Fieldtripping with AI

When the Tools Don't Exist, Build Them

Brett Moller

March 2026

An educator's account of building custom software for a school trip — with artificial intelligence as co-developer. Including companion documents on implementation and technical deployment.

Contents

Part One

Fieldtripping with AI

The main essay — what was built, how it was built, and why it matters that the person who understood the learning goals most deeply was the one building the tool.

> The tools I needed didn't exist. So I built them. The entire digital infrastructure supporting a week-long school trip — a native iOS staff app, a GPS boundary system, a real-time scavenger hunt platform — was designed and built not by a software development team, but by me, their teacher, working in partnership with an artificial intelligence.

In March 2026, thirty-five Year 9 students stepped off a plane and into the streets of Melbourne. They came from a regional coastal community on the Sunshine Coast, where life moves at a different pace, where you know your neighbours, and where navigating the world mostly means driving familiar roads between familiar places. Melbourne was something else entirely. A sprawling, fast-moving, culturally rich city of five million people. For many of these students, it was the biggest, busiest, most diverse place they had ever been asked to navigate on their own.

The camp was designed to stretch them. To push them into unfamiliar territory, literally and figuratively. To help them explore culture, diversity, and the functioning of a major city. To give them opportunities to work as a team, make decisions, engage with strangers, and discover that they are more capable than they thought.

What the students didn't know was that the entire digital infrastructure supporting their experience had been designed and built not by a software development team, but by me, their teacher, working in partnership with an artificial intelligence. This paper describes what I built, how I built it, and why it matters that the person who understood the learning goals most deeply was the one building the tool.

How It Works

Want to see exactly how these tools work in practice? The companion document, *How It Works*, walks through every workflow, screenshot, and real-time build from the trip — including twelve features conceived and shipped to production in a single day.

For IT Teams: Technical Deployment Guide

Deploying AI-built tools into school infrastructure raises legitimate questions about security, authentication, and data handling. The companion document, *Technical Deployment Guide*, provides a detailed architecture overview covering Microsoft Entra ID integration, API security design, reverse proxy configuration, and the case for purpose-built systems over traditional spreadsheets and printed camp books carrying sensitive student information.

> *"The role of the teacher is to create the conditions for invention rather than provide ready-made knowledge." — Seymour Papert*

Papert wrote those words about children and computers. But they apply with equal force to what happened here: an educator creating the conditions for an experience that no off-the-shelf product could provide.

The Dilemma

Every educator who has taken students into the world beyond the classroom knows the tension. On one side sits the learning experience: the growth that happens when young people encounter the unfamiliar, take risks, and discover new things about themselves and the world. On the other side sits the weight of responsibility: safety, logistics, duty of care, risk management, the knowledge that thirty-five families have entrusted you with their children in a city far from home.

This tension is not new, but it is intensifying. In an increasingly complex world, the risks that educators must manage have grown more numerous and more visible. Allergies, medical conditions, safeguarding protocols, emergency communications, real-time location awareness. Each new layer of risk management is entirely justified, but the cumulative effect can be paralysing. The dilemma is that the very systems designed to keep students safe can, if poorly implemented, constrain the experience to the point where the learning disappears.

> *"While those of us in schools may wish to blame others for imposing constraints, many of our constraints are self imposed. Are we able to escape from these and respond to what we believe is best for our students?" — David Loader, The Inner Principal*

The question Loader poses is the right one. The answer, in this case, was to build software that didn't force a choice between safety and experience. Software that put critical information at staff fingertips so they could manage risk confidently and still give students the freedom to explore. Software designed by someone who understood both sides of the dilemma because he lived it every day.

What We Built

The suite of applications I built for the Melbourne trip comprised two integrated platforms, each designed to solve a specific dimension of the challenge. Together, they formed a comprehensive digital ecosystem for a complex, week-long learning experience.

Melbourne Connections: The Staff Companion

Melbourne Connections is a native iOS application I built in Xcode with the help of Claude Code, installed on each staff member's iPhone. It provides offline-capable access to every piece of information a supervisor might need at any moment of a six-day trip.

The complete itinerary lives in the app. Fifty activities across six days, with flight details, coach transfers, venue information, and timing. The app detects the current and next activity automatically, so a staff member glancing at their phone at any moment knows exactly where they should be and what's coming next.

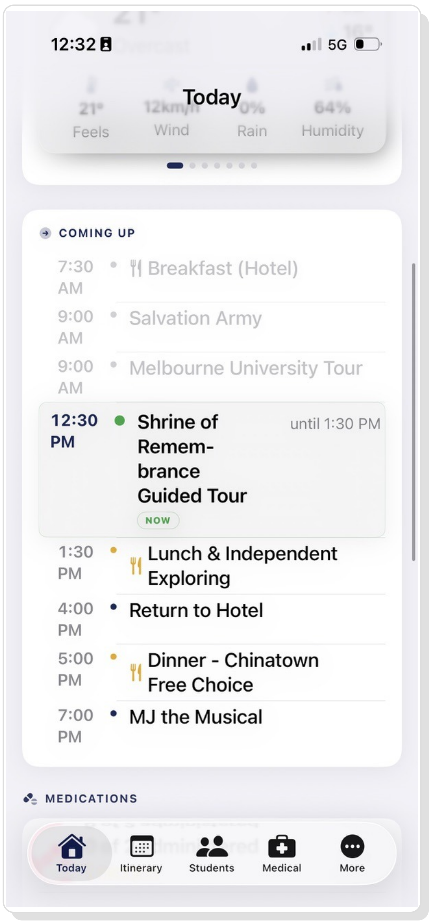
Roll call management tracks attendance by group, with check-all buttons for efficiency, notes fields for recording absences or issues, and automatic history logging with timestamps. Staff can email roll call summaries directly from the app, creating a documentary record without paperwork.

The student directory holds every student's details: group allocation, room assignment, roommate information, and critically, dietary and medical data. When a staff member walks into a restaurant with eighteen students, they can tap their phone and show the chef exactly which students have allergies or dietary requirements. No printed lists to lose. No trying to remember who is gluten-free. The information is there, instantly, at the moment it matters.

Emergency contacts are accessible with a single tap. Staff contacts have direct call and SMS buttons. If something goes wrong, the path from "I need to contact someone" to "I'm speaking to them" is measured in seconds, not minutes of scrolling through phone contacts or searching through paperwork.

Location information for all nineteen venues is integrated with GPS mapping, so staff can navigate to any destination without switching apps. The entire system works offline, because mobile signal in the Melbourne Metro or inside large venues is unreliable, and the moment you most need critical information is often the moment your internet connection is weakest.

Critically, the sensitive medical and personal information powering the app was not manually typed from spreadsheets. Our school's core systems already hold dietary requirements, medical information, and emergency contacts in structured databases with APIs available for integration. The AI helped me connect to these existing systems securely, ingesting the data in encrypted form and transforming it into the formats the app needed. This is a key enabler. Schools already maintain this sensitive data. When an educator can leverage those existing APIs with the help of an AI development partner, the data flows securely from the authoritative source into the purpose-built tool without re-entry, without transcription errors, and without the stale-data problem that plagues printed lists — not to mention the significant security risk of staff carrying printed booklets of student and parent PII throughout a trip. I have been on a trip where a staff member left the camp booklet in a café. In a data-privacy-aware society, this is a complex problem, and one that purpose-built, authenticated digital tools solve comprehensively.


12:32 5G

Today

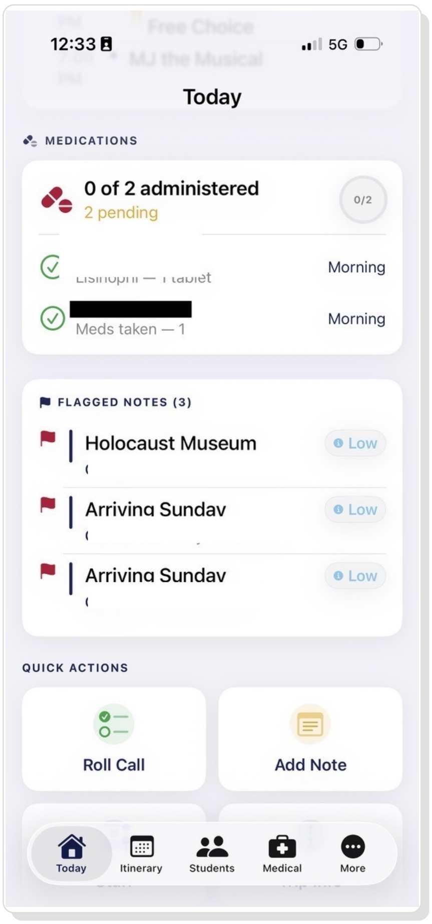
21° Feels · 12km/h Wind · 0% Rain · 64% Humidity

COMING UP

7:30 AM	Breakfast (Hotel)
9:00 AM	Salvation Army
9:00 AM	Melbourne University Tour
12:30 PM	Shrine of Remem-brance Guided Tour NOW — until 1:30 PM
1:30 PM	Lunch & Independent Exploring
4:00 PM	Return to Hotel
5:00 PM	Dinner - Chinatown Free Choice
7:00 PM	MJ the Musical

MEDICATIONS

Trip itinerary display showing activities, timing, and venue information


12:33 5G

Today

MEDICATIONS

0 of 2 administered
2 pending
0/2

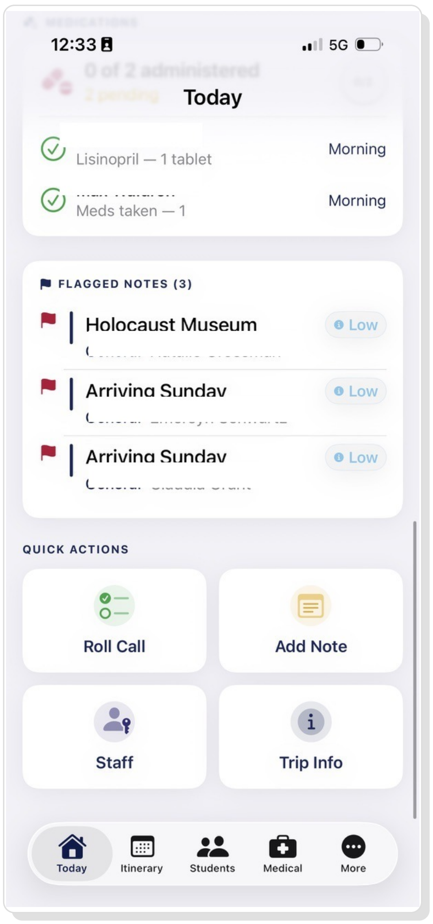
Lisinopril — 1 tablet	Morning
[redacted] Meds taken — 1	Morning

FLAGGED NOTES (3)

Holocaust Museum	Low
Arriving Sunday	Low
Arriving Sunday	Low

QUICK ACTIONS

Medication and dietary management access for student care



12:33 5G

Today

Lisinopril — 1 tablet Morning

Meds taken — 1 Morning

FLAGGED NOTES (3)

Holocaust Museum Low

Arriving Sunday Low

Arriving Sunday Low

QUICK ACTIONS

Flagged notes and student wellbeing information at a glance

Melbourne Connections: Explore Zones

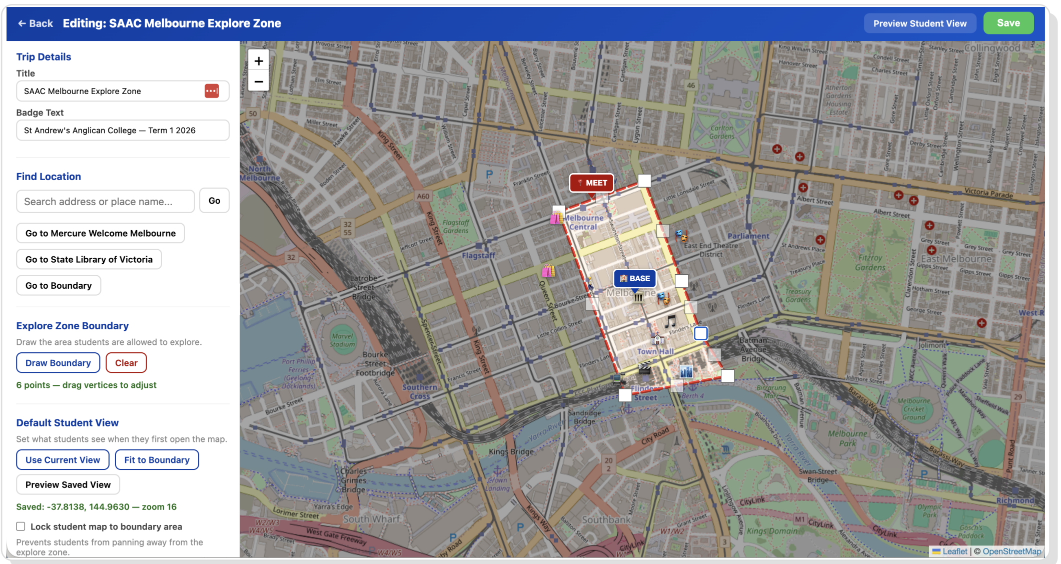
The second component of Melbourne Connections addresses one of the most anxiety-inducing aspects of taking regional students into a major city: free exploration time. Students need the freedom to explore, to make choices, to navigate independently. Staff need confidence that students understand their boundaries and can find their way back.

Explore Zones is a GPS boundary system I built that lets staff draw a polygon on a map defining the safe exploration area. Students access a live map on their phones showing their real-time position, the boundary, the hotel location, and a designated emergency meeting point. If a student drifts outside the zone, the interface immediately shifts to alert them. The status bar turns red. The message is unambiguous.

Staff configure landmarks within the zone so students can see points of interest. The visual theme matches the program's branding. QR codes provide instant student access. The system gives students genuine autonomy within clear, visible boundaries, and gives staff confidence that the technology is reinforcing the safety framework without requiring constant supervision.

The origin of Explore Zones illustrates the speed at which ideas can move from conversation to working prototype with AI. The feature was not planned before the trip. It was born during the trip itself, while I was demonstrating the existing tools to a colleague. As I walked through what the platform could do, the colleague suggested a GPS boundary feature for student free-time. I didn't have my laptop, but I had Claude on my iPhone. Standing in the city, I opened a voice conversation and described the idea aloud. Claude transcribed the conversation, thought through the technical approach, and generated a working prototype — an HTML file using open-source mapping libraries with GPS positioning and boundary detection — right there on my phone. That evening, back at the hotel with my MacBook, I refined the concept and deployed it as a full feature. From a spoken idea to a working tool in hours, conceived and prototyped without writing a single line of code by hand.

For students from a regional coastal town, this is transformative. They can walk the streets of Melbourne independently, make decisions about where to go, discover things on their own, and still have a digital safety net that keeps them oriented and within bounds. The technology doesn't replace the trust placed in students. It supports it.


Editing: SAAC Melbourne Explore Zone

Trip Details

Title: SAAC Melbourne Explore Zone

Badge Text: St Andrew's Anglican College — Term 1 2026

Find Location

Explore Zone Boundary

Draw the area students are allowed to explore.

6 points — drag vertices to adjust

Default Student View

Set what students see when they first open the map.

Saved: -37.8138, 144.9630 — zoom 16

☐ Lock student map to boundary area

Prevents students from panning away from the explore zone.

Explore Zones admin interface: staff draw the safe boundary, set the base and meeting point, and add landmarks for student orientation

Melbourne Icons: The Scavenger Hunt

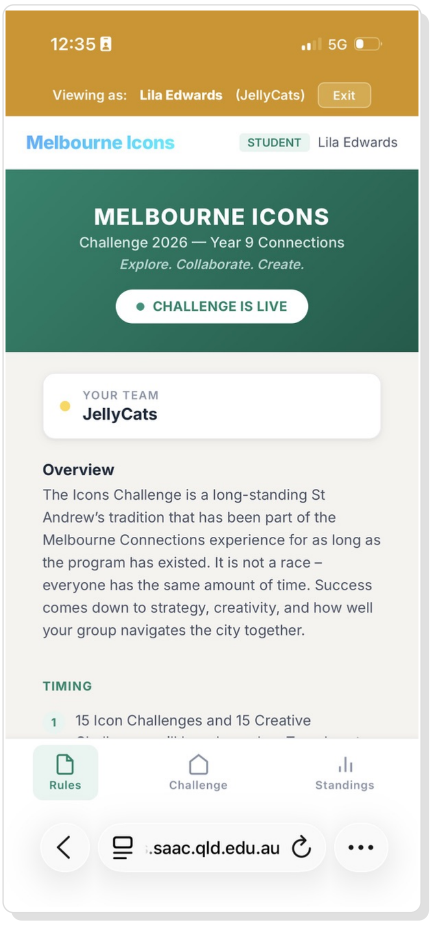
Melbourne Icons is a real-time team-based challenge platform. Seven teams competed across thirty-one items spanning landmarks, cultural challenges, and surprise bonus rounds. Students submitted photos and videos from the field. Staff reviewed and scored submissions in real time through a dedicated dashboard.

I designed the system to push students out of their comfort zones through creative challenge design. High-fiving a stranger. Singing a song at an iconic location. Asking a member of the public for directions while being filmed. Dancing to a busker. Finding a famous person for a photograph. These are not typical school activities. They require courage, humour, and the willingness to look a little foolish in pursuit of something meaningful.

On the technical side, the platform features multi-judge scoring with configurable judge panels, live leaderboards, client-side image compression to handle mobile uploads on cellular networks, a leaderboard freeze-and-reveal mechanism for dramatic winner announcements, and a complete administrative interface for challenge design. Staff can add bonus challenges mid-event, creating spontaneous moments of excitement. A staff member can hide somewhere in the city, release a riddle as a bonus challenge, and award points to the first team that finds them.

On challenge day, the system processed eighty-three submissions from seven teams across a three-hour window. Zero downtime. Zero upload failures. Average review turnaround of 2.2 minutes. The architecture handled every request without stress, running on a single Node.js process with SQLite. The largest video uploaded was 6.58 megabytes. Combined, the seven teams walked an estimated nineteen kilometres through the city.

Students didn't think about any of this. They opened a link, chose a challenge, took a photo, and submitted it. The complexity was invisible. The experience was everything.


12:35 5G

Viewing as: **Lila Edwards** (JellyCats) Exit

Melbourne Icons STUDENT Lila Edwards

MELBOURNE ICONS

Challenge 2026 — Year 9 Connections

Explore. Collaborate. Create.

● CHALLENGE IS LIVE

YOUR TEAM

JellyCats

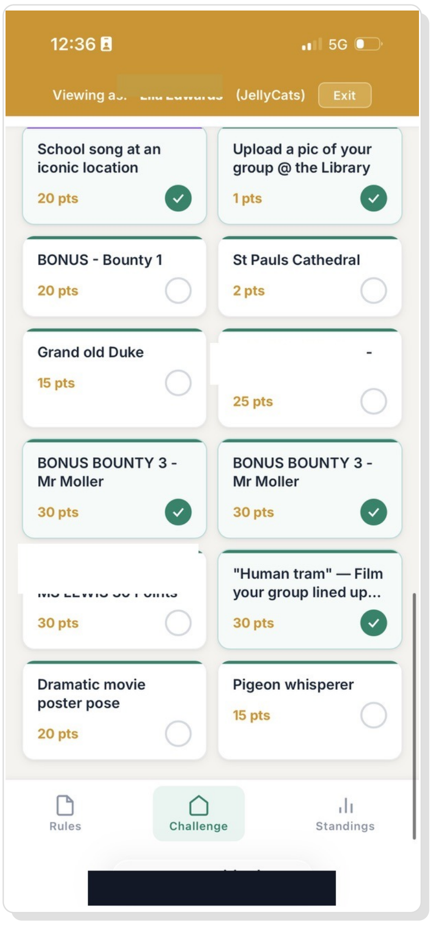
Overview

The Icons Challenge is a long-standing St Andrew's tradition that has been part of the Melbourne Connections experience for as long as the program has existed. It is not a race – everyone has the same amount of time. Success comes down to strategy, creativity, and how well your group navigates the city together.

TIMING

1 15 Icon Challenges and 15 Creative

Student interface for accessing challenges

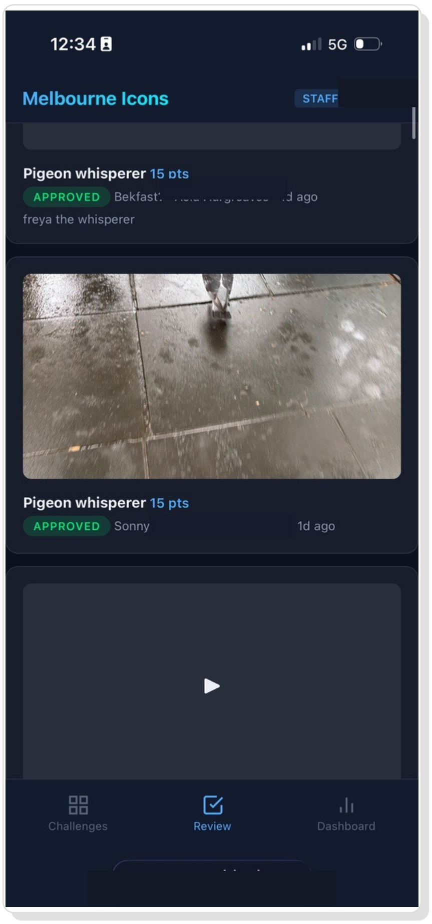

12:36 5G

Viewing as: [redacted] (JellyCats) Exit

School song at an iconic location 20 pts ✓	Upload a pic of your group @ the Library 1 pts ✓
BONUS - Bounty 1 20 pts ○	St Pauls Cathedral 2 pts ○
Grand old Duke 15 pts ○	[redacted] - 25 pts ○
BONUS BOUNTY 3 - Mr Moller 30 pts ✓	BONUS BOUNTY 3 - Mr Moller 30 pts ✓
[redacted] 30 pts ○	"Human tram" — Film your group lined up... 30 pts ✓
Dramatic movie poster pose 20 pts ○	Pigeon whisperer 15 pts ○

[redacted]

Challenge list with details and submission options



12:34 5G

Melbourne Icons STAFF

Pigeon whisperer 15 pts

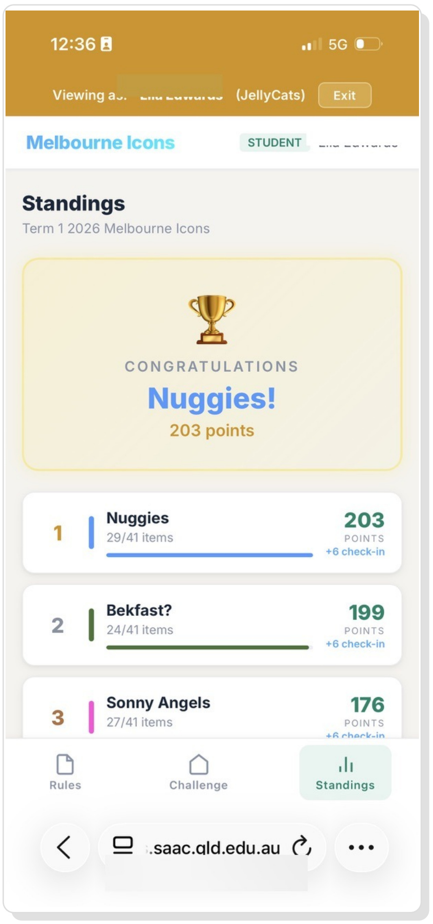
APPROVED Bekfast? [redacted] ·d ago

freya the whisperer

Pigeon whisperer 15 pts

APPROVED Sonny 1d ago

Staff dashboard for real-time submission review and scoring



12:36 5G

Viewing a[redacted] (JellyCats) Exit

Melbourne Icons STUDENT [redacted]

Standings

Term 1 2026 Melbourne Icons

CONGRATULATIONS

Nuggies!

203 points

1	Nuggies 29/41 items	203 POINTS +6 check-in
2	Bekfast? 24/41 items	199 POINTS +6 check-in
3	Sonny Angels 27/41 items	176 POINTS +6 check-in

Live leaderboard tracking team progress and scores

The Partnership: Human Intent, AI Capability

The development process was not what most people imagine when they hear "AI-generated code." There was no prompt that said "build me a trip management app." The process was iterative, conversational, and deeply grounded in my understanding of the problem domain.

A typical development session began with me describing a need rooted in lived experience. Not a technical specification, but a human problem: "When I walk into a restaurant with eighteen students, I need to show the chef every allergy and dietary requirement instantly." Or: "Staff need to see which multi-judge submissions they haven't scored yet, right at the top of the review screen." Or: "Students need to know immediately if they've wandered outside the safe zone."

The AI, working through Claude Code, would explore the existing codebase, understand the data model, propose an approach, and implement the changes. I would test, refine, redirect. The AI would adapt, learning the conventions of the project and building on previous decisions. Over time, the AI developed a deep familiarity with the architecture, the coding style, and the pedagogical intent behind every feature.

This is a fundamentally different relationship than either "the AI writes the code" or "the human writes the code with AI assistance." Ken Kahn, whose work at MIT's AI Laboratory alongside Papert and Minsky spans decades, describes this kind of AI as a "learner's apprentice" — not a tutor delivering answers, but an intellectual ally that amplifies what the human already knows. That framing captures exactly what I experienced. I understood the problem in the way only someone who has supervised teenagers in a foreign city can understand it. The AI understood software architecture, database design, and the mechanics of turning intent into functioning code. Neither understanding alone was sufficient. Together, we produced something neither of us could have built independently.

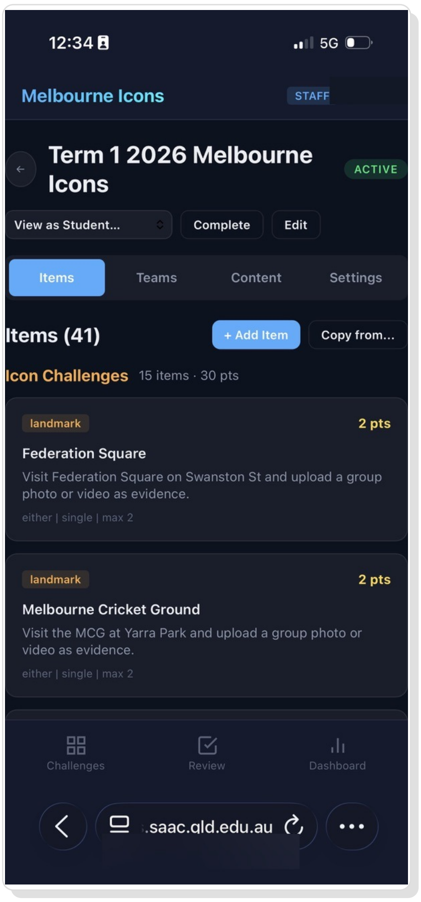
A key enabler of this partnership is the tooling itself. Claude Code runs directly inside the terminal, but its real power emerges when it operates within a professional development environment like Apple's Xcode. Xcode is the platform used to build native iOS applications — the same tool that professional developers use to create the apps on your phone. Traditionally, the learning curve is steep: Swift programming, interface design, build systems, device deployment. Claude Code collapses that curve dramatically. I describe what I need in conversational language — "add a screen that shows dietary requirements grouped by meal" — and the AI reads the existing project structure, understands the patterns already established, and writes the Swift code, the interface layout, and the data connections. I can see the changes appear in Xcode in real time, build the app, and test it on my phone within minutes. The conversation continues iteratively: I test, describe what needs adjusting, and the AI refines. The entire development cycle feels less like programming and more like a design conversation with a technically fluent colleague who happens to type very fast.

The partnership extended beyond writing code. I shared the program's risk assessment with the AI, discussing the specific concerns and challenges of taking regional students into a major city. The AI helped think through how technology could address those risks without diminishing the experience. It was a genuine dialogue about design, safety, and purpose.

During the live challenge day itself, the partnership continued in real time. When a staff member encountered an error while scoring submissions, I described the problem to the AI, which diagnosed the issue and implemented a fix within minutes. When I wanted to pause the leaderboard for a dramatic winner reveal, the feature was designed, built, deployed, and working within the same session. The AI wasn't writing code in isolation. It was responding to the lived reality of an event in progress, guided by me — someone who understood exactly what was needed and why.

After the challenge, the AI analysed the technical performance data from the day, examining submission rates, upload sizes, review turnaround times, and system load patterns. But more than producing a technical report, it helped me understand what that data meant for the quality of the learning experience. A 2.2-minute average review turnaround meant students got feedback fast enough to stay motivated. Zero upload failures meant no student lost a moment of creative courage to a technical glitch. The gap analysis showed the system was available every second of the three-hour window, meaning the technology never interrupted the flow of the experience. Technical metrics became pedagogical insights.

Even this paper itself is a product of the partnership. The AI helped me move from conversational reflection to articulated ideas, drawing out the connections between lived experience and educational theory, structuring thoughts that had been forming across weeks of building and deploying into a coherent narrative. The process of writing became another form of construction: building meaning from experience with the help of a capable partner.


12:34 5G

Melbourne Icons STAFF

Term 1 2026 Melbourne Icons ACTIVE

View as Student... Complete Edit

Items Teams Content Settings

Items (41) + Add Item Copy from...

Icon Challenges 15 items · 30 pts

landmark 2 pts
Federation Square
Visit Federation Square on Swanston St and upload a group photo or video as evidence.
either | single | max 2

landmark 2 pts
Melbourne Cricket Ground
Visit the MCG at Yarra Park and upload a group photo or video as evidence.
either | single | max 2

Challenges Review Dashboard

.saac.qld.edu.au

Staff interface for real-time challenge creation and management

The Constructionist Tradition

What happened in this project sits within a tradition that stretches back more than four decades. Seymour Papert, working at MIT's Artificial Intelligence Laboratory alongside Marvin Minsky, articulated a vision of computing in education that was radical in the 1980s and remains radical today:

> *"The child programs the computer and, in doing so, both acquires a sense of mastery over a piece of the most modern and powerful technology and establishes an intimate contact with some of the deepest ideas from science, from mathematics, and from the art of intellectual model building." — Seymour Papert, Mindstorms*

Papert's constructionism, the idea that people learn most powerfully when they are building personally meaningful artifacts, was never just about children. It was about the relationship between the builder, the tool, and the knowledge that emerges through the act of construction. The same principle applies when an educator builds software to serve their students. The act of building forces a depth of engagement with the problem that no amount of vendor evaluation or requirements specification can match.

When I designed the scoring system for Melbourne Icons, I wasn't configuring someone else's software. I was making decisions about how judging should work based on my knowledge of the staff who would be scoring, the time pressures of a live event, and the goals of the challenge. When I realised mid-event that the "required judges" model was too rigid, because some judges wouldn't reach every submission, I could immediately redesign it to an "up to five judges, average as you go" model. The change was implemented and deployed within the hour. This responsiveness is only possible when the person who understands the educational context is the same person shaping the technical solution.

As early as 1977, Kahn identified three ways AI and education interact: students using AI tools in their projects, students creating artefacts by interacting with AI, and using computational models of thinking to design learning itself. The Melbourne project touched all three. I used AI tools to build software. I created artefacts — functioning applications — through interaction with AI. And the process itself forced me to think computationally about learning design: how students navigate a city, how staff manage risk, how a challenge sustains engagement over three hours. Mitchel Resnick's "Four P's" framework — projects, passion, peers, and play — offers another lens. The scavenger hunt was a project driven by passion, experienced with peers, in a spirit of play. But so was the act of building it. I was learning through making, just as Papert always argued we should.

AI-assisted development positions the educator at the intersection of deep domain knowledge and technical capability, providing access to the craft of software construction without requiring years of formal training in computer science. I bring the "what" and the "why." The AI provides the "how."

Risk, Safety, and Pushing the Boundaries of Experience

David Loader, the pioneering Australian principal who created the world's first one-to-one laptop program at Methodist Ladies' College in 1990, understood something essential about the relationship between risk and learning:

> *"To not risk is to not learn. Or to state this more positively; risk taking can make one open to learning and can ensure that this learning is retained." — David Loader, The Inner Principal*

Taking thirty-five regional students into Melbourne involves real risk. Traffic, public transport, strangers, unfamiliar environments, students navigating independently for the first time in a city of five million people. Every year, the risk assessment grows longer. Every year, the duty of care expectations become more detailed. The danger is not that these requirements are wrong. They are essential. The danger is that without the right tools, managing the risk becomes so consuming that it crowds out the very experience the trip exists to provide.

This is where purpose-built software changes the equation. When every student's allergy information is one tap away, a restaurant meal becomes a manageable moment rather than a source of anxiety. When emergency contacts are accessible in seconds, not minutes, the margin of safety widens. When students carry a live GPS map showing their boundaries, the meeting point, and their own position, free exploration time becomes genuinely free rather than a source of constant worry for supervising staff.

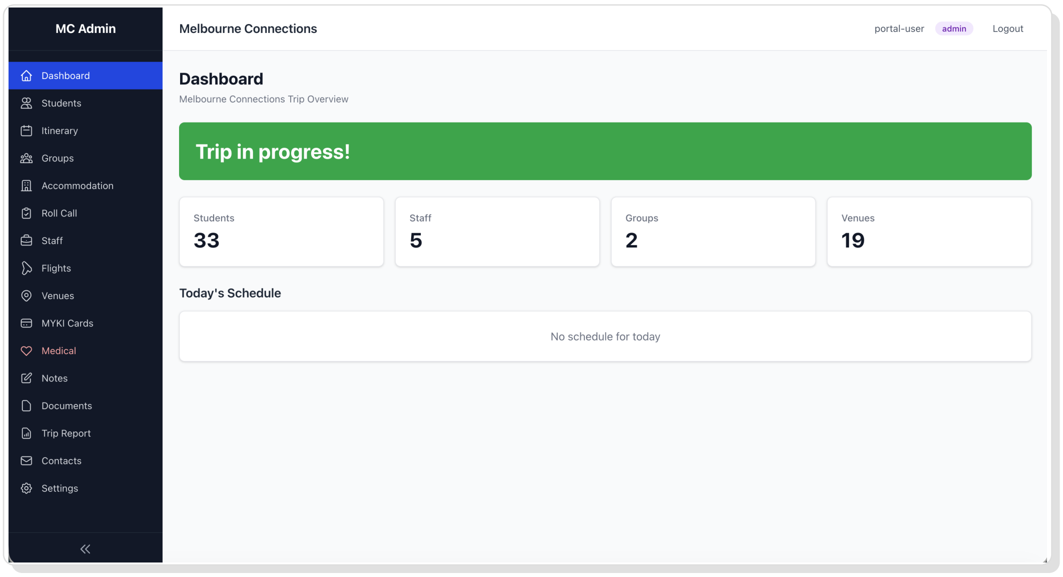
The technology doesn't eliminate risk. It manages risk so efficiently that educators can focus on what matters: the learning. A staff member who isn't anxiously searching through printed lists for a student's medical information is a staff member who is present, attentive, and able to engage with the educational moment unfolding in front of them.

The AI played a direct role in this thinking. I shared the program's risk assessment and discussed the specific concerns of the trip. Together, we worked through how each feature could address a real safety need without constraining the experience. The GPS boundary system emerged from this dialogue. So did the offline-first architecture, because the moment you most need safety-critical information is often the moment your internet connection fails. So did the one-tap emergency contacts, because in a genuine emergency, every second of searching is a second too many.

Loader also wrote:

"Learning is not some technical task like computer programming; it is integral to the person. It is part of the spirit, the soul and the heart of a human being." — David Loader, Jousting for the New Generation

The software I built for Melbourne was technical, but the purpose it served was anything but. Every feature traced back to a human intent: help students be brave, help staff feel confident, create space for young people to surprise themselves. The technology was always in service of the spirit of the experience.



Dashboard
Melbourne Connections Trip Overview

Trip in progress!

Students	Staff	Groups	Venues
33	5	2	19

Today's Schedule

No schedule for today

Web-based management backend for the Melbourne Connections staff trip management platform

AI as the Educator's Apprentice

The model of human-AI collaboration demonstrated in this project aligns with an emerging understanding of AI's role in education, one that resists both the utopian narrative of AI as an all-knowing tutor and the dystopian narrative of AI as a threat to human agency. Kahn's framing of AI as the learner's apprentice — a co-thinker, pair coder, brainstorming buddy, and intellectual ally that amplifies human potential rather than replacing it — describes exactly what happened in this project. The AI was never in charge. It was capable, fast, and tireless, but it waited for direction. The intent, the judgement, and the accountability were always mine.

In the constructionist tradition established by Papert and extended by Kahn, Resnick, Stager, and others, the most powerful educational technology is not the one that delivers content most efficiently but the one that enables the learner to build something meaningful. When that principle is applied to educators themselves, a profound shift occurs: the teacher is no longer a consumer of educational technology but a creator of it.

This shift matters because educators possess a form of knowledge that no technology company can replicate. I know my students. I know the specific anxieties of a Year 9 student being asked to approach a stranger in a foreign city. I know which staff member will be meticulous about scoring and which will need a streamlined interface. I know that the moment of revealing the winner needs to be theatrical, not transactional. I know that students from a regional coastal town experience Melbourne differently than students from Sydney or Brisbane. This contextual, embodied, relational knowledge is precisely what makes educator-built software qualitatively different from generic platforms.

Minsky argued that intelligence itself is not a singular mechanism but emerges from the interaction of many diverse agents, each contributing a different form of understanding:

> *"What magical trick makes us intelligent? The trick is that there is no trick. The power of intelligence stems from our vast diversity, not from any single, perfect principle."*
> *— Marvin Minsky, The Society of Mind*

The partnership between an educator and an AI embodies this principle. I contribute pedagogical knowledge, relational understanding, institutional context, and creative vision. The AI contributes technical fluency, architectural patterns, rapid implementation, and the ability to hold a complex codebase in working memory. What emerges from the collaboration is greater than either party could produce alone.

What the Students Actually Experienced

On 2 March 2026, seven teams dispersed from the State Library of Victoria into the streets of Melbourne. Over the next three hours, every single team completed the challenge of high-fiving a stranger. Seven groups of fifteen-year-olds approached members of the public, introduced themselves, and slapped palms. They sang songs at iconic locations. They asked strangers for directions while being filmed. They danced to buskers. They tracked down famous people and asked for photographs. One team walked all the way to the MCG, a five-kilometre round trip, to film their team song. Another team earned a perfect score for singing a nursery rhyme while skipping along the Yarra River.

The team that completed the most challenges, fourteen out of thirty-one, maintained a submission rate of thirty-one per hour, uploading a new photo or video every two minutes for thirty-three straight minutes. The team with the highest quality scores averaged 93% across all judged submissions, with four perfect scores. Combined, the seven teams walked an estimated nineteen kilometres through the city.

None of this required the students to think about technology. They opened a link on their phones, chose a challenge, took a photo or video, and submitted it. The technology was invisible. The experience was everything.

Melbourne Icons was competitive, creative, physical, social, and occasionally terrifying. It asked students to do real things in a real city with real people. The technology simply made it possible to orchestrate, track, judge, and celebrate those real things at scale.

But the scavenger hunt was only the visible peak. Underneath it, Melbourne Connections was quietly doing its work throughout the entire week. Every restaurant meal went smoothly because dietary information was instantly accessible. Every free exploration session was supported by GPS boundaries and meeting points. Every roll call was logged, every venue navigated, every emergency contact one tap away. The students experienced a trip that felt free and adventurous. The staff experienced a trip that felt manageable and well-supported. Both perceptions were accurate, and both were enabled by the same underlying technology.

What Happens Next: The Collaborative Horizon

As I write this, the Melbourne trip is being experienced again, this time with three incredible colleagues. And something is happening that was not fully anticipated.

The colleagues are curious. They are using the tools, and they are seeing possibilities. They notice things that weren't apparent before. They have insights born from their own expertise and their own relationships with students. One suggests a refinement to how roll calls work. Another sees an opportunity to use the platform for student reflection, giving students a way to easily record and capture their thoughts in the moment, not as homework days later but as a living response to what they are experiencing. A third identifies a workflow improvement in the review process that would have taken weeks to discover on my own.

The technical platform makes these refinements straightforward. What once required a software development cycle now requires a conversation. An educator describes what they need, and within hours, the tool evolves to match their insight. The system improves not because of a product roadmap designed by people who have never supervised a school trip, but because the people who live the experience every day are shaping the solution.

This points toward something genuinely significant. Imagine if the diverse insight and expertise of a broad range of professional educators could flow directly into the tools they use. Not through feature requests submitted to a vendor. Not through workarounds and compromises. Directly. Teachers, counsellors, outdoor education specialists, learning support staff, each contributing their unique understanding to software that serves their students.

> *"The scandal of education is that every time you teach something, you deprive a student of the pleasure and benefit of discovery." — Seymour Papert*

The same principle applies to the tools educators use. Every time a vendor decides how a feature should work, they deprive an educator of the insight that comes from designing it themselves. When educators build, they discover. They discover what matters, what works, and what their students actually need. That discovery is valuable. It is, in Papert's sense, a form of learning.

What does education look like when the people who live it build the solutions? It is possible now. Not as a distant aspiration, but as a practical reality. An educator with domain expertise, an AI development partner, and the willingness to build can create tools that are more responsive, more contextual, and more aligned with the purpose of education than anything available on the market. And when that capability is distributed across a team of educators, each contributing their own professional knowledge, the potential is extraordinary.

Implications

Educators as domain experts deserve builder tools. Teachers possess deep, contextual knowledge of their students, their programs, and their institutional cultures. When AI lowers the technical barrier to software creation, this knowledge can be expressed directly in the tools that shape student experiences. The result is software that fits like a bespoke suit rather than an off-the-rack uniform.

The process of building is itself valuable. Papert's constructionism applies to educators as much as to students. The act of building the Melbourne trip platform forced a depth of thinking about challenge design, user experience, staff workflow, and pedagogical intent that would not have occurred through merely configuring an existing product. I understand the system intimately because I made every decision that shaped it.

AI partnership is not automation. At no point in this project did the AI make pedagogical decisions. It did not choose which challenges to include, how scoring should work, or what the student experience should feel like. It implemented, suggested, debugged, and refined. I remained the designer, the decision-maker, and the person accountable for the outcome. This is a partnership model, not a replacement model.

Safety and experience are not opposites. Purpose-built software can manage risk so efficiently that it creates more space for learning, not less. When safety information is instant, when boundaries are visible, when communication is one tap away, educators can push the boundaries of what they offer students because they have genuine confidence in their ability to manage whatever arises.

Real-time adaptability changes what is possible. The ability to modify software during a live event, adding features, fixing issues, adapting to unexpected needs, represents a qualitative shift in what educator-built tools can achieve. When I conceived, designed, and deployed the leaderboard freeze feature within minutes, the boundary between software development and responsive teaching effectively dissolved.

None of this happens without the right IT team. The educator-as-builder model only reaches production when there is an IT team willing to deploy, secure, and maintain the infrastructure behind it. They are the roadies who make the rockstars possible. I am fortunate to lead an IT team who share this vision for learning and are proud to put their technical expertise in the service of students and educators. The final hurdle of any project like this is not writing the code — it is deploying it into a school's production environment with enterprise-grade authentication, access control, and audit logging. That requires an IT team who understand not just the technology, but the philosophy of what school IT exists to serve.

Conclusion

Loader wrote in The Inner Principal about the constraints we impose on ourselves as educators, the things we assume we cannot do, the boundaries we accept without questioning. For decades, custom software has been on the other side of that boundary for most teachers. You could use technology. You could integrate technology. You could even advocate for technology. But building it? That was for other people.

AI-assisted development tools like Claude Code are dissolving that boundary. Not by making software development trivial, because it isn't, but by making it accessible to people whose primary expertise lies elsewhere. An educator who understands their students, their program, and their context can now translate that understanding directly into functioning software, working in partnership with an AI that handles the technical craft.

> *"We imagine a school in which students and teachers excitedly and joyfully stretch themselves to their limits in pursuit of projects built on their own visions." — Seymour Papert*

On a warm March afternoon in Melbourne, thirty-five teenagers stretched themselves to their limits: singing in public, high-fiving strangers, sprinting across a city they had never navigated alone. Behind the scenes, I had stretched myself too, working with an AI to build the infrastructure that made the experience possible. Both acts of stretching mattered. Both produced learning that will last longer than any individual lesson.

The Melbourne program was not a technology project. It was a human project, enabled by technology.

References

Loader, D. (1997). The Inner Principal: Reflections on Educational Leadership. Routledge. Reissued 2016 by Constructing Modern Knowledge Press.

Loader, D. (1997). Jousting for the New Generation: Challenges to Contemporary Schooling. ACER Press.

Minsky, M. (1986). The Society of Mind. Simon & Schuster.

Papert, S. (1980). Mindstorms: Children, Computers, and Powerful Ideas. Basic Books.

Papert, S. (1993). The Children's Machine: Rethinking School in the Age of the Computer. Basic Books.

Kahn, K. (1977). Three Interactions between AI and Education. MIT Artificial Intelligence Laboratory.

Kahn, K. (2025). The Learner's Apprentice: AI and the Amplification of Human Creativity. Constructing Modern Knowledge Press.

Resnick, M. (2017). Lifelong Kindergarten: Cultivating Creativity through Projects, Passion, Peers, and Play. MIT Press.

Stager, G. & Martinez, S. (2013). Invent to Learn: Making, Tinkering, and Engineering in the Classroom. Constructing Modern Knowledge Press.

Part Two

How It Works

A companion walkthrough of every workflow, screenshot, and real-time build from the trip — including twelve features conceived and shipped to production in a single day.

Workflows, Screenshots, and What We Built on the Fly

This companion document provides the practical detail behind *Fieldtripping with AI*. It covers three things: the workflow for each platform, the development workflow with AI, and a detailed account of what was designed, built, and deployed in real time during the live event.

Part 1: Melbourne Connections — Educator Workflow

Melbourne Connections is the staff-facing trip management platform. It comprises two components: the Staff Companion app (a native iOS app for staff iPhones) and Explore Zones (a GPS boundary tool for student free-time). Here is how I set up and use each.

Staff Companion App

The Staff Companion is a native iOS app, built in Xcode with the help of Claude Code. It runs directly on staff iPhones with full offline capability, putting every piece of trip information in a staff member's pocket without relying on an internet connection.

Step 1: Ingest medical and personal data from school systems

The school's core systems already hold sensitive medical information, dietary requirements, and emergency contacts. Every modern school information system provides APIs — programmatic interfaces that allow other software to request data securely. Rather than manually retyping this information into spreadsheets (the traditional approach, which introduces transcription errors and produces documents that are stale the moment they are printed), the AI helped me connect to these APIs directly, pulling the data encrypted and transforming it into the formats the app needs.

The security architecture for this integration was itself designed in partnership with the AI. Claude helped design a workflow where API authentication credentials are stored securely in environment variables on my local machine — never hardcoded into the project files, never committed to version control, and never visible in the codebase itself. The AI guided best-practice deployment procedures: the data ingestion script runs locally on my secured machine, authenticates against the school's API using tokenised credentials, retrieves only the specific data fields required (medical, dietary, emergency contacts), encrypts the output, and bundles it into the Xcode project for compilation. At no point does confidential authentication information pass through the AI or appear in any shared file. The AI designed the security model, wrote the integration code, and advised on deployment — but the actual credentials remained solely under my control. This means the sensitive data flowing into the app is always current, always accurate, and always handled according to the same security standards the school applies to its own systems.

Step 2: Prepare the data files

The app is built around structured data files bundled into the Xcode project: students, staff, schedule, and locations. Sensitive medical and personal data flows securely from the school's existing systems. Staff details, the itinerary, and venue information are maintained directly. I can describe changes to the AI partner who updates the data and rebuilds the app, or edit the data files directly in Xcode.

Step 3: Customise the itinerary

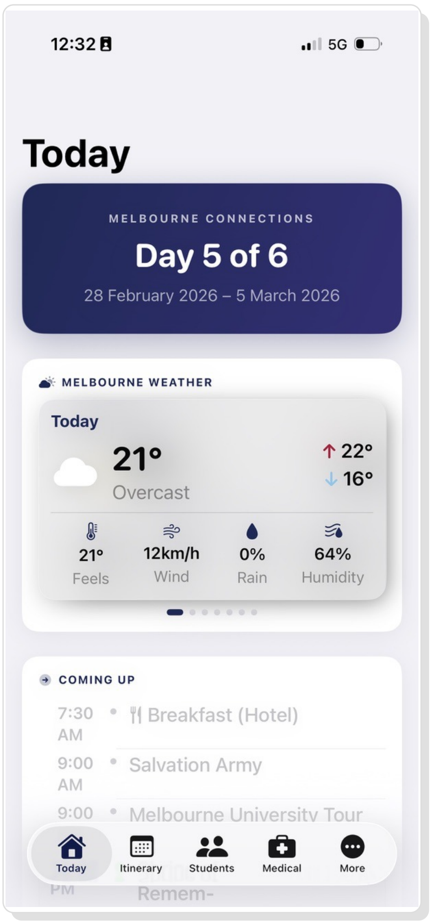
The schedule file holds every activity across the trip. Each entry has a date, start time, end time, location, description, and type (travel, activity, meal, free-time). The app automatically highlights the current and next activity based on the device clock, so staff always know where they should be.

Step 4: Review student information

Student records include group allocation, room number, roommates, dietary requirements, medical notes, and emergency contact details. Review the data pulled from the school systems for accuracy. This information is bundled into the app and available to every staff member on every device, instantly, even offline.

Step 5: Install on staff devices

The app is built in Xcode and deployed to staff iPhones. As a native iOS app, all data is stored on-device and available offline from the moment of installation. Staff can test it before departure to confirm everything is correct. The native platform also enables direct integration with iOS features like phone calls, messages, and Apple Maps.


12:32 5G

Today

MELBOURNE CONNECTIONS
Day 5 of 6
28 February 2026 – 5 March 2026

MELBOURNE WEATHER
Today
21°
Overcast
22°
16°

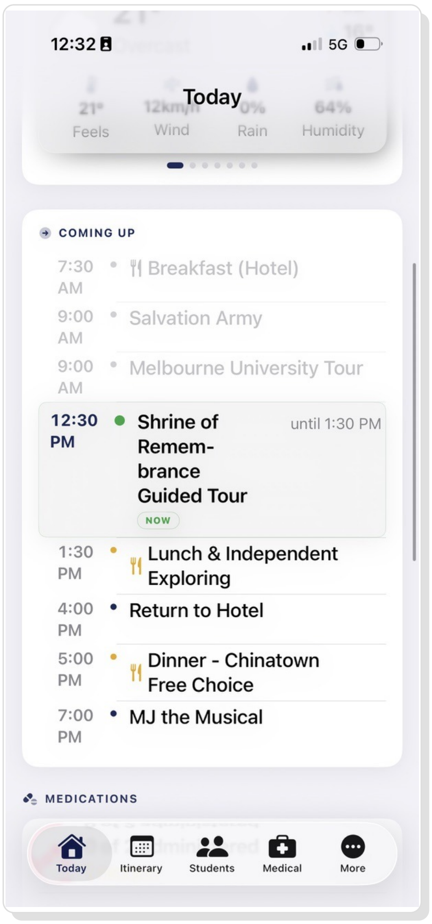
21°	12km/h	0%	64%
Feels	Wind	Rain	Humidity

COMING UP

7:30 AM	Breakfast (Hotel)
9:00 AM	Salvation Army
9:00	Melbourne University Tour

PM Remem-

Daily overview showing current and next activities


12:32 5G

Today

21°	12km/h	0%	64%
Feels	Wind	Rain	Humidity

COMING UP

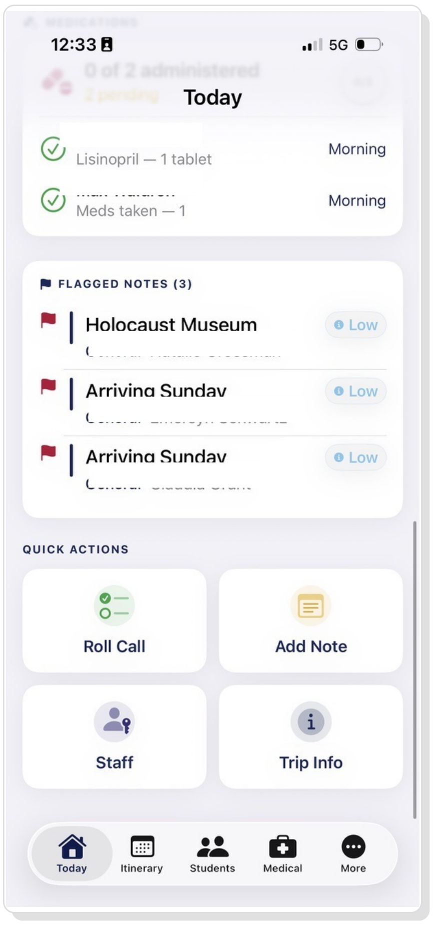
7:30 AM	Breakfast (Hotel)
9:00 AM	Salvation Army
9:00 AM	Melbourne University Tour
12:30 PM	Shrine of Remem-brance Guided Tour until 1:30 PM NOW
1:30 PM	Lunch & Independent Exploring
4:00 PM	Return to Hotel
5:00 PM	Dinner - Chinatown Free Choice
7:00 PM	MJ the Musical

MEDICATIONS

Full trip itinerary with times and locations

Step 1: Morning roll call

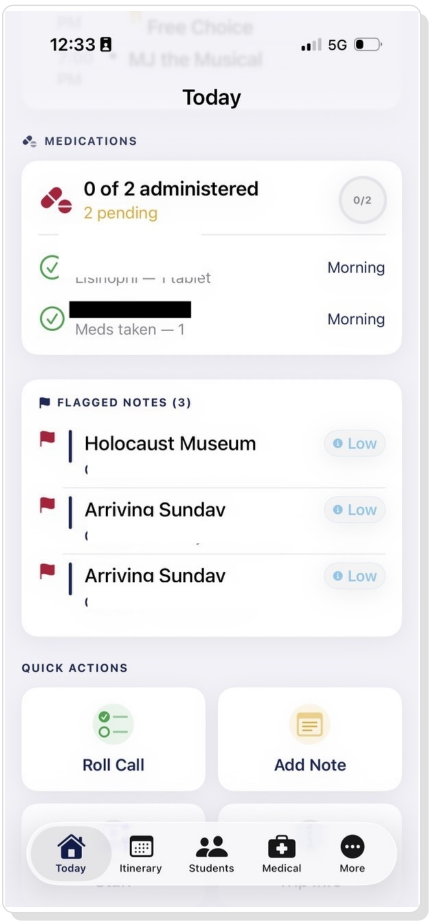
Open the Roll Call screen. Students are grouped by their assigned group (Group 1, Group 2). Tap individual names or use the "Check All" button. Add notes for any absences or issues. Tap "Save" to log the roll with a timestamp. The roll call history is stored on the device and can be emailed to the school office directly from the app.



Roll call with flagged notes and student wellbeing indicators

Step 2: Restaurant arrival

Open the Student Directory. Filter by group if needed. The dietary and allergy information is displayed alongside each student's name. Hand the phone to the chef or restaurant manager to show them the complete list. No printed sheets required.


12:33 5G
Today
MEDICATIONS
0 of 2 administered
2 pending
0/2
Lisinopril — 1 tablet Morning
[redacted]
Meds taken — 1 Morning
FLAGGED NOTES (3)
Holocaust Museum Low
Arriving Sunday Low
Arriving Sunday Low
QUICK ACTIONS
Roll Call
Add Note

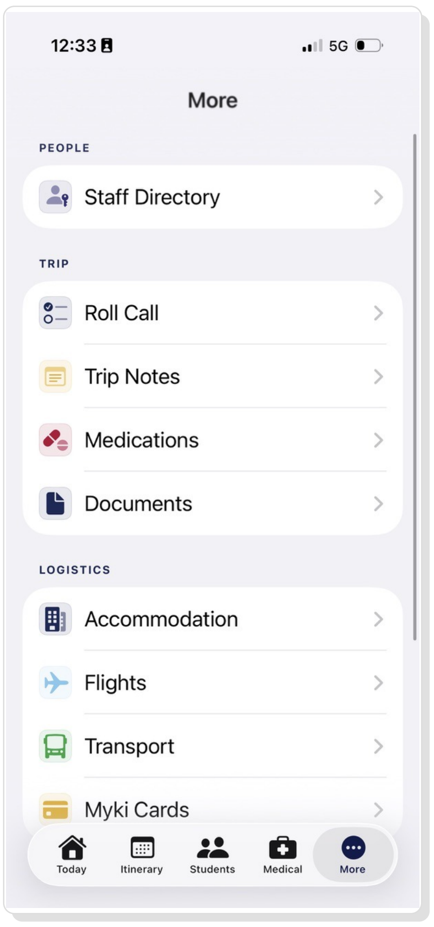
Student directory with dietary requirements and medical information

Step 3: Navigating to a venue

Open the Locations screen. Tap any venue to see its address, phone number, and a "Navigate" button that opens Apple Maps (or Google Maps) with the GPS coordinates pre-loaded. Staff don't need to search for the address.

Step 4: Emergency contact

Open the Contacts screen. Tap the phone icon next to any staff member's name to call them directly. Tap the message icon to send an SMS. One tap from the app to a live call.



Quick access to relevant trip information and key contacts

Explore Zones

Explore Zones gives students a live GPS map during free exploration time, showing their position within a defined safe boundary. Staff create and manage the zones through an admin interface.

Explore Zones was not planned before the trip. It was born during the trip itself. While demonstrating the existing platforms to a colleague in Melbourne, the conversation turned to student free-time and the colleague suggested a GPS boundary feature. I didn't have my laptop, but I had Claude on my iPhone. Standing in the city, I opened a voice conversation and described the idea aloud. Claude transcribed the discussion, worked through the technical approach, and generated a working prototype right there on my phone — an HTML file using Leaflet (an open-source mapping library) with real-time GPS positioning, polygon boundary detection, and status alerts. That evening, back at the hotel with my MacBook, I refined the prototype into a full-featured admin and student interface and deployed it as a live feature. The entire journey from a spoken idea to a deployed tool took hours, not weeks.

Step 1: Create a new zone

Open the Explore Zones admin panel. Tap "Create New Map." Enter a title (e.g. "Melbourne CBD Explore Zone"), a badge line (e.g. "Year 9 Melbourne Trip — Term 1 2026"), and a URL slug that becomes the student link.

Step 2: Draw the boundary

The admin interface shows a Leaflet map of Melbourne. Use the polygon draw tool to trace the boundary of the safe exploration area. Click to place points. Close the polygon. The boundary appears as a shaded region. Staff can edit the shape by dragging points, or redraw it entirely.

Step 3: Set the base location

Mark the hotel (or accommodation) on the map. Give it a label, name, and emoji. This appears as a prominent marker on the student map, so students always know how to get back.

Step 4: Set the meeting point

Mark the emergency meeting spot. Add help text (e.g. "Come here if you are lost or need help"). This is the fallback location if a student is disoriented or separated from their group.

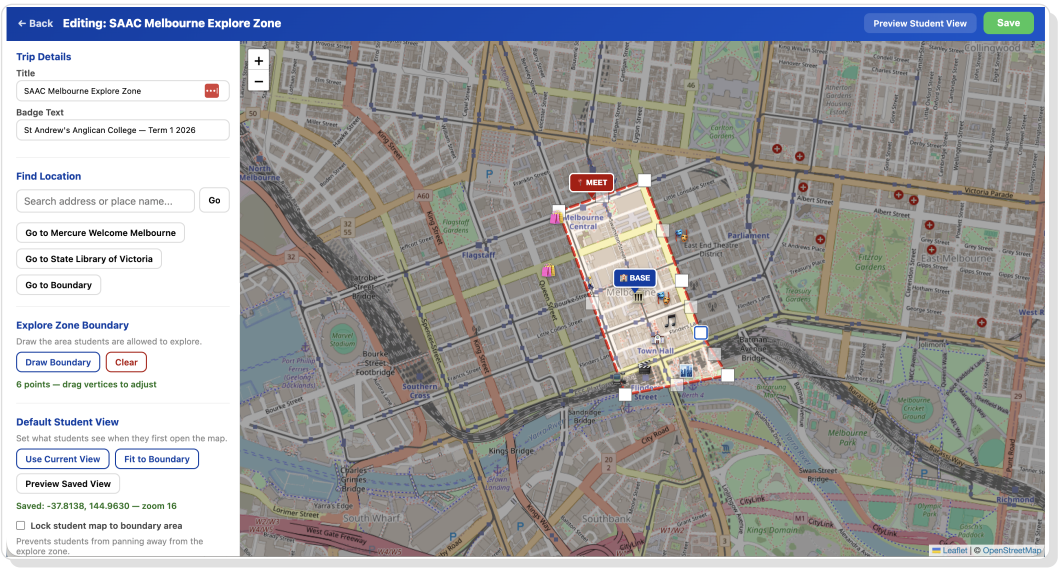
Step 5: Add landmarks

Drop pins for points of interest within the zone: shops, attractions, restaurants, train stations. Each gets a name and emoji icon. These help students orient themselves and discover the area.

Step 6: Share with students

Copy the student link or generate a QR code. Display the QR on a screen, print it on a handout, or send the link via your school's messaging system. Students open the link on their phones. No app install required.

Students open the link on their phones and see a live map. A blue dot shows their real-time GPS position, updating every four seconds. The boundary is drawn as a red dashed line. A status bar at the top shows green ("Inside explore zone") or red ("Outside explore zone!") depending on their position. Tapping the hotel card flies the map to the base location. Tapping the meeting spot card flies to the emergency rendezvous. Landmarks are displayed as emoji markers with labels. The entire interface is designed to be immediately understandable with zero instruction.



Editing: SAAC Melbourne Explore Zone

Trip Details

Title: SAAC Melbourne Explore Zone

Badge Text: St Andrew's Anglican College — Term 1 2026

Find Location

Search address or place name...

Explore Zone Boundary

Draw the area students are allowed to explore.

6 points — drag vertices to adjust

Default Student View

Set what students see when they first open the map.

Saved: -37.8138, 144.9630 — zoom 16

☐ Lock student map to boundary area

Prevents students from panning away from the explore zone.

Explore Zones admin interface: staff draw the safe boundary, set the base and meeting point, and add landmarks for student orientation

Part 2: Melbourne Icons — Educator Workflow

Melbourne Icons is the real-time scavenger hunt platform. The workflow has three phases: challenge setup (before the event), live management (during the event), and results (after the event).

Challenge Setup (Days or Weeks Before)

Step 1: Create a challenge

Open the Melbourne Icons staff dashboard. Navigate to Settings. Create a new challenge with a name, description, and optional start/end dates. The challenge starts in draft mode, invisible to students.

Step 2: Design the challenge items

Navigate to the Items tab. Add challenge items one by one. Each item has a title (e.g. "High-fiving a member of the public"), a category (landmark, culture, surprise, general), a point value, and a media type requirement (photo or video). For items that need subjective judging (e.g. "Singing a song at an iconic location"), set the scoring mode to "multi-judge" and configure the maximum number of judges (up to 5). The points for multi-judge items come from the average of all judge scores rather than a flat value.

Step 3: Configure bonus/hidden items

Any item can be marked as "bonus." Bonus items are hidden from students when the challenge is active but visible to staff with a [BONUS] tag. Staff can reveal bonus items at any time during the challenge by unticking the bonus checkbox. This enables surprise rounds, staff scavenger hunts (a staff member hides in the city and releases a riddle), or timed bonus windows.

Step 4: Set up teams

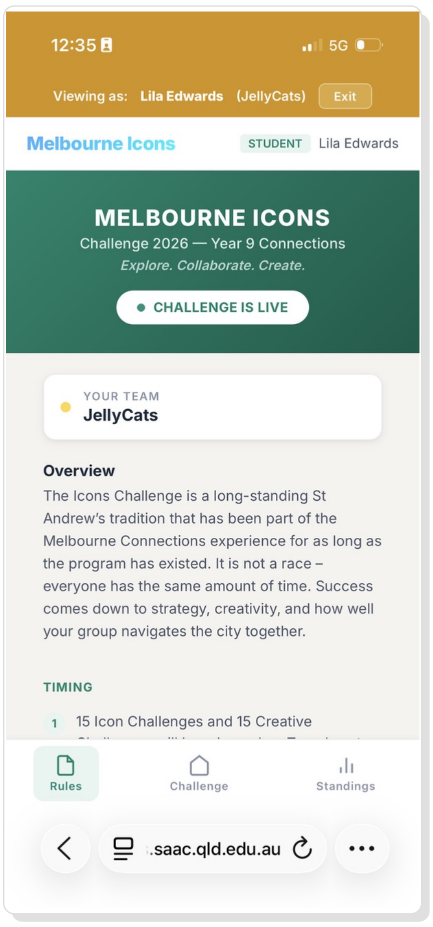
Navigate to the Teams tab. Create teams with names and colour assignments. Add students to each team. Team names are editable at any time (students can rename their own teams for fun). Each team gets a unique colour that appears on the leaderboard and throughout the interface.

Step 5: Customise the student landing page

The student-facing page has a content editor where staff can write instructions, rules, and encouragement. This content appears when students first open the app, before they start submitting.

Step 6: Test with a practice submission

Create a test item (e.g. "Upload a pic of your group at the library") in the "test" category. Have a student or staff member submit a photo to confirm the full pipeline works: upload, review, scoring, leaderboard update. This can be done the day before or even the morning of the event.



MELBOURNE ICONS

Challenge 2026 — Year 9 Connections

Explore. Collaborate. Create.

YOUR TEAM
JellyCats

Overview

The Icons Challenge is a long-standing St Andrew's tradition that has been part of the Melbourne Connections experience for as long as the program has existed. It is not a race – everyone has the same amount of time. Success comes down to strategy, creativity, and how well your group navigates the city together.

TIMING

1 15 Icon Challenges and 15 Creative

Student view of available challenges

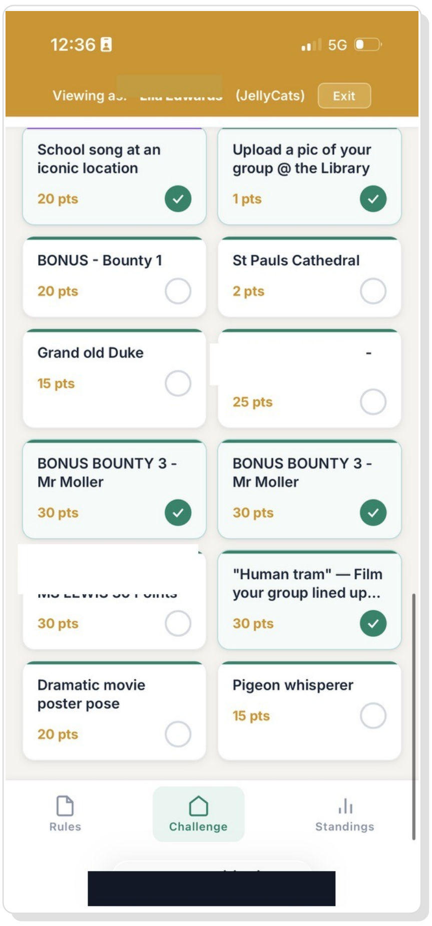
Live Event Management

Step 1: Activate the challenge

Set the challenge status to "active." Students can now see the challenge items and begin submitting. The Dashboard shows real-time statistics.

Step 2: Review submissions as they arrive

The Review screen shows incoming submissions in chronological order. Each submission card shows the team name, challenge item, media type, and a thumbnail. Tap to view the full photo or video. Approve or reject with a single tap. Add an optional rejection reason if declining a submission (e.g. "Can't see a high five"). For multi-judge items, the first approval triggers the item to count on the leaderboard immediately.



12:36 5G

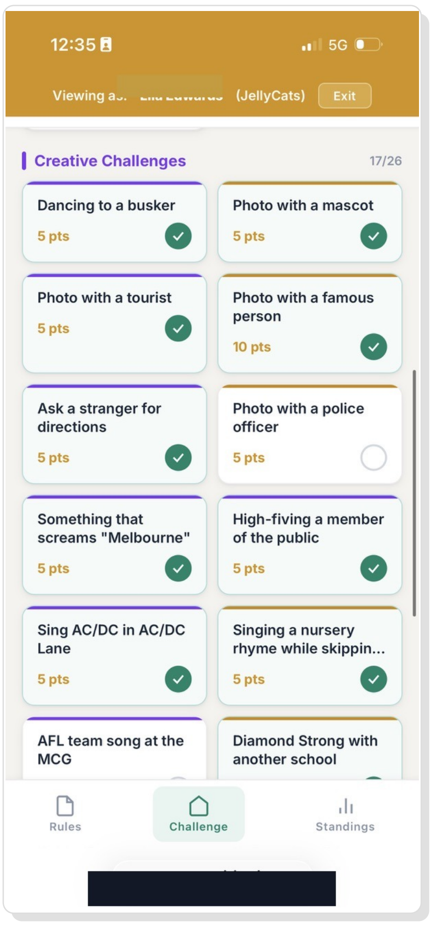
Viewing as [redacted] (JellyCats) Exit

School song at an iconic location 20 pts ✓	Upload a pic of your group @ the Library 1 pts ✓
BONUS - Bounty 1 20 pts ○	St Pauls Cathedral 2 pts ○
Grand old Duke 15 pts ○	[redacted] - 25 pts ○
BONUS BOUNTY 3 - Mr Moller 30 pts ✓	BONUS BOUNTY 3 - Mr Moller 30 pts ✓
[redacted] 30 pts ○	"Human tram" — Film your group lined up... 30 pts ✓
Dramatic movie poster pose 20 pts ○	Pigeon whisperer 15 pts ○

Rules Challenge Standings

[redacted]

Challenge list showing available items



12:35 5G

Viewing as [redacted] (JellyCats) Exit

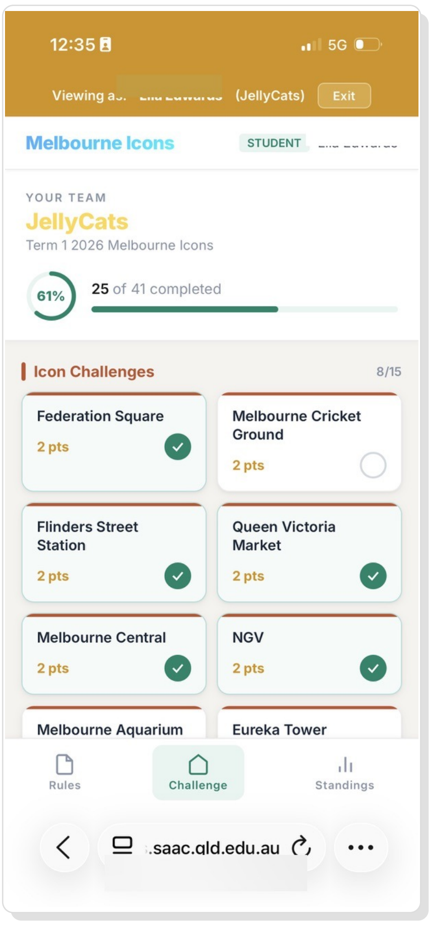
Creative Challenges 17/26

Dancing to a busker 5 pts ✓	Photo with a mascot 5 pts ✓
Photo with a tourist 5 pts ✓	Photo with a famous person 10 pts ✓
Ask a stranger for directions 5 pts ✓	Photo with a police officer 5 pts ○
Something that screams "Melbourne" 5 pts ✓	High-fiving a member of the public 5 pts ✓
Sing AC/DC in AC/DC Lane 5 pts ✓	Singing a nursery rhyme while skippin... 5 pts ✓
AFL team song at the MCG	Diamond Strong with another school

Rules Challenge Standings

[redacted]

Challenge list with point values



12:35 5G

Viewing as [redacted] (JellyCats) Exit

Melbourne Icons STUDENT [redacted]

YOUR TEAM

JellyCats

Term 1 2026 Melbourne Icons

61% 25 of 41 completed

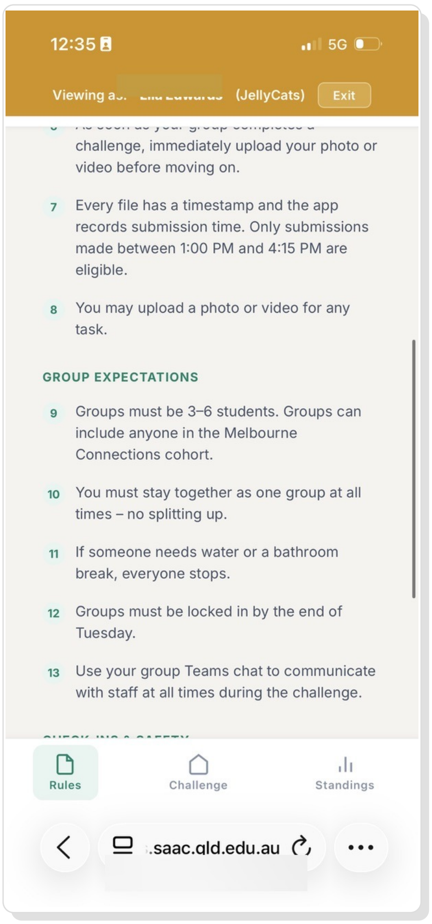
Icon Challenges 8/15

Federation Square 2 pts ✓	Melbourne Cricket Ground 2 pts ○
Flinders Street Station 2 pts ✓	Queen Victoria Market 2 pts ✓
Melbourne Central 2 pts ✓	NGV 2 pts ✓
Melbourne Aquarium	Eureka Tower

Additional challenge items and categories

Step 3: Score judged items

Items with multi-judge scoring show a "Score" button. Tap to open the scoring modal, which displays any existing scores from other judges. Enter your score out of the maximum and an optional comment. The leaderboard updates automatically with the running average. The "Needs Your Score" section at the top of the Review screen highlights items where you haven't yet added your score, so nothing gets missed.


12:35

Viewing as [redacted] (JellyCats) Exit

challenge, immediately upload your photo or video before moving on.

7 Every file has a timestamp and the app records submission time. Only submissions made between 1:00 PM and 4:15 PM are eligible.

8 You may upload a photo or video for any task.

GROUP EXPECTATIONS

9 Groups must be 3–6 students. Groups can include anyone in the Melbourne Connections cohort.

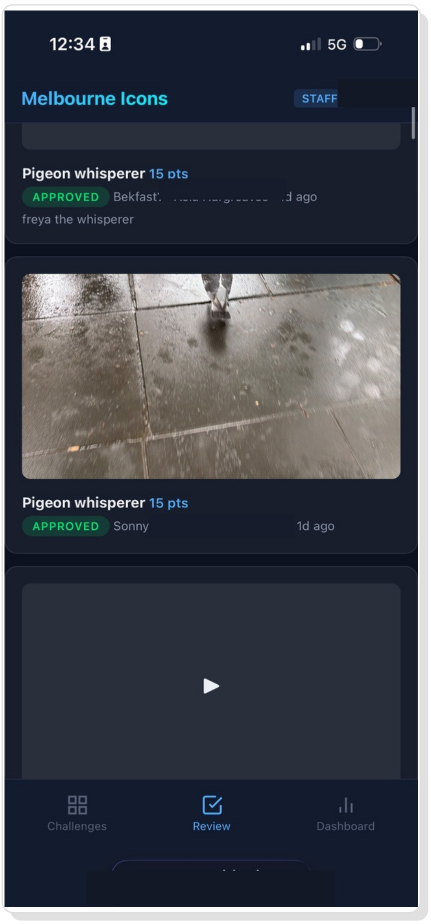
10 You must stay together as one group at all times – no splitting up.

11 If someone needs water or a bathroom break, everyone stops.

12 Groups must be locked in by the end of Tuesday.

13 Use your group Teams chat to communicate with staff at all times during the challenge.

Student expectations and challenge guidelines

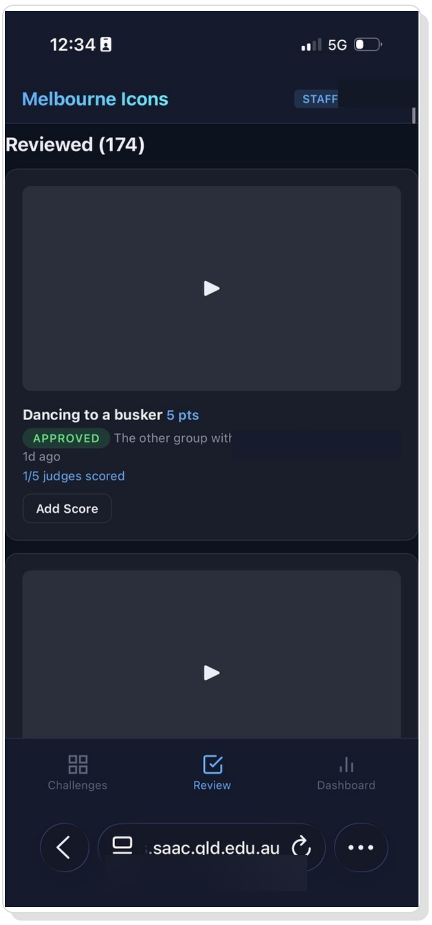

12:34

Melbourne Icons STAFF

Pigeon whisperer 15 pts
APPROVED Bekfast' [redacted] d ago
freya the whisperer

Pigeon whisperer 15 pts
APPROVED Sonny 1d ago

Staff review of incoming submissions



Additional staff review controls

Step 4: Release bonus challenges

At any point during the event, navigate to the Items tab and untick the "bonus" checkbox on a hidden item. It immediately becomes visible to all students. Use this for surprise rounds, timed bonuses, or to inject energy when the pace slows.

Step 5: Monitor the Dashboard

The Dashboard shows total submissions, submissions by team, approval/rejection counts, and the current leaderboard standings. Use this to gauge how the event is progressing and whether teams are engaging.

Step 6: Record safety check-ins

The check-in feature lets staff log when teams check in at designated points. Check-in bonus points are added to the team's total automatically.

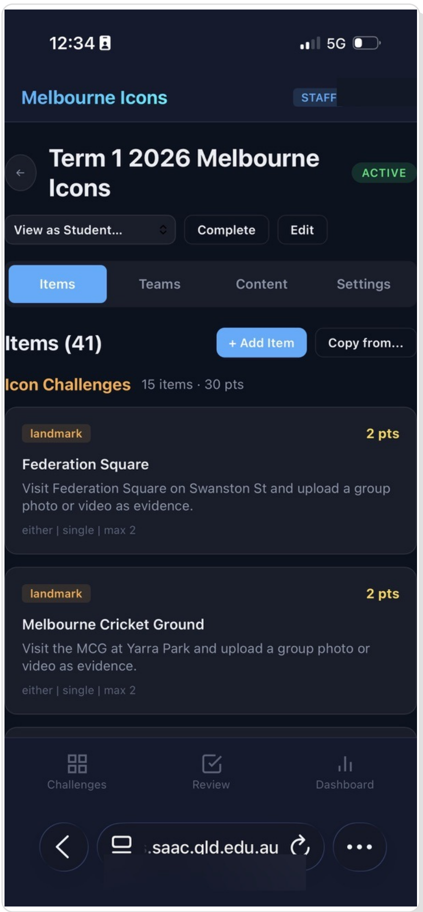
Results and Reveal

Step 1: Freeze the leaderboard

When the challenge window closes, navigate to Settings and tick "Pause Leaderboard." Students immediately see a "Final results are still being calculated... Stay tuned!" message with a pulsing animation instead of the standings. Staff still see the full leaderboard on the Dashboard and can continue reviewing and scoring.

Step 2: Complete final reviews and scoring

Work through any remaining pending submissions. The "Needs Your Score" section helps judges catch any items they missed. Take the time to get scores right without students watching numbers change in real time.



12:34 5G

Melbourne Icons STAFF

Term 1 2026 Melbourne Icons ACTIVE

View as Student... Complete Edit

Items Teams Content Settings

Items (41) + Add Item Copy from...

Icon Challenges 15 items · 30 pts

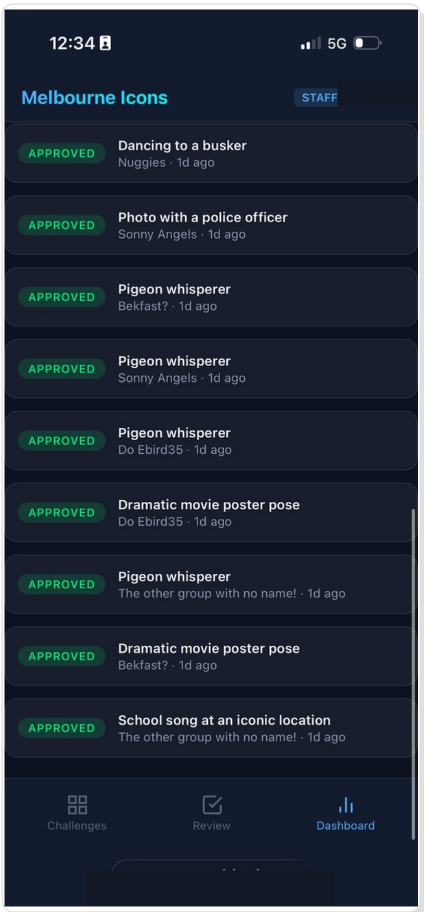
landmark 2 pts
Federation Square
Visit Federation Square on Swanston St and upload a group photo or video as evidence.
either | single | max 2

landmark 2 pts
Melbourne Cricket Ground
Visit the MCG at Yarra Park and upload a group photo or video as evidence.
either | single | max 2

Challenges Review Dashboard

.saac.qld.edu.au

Staff interface for setting up challenges



12:34 5G

Melbourne Icons STAFF

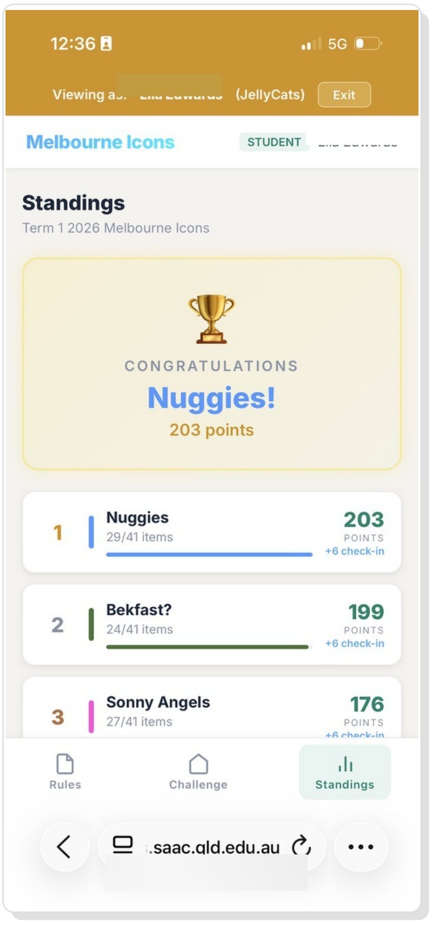
APPROVED	Dancing to a busker Nuggies · 1d ago
APPROVED	Photo with a police officer Sonny Angels · 1d ago
APPROVED	Pigeon whisperer Bekfast? · 1d ago
APPROVED	Pigeon whisperer Sonny Angels · 1d ago
APPROVED	Pigeon whisperer Do Ebird35 · 1d ago
APPROVED	Dramatic movie poster pose Do Ebird35 · 1d ago
APPROVED	Pigeon whisperer The other group with no name! · 1d ago
APPROVED	Dramatic movie poster pose Bekfast? · 1d ago
APPROVED	School song at an iconic location The other group with no name! · 1d ago

Challenges Review Dashboard

Mobile approval workflow for staff

Step 3: Reveal the winner

When ready, untick "Pause Leaderboard" in Settings. Students refresh and see a celebration banner with the winning team's name, colour, and a glowing animation, followed by the full final standings. Time this for maximum impact: gather students together, count down, then unfreeze.


12:36 5G

Viewing as [redacted] (JellyCats) Exit

Melbourne Icons STUDENT [redacted]

Standings

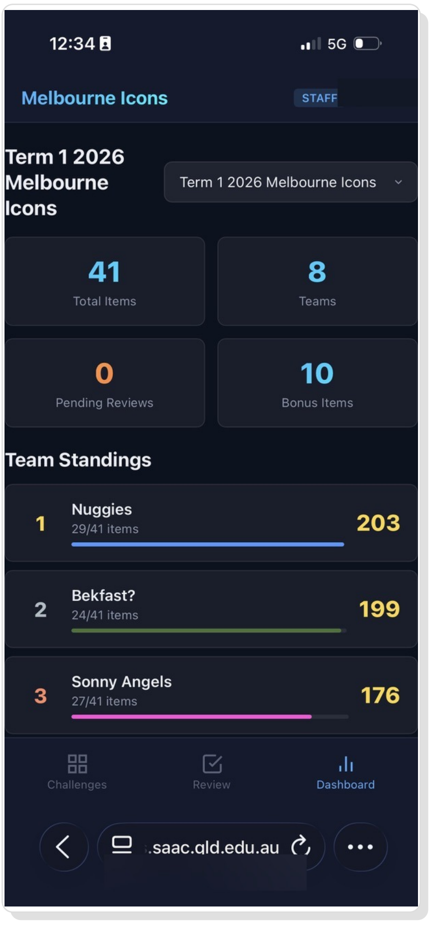
Term 1 2026 Melbourne Icons

CONGRATULATIONS

Nuggies!

203 points

1	Nuggies 29/41 items	203 POINTS +6 check-in
2	Bekfast? 24/41 items	199 POINTS +6 check-in
3	Sonny Angels 27/41 items	176 POINTS +6 check-in


12:34 5G

Melbourne Icons STAFF

Term 1 2026 Melbourne Icons

Term 1 2026 Melbourne Icons

41 Total Items	8 Teams
0 Pending Reviews	10 Bonus Items

Team Standings

1	Nuggies 29/41 items	203
2	Bekfast? 24/41 items	199
3	Sonny Angels 27/41 items	176

Student leaderboard showing team standings *Leaderboard with final standings*

Part 3: The Development Workflow with AI

For readers wondering what it actually looks like to build software in partnership with AI, here is the workflow.

The Tool: Claude Code

Claude Code is a command-line AI development tool that runs inside a terminal window. What makes it powerful is not just that it can write code, but that it operates directly inside the developer's own environment. For the web platform (Melbourne Icons and the Explore Zones system), Claude Code worked across the full stack: database, backend, frontend, and deployment scripts. But it was the native iOS development — building the Staff Companion app in Apple's Xcode — where the power of this approach became most apparent.

Xcode is the professional development environment used to build every app on an iPhone. Traditionally, the learning curve is steep: the Swift programming language, Apple's interface frameworks, build configurations, provisioning profiles, device deployment. It is not a tool designed for casual users. Claude Code changes that equation entirely. It reads the existing Xcode project — every Swift file, every interface definition, every data model — and understands the patterns and conventions already established. When I describe a new feature in plain language, it writes Swift code that fits seamlessly into the existing architecture. The changes appear in Xcode in real time. I build the app, deploy it to my phone, and test it. If something needs adjusting, I describe what I see and the AI refines. The entire cycle feels less like programming and more like a design conversation with a technically fluent colleague who happens to type very fast.

This matters because it means an educator can build a native iOS app — not a web page pretending to be an app, but a genuine native application with offline capability, direct access to phone features like calling and GPS, and the performance and reliability that comes with running natively on the device. The learning curve that would normally take months of formal study is compressed into a conversational workflow where the educator focuses on what the tool should do and the AI handles how to make it happen in Swift and Xcode.

The Workflow

Step 1: Describe the problem in plain language

I open Claude Code in a terminal window alongside Xcode (for the iOS app) or my code editor (for the web platform) and describe what I need. Not in code. Not in technical jargon. In the language of someone who lives the problem: "I need staff to be able to freeze the leaderboard so students see a holding message while we finish scoring, then reveal the winner with a big celebration banner when we're ready."

Step 2: The AI explores and proposes

Claude Code reads the existing codebase, understands the database schema, the API routes, the frontend structure, and the coding conventions already established. It proposes an approach: "I'll add a leaderboard_frozen column to the challenges table, update the settings form with a toggle, modify the leaderboard API to return an empty standings array for students when frozen, and add a winner banner component to the frontend." I review the approach and agree, adjust, or redirect. For the iOS app, the same principle applies: the AI reads the Xcode project, understands the existing Swift views and data models, and proposes changes that are consistent with the app's architecture.

Step 3: The AI implements across the full stack

Claude Code writes the code — whether that's database migrations and JavaScript for the web platform or Swift views and data models for the iOS app. It modifies multiple files in a single pass, maintaining consistency with the existing code style. I watch the changes appear in real time.

Step 4: I test

For the web platform, I open the browser, click through the new feature, and test it as a staff user and as a student user. For the iOS app, I build in Xcode and deploy to my phone. If something isn't right, I describe what I see: "The frozen message looks good but the winner banner needs to use the team's colour, not just gold." The AI adjusts.

Step 5: Deploy

For the web platform, I copy the files to the production server. The process manager restarts the application. For the iOS app, I rebuild in Xcode and deploy to staff devices. The feature is live. The full cycle from "I have an idea" to "students can see it" was often under thirty minutes.

Part 4: What We Built on the Fly

This is the part that is hard to believe until you've lived it. The following features were not planned weeks in advance. They were conceived, designed, built, tested, deployed, and used during the live event or in the hours immediately surrounding it. Each one emerged from a real problem encountered in real time by real people managing a real experience.

The timeline below covers approximately eight hours, from the morning preparations through the post-challenge analysis. Every entry represents a working feature that went from idea to production.

Challenge Day: 2 March 2026

8:30 AM Student Guide Text Refinements

While reviewing the student-facing guide one last time before the challenge, I noticed the heading "Before You Leave School" didn't make sense for students already in Melbourne. I described the issue to the AI. Within minutes, the heading was changed to "Before You Leave" and the test submission instructions were reordered to appear right after the "Get Set Up" section, so the first thing students see after setup is how to do a test upload. Deployed before students woke up.

9:00 AM Staff Guide Update: Bonus Challenge Instructions

I realised the staff guide didn't explain the bonus/hidden item feature clearly enough. I described to the AI how the feature works and asked it to add a section with a concrete example: "A staff member hides in the city and releases a bonus challenge with a riddle. Students solve the riddle to find the staff member and earn bonus points." The AI added two new sections to the staff guide, complete with step-by-step instructions, and renumbered the existing steps. I deployed to production.

10:00 AM Multi-Judge Scoring Overhaul

A fundamental design change I made the morning of the event. The original scoring system required a fixed number of judges (e.g. 3 out of 3) before a multi-judge score would count on the leaderboard. Thinking through the reality of four staff members trying to judge submissions while also supervising students in a busy city, I realised this was too rigid. If even one judge didn't reach a submission, the score would never count. I described the problem to the AI: "Instead of required judges being a set limit, it needs to be up to 5 judges. Post the score as each judge scores it. If one doesn't get to the final score, just average it out." The AI redesigned the scoring engine. Changes spanned four files: the scoring service (changed the threshold from "scores >= required" to "scores > 0"), the submissions route (auto-approve on first score, enforce maximum judges, add judge count tracking), and the frontend (show judge count badges, enable scoring on already-approved items, display existing scores in the score modal). Tested, deployed, and working before the challenge started.

10:30 AM Microsoft Forms Backup Plan

I created a plain-text list of all thirty-one challenge items with point values as a backup, in case the primary system had issues and submissions needed to be collected via Microsoft Forms. The AI queried the challenge database and generated the formatted list. Never needed, but ready.

1:00 PM Challenge Goes Live

Seven teams depart. First submissions arrive within two minutes. The system handles everything smoothly.

1:50 PM Live System Health Check

Fifty minutes into the challenge, I asked the AI to run a health check on the production system. The AI examined the database files (128 KB main database, 3.9 MB write-ahead log), counted uploads (99 files across 7 team directories, 261 MB total), verified the most recent upload timestamp, and checked the server proxy was responding. Confirmed: zero issues, all seven teams actively submitting, the last upload was two minutes ago. Full confidence to keep pushing the experience.

2:00 PM Bonus Challenge Ideas Generated on the Spot

As the challenge progressed, I wanted to inject more energy. I asked the AI for quick bonus challenge ideas that students could complete in the final stretch. The AI generated a batch of creative options tailored to Melbourne locations and the program's "step outside your comfort zone" ethos. Staff selected their favourites and released them as surprise bonus items through the existing bonus challenge feature.

2:30 PM "Already Scored" Error Fix

A staff member tried to update their score on a submission and hit an error: "You have already scored this submission. Use PUT to update." I screenshot the error and sent it to the AI. The AI diagnosed the issue immediately: the frontend was using a POST request (create new score) when it should use PUT (update existing score) for judges who had already scored. Within minutes, the AI implemented an automatic retry mechanism. If the POST fails with "already scored," the frontend silently retries with PUT to update the score instead. No more error messages. Deployed while the challenge was still running. Staff never saw the error again.

3:00 PM "Needs Your Score" Review Section

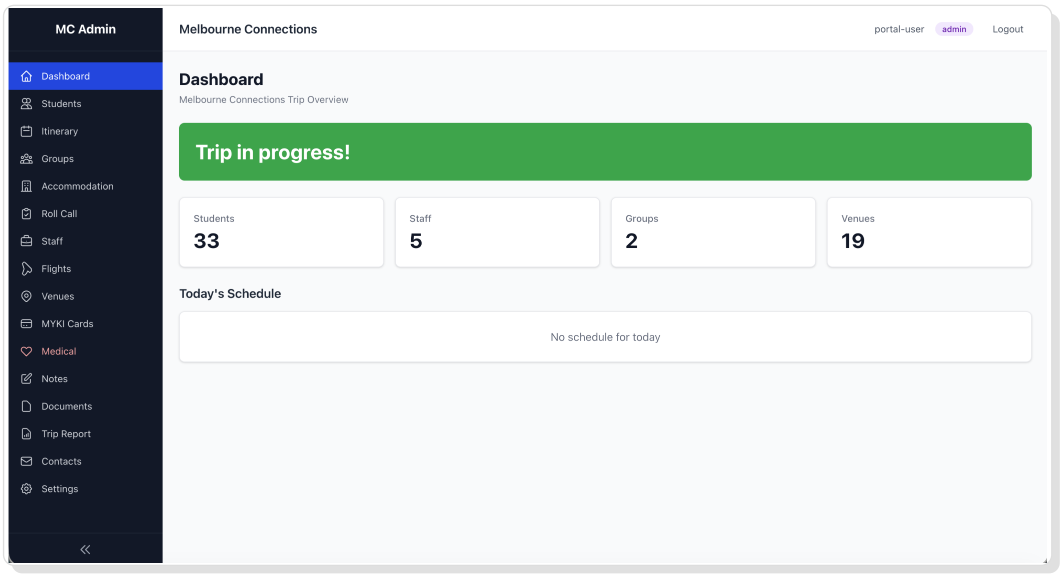
With multiple judges scoring different submissions at different times, it was hard for staff to know which items they still needed to score. I described the problem: "Put at the top of the review screen all of the submissions where the current user hasn't given a score yet." The AI added a current_user_scored flag to the backend API response, then built a "Needs Your Score" section at the top of the Review screen that filters for multi-judge items where the logged-in staff member hasn't scored yet. Each card shows how many judges have scored so far (e.g. "2/5 judges scored") with a prominent "Add Score" button. Deployed while staff were actively scoring.

3:30 PM Leaderboard Freeze Feature

The challenge window was closing. Staff needed time to finish reviewing and scoring the final submissions, but I didn't want students watching the leaderboard shift in real time as the last scores trickled in. The reveal should be a moment, not a gradual trickle. I described the idea: "Can we add a toggle for staff to pause the leaderboard? Students see 'Final results still being calculated' and then when I toggle it off, it's live with the final results and a big flashing banner congratulating the winning team." The AI designed and built the entire feature from scratch:

- Database: Added a leaderboard_frozen column to the challenges table
- Backend API: Modified the leaderboard endpoint to return empty standings for students when frozen, while staff still see full data
- Settings UI: Added a checkbox toggle ("Pause Leaderboard") in the staff settings panel
- Student view: Built a "Results being calculated" card with a pulsing timer animation
- Winner banner: Built a celebration component that detects the first-place team, displays their name in their team colour with a glowing gold border animation
- CSS animations: Created keyframe animations for both the frozen state and the winner reveal

Six files modified. Tested across staff and student views. Deployed. I froze the leaderboard while judges finished scoring. When the final scores were in, the leaderboard was unfrozen in front of the gathered students. The winning team's name appeared in a pulsing gold banner. That moment, the one that made the whole day feel like it had a proper ending, was built in under thirty minutes during the event itself.



Dashboard

Melbourne Connections Trip Overview

Trip in progress!

Students 33

Staff 5

Groups 2

Venues 19

Today's Schedule

No schedule for today

Web-based management backend for the Melbourne Connections staff trip management platform

5:00 PM Production Database Backup and Analysis

With the challenge complete, the AI helped me secure a snapshot of the production database for analysis. The production database runs on a remote Windows server, and direct access over the network mount was blocked by SQLite's file locking. The AI worked through the problem, copying the database locally and then running comprehensive queries against the snapshot.

5:30 PM Challenge Day Review: Written from Data

I asked the AI to write an exciting review of the challenge day, "like it was an amazing sporting battle between the teams." The AI queried the database for every submission timestamp, score, team, category, and landmark. From this raw data, it reconstructed the narrative of the day: which teams started fastest, who went to the furthest landmarks, which teams dominated the culture challenges, who pushed to the final minute. The result was a detailed, engaging, sports-commentary-style review grounded entirely in real data. Saved to my desktop, ready to share with students.

6:00 PM Technical Health Report

I asked for a developer-focused performance analysis. The AI examined submission rates (peak: 6 per minute), file sizes (average photo: 0.62 MB, average video: 1.37 MB), review turnaround distribution (40% reviewed in under 1 minute), rejection analysis (2 out of 83, a 2.4% rate), judge scoring patterns (one judge handled 67% of all scores), database performance (WAL mode, healthy, no contention), and network reliability (zero upload failures). The report translated these metrics into actionable recommendations for the next deployment. It also identified what the data meant for the quality of the student experience: the 2.2-minute average review meant students stayed engaged because feedback was fast; zero upload failures meant no moment of creative courage was lost to a technical glitch.

7:00 PM Per-Team Statistics for Students

I wanted fun team statistics to share with students the next day. The AI ran a detailed analysis of each team: submissions per hour, fastest back-to-back submission (one team managed 11 seconds between uploads), categories completed, landmarks visited, judge score averages, total data uploaded, and estimated walking distance based on which Melbourne landmarks each team tagged. The document included superlatives: "Most Challenges Completed," "Furthest Distance Walked," "Fastest Back-to-Back," "Highest Judge Scores." Ready to read out at breakfast the next morning.

The Count

In a single day, the following was conceived and shipped to production:

- Student guide text and layout refinements
- Staff guide bonus challenge documentation (2 new sections)
- Complete multi-judge scoring engine redesign (4 files, new scoring logic)
- Backup submission plan (challenge list export)
- Live system health check and analysis
- On-the-spot bonus challenge content creation
- Auto-retry scoring fix (frontend error handling)
- "Needs Your Score" review prioritisation (backend + frontend)
- Leaderboard freeze and winner reveal (6 files, database change, 2 CSS animations)
- Challenge day narrative review (data-driven, 2,000+ words)
- Technical health report with recommendations (3,000+ words)
- Per-team statistical analysis with walking distance estimates

Twelve distinct deliverables. Three of them (the scoring fix, the "Needs Your Score" section, and the leaderboard freeze) were built and deployed while students were actively using the system. The students never noticed a thing. The staff barely paused. The system kept running.

This is what becomes possible when the person who understands the experience is the same person shaping the tool, with an AI partner capable of turning intent into working software in real time.

A Note on Process

This document, like the main paper it accompanies, was itself produced through my partnership with AI. The timelines and technical details were reconstructed from the actual development conversation logs. The AI had access to the full history of every feature request, every bug fix, every deployment, and every database query from the day. It helped me identify which moments were most significant, organise them chronologically, and articulate why each one mattered.

Part Three

Technical Deployment Guide

For IT teams: a detailed architecture overview covering Microsoft Entra ID integration, API security design, reverse proxy configuration, and the case for purpose-built systems.

Fieldtripping with AI — Backend Infrastructure for School IT Teams

The People Behind the Infrastructure

The final hurdle of technical deployment cannot be achieved without the assistance of an IT team who understand the philosophy of what IT in schools should be there to serve. They are the roadies who make the educators and experiences like this become true rockstars.

I am very fortunate to lead an incredible IT team who share this vision for learning and are proud to put their technical expertise and desire to keep learning in the service of our students and educators. A special thanks to Tom Robinson, our team's system engineer — he is not just technically brilliant, he possesses the mindset and attitude that school IT teams need in order for students and educators to flourish in these areas.

If you are an IT professional reading this guide, know that your role is not simply to maintain infrastructure. It is to enable the kind of work described in these pages. The architecture matters, the security matters, the uptime matters — but what matters most is the willingness to say yes when an educator walks in with an idea that has never been done before, and to bring your expertise to bear in making it real.

— Brett Moller

---

1. Purpose of This Document

This guide provides the technical detail that a school IT team needs to set up, secure, and maintain the backend infrastructure for the platforms described in *Fieldtripping with AI*. It covers the server environment, authentication architecture, data ingestion from school information systems, deployment procedures, and security posture. It is written for IT administrators who may be deploying these tools in their own school context, or supporting an educator who has built them in partnership with AI.

The guide reflects the infrastructure deployed at Your School, but is written to be adaptable. Where our setup uses specific products (Windows Server, IIS, Fortigate), we note where alternatives exist and how the same principles apply in different environments.

2. The Security Problem This Solves

Before discussing the technical architecture, it is worth understanding the security problem that existed before this infrastructure was built — because it is the same problem that exists in almost every school running camps, excursions, and off-site programs.

The Traditional Approach: PII Everywhere

The traditional method of managing student information on a school trip involves printing staff camp books — bound documents containing student medical records, dietary requirements, allergy alerts, emergency contacts, room allocations, and parent phone numbers. These books are photocopied and handed to every staff member. They travel in backpacks, sit on restaurant tables, get left in hotel rooms, and are carried through busy city streets. When the trip is over, they are supposed to be collected and destroyed. In practice, they end up in desk drawers, car boots, and recycling bins. This is not a theoretical risk: the author has personally experienced a trip where a staff member left the camp booklet — containing the full medical, dietary, and emergency contact details of every student — in a café. The booklet sat on a public table, accessible to anyone, until another staff member happened to notice it was missing.

Alongside the printed books, the same information lives in spreadsheets on shared drives, in email attachments sent to staff before departure, in WhatsApp group chats where dietary requirements are discussed, and in paper notes taken during the trip. Every copy is unencrypted, untracked, and beyond the control of the school's data governance framework.

The core problem: Schools hold some of the most sensitive personal information about children — medical conditions, allergies, mental health notes, family circumstances, emergency contacts — and the traditional approach to excursion management scatters this data across printed documents, spreadsheets, emails, and messaging apps with no encryption, no access control, no audit trail, and no ability to revoke access.

What Purpose-Built Infrastructure Solves

The platforms described in this guide consolidate all sensitive student data into purpose-built applications with the same authentication, access control, and audit logging that the school applies to its core systems. Medical data is never printed. It is never emailed. It is never in a spreadsheet on someone's laptop. It lives in an encrypted, access-controlled application that requires Microsoft Entra ID (Azure AD) authentication to access, that logs every access event, and from which access can be revoked in real time by removing a user from an Azure AD security group.

This is not a marginal improvement over the traditional approach. It is a fundamentally different security posture — one that brings excursion data management into line with the standards schools already apply to their student information systems but have historically failed to extend to off-site programs.

3. Architecture Overview

The infrastructure comprises three distinct components, each with its own deployment model:

Network Topology

All web-facing applications run on a single Windows Server 2022 virtual machine (VMware vSphere) on the school's internal network. The server is not directly internet-accessible. External access is routed through a Fortigate firewall via a port-forward rule to the server's internal IP. All traffic is SSL-encrypted via a certificate bound in IIS.

Backend applications (Node.js processes) run on localhost ports and are never directly exposed to the internet. They are accessible only through the AppPortal reverse proxy, which handles authentication before forwarding any request. This means a backend application does not need to implement its own authentication — it receives requests only after AppPortal has verified the user's identity and group membership.

4. AppPortal: The Authentication & Proxy Layer

AppPortal is the centrepiece of the security architecture. It is a custom-built web application dashboard that provides a single, secure entry point for all school web applications. It was designed in partnership with Claude Code and serves two critical functions: it authenticates every user via the school's existing Microsoft Entra ID tenant, and it acts as a reverse proxy that routes requests to backend applications only after verifying identity and group membership.

4.1 Authentication: Microsoft Entra ID (Azure AD)

AppPortal uses Microsoft's OpenID Connect (OIDC) protocol integrated with the school's Azure AD / Entra ID tenant. This means:

- No separate usernames or passwords — users log in with their existing school Microsoft account
- Multi-factor authentication (MFA) is enforced if the school's Azure AD conditional access policies require it
- Session tokens are managed securely and expire automatically
- Works on any device or browser — staff on school computers, students on personal phones

This is the same authentication layer that protects the school's email, Teams, SharePoint, and every other Microsoft 365 service. By using it for custom-built applications, those applications inherit the same security posture — including conditional access policies, sign-in risk detection, and centralised audit logging — as the school's enterprise systems.

4.2 Two-Layer Access Control

Every application registered in AppPortal has two independent access gates, both of which must pass before a user can reach the backend:

Gate 2 runs on **every request**, not just at login. This means access can be revoked in real time by removing a user from an Azure AD security group — the next request they make will be denied. This is a stronger access control model than many commercial SaaS products, which typically check permissions only at login.

4.3 Reverse Proxy (YARP)

AppPortal uses YARP (Yet Another Reverse Proxy), a Microsoft-built proxy library, to forward authenticated requests to backend applications. Backend apps run on localhost ports and are never directly internet-accessible. URL paths are rewritten so users see clean URLs (`/tools/icons/`) while backend apps receive requests at their own root.

4.4 Role-Based Access

AppPortal uses Microsoft Graph API to resolve the authenticated user's Azure AD group memberships, then matches these against the `allowedGroups` configured for each application. The audience model supports staff, primary students (Years 1–6), secondary students (Years 7–12), all students, all users, and admin-only access. Group membership is resolved from Azure AD in real time using the school's existing group structure — no additional directory or user database is required.

4.5 Configuration

Applications are registered in a JSON configuration file on the server. Adding, removing, or modifying applications requires editing this file — no code changes or redeployment needed. A new proxied application requires an app pool recycle command (`Restart-WebAppPool -Name "AppPortal"`). Changes to name, description, audience, or groups take effect immediately.

4.6 Logging and Audit

AppPortal logs all activity using Serilog to daily rolling log files. Logs capture every login event, every dashboard load (with user identity and number of tiles shown), every application access (with username, path, status code, and response time), and all errors with full exception details. This provides a complete audit trail of who accessed what, when, from which device — the same level of audit logging that schools expect from their student information systems.

5. School Information System Integration

The Staff Companion iOS app is powered by data ingested from the school's existing information systems via their APIs. This section describes how the integration was designed, how security was handled, and how other schools can adapt the approach for their own systems.

5.1 Why API Integration Matters

Every modern school information system — whether it is Synergetic, TASS, Compass, Sentral, SIMON, or another platform — provides APIs (Application Programming Interfaces) that allow authorised software to request data programmatically. Schools already use these APIs to feed data into timetabling systems, learning management systems, finance platforms, and reporting tools. The same APIs can be used to feed data into purpose-built educator tools.

The alternative — manually exporting data to spreadsheets and retyping it into applications — introduces transcription errors, creates stale data (the spreadsheet is out of date the moment it is exported), and produces uncontrolled copies of sensitive information on local machines, USB drives, and email attachments.

5.2 How the Integration Was Designed (with AI)

The API integration was co-designed with Claude Code. The AI helped with:

- **Identifying available endpoints** — reading the API documentation for the school's information system and identifying which endpoints provide medical data, dietary requirements, emergency contacts, and enrolment information
- **Designing the authentication flow** — implementing OAuth 2.0 client credentials flow to authenticate against the API without user interaction
- **Writing the ingestion script** — a Python script that authenticates, requests specific data fields, transforms the response into the format required by the iOS app, encrypts the output, and writes it to the Xcode project directory
- **Designing the security model** — ensuring credentials are stored as environment variables on the developer's local machine, never appear in source code, and never pass through the AI

5.3 Security Architecture for API Credentials

The credential management follows industry best practice:

5.4 Adapting for Other School Information Systems

The approach generalises to any school information system with an API. The key steps for an IT team are:

1. **Identify the API** — check your SIS documentation for REST API endpoints. Most modern systems (Synergetic, TASS, Compass, Sentral) provide these. If your system only provides CSV exports, the same ingestion script can read CSV files instead of calling an API.
2. **Request API credentials** — work with your SIS vendor to obtain API client credentials with read-only access scoped to the minimum required data fields.
3. **Set up the ingestion environment** — a secured machine (the developer's workstation or a dedicated build server) with credentials stored as environment variables. The AI partner can help write the ingestion script for your specific API.
4. **Schedule or trigger ingestion** — run the script before each trip to pull current data. For a week-long trip, running it the day before departure ensures data is current. For longer programs, consider scheduling daily ingestion.
5. **Audit the data flow** — document which fields are extracted, where they are stored, and who has access. This documentation supports compliance with your school's data governance and privacy obligations.

6. Deployment Procedures

6.1 Server Setup (One-Time)

The following describes the initial server setup for a school deploying this infrastructure for the first time:

1. **Provision a Windows Server VM** — Windows Server 2022 on VMware vSphere (or Hyper-V, or any hypervisor). Allocate a minimum of 2 CPU cores, 4 GB RAM, and 100 GB disk. Assign a static internal IP.
2. **Install IIS** — enable the Web Server role with the ASP.NET Core Hosting Bundle (version 8.x or later).
3. **Install Node.js** — LTS version (currently 22.x). Install PM2 globally (`npm install -g pm2`) for process management of backend Node.js applications.
4. **Configure SSL** — purchase an SSL certificate (or use a wildcard certificate if available) and bind it in IIS to port 443. All traffic must be HTTPS.

5. **Configure the firewall** — create a port-forward rule on the school firewall (Fortigate, pfSense, SonicWall, etc.) routing external HTTPS traffic to the server's internal IP on port 443.
6. **Register an Azure AD application** — in the school's Entra ID (Azure AD) portal, register a new application with OpenID Connect. Configure the redirect URI to match the public URL. Grant the application the necessary Microsoft Graph permissions for user identity and group membership resolution. Record the Client ID, Tenant ID, and Client Secret for AppPortal configuration. Consult Microsoft's documentation for the minimum required permission scopes.
7. **Deploy AppPortal** — extract the AppPortal build to `[install directory]\AppPortal\`. Configure the application settings with the Azure AD credentials. Create an IIS application pool named "AppPortal" and an IIS site pointing to the AppPortal directory.
8. **Configure `apps.json`** — register each backend application with its ID, name, description, audience, allowed Azure AD groups, and localhost port. See the companion document *How to Add a New App* for detailed instructions.

6.2 Deploying a Backend Application (Melbourne Icons, Explore Zones)

1. RDP into the application server
2. Copy the application files to a directory under `[install directory]\apps\` (e.g. `[install directory]\apps\[app-name]\`)
3. Install dependencies: `npm install --production`
4. Start the application with PM2: `pm2 start server.js --name [app-name] -- --port [port]`
5. Save the PM2 process list: `pm2 save`
6. Register the application in the configuration file with the assigned localhost port
7. Recycle the AppPortal app pool: `Restart-WebAppPool -Name "AppPortal"`
8. Test: log in to AppPortal with a user account in the configured `allowedGroups` and verify the tile appears and the application loads

6.3 Deploying the Staff Companion iOS App

1. Run the data ingestion script on the secured build machine to pull current student data from the school information system
2. Open the Xcode project on a Mac with the latest Xcode installed
3. Verify the data files in the project are up to date (students, staff, schedule, locations)
4. Build the project (`Cmd+B`) and resolve any warnings
5. Connect each staff iPhone via USB and deploy the app to the device
6. Have each staff member open the app and verify their data access

Note on Apple Developer Program: To deploy to staff devices without the App Store, the school needs an Apple Developer account (A$149/year). Apps are deployed directly via Xcode to registered devices. For schools with a large number of staff devices, consider Apple's Developer Enterprise Program or TestFlight for distribution.

6.4 Deploying Explore Zones (Student GPS Maps)

Explore Zones is a lightweight web application with no student authentication requirement — it does not contain any personally identifiable information. Students access it via a URL or QR code on their phones. The admin interface (where staff create and manage zones) is protected behind AppPortal's authentication layer. Deployment follows the same Node.js/PM2 process described in 6.2.

7. Security Posture

This section summarises the security measures across the entire platform, from network layer to application layer to data handling.

7.1 Network Security

- All traffic is SSL/TLS encrypted (HTTPS enforced)
- Backend applications run on localhost and are never directly internet-accessible
- The only public-facing component is the AppPortal reverse proxy, which authenticates before forwarding any request
- Fortigate firewall rules restrict inbound traffic to HTTPS on port 443 only
- The server has no other public-facing services

7.2 Authentication and Authorisation

- Microsoft Entra ID (Azure AD) provides enterprise-grade authentication via OpenID Connect
- Multi-factor authentication (MFA) is enforced where configured in the school's Azure AD conditional access policies
- Two-layer access control (tile visibility + proxy access) verified on every request
- Access revocable in real time by changing Azure AD group membership
- Session tokens expire automatically
- No separate user database — all identity management through the school's existing Azure AD tenant

7.3 Data Security

- Sensitive student data (medical, dietary, emergency contacts) is never printed, never emailed, never stored in spreadsheets
- Data is ingested from the school information system via authenticated API calls, encrypted, and bundled into the iOS app at build time
- API credentials are stored as environment variables, never in source code or version control
- The iOS app stores data on-device (no cloud sync, no runtime API calls)
- SQLite databases for web applications are stored on-server with no external access
- No student PII is stored in or transmitted through the scavenger hunt or GPS boundary systems

7.4 Audit and Compliance

- Every authentication event, dashboard load, and application access is logged with user identity, timestamp, path, and response code
- Logs are stored on-server in daily rolling files, accessible to IT administrators
- The data flow from school information system to iOS app is documented and auditable
- The platform meets the practical requirements of the Australian Privacy Principles (APPs) as they apply to educational institutions handling student data

7.5 Comparison: Traditional vs. Purpose-Built Security

8. The Role of AI in Designing This Infrastructure

The infrastructure described in this document was co-designed with Claude Code, an AI development tool. It is important for IT teams to understand exactly what the AI did and did not do, and how security was maintained throughout the process.

8.1 What Claude Code Did

- Designed the AppPortal architecture — the authentication flow, the YARP reverse proxy configuration, the two-layer access control model, and the role-based dashboard
- Wrote the ASP.NET Core application code, including the OpenID Connect integration, the Microsoft Graph API calls for group resolution, and the Serilog logging configuration
- Designed the API integration architecture — the credential management model, the ingestion script, the data transformation pipeline, and the encryption layer

- Wrote the backend Node.js applications (Melbourne Icons, Explore Zones) including the database schema, API routes, and frontend interfaces
- Built the Staff Companion iOS app in Xcode — the Swift code, the interface layouts, the offline data model, and the integration with iOS features (phone calls, SMS, Apple Maps)
- Helped write this documentation and the associated IT procedures

8.2 What Claude Code Did Not Do

- At no point did the AI have access to actual API credentials, student data, or any personally identifiable information
- The AI did not access the school's Azure AD tenant, information system, or any production system
- The AI did not make decisions about which data to collect, who should have access, or how the tools should be used pedagogically — these decisions were always made by the educator
- Deployment to the production server was performed manually by the developer — the AI wrote the code, the human deployed it

8.3 Security Testing with AI

Claude Code was also used to review and test the security of the deployed applications. This included:

- Reviewing the authentication flow for common vulnerabilities (session fixation, token leakage, CSRF)
- Testing the access control model by simulating requests from unauthorised users and verifying that Gate 2 correctly denied access
- Reviewing the SQL queries in the scavenger hunt platform for injection vulnerabilities (all queries use parameterised statements)
- Checking file upload handling for path traversal and size-based denial of service
- Reviewing the ingestion script for credential leakage, ensuring no secrets appear in logs, error messages, or stack traces
- Producing a system health report after the live event, verifying database integrity, checking for failed requests, and analysing error logs

9. Alternative Deployment Models

The architecture described above reflects a specific school environment. Other schools may have different infrastructure, and the same applications can be deployed in several alternative configurations.

9.1 Cloud Deployment (Azure / AWS / GCP)

Schools without on-premise Windows servers can deploy the entire stack to a cloud virtual machine. Azure Virtual Machines are a natural fit given the existing Azure AD integration. The VM runs the same IIS + AppPortal + Node.js stack. The main consideration is network latency — for applications used primarily while students are in a specific city, an Australian data centre (Azure Australia East or Southeast) is recommended.

9.2 Linux Server

AppPortal can be deployed on Linux using Kestrel (the built-in ASP.NET Core web server) behind Nginx or Caddy as the reverse proxy, replacing IIS. Node.js and PM2 run natively on Linux. This eliminates the Windows Server licence requirement and may suit schools that prefer Linux infrastructure.

9.3 Container-Based Deployment

For schools with Docker or Kubernetes infrastructure, each component can be containerised. AppPortal, each backend application, and the database can run in separate containers orchestrated by Docker Compose or Kubernetes. This provides deployment reproducibility and simplifies scaling if the platform is used across multiple simultaneous trips.

9.4 Managed PaaS

Azure App Service or AWS Elastic Beanstalk can host the .NET and Node.js applications without managing a virtual machine. Azure App Service has native Entra ID integration, making the authentication setup straightforward. This is the lowest-maintenance option but may have higher running costs than a single VM.

9.5 Google Workspace Schools

Schools using Google Workspace instead of Microsoft 365 can adapt the authentication layer to use Google OAuth 2.0 and Google Cloud Identity for group-based access control.

The AppPortal codebase would need modification to use Google's identity provider instead of Microsoft's, but the architectural principles — SSO, group-based access, reverse proxy, audit logging — remain identical.

10. Ongoing Maintenance

11. Continuous Improvement

Like any production system, the infrastructure should be reviewed regularly for opportunities to improve resilience, automation, and security posture. Areas to consider include: token cache persistence, automated deployment pipelines, uptime monitoring and alerting, Mobile Device Management (MDM) integration for app distribution, and automated database backups. Work with your IT team and, where appropriate, your AI development partner to identify and prioritise enhancements specific to your environment.

Appendix A: Port Allocation

Each backend application is assigned a unique localhost port number. Maintain a port allocation register as part of your server documentation. Recommended practice is to use a contiguous range (e.g. 5000–5090) with each application assigned the next available port.

Appendix B: Administration Commands

Your IT team should maintain a runbook of common administration commands for your specific environment. Key operations to document include: recycling the AppPortal application pool after configuration changes, starting and managing PM2 processes, viewing application logs, checking port usage, and testing SSL certificates. These commands will vary depending on your server operating system and firewall vendor.

Fieldtripping with AI — Technical Deployment & Infrastructure Guide
Technical Reference Document — March 2026
Your School