User Story Template

<Subject/User> <Action> <Outcome/Reason>

Comparing User Stories

Below are examples of progressively better user stories due to addition of relevant business context:

	User Story	Developer Comment
A	Restaurant Search feature	No context provided at all
B	**As a user**, I can search for a restaurant.	Identified "user" business context
C	As a user, I can search for a restaurant **close to me**.	Identified "close to me" business rule
D	As a user, I can search for a restaurant close to **my current location**.	Clarified what "close" means, but still ambiguous
E	As a user, I can search for a restaurant **within a 5-mile radius**.	Quantified "close" to a business rule
F	As a user, I can search for **any open** restaurants within a 5-mile radius.	Identified "open" as a filter rule
G	As a user, I can search for any open restaurant within a 5-mile radius **to select from**.	Identified purpose of search action
H	As **an unauthorized** user, I can search for any open restaurant within a 5-mile radius to select from.	Clarified that system has a security model and/or different types of users

Below are examples of questionable user stories that provide too much implementation/technical details. User stories should only focus on the business context; leave the technical details to the developer as they are the technical expert.

	User Story	Developer Comment
A	As a user, I can search for any open restaurant within a 5-mile radius using an **autocomplete field**.	Is an "autocomplete field" required or a nice to have? Search results may take a long time to appear when data is unfiltered/large.
B	As a user, I can search for any open restaurant within a 5-mile radius **using a slider** to filter by 0 to 5-mile radius.	Is a "slider" UI required? Can a radio button suffice? • <= 1 miles • <= 2 miles • <= 5 miles
C	As a user, I can search for any open restaurant within a 5-mile radius **with search results displayed in a table**.	A "table" UI will cause large performance issues when a search results data is large. Even with pagination (btw: difficult to implement with tables) will inflict needless slowdowns. Will a card view or a list view suffice?
D	As a user, I can search for any open restaurant within a **5.00000000**-mile radius.	Requires 8-digit precision. Need to store coordinates as double data type, which requires more memory and computation time needed to versus a float data type. All these penalties just to achieve location accuracy to within centimeters, but the average human only needs within 10 meters accuracy to find a location.

A User Story is good for identifying a scenario that you want to implement. It defines the user, the action, and the outcome of said action. However, it alone is not enough for a developer to implement. A COMPLETE requirement will also need to define the feature's non-functional requirements (NFR) and the scenario's acceptance criteria & failure behaviors. NFRs defines the aspects that will make the feature high quality, while the acceptance criteria & failure behaviors provide details into how the feature can be tested.

Detailed Restaurant Search Example

As **an authenticated non-admin user**, I can search any restaurant that **meets all my filter criteria** so that I can select from the result.

- Acceptance Criteria for User Story:
    1. Search results are loaded within 5 seconds of search invocation.
    2. Results are sorted in ascending order of distance from user's current location.
    3. When user selects a single result entry, the user will be **displayed** additional info about the restaurant
    4. Available Filters are:
        - Restaurant status
            - Status options
                - Open
                - Closed
        - Distance
            - Distance options
                - <= 1 mile
                - <- 5 miles
    5. Filters can be toggled on/off on demand, thus the order of operation on search result is Restaurant status, then distance

Justifications

Blue: User stories should always distinguish which feature requires signing in to use versus can be used by anyone. For example, to watch a video on YouTube, you do not need to sign-in, but to upload a video you do need to sign-in.

Yellow: I am explicitly pointing out to the developer that there is more than one filter option. The order of precedence of filter options needs to be defined in the Acceptance Criteria. Defining all of them in the user story itself would make the sentence very cumbersome to read and understand. Acceptance Criteria entry 4 and 5 provides this information.

Acceptance Criteria Breakdown:

- Entry 1 is a non-functional requirement. "5 seconds" is an arbitrary value defined by the business side as a business rule. Without Entry 1 implemented, the search feature would still work, but it would be sluggish to the user leading to abandonment, thus it improves the robustness of the search feature.
- Entry 2 is another NFR. Random results will not only make it difficult to test, but a lack of consistency will make the user think that the search feature is volatile and unstable. Again, the search feature will work without Entry 2 being implemented, but at the cost of user trust in the system.
- Entry 3's purpose is to define the behavior of the main goal of the search feature--to ultimately pick a restaurant. Notice the key choice of words used. "Displayed" does not dictate how the developer should implement showing the user the "additional info". It just informs the user that they need to provide content to the user, but it's up to the developer to decide if that content is presented a modal window, a pop-up, a completely new page, etc.
- Entry 4 is solely specifying business rules and for improving feature validation during testing.
- Entry 5 is required and not deemed as providing technical/implementation details due to the search feature having on demand filter toggling. To properly test on demand toggling, the order of operation must be deterministic

Acceptance Criteria

One missing piece of the "Restaurant Search" requirements is how the system should behave when errors occur. These are typically referred to as failure scenarios and often correlate to an entry in the acceptance criteria.

Define how the system should behave when each criteria fail:

Failure Behavior for restaurant search:

1. If Acceptance criteria entry 1, took longer than 5 seconds
2. If Acceptance criteria entry 2, were not sorted or sorted in descending order instead of ascending
3. If Acceptance criteria entry 3, did not display the additional info about the restaurant selected or displayed the wrong info (i.e., another restaurant's info) instead
4. If Acceptance criteria entry 4, had missing filters
5. If Acceptance criteria entry 5, the order of operation is not respected

All these details combined will make a fully flushed out set of requirements for a feature.

Example User Story Template

Feature	
User Story	
Effort Points	
Data Source	**Origin=(Internal or External)** **Type=(Web API or data store)** **Details=** **AuthN=**
App Permissions	
Target Audience	
Pre-conditions	
Success Outcome(s)	
Failure Outcome(s)	

Feature	Restaurant Search
User Story	As an unauthenticated user, I can search for an opened restaurant that is within 5 miles
Effort Points	15
Data Source	**Origin=**External **Type=**API **Details=**https://api.yelp.com/restaurants **AuthN=**YelpAccount4AppUser
App Permissions	Permission=Search Scope=Restaurant
Target Audience	Any end-user
Pre-conditions	1. User is using an IP address within U.S. to connect to app 2. User is currently on the Restaurant Search View
Success Outcome(s)	• Results are displayed in order of distance to user's current location within 5 seconds
Failure Outcome(s)	• User did not see any results • Results took longer than 5 seconds to display • Results are displayed out of order