Automating Checking - Phase 1 - Test Planning and Execution

Test Planning and Strategy

- have already done some exploratory testing

More Planning

- external influences
  - can't control everything
- plans are dynamic
  - stuff changes
- content and structure
  - what - scope and objective setting
    - what is in scope
    - what is not in scope?
    - useful for...
      - Testers, who are executing the plan, to know without ambiguity which areas are "off charter" and they shouldn't need to look into.
      - Stakeholders, test leads, or anyone else reviewing the plan or its results, to know which areas will not or have not been included. In this case, you're managing expectations to avoid ending up in the position where the stakeholder/s expected something to be tested, which was not tested.
  - what identifying what is testable
    - White box testing: as we've seen previously, this is where you have access to the codebase and can plan your tests based on what you find there.
    - Change-related testing will be explained in detail later on but, in short, it involves writing tests to ensure that a given bug has been removed and, then, doesn't get re-introduced by any subsequent changes.
    - Non-functional testing was also covered a previous module (Intro to Testing). These tests relate to aspects of the application that extend beyond its basic feature set. For example, you may wish to test the security and performance of an application. You'll see much more on non-functional testing, and apply some non-functional testing techniques, in a later module.

    - what - ensuring traceability
        - map your tests to the info you have been given - user stories? or feature spec? whatever it is make it link back to that
        - make this mapping clear and readable for others
    - when prioritization and scheduling
        - order things by what needs to be donw first
        - business need
        - test leads/managers give direction
        - resources as well, people, time etc
        - take into account risk
        -
    - How - execution planning
        - how are you actually going to do the tests
        - metrics to show monitoring
        - exit criteria?
    - How - formatting
        - how much detail?

Test plan formats

Formatting options

- Spreadsheets
    - tables and good functions already built in
- mindmaps
    - visual rep - visual grouping
    - or maybe communicate how a test becomes increasing in depth
- structured text docs

Test Case Formatting

Each test plan will include a number of 'test cases', which describe each individual test to be carried out. These can be formatted in many different ways and your company or team might dictate what you do. We recommend that you start by using this really common format.

- Steps list what actions a tester should take in order to execute the test case

- Expected Result detailing what the designer of the test case would expect the outcome to be, were it to be considered a "pass"
- Actual Result for recording what the tester actually got back or observed when running through the Steps

Maybe add in here a brief description
maybe a risk explanation and a prioritization as well

Executing Test Plans

Timeboxing

- estimate the time needed
- work from experience
- set that time and stick to it

Notes

- make notes as you go
- what is being done
- by whom
- findings of any sort?
- issues?
- anything else

As you progress with the project:

- update your notes/ provide updated notes
- try to determine how close you are to 'exit criteria'
    - are you on track for completion?
    - if not - what is the contingency plan?
- Provide a summary/picture of the project under test

Story of product and story of testing

Focusing and Defocusing

The first technique, focusing, is useful in situations where the tester has too many choices, inputs, information, etc. for example when trying to work out what steps are causing a particular behaviour, or for trying to understand a single area within an application. By removing variables, simplifying the environment, reducing the scope, testing can become more focused.

The second technique, defocusing, is the opposite and is useful in situations where the tester wants to break out of a pattern of behaviour or thought process, potentially to uncover new test ideas and defects. When you want the testing to defocus, you might try to broaden the scope, introduce new ideas and possibilities, or come up with different things to try.

Off charter

- same as 'thinking outside the box' this case it's a map i guess
- This should be a conscious move - not just a tangent
- if you do plan to do this:
    - Add a note about what you were doing when you paused your planned testing and deviated. Include what would've been next and what was on your mind at the time.
    - Set a new timebox - how long will you go off charter for, before definitely returning?

Test Reports

- Progress report - when part way
- Summary report - when you are done
- These reports help for *monitoring* and *controlling* work
- Likely to include:
    - In **progress reports**:
      - Impediments/blockers which may need other people within the organisation to remove (e.g. lack of access to environments on which to test)
        - What's next in the plan
        - Revised plans
- In **status reports**:
    - Additional details on what was done e.g. off-charter/out of scope work
    - Problems encountered during testing that should be reviewed for future activities

Detail, format, and formatity will be context dependant

Accuracy and Clarity

- important
- Short clear summaries => quick to read about the bug and understand some details
- Observations vs Expectations => what the tester found that differed, why they think it's important
- Reproduction steps => someone can easily repeat the steps to debug, fix, and retest

Consider language used always.

Exercise:

Consider these 3 options (as a stakeholder) which would you prefer and why?

> 1. Testing is finding loads of big issues at the moment and the team are making slow progress - I think they're starting to tire and miss things like the minor installation bug I accidentally encountered earlier that I told you about in our meeting. We'll carry on tomorrow though and see what happens.
>   1. very informal, this is more like a casual slack message to a colleague, not an update - would expect an update in person following this
> 2. We're half way through the planned tests, starting with installation which was deemed to have the next highest risk. All observed issues have already been added to the bug tracker for review. The team believes at least one could stop the ship but it needs review - it's an XSS attack from our automated ZAP run. At the current rate, we're unlikely to reach the end; we'll continue with the remaining tests as originally prioritised, unless requested otherwise.
>   1. This one is clear, redirects to where more info can be added - is formal enough that you are comfortable they are logging everything, brief extra detail on the main issue and that this will effect timing - given indication of what is happening with that and gave ongoing plan
>   2. pretty good overall
>   3. what is xss and zap who knows
> 3. Day two of three is now all wrapped up and the testers have had a busy day. In the morning, the team were busy working through the various installation tests that we'd discussed and agreed with you beforehand - essentially covering both Linux (Red Hat and CentOS) and Windows across all our company's supported versions, as we would usually do. It ended up being a very, very interesting session with a lot of discoveries! The afternoon was just as busy, but while we've been working hard we've got 48% of the tests remaining to finish off tomorrow - we'll do that and let you know when we've completed it by sending over a final report.

> 1. you are going to do 48% of the work in 33% of the time?
> 2. emotive language is not helpful - hopefully we are friends so this makes more sense
> 3. longer than 2 but less info given

Ilities:

- 1. Usability.
- 2. Reliability.
- 3. Availability.
- 4. Portability.
- 5. Testability.
- 6. Scalability.
- 7. Flexibility.
- 8. Reusability.
- 9. Maintainability
- 10. Supportability
- 11. Interoperability
- 12. Performance
- 13. Security

Exercise 1: Evaluating a test plan

Scenario

Your team has been tasked with helping to determine whether the site is ready to be launched in the next few days, and somebody in your team has produced this test plan. They have asked the rest of your team to evaluate whether the plan is suitable.

- existing test plan
  - https://github.com/makersacademy/automating-checking/blob/main/phase1/05_resources/todomvc_test_plan.md
- 'few days' till launch
- is the plan up the the task?

Step 1 (challenge)

Within your group, spend a maximum of 1 hour writing some notes about the quality of the test plan.

You'll be referring to these notes in future exercises, so make sure you can understand them!

- What do you like about the plan?
  - feature list and some good sections
- Does it feel like anything important is missing from the plan?
  - test cases
  - realism
- Does any of the testing feel unnecessary?
  - not all the browsers? def not IE
  - not sure about the version bleed
  - not sure about the Json part in local storage?
  - Mobile...? ha
  -
- Does any of the testing seem unrealistic? Remember, your team wants to launch this site in the next few days!
  - see above
  - staff
  - time
  - resource
  - all unrealistic
- If you were the tester who was asked to execute this plan, would you be able to follow it? Where might you struggle?
  - some areas unclear
  - lots of unrealistic stuff

There aren't any right or wrong answers. Different people will have differing opinions, influenced by their personal experience. This activity is all about debating diverse ideas and using them to drive future improvements.

If there is no debate in your group, nominate someone to play devil's advocate.

take a break then move to next exercise

--

Read through notes:

- do i care about the todoMVC test plan?
- objective and scope seem clear
- test environment is being rebuilt?
- testing on internet explorer...? why...?
- test team - have 4/7 not available due to other products being released in the coming MONTHS - not few days - move them.
- Adams - lead - experienced
- Barker - prob the most useful - primary tester on this whole thing
- Chapel - noob - good for new looks tho
- Features to be tested - lots of stuff most of which seems fair enough, multi browser test will be slow
    - Has this stuff already been made? if not then with only a couple days and basically just 2 testers they are in trouble - MAYBE if everything works fine it will be ok - but if there are any issues at all it will end up going live with problems (since we cannot move the launch)
    - not sure what the diff verisons react vs vue.js are
    - no idea how to test the last part the inficidual todo items UUID - `b3a592f9-fbfb-6461-4eef-fba1274b5868`
- Features not to be tested
    - ok fair dont test links
    - keyboard navigation not yet implemented...? seems odd
    - why is there a question about mobile devices in there - this should be a statement instead - if it does need mobile that adds a whole new thing
- test estimation claiming 216 hours
    - claiming divided between 3 testers this is 72 working hours - so two weeks solid
    - 1 of the testers is a noob so not gonna happen
    - this is absolute so not taking into account issues sickness problems meetings anything
    - and it needs to launch in 'a few days'
    - Impossible
    - taking into account bug writing? depends on how many
- Defect management
    - the prioritization list is awful

    - high immediate, medium fixed today (even faster than high prob), low - asap - they're all the same and either too specific or vague
    - test team cannot be responsible for fixing the bugs
    - it is not possible to 'find them all'
- no more bugs remaining is a joke
- entry criteria section is odd - dunno what some of it means - it is clearly incomplete, surely they need to start now
- exit - date and or no bugs -
    - no bugs is not possible
    - the date being a few days is really soon
- risks
    - internet is an issue...
    - test team will work overtime until tests are finished - no thanks
    - performance testing after launch is very risky

Extra -

- no test cases
- no starter plan like first 90 mins
- no prioritisation

> Automating Checking - Phase 1 - TodoMVC Test Plan
>
> ## Objective
>
> This test plan intends to outline the testing which will be performed to confirm that the TodoMVC website is deployable before it is given to customers later this week.
>
> We have been provided with a feature list to be tested, which has been modified to make it as realistic as possible given the time constraints.
>
> ## Overview
>
> TodoMVC is a to-do list application doesn't have any persistent storage; it stores its data in the user's local storage. Consequently, the application is not designed to be a real-world to-do list manager, as there's no way to share the data between sessions, browsers or machines.
>
> ## Scope

In scope

- we will be testing the feature list across the frameworks listed on TodoMVC
- The links on the page
- The tab system that filters JavaScript, Compile-toJS and Labs
- Mobile browsing experience
- General performance

Out of scope

- Security
- Large scale performance (such as high level stress testing)
- Sidebar links (Official Resources, Community, etc) for each template
- Keyboard navigation - not yet implemented

Team

We have 7 testers in our test team, however 4 of them are currently allocated to other projects which are due for release in the coming months. This leaves us with 3 team members for TodoMVC testing.

The testers assigned to this project are:

- `A.Adams` - Test Team Lead, the most experienced member of the team.
- `B.Barker` - Test Engineer who's been the primary tester on TodoMVC for the entire project.
- `C.Chapel` - Test Engineer who joined the company last week and is primarily still completing onboarding.

Targets for team

- Write automated testing for the chosen starter language (JavaScript, React)
- apply these tests to the other frameworks
- to be completed alongside exploratory testing
- submission of all test reports to Product owner stand-in

Day 1/4

- Write the automated tests for JS React in Chrome (Adams and Barker)
- Exploratory tests - free roam and then structured to the feature list (Chapel)

- Run these tests and make sure they are running correctly (ALL)

Day 2/4

- clear up any left over checks from day 1 tests
- modify/create the same tests for the other frameworks and browsers available (ALL staff)
  - Browsers - Chrome, Safari, Firefox, Edge
  - Frameworks - those listed on the MVC website - starting with JavaScript tab, then moving to the Compile-to-JS, then Labs as time allows
- Run all tests on all variants and list any areas of note
  - failed tests
  - bugs
  - etc
- Provide all gathered information and bug reports to the other team members - Product owner stand in, and devs
  - This then allows the Product owner stand in to decide what to do with regards to instructing the devs and the deployment
  Day 3
- Review tests (ALL)
- review any that came back with errors
- creation of documentation
- Chapel - to do exploratory testing with Mobile and write summary reports of experience

Day 4

- Any essential catchup(Ideally this is not the case)
- Test sign off report and submission (ALL)

Features to be tested

- Add a new ToDo item
  - Can't add an item with an empty value
  - Can add a value with a single character
  - Check that every single accented character and symbol is supported
  - Check that every single emoji character is supported

- Modify a ToDo item (by double-clicking)
- If you modify a ToDo item and click Escape during edit, it should cancel the modification
- A ToDo item can be ticked-off (it will appear with a line through it)
- A completed ToDo item can be unticked again
- A ToDo item can be reordered by dragging it up or down in the list
- Delete an incomplete ToDo item
- Delete a completed ToDo item
- Status bar displays "0 items left" when there are no items left
- Status bar displays "1 item left" when there is 1 item left
- Status bar displays "2 items left" when there are 2 items left
- Status bar displays "3 items left" when there are 3 items left
- Status bar is hidden when there are no ToDo items in the system
- ToDo items have a 248 Character limit
- When there are any completed items, a "Clear completed" link appears in the status bar
- When the "Clear completed" link is clicked, all completed items are deleted

- Status bar is hidden when there are no ToDo items in the system

- Clicking the down arrow symbol next to the "What needs to be done?" box will toggle all items to Completed or Not Completed

- ToDo items from one variant do not "bleed over" into other applications (e.g if you create a ToDo item in the React version, it is *not* visible in the Vue.js version)

- clicking on the links on the pages take you where expected

Assumptions

- The deadline for this project is 'the end of this week' which we are assuming to be 4 days of working time
- Testers will run and give severity and priority ratings to bugs found, but the decisions around development and deploying with or without fixes is not down to the testers
- We are assuming that with the Product owner being on holiday that there is someone in post other than the testers who are available to make decisions. They are fully up to speed and they are invested in the product

Defect management

Bugs should be reported with:

- a brief description
- steps to reproduce
- expectation vs actual
- priority/severity

High - Requires immediate attention
Medium - Ideally to be fixed before deployment (would be nice to be fixed by deployment)
Low - could be fixed after deployment

Exit criteria

Testing will come to an end when the testers have completed the planned activities above, it will then be over to the PO stand-in to determine launch

---

First 60 minute exploratory testing guide:

https://todomvc.com/

Things to look for:

- The features and general feel as listed below
- View and formatting
- general performance

Initial view:

- [ ] Click through the site, providing narrative about how you think it works, what you think its aim is and what you think you should click to view the website

---

Features to test

- [ ] Add a new ToDo item
- [ ] Can't add an item with an empty value
- [ ] Can add a value with a single character
- [ ] Check that every single accented character and symbol is supported
- [ ] Check that every single emoji character is supported
- [ ] Modify a ToDo item (by double-clicking)
- [ ] if you modify a ToDo item and click Escape during edit, it should cancel the modification
- [ ] A ToDo item can be ticked-off (it will appear with a line through it)
- [ ] A completed ToDo item can be unticked again
- [x] ~~A ToDo item can be reordered by dragging it up or down in the list~~
- [ ] Delete an incomplete ToDo item
- [ ] Delete a completed ToDo item
- [ ] Status bar displays "0 items left" when there are no items left
- [ ] Status bar displays "1 item left" when there is 1 item left

- [ ] Status bar displays "2 items left" when there are 2 items left
- [ ] Status bar displays "3 items left" when there are 3 items left
- [ ] Status bar is hidden when there are no ToDo items in the system
- [ ] Status bar can toggle between Active, All and Completed
- [x] ~~ToDo items have a **248-character limit~~
- [ ] When there are any completed items, a "Clear completed" link appears in the status bar
- [ ] When the "Clear completed" link is clicked, all completed items are deleted
- [ ] When clicking on All in the status bar it shows all
- [ ] when clicking on active it shows all incompleted
- [ ] when clicking on completed it shows all completed
- [ ] Clicking the down arrow symbol next to the "What needs to be done?" box will toggle all items to Completed or Not Completed
- [ ] ToDo items from one variant do not "bleed over" into other applications (e.g if you create a ToDo item in the React version, it is *not* visible in the Vue.js version)
- [ ] clicking on the links on the front page take you where you would reasonably expect

Steps taken:

- discussion around the the site in general
- used 'mythril' (in JavaScript) as an example framework to run through the feature list (firefox)
- briefly looked at mobile version on firefox and safari
- Ran through the feature list on chrome with Elm (in Compile-to-JS)
- wrote up bugs and progress report for the exploratory testing

Bugs

Description

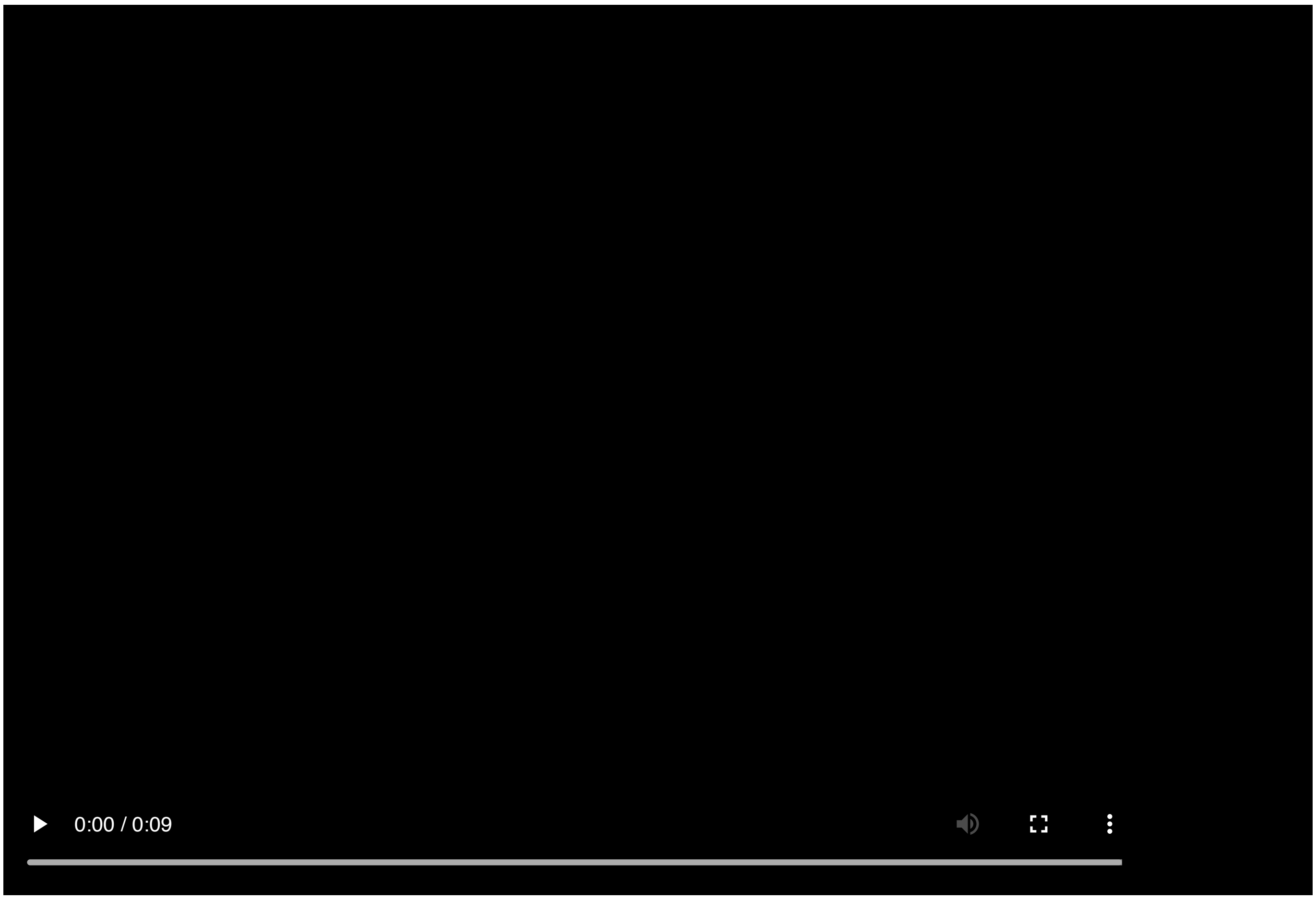
Unable to 'reorder by dragging it up or down in the list'.

We were unable to re-order items

Steps to reproduce

1. In this case we were using Firefox and 'Mythril'
2. entered in the to do items as shown in the video

3. tried 'dragging to move' but was unable to re-order the items



Risk/priority

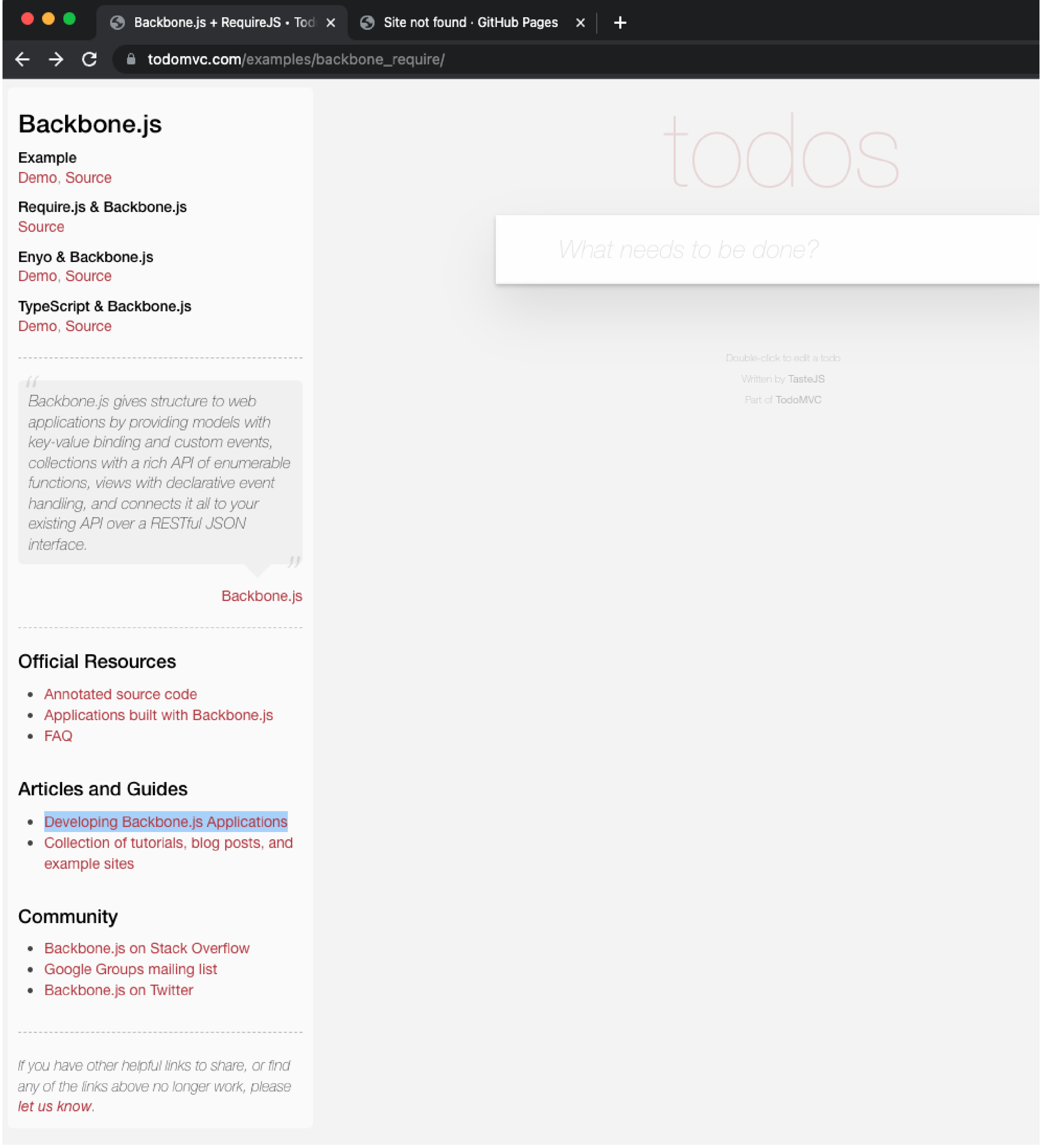
High - This is due to it being one of the specified features of the testing list - that is not working, as far as our exploratory test showed.
Would need time to retest before deployment

Description

Broken link is leading to a 404 page on the Backbone.js page

Steps to reproduce

1. go to start page
2. click on 'Backbone.js'
3. in the 'articles and guides' section on the left - click on 'Developing Backbone.js Applications'
4. This leads to a 404 page


Backbone.js + RequireJS • Tod
Site not found · GitHub Pages
todomvc.com/examples/backbone_require/

Backbone.js

Example
Demo, Source

Require.js & Backbone.js
Source

Enyo & Backbone.js
Demo, Source

TypeScript & Backbone.js
Demo, Source

Backbone.js gives structure to web applications by providing models with key-value binding and custom events, collections with a rich API of enumerable functions, views with declarative event handling, and connects it all to your existing API over a RESTful JSON interface.

Backbone.js

Official Resources

- Annotated source code
- Applications built with Backbone.js
- FAQ

Articles and Guides

- Developing Backbone.js Applications
- Collection of tutorials, blog posts, and example sites

Community

- Backbone.js on Stack Overflow
- Google Groups mailing list
- Backbone.js on Twitter

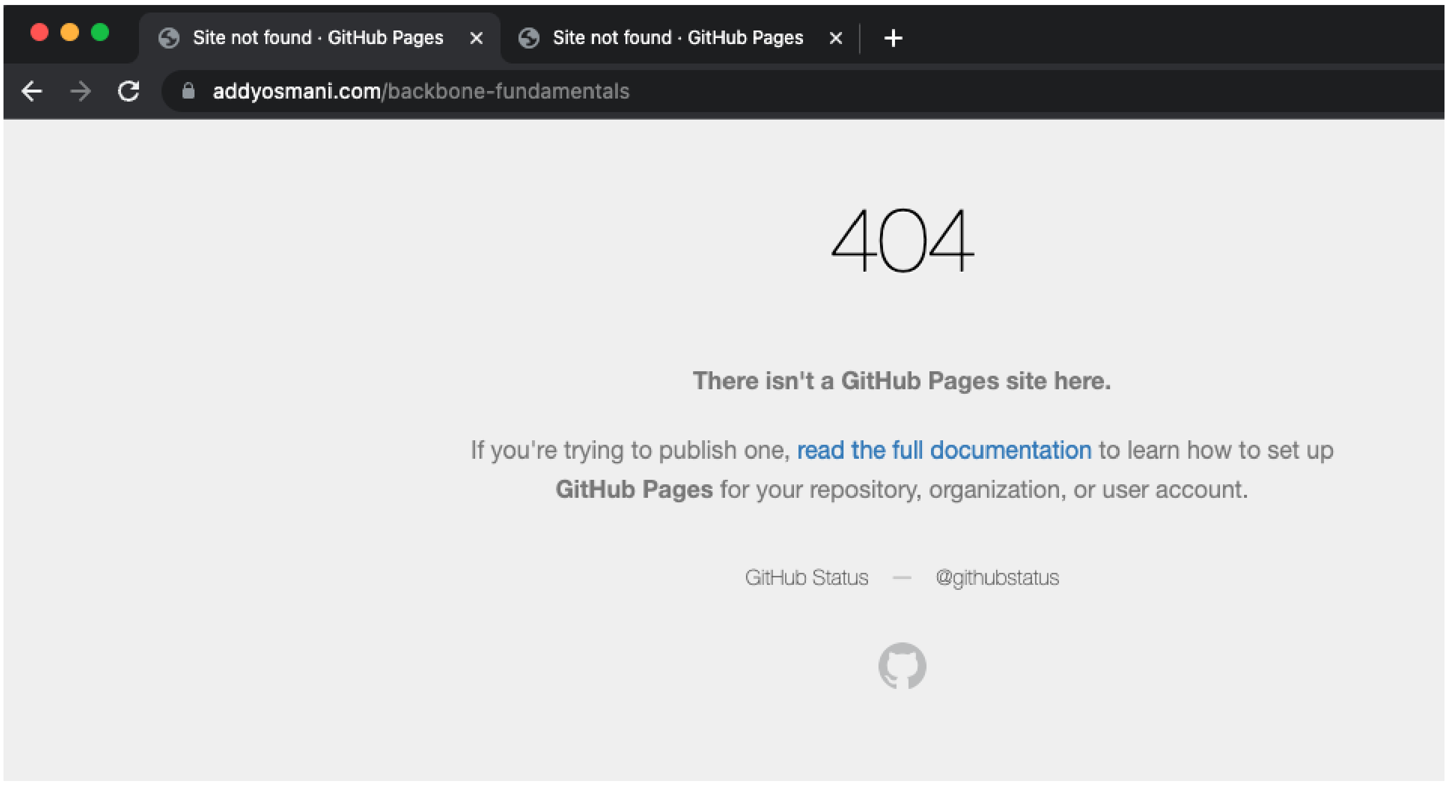
If you have other helpful links to share, or find any of the links above no longer work, please let us know.

todos

What needs to be done?

Double-click to edit a todo
Written by TasteJS
Part of TodoMVC

After clicked:



Site not found · GitHub Pages

addyosmani.com/backbone-fundamentals

404

There isn't a GitHub Pages site here.

If you're trying to publish one, read the full documentation to learn how to set up GitHub Pages for your repository, organization, or user account.

GitHub Status — @githubstatus

Risk/priority

Low - These links' destinations are not under our control. Ideally, we would check this at some point, but we do not believe this would be within scope given our deadline.

Extra Observations

- No way to add an item with mouse
- Mobile page loads as a 'desktop' version of the site (using safari and Firefox tested)
  - can still add items using keyboard
- No clear 'navigate back' button when within a framework - had to use browser buttons
- Note that the site allowed entries of well over 248 characters. for example, we copy pasted the whole of the wiki article on WW2 and it accepted it as one item
- In general the format and the view is good - all clear, no obvious out of place formatting
- The general performance was good - pages loaded fine, not lagging or delays in using the frameworks we tested on the site
- the features we tested worked as described - other than those listed and/or highlighted in the 'Bug' section.