

From Job Stories to User Stories

How the Level 4 exemplar was built · Sunday Mall Shuttle interview · ICS4U Requirements Discovery

The planner exemplar has three job stories and three user stories. This sheet shows where each one came from, and why the two kinds of story are worded differently.

A **job story** describes the problem, in the moment it happens. A **user story** describes one way to solve it.

	Job story	User story
Describes	The problem, in the moment it happens	One way to solve it
Starts from	A situation (“When... ”)	A person (“As a... ”)
Names a user?	No. Anyone in that situation has the job.	Yes. You decide who.
Mentions a feature?	No. Nothing is designed yet.	Yes. The feature first appears here.
Comes from	The Quote, Need, Impact and Rule columns of the matrix	The job story, plus two decisions: who, and how

The path each story followed

1. Pick one row of the matrix: a real situation that causes a pain.
2. Write the job story from the situation, the need and the outcome. Leave the *Opportunity or Idea* column out of it.
3. Run the quick test: *would this job story still be true with no app at all?*
4. Make two decisions. **Who** is the user? **How** will they complete the job?
5. Write the user story. The ideas you held back in step 2 can come in now.

Story A: recording riders at the bus door

From the notes	“... there’s a huge group of people who are standing there waiting for the bus and have to sign their name and then write their signature...”
Job story	When a crowd of students is getting on the Sunday mall shuttle, I want to quickly record who is boarding and that each one agrees to the \$10 charge, so I can get the bus away on time without missing anyone.
Quick test	Passes. The job is done today with a clipboard and a pen. It is slow, but it exists without an app. No role is named and no feature is named.
Decision: who?	The supervisor on duty at the bus. Ms. Reid: “one of our school block supervisors will actually arrive at the bus on the day, and I think we’d like to keep it so that they arrive on that day.”
Decision: how?	Scan a QR code, then the student accepts the charge on the spot. Ms. Reid raised breakfast check-in herself. This keeps sign-up at the bus, which is a rule from the matrix.
User story	As a supervisor on duty at the bus, I want to scan each student’s QR code as they board and have them accept the \$10 charge on the spot, so that every rider is recorded in a few seconds with no handwriting.
Same job, another user story	As a boarding student, I want a prepaid shuttle pass that is marked off when I get on, so that I do not sign anything at the bus. (The “passport” idea from the interview.)

Story B: getting the charges to the business office

From the notes	<i>“Not everyone writes the same equally . . . A student might have two last names but only write one last name down.”</i>
Job story	When the trip is over and the riders need to be charged, I want to match every rider to the right student account without re-reading or retyping names, so I can pass accurate charges to the business office quickly.
Quick test	Passes. Today someone reads the sheet, types the names into Excel and looks up each account number. The job is real with or without software.
Decision: who?	The shuttle organizer: the staff member who takes the sheet after the trip and prepares it for the business office. This is a different person from the supervisor in Story A.
Decision: how?	Download a charge list built from the boarding records. Names then come from school records, not handwriting.
User story	As a shuttle organizer, I want to download a charge list that shows each rider’s full name, account number and acceptance of the charge, so that the business office can charge accounts without anyone retyping names.
Same job, another user story	As a business office clerk, I want shuttle charges to arrive in the same form as ski-day charges, so that I can post them the same way. (From Student D’s point about ski days.)

Story C: telling the houses who is on the bus

From the notes	<i>“So the list is sent back to the houses, but again, if it’s a hard list to read, that’s a challenge point.”</i>
Job story	When the shuttle has left campus, I want to know which students from each house are on it, so I can compare that with who asked for leave and be ready to check them in when they return.
Quick test	Passes. The paper list is already sent back to the houses each Sunday. The job exists; the list is just hard to read.
Decision: who?	A head of house (or the house staff member on duty). They approve leave and do the face-to-face check-in.
Decision: how?	A rider list that shows only the students from one house.
User story	As a head of house, I want to see today’s rider list showing only students from my house, so that I know exactly who to expect at face-to-face check-in.
Same job, another user story	As a supervisor on duty at the bus, I want each house to be sent its riders when the bus leaves, so that I do not have to deliver a paper list.

A common slip: the user story in disguise

“When students are getting on the bus, I want to scan their QR codes, so I can record them quickly.”
This starts with “When”, so it looks like a job story. It fails the quick test. Take the app away and there is nothing to scan, so the sentence is no longer true. Scanning is a solution, and solutions belong in the user story.

The fix: replace the feature with the need behind it. “Scan their QR codes” becomes “record who is boarding”.

Why keep the two apart?

In the interview, Student A asked whether the business office would accept an e-signature. Ms. Reid answered: “I’m going to have to get back to you about that.”

Suppose the answer is no. **User story A** has to change, because accepting the charge on a screen is no longer allowed. **Job story A** does not change by one word: a crowd still needs to be recorded quickly at the bus door.

The same thing happened live in the interview. When the passport idea came up, Ms. Reid said: “Yes, we’d still need attendance.” The solution changed, and the job stayed.

That is the reason to write the job story first. It records what you learned from the stakeholder, and it stays true. The user story records a design decision, and you can change it. One job story can lead to several different user stories, as the last row of each table shows.

Checking the exemplar against the rubric

The tables below use the look-fors from the feedback sheet. Each one is matched to the place in the exemplar that earns “Always”.

Note-Taking Matrix	Where the exemplar shows it
Quotes are relevant to the overall challenge presented	All four quotes are Ms. Reid’s own words about the shuttle sign-up: the line-up, the handwriting, the list for the houses, and the Google form that was tried before.
Needs or pains accurately summarized	Each need restates its quote in plain words. Nothing is added that Ms. Reid did not say.
Frequency (how often) is accurate	“Every Sunday, all year” comes straight from the interview. Row 4 says “whenever advance sign-up was tried”, because that pain is not weekly.
Impacts are reasonable and clear	Each impact says who is affected and how: students wait, staff retype names, house staff cannot tell who is away.
Constraints are accurate	Each one is a rule Ms. Reid stated: sign-up at the bus, a signature for the \$10 charge, leave requests and face-to-face check-in, limited seats.
Opportunities or ideas are meaningful and connected to other items in same row	Each idea answers the pain in its own row, and stays loose. In row 2 the pain is handwriting, and the idea is to take names from school records.

Job Stories	Where the exemplar shows it
Situation is clearly defined	Each “When” is a moment you could point to on a Sunday: students boarding, the trip being over, the bus having left campus.
Motivation – what the user is trying to get done – is plainly identified	A: record who is boarding and that they agree to the charge. B: match riders to accounts. C: know which students from each house are on the bus.
Outcome – what changes – is defined	A: the bus leaves on time with no one missed. B: accurate charges reach the business office quickly. C: house staff are ready for check-in.
Job stories are not tied to a particular user	No role and no feature appears in any of them. Each one passes the quick test.

User Stories	Where the exemplar shows it
User (who) is plainly identified	Supervisor on duty at the bus, shuttle organizer, head of house. These are three different people, and each one was named in the interview.
Task (what) is understandable and tied to the user's role	Each task happens where that person already works. The supervisor is at the bus door, the organizer prepares the charges, and the head of house does the check-in.
What gets done (why) is reasonable & clear	Each “so that” points back to a pain in the matrix: no handwriting, no retyping, knowing who to expect.
Acceptance Criteria	Where the exemplar shows it
At least one happy path acceptance criteria provided for each scenario (A, B, and C)	A, B and C each have a happy path first, then one edge case.
Context or pre-condition stated plainly	For example: “Given boarding is open and Student X is not on the rider list”.
Action is specific	Each “when” is a single action: X’s code is scanned and X accepts the charge; I download the charge list; I open today’s rider list.
Observable outcome is specific and measurable with clear pass/fail conditions	“The rider count goes up by one.” “Exactly 23 riders.” “Those four names and no students from other houses.” Anyone watching could say pass or fail.
Avoids spelling out how functionality is implemented in UI or coding terms	The criteria say what the user sees. They do not mention buttons, screens, files or database tables.
Hits the right notes for audience and purpose	They use the shuttle’s own details: the \$10 charge, houses, face-to-face check-in. They could not be pasted into another project.