like to interact with your older family member?"

Learning card

Four lines, filled in after:

```
We believed that         ______________________
We observed              ______________________
From that we learnt that ______________________
Therefore, we will       ______________________
```

Worked example — hypothesis accepted

We believed that our end users are people over 65 who live on their own who would like to interact digitally with their family more often because they do not see them much.

We observed that 44 out of 50 people we interviewed confirmed that they would like daily digital interaction with their family.

> **From this we learnt** that our end users would consider using a product that supported this.
>
> **Therefore, we will** accept this hypothesis.

44/50 is 88%, over the 70% threshold the test card committed to — so the decision is already made by the card, not by argument.

Worked example — hypothesis rejected

This is the more instructive one:

> **We believed that** close family members would like to interact digitally with older family members using a photo-sharing app.
>
> **We observed** that 12 out of 50 people we interviewed confirmed that they would use such an app.
>
> **From this we learnt** that family members would not use a new product. **However**, we learnt that some family members would send photos from their smartphone to such an app.
>
> **Therefore, we will** reject this hypothesis, and test a new hypothesis about how family members would take and send photos for their older family member.

Two things to copy from it. First, the rejection produces a *replacement hypothesis*, not a dead end. Second, the "however" — the most valuable finding was a side observation that the headline number would have buried. This is what [experimentation] means by paying attention to outliers.

The templates are in the course's business model and testing templates.

The Value Proposition Canvas

Value proposition

> A **value proposition** is a feature or service that makes a product attractive to customers.

It sits at the intersection of three things:

- What your **customer wants**
- What **you do well**
- What your **competitors do well**

Your value proposition is what your customer wants that you do well — and, ideally, that your competitors don't. The competitor axis matters more than teams expect; it reappears in minimum-viable-products[69] when deciding what the minimum actually has to include.

The canvas

> A **value proposition canvas** is a visual description of our customer segments and our value propositions to solve those customers' problems.

It has two halves that are meant to be matched against each other:

- the **value map** — our side, what we offer
- the **customer profile** — their side, the jobs, pains and gains

Full detail in the course notes, chapter 4.

Six ways to innovate from a customer profile

Once you have a customer profile, it suggests its own moves:

Strategy	Explanation
Address more jobs?	Can we help customers complete more jobs in their day?
More important jobs?	Can we help customers do something more valuable?
Beyond functional jobs?	Can we help customers create value by fulfilling important social and emotional jobs?
Help more customers?	Can we help more customers by making jobs easier or cheaper?
Do a job better?	Can we help customers achieve a higher quality on existing jobs?

Only five of the six are listed here.

[69] courses/comp1100/minimum-viable-products.pdf

Ground rules for interviewing

Section 4.5 of the course notes, and the most immediately practical page in the course — this is how the customer profile actually gets filled in.

Ground rule	What it means
Adopt a beginner's mindset	Listen with a "fresh pair of ears". Avoid trying to interpret — ask for clarification. Explore things you did not expect.
Listen more than you talk	Listen, don't tell. Do not talk about what you believe. Clarify misunderstandings, but don't try to convince.
Get facts, not opinions	Don't ask "Would you…?" Do ask "When is/was the last time you…?" and "How much did you…?"
Ask "why" questions	Motivation matters and helps uncover new facts. Ask "Why do you need…?", "Why is this a pain?"
Don't sell	We are not interviewing to sell a solution. Don't explain our solution ("Wouldn't you like…?"). Ask "What are the most painful…?"
Follow up	Ask for contact details, so you can come back with more questions later.
Always open doors at the end	Ask "Who else should I talk to?" **Always.**

The "get facts, not opinions" rule is the one that most changes an interview. "Would you use an app that…?" reliably produces a polite yes that means nothing; "when did you last…?" produces a fact.

Key takeaway

When we interview customers to construct profiles, **we need to listen**. See customer-discovery[70] for what the interviews are feeding, and [experimentation] for how the answers get turned into evidence.

[70] courses/comp1100/customer-discovery.pdf