Functional Specification: CityJogger App

Overview

This is a functional specification document for a software product, written the Ordinary Objects way. It begins with a very brief overview of the product's business domain and motivations.

In this instance, we're building an early version of a made-up app called **CityJogger** for use by joggers and run commuters in London. This will be valuable to Londoners increasingly invested in their health and increasingly frustrated with traffic and pollution.

Functionality

Then we dive right into it. Software engineers won't benefit from reading 10 pages of preamble, and this document is for them. If needed, place some definitions upfront to align meanings, so the development team can refer back:

Definitions

- *Jogger Cadence*: The number of footsteps detected per minute.
- *Swerve*: A non-180° direction change.
- *Run Pause*: An absence of footstep input for two seconds or more.
- *Weaving*: A jogger behaviour pattern defined by at least four swerves in a minute, or two swerves and two pauses in a minute.

Basic Details

The Functionality section is the heart of the functional specification. It lays out the product's desired business logic. We like bulletpoints, but don't worry too much about layout. Start with describing the basics of the desired product or feature, and the detail will come. If you miss something, presenting work in this way makes it likelier that your software engineers will catch omissions. This living document should evolve to cover more cases and detail over time.

The Functionality section should be written in *rigorous*, *conditional*, *plain* language, to prevent confusion or alternate readings from sneaking in. Mockups and sensible graphs (not flow diagram spaghetti!) are never a bad thing.

- ***Rigorous***: Don't say 'greater than' if you mean 'greater than or equal.' Use words like 'must,' 'should,' and 'could' deliberately, not accidentally. Qualify your ifs: Don't use 'if' when you mean 'only in this one case'—use 'only in this case,' or 'only if'.
- ***Conditional***: Construct conditional sentences whenever possible: **the product will display behaviour ABC *when* this condition occurs.** These translate more directly into unit tests and functional test cases.
- ***Plain***: Don't be stuffy or formal or use jargon that isn't known to your team without defining it in the same document. No one wants to read stuffy documentation.

Here's an example feature description in a Functionality section:

(1) CityJogger RoutePicker

- Once the CityJogger app's splash screen fades, the RoutePicker screen is displayed.
- The RoutePicker feature prompts the user for an Origin, Destination, and desired jogging speed in minutes/mile.
    - (**In future phases**, RoutePicker will allow choices in minutes/km and miles/hour).

- The ‘Pick Route!’ button submits Origin, Destination, and jogging speed selections.
- The button should be large and easy to tap as it may be used while running.
- When the user taps ‘Pick Route!’, RoutePicker displays an ordered list of jogging routes with icons.
    - The Origin, Destination, and jogging speed selector should still be accessible at the top of the screen.
- The route icons are as follows (these bold text descriptions should be displayed when the respective icons are tapped):
    - Heart Icon: **this route is health friendly due to a lower than average pollution index**
        - The Heart Icon is displayed if the pollution index at sampled postcodes at 0.5 mile intervals along the route is, on average, lower than the citywide average for that day at that time.
    - Zoom Icon: **this route is likely to be faster than other routes**
        - The Zoom Icon is displayed if the net elevation along the route is less than or equal to 10 meters (indicating the route is flat) and there are no walking route obstructions retrieved via the CityRoute API for that day at that time.
- RoutePicker results are ordered by distance weighted by a RoutePick multiplier (software engineers will define this!). The multiplier up-weights any route carrying a Heart Icon or a Zoom Icon. Routes with both Heart and Zoom should be strongly weighted to display at the top of the result list.
- During a Run Pause, the RoutePicker results should be re-calculated in expectation of the user attempting to re-plan their commute.

Solutionising

Definitions

- *Solutionising:* Prescribing features in a functional specification or a requirements document at too fine a level of detail.

  You should avoid solutionising. And not because many developers dislike it.

Solutionising is dangerous because it makes you accountable for a level of detail you might not understand as well as someone else on your team. And because it's written in a specification, that detail may bypass the built-in checks and balances development and test teams have on the quality of a technical solution.

If you are authoring this document as a technical team member, write the algorithmic detail or formula in a separate technical specification hashed out with your team, not in a functional specification.

If you are a Product Owner, only go to the level of detail you understand more thoroughly than your team. Usually this is a mix of business-level detail and functional precision. Not database architecture. Not a massive RoutePick ordering formula using logarithms and square roots. Let the experts come up with the technical logic. Weigh in only on the touchpoints where a proposed solution makes a functional difference for users. You will end up with a better product and a more bought-in team.

(2) CityJogger Bop

- CityJogger Bop is a feature that allows a musical selection to be customised to Jogging Cadence and other jogging behaviour patterns.

- To set up Bop, a user must connect a music library. The Bop onboarding screen must display the selection of the device's installed music apps that CityJogger is able link to.
- Once Bop is linked to a music library, a modal displaying 'Ready to Bop!' appears at the top of the screen and disappears after 3 seconds.
- Bop should work automatically in the background during a run. A Bop icon appears on the activity screen to indicate Bop is working.
- Clicking the Bop icon toggles CityJogger Bop off and on during a run.
- When Bop is on, Beat Analysis should occur (software engineers will define this!).
- Via Beat Analysis, the cued musical selection should match Jogging Cadence.
    - (**In future phases**, Bop will be able to switch songs if the current beat is too far out of sync with Jogging Cadence for too long.)
- Swerves and Weaving patterns should ideally be used in Beat Analysis too.
- Run Pauses should pause the current song.

Other Detail

Once the general Functionality section is done, you may wish to break out a series of sections for specific areas of business logic. In the CityJogger example, this could be something like Music Licensing.

These separate sections look and feel like the Functionality section above.

Why break them out? Maybe there's a need for a deeper level of detail or a series of visual examples. Maybe you want to indicate these will be a different Agile backlog Epic / Story. Or maybe they belong to a future Sprint or Release. Sometimes a separate section comes out of a corner case or logical bug encountered while building core functionality—remember this is a living document.

This Document in the Software Development Cycle

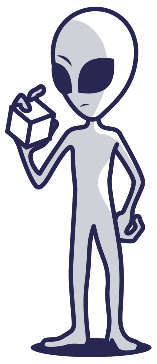
In an Agile framework, this document should be used as the source of your User Stories. It should be taken to refinement sessions and challenged and iterated as the Stories related to it evolve.

If this document derives from a business requirements document, it is a good idea to call out the source business requirement, if any, next to each listed bit of functionality.

No matter the methodology, this document should—to the extent possible—be a living specification for a feature. Keep it up-to-date as a product changes.

Did You Like This?

Let us know! hello@ordinaryobjects.net



Document v1.0

Released February 2017