Mattias Wickberg 1dv600 Assignment 1

Vision for library system

The system is supposed to be a library system, so the end vision must of course be a system which would help libraries in their day-to-day work of cataloguing, lending and retrieving books.

This system requires a few things for basic functionality from the client perspective. First of all, there needs to be a way of adding new books to the system, as well as modifying and deleting existing books. Each book needs to have a unique id to easily separate books from the same author or with identical titles, different printings of the same book or just multiple copies. Other than these, there need to be some fields to provide information about the nature of the book, such as genre. Since the system is, at least at this time, not aimed for commercial use, implementation of a standard library cataloguing system is not planned. A place for a description would however be helpful as well. The functions needed for this system to be working as a personal library is adding, removing, editing, and listing books both as a whole, and searching by author or title.

Current status

The status of the application as the project starts off is that the client side is finished and has an interface that will work, as well as an API for the server to work with. The server side is however non-functional, with only the very bare bone structure of code in place.

Stakeholder goals

The customer of a system like this wants a library system for personal use with the functionality stated above, within the dealine.

Developer goals: Meet deadline, with functionality in place and a robust code base, tests and documentation that would allow for refactoring or expanding the system later. Also, pass the course.

Tasks

In each iteration of teh project:

- Plan the project
- Design the project
- Plan and write tests
- Implement functionality
- Make sure everything works

Mattias Wickberg, 1dv600

Project plan for library system

Introduction

The project is to build the backend for a working library system from a framework supplied by the course managers. The work started in the end of January 2018, and the project is due March 18th 2018, although there will be a fall-back option later on during the spring in case the project plan fails and the time-frame needs to be extended.

Purpose

The primary purpose of the project is primarily to learn the methodology for developing software, with focus on the planning, testing and documentation parts. The secondary purpose of the project is to build a working library system for personal use.

Stakeholders

The primary stakeholder in this project is the developer - the student working on the project, i.e. me. This stakeholder is affected by the success of the project in whether or not I pass the course, and eventually whether I get my teacher's license in Interface Design - which is the reason I'm doing this.

Other important stakeholders in this projects are the teaching staff for the course, being affected in the way that they want as many students to succeed as possible, with good results, and want their project to be percieved as educational, challenging and rewarding.

If this was a real library system project, the stakeholders would also be the customers asking for the system.

Scope

The scope of this project is limited to developing a working library system aimed at personal use rather than commercial, given that the functionality is limited to adding, editing, removing and listing books without adhering to any existing library conventions. The front end in this system is already provided, and so the project is limited to coding the back end.The time available is limited to January 15th 2018 - March 18th 2018.

Resources

Resources required for this project:

- Lectures provided by LNU for theoretical background
- Client side code provided by LNU, as well as skeleton for back end.
- Software Engineering 10th ed. by Ian Sommerville
- Project instructions on github

Project Goals

- Finish the project within time allotted
- Have a working product with the ability to add, delete, edit, list and search for books by title and author.
- Finish the project with enough quality to get a passing grade in the course.

Risk analysis

Risk	**Probability**	**Effects**
Crisis at work means I won't have time for studies	Moderate	Catastrophic
Crisis in own life means I won't be able to complete project	Low	Catastrophic
Illness means I miss study time	Low	Serious
Time allotted for study isn't enough to finish project	Moderate	Serious

Complexity of project is underestimated	Moderate	Tolerable
Complexity of other course demands I take study time from this project	Moderate	Tolerable
Computer breaks down	Low	Tolerable

Overall, the largest risks that I can see to me failing to complete this project is that my time will have to be devoted to my job instead of my studies - working full time as Lead Teacher at a school I've helped found means I tend to be the one having to jump in whenever something goes awry, and working with people, that tends to happen quite frequently. However, this time of year tends to be calmer that most, so hopefully I'll be able to only work full time and not more.

The other risks are that I have underestimated the complexity, or overestimated the time I can give to this project, but I count those as more moderate risks, since they can be mitigated by reevaluating and reworking the plan.

Work breakdown

Assignment 1

- Task 1: Personal planning

1. Subtask A: Books
2. Subtask B: JSON
3. Subtask C: Web

- Task 2: Vision
- Task 3: Project plan
- Submit assignment for review by 2018-02-04

Iteration goal

Have an understanding of what the product is and what the specific goals of the project are.

Assignment 2

- Task 1: Analysis

1. Subtask A: Use cases
2. Subtask B: Diagrams
3. Subtask C: Use case realization

- Task 2: Design
- Task 3: Implementation
- Submit assignment for review by 2018-02-18

Iteration goal

Have an understanding of system analysis and design, and design documents for a few of the functions to be implemented.

Assignment 3

- Task 1: Test plan
- Task 2: Test cases
- Task 3: Unit tests
- Task 4: API tests
- Submit assignment for review by 2018-03-04

Iteration goal

Have a plan for the testing of the product, and the first few tests in place.

Assignment 4

- Task 1: Planning

Iteration goal

Have a plan for the different iterations of the last stage of development, including the plan, design, testing and implementing of each interation.

- Task 2: Iteration #1

Iteration goal

Having designed, tested and implemented the search for specific book function, working with both title and author search options.

- Task 3: Iteration #2

Iteration goal

Having designed, tested and implemented the Add book function of the system.

- Iteration #3

Iteration goal

Having designed, tested and implemented the edit book function of the system, and tested that everything works from all previous iterations.

- Submit assignment for review by 2018-03-18

Project schedule

Assignment:Task	**Start**	**Done**
Assignment 1: Plan		
Task 1	2018-02-01	2018-02-02
Subtask A	2018-02-01	2018-02-01
Subtask B	2018-02-02 AM	2018-02-02 AM
Subtask C	2018-02-02 PM	2018-02-02 PM
Task 2	2018-02-01	2018-02-03
Task 3	2018-02-01	2018-02-04
Assignment 2: Design		
Task 1	2018-02-08	2018-02-09
Subtask A	2018-02-08	2018-02-08
Subtask B	2018-02-09 AM	2018-02-09 AM
Subtask C	2018-02-09 PM	2018-02-09 PM
Task 2	2018-02-08	2018-02-10
Task 3	2018-02-08	2018-02-17
Assignment 3: Testing		
Task 1	2018-02-20	2018-02-22
Task 2	2018-02-20	2018-02-24
Task 3	2018-02-20	2018-02-26
Task 4	2018-02-20	2018-02-28

Assignment 4: Project		
Task 1	2018-03-10	2018-03-10
Task 2	2018-03-11	2018-03-15
Task 3	2018-03-15	2018-03-18

February 1st - 4th: Assignment 1.1-1.3

Monitoring

Revise and update plan every Sunday.

Plan for project stage

Project goal: Have a working library system with all basic functionality implemented. Basic functions are:

- View a book
- View a list of books
- Add a book
- Edit a book
- Delete a book
- Search a book by author or title (this one is optional)

Of the functions above, the view list of books, view book, and delete book are already implemented at this stage of development. Thus there are two functions that are imperative that they get finished, and then one that is optional, but needed for the project to feel complete.

Milestones:

Iteration 1 milestone: Get book (by id) implemented.

Iteration 2 milestone: Add book function fully implemented and tested.

Iteration 3 milestone: Edit book function fully implemented and tested Stretch Goal: Search for specific book fully implemented and tested, allowing search by both author and title.

Iteration time plan

Iteration 1: 8 hours Iteration 2: 8 hours Iteration 3: 10 hours

Iteration #1

Tasks in I1:

- Create use case diagram for project 30 minutes
- Create sequence diagram for Get book by id function: 45 minutes
- Describe design: 30 minutes
- Write a test plan for I1: 30 minutes
- Write manual tests: 30 minutes
- Write unit tests for I1: 60 minutes
- Write code: 60 minutes
- Test functionality: 15 minutes
- Document process: 30 minutes
- Reflect: 30 minutes

Requirements

Function GetBookResourcse should return a book object when called with an id that corresponds to a book id in the database. If id does not correspond to an id in the database, function should return an empty object.

Design comments

The choice of designing the GetBookResource was not my original plan - after all, the function seems not to be used at all by the client, and as such seemed unneccessary. However, it dawned on me that while the function is not at this point useful for the API, I still need a function that does this job to be able to test the other functions properly, and so the most logical place to start would be this one, even though it makes a poorer example for some of the documentation needed for this course.

Test plan

This is probably when choosing this function may lead me to a failing grade - I don't really see a way of properly making manual tests for a function that is not used in the client API in any way that I can see. Thus, I can't make proper manual tests for this iteration.

Unit tests

Two unit tests seem logical for the implementation of this method. One to make sure that the id of the returned book is the one expected, and one checking that if an id that doesn't correspond to any book, then the function returns an empty object.

Process

This first iteration started with me having to replan the whole things, since I realised I'm better off starting with the implementation of GetBookResource rather than Add Book, since I realised that I would need the former to write the unit tests that I want for Add book. Thus after redoin the plan for the project phase and switching things around, I did the sequence diagram for GetBook, then planned my tests. I realised that I didn't quite know how I would go about doing manual tests, since I don't quite understand how the client uses the GetBookResource - it doesn't really seem to at all. I read up a bit on the Router.route, then looked through the slack channel to realise a lot of people are wondering about this function. There seemed to be no answers forthcoming from anyone in charge of the course though, so I decided to focus on the unit tests this time around.

Two unit tests were coded to check that the function responds with the correct book if provided with an id of a book in the database, and with an empty object if not.

Reflection

My main profession is working with people rather than code, but it seems to me to be at least to some degree a similar principle - the plan you make never holds longer than when you start working by that plan. Of course, in the world of software development the plans you make will probably hold better and better the more experience you have, and the better you know the capabilities of your team. However, your plans will probably be continously revised as you discover that you need something that you missed in the planning stage, or that you probably should switch the order of some parts of development.

So at this stage - do I feel that the implementation was easier because of all this planning, designing, writing unit tests and diagrams? Slightly yes. Is it worth the time invested? Not by any means! The time invested in these iterations seem to me to be a bit much for the simple system being built, and I guess that's why I have some trouble taking some parts of the assignments seriously. I get that it has to be this way, though, and I am very much trying to be serious about my work.

Iteration #2

Tasks in I2:

- Review and update use case diagram for Add book function: 15 minutes
- Create sequence diagram for Add book function: 45 minutes
- Describe design: 30 minutes
- Write a test plan for I1: 30 minutes
- Write manual tests: 30 minutes
- Write unit tests for I1: 60 minutes
- Write code: 60 minutes
- Test functionality: 15 minutes
- Document process: 30 minutes
- Reflect: 30 minutes

Requirements

Function add book should take a JSON-object as input and given that object has a title that is a string of at least a length of 1, give that book a unique id and save it to the database. If title is missing, Add book should ignore the input and not save anything.

Design comments

The functionality to add a new book into the library systam database is a primary one of course. The API for this functionality specifies that the client will send a reqest with a json object to the server. This json object will on the server side be sent from books.js to the AddBookResource method, which in turn will fetch the current list of books, convert these into json format - functionality that is already in place from the GetBooksResource method - and add in the new book into this list. The book list will then be converted back into xml format - also in place from earlier stages of development, and write the expanded book list back to the file books.xml

Test plan

In this iteration I want to make two manual test cases: One adding a new book with all fields filled out, and one adding a new book with no title or author given. The first should result in a book being added, but the second should not (title at least should be mandatory, though author could be implemented to setting a default of unknown if none is provided).

Unit tests

Two unit tests should be implemented, one to check that AddBookResource checks that book data is valid, and one to check that if id is already taken, AddBooksResorce responds with a message indicating that.

Process

For this iteration I started by doing the sequence diagram, and then spent some time deciding on the manual tests and unit tests. Then I wrote a code for add book, which at first run didn't pass the unit tests, which meant some time figuring out what I had missed. Turns out I didn't have my types right in my id. I did some work and ran again, and this time passed the unit tests. Moving on to the manual tests, these, however, did not pass. The first one - to check if a book was added with the fields filled in, caused the system to crash when I reloaded the page. Some bug tracking later I realised that the implementation I had made couldn't gracefully handle empty fields converted to xml, so I added a check for empty fields to add a string with just a space to those.

Once done, only the second manual test failed, which was that a book without title was still added, which makes no sense for a library system. Ideally, this would of course be checked client side rather than server side, but the situation being what it is, I had added a check for title that gave a false positive. I changed this check to look for a string, and the test passed.

Reflections

This iteration the tests really did their work! I haven't had much use for the diagrams yet in this course, but the tests are definitely helpful, and I must say that I think that at least to some degree, I'm going to move towards TDD and BDD. When it comes to diagrams, I'll look into it further at some point, but for this particular project, they haven't been very useful. They don't really seem meant for a small solitary project.

Iteration #3

Tasks in I3:

- Review and update use case diagram for Edit book functionality: 15 minutes
- Create sequence diagram for Edit book functionality: 60 minutes
- Describe design: 30 minutes
- Write a test plan for I1: 30 minutes
- Write manual tests: 60 minutes
- Write unit tests for I1: 60 minutes
- Write code: 120 minutes
- Test functionality: 15 minutes
- Document process: 30 minutes
- Reflect: 30 minutes

Requirements

Edit book should take a JSON-object as input and look for a book entry with the corresponding id, and if input has a title of a string of at least length one, change the details of the book in the database to the details of the incoming object and save the new book to file. If no title, function should not update.

GetBooksResource should take a search term as input and filter the boks in the database by that search term as author or title, and return the filtered list even if that list is empty.

Design comments

The design of edit book should be fairly straight forward, since I have the functions for GetBook and add book finished. Edit book gets info from client, and needs to validate that information to make sure that there is still a title in there, and that everything is a string - this functionality is already in AddBook, so no need to duplicate that code. However, I don't want to add a book since there is already one, so what I want to do is get the existing book list, find the instance of the book that is in there, and update the fields in that book to match the changes that the user saved.

For the search books by title or author, this of course will require a bit more work. I'm thinking, that other than changing books.js to allow this functionality, the search terms is forwarded from books.js to GetBooksResource, where I before the current code returns the book list will have to do a check if that information was sent, and if it was call a function FilterBookList that will go through the current book list and make a new list including books that match the serach terms either in teh author field or in the title field, and send that list to the client. If the list is empty, then that should be sent to, to indicate that no books matching that title/author was found.

Test plan

For the Edit book function I want two manual test cases, one that check if a book that is edited also shows the changes that I make, and one where I completely remove the title, which the system shouldn't let me.

For the search by author, I need one test case to check if a search by title works, one for an author with more than one entry, to check that the system lists all the books expected, and one with a search term that doesn't match any books, where I expect an empty page as a result.

Unit tests

For the edit books function, I want a unit test that checks that the id sent in is the same as one after editing, and I want one that checks if the function can handle empty or fields with the wrong data type.

For the search books by title/author function I'm going to want ???

Process

I started by implementing EditBookResource, as that was the primary requirement. As with previous iterations, I made a sequence diagram and unit tests first, and then wrote the function. Not much trouble this time, other than discovering a typo in a previous function that had some impact even though it did not break functionality (published date was not displayed).

Once the primary functionality was all implemented, I wanted to reflect on what I was missing toi get a passing grade, and realized I probably need to make activity diagrams, and add API tests. I have been slightly impaired this iteration by illness, augmented by a bout of panic attacks, but the show must go on, so I have been trying to fill in the gaps.

After this, I went on to implement the added functionality of searching by title or author to the GetBooksResource function, because it bugged me not to have it in there.

Finally, I added some more api tests.

Reflection

On a personal level, since I never intend to make this my profession I don't much see the need to master UML (I'm doing this to get my teacher's license in interface design), and for these smaller projects I see limited use of so much time spent on making diagrams, while not even mentioning usability. It seems like bad development practice to me to devote so much energy to the former while completely ignoring the latter. However, this is a university course, and I fully understand that the scope of this course is limited to the inner workings of development, and that for at least projects with some complexity, or systems that are critical, these diagrams are vital.

Mattias Wickberg, 1dv600

Personal reflections on Project 1dv600

#Assignment 1

## Task 1

Subtask A

The most complicated aspect of this task was really that I don't understand what the book object in the dao folder should be? In the end, I just made a file there and went on to create the book list in GetBooksResource. Again, I made the mistake of adding the book list to the callback to make it show on the website instead of console.log, so had to retrace and log it to the terminal instead. As always, the hardest part of any task is usually to actually understand what the task giver is saying/writing. This is something I try to remind myself of every day as a teacher, because it's often the most challenging task for both me and my students.

Subtask B

Converting to JSON is the work of a few minutes, so not much problem with the work this time.

Improvements: Firstly having a list of books in a function is really not a workable strategy for a site where you'll need to add new books through an interface, so first of all I need to separate my books from my functions, and then require that json file into my function. This is probably not good enough for the final version, but it's better than what's there now.

Secondly, I need to comment my code so that I know what I'm doing, and add more details to my books.

Subtask C

Really didn't need much time for this part, and I think maybe I should have come up with more complex improvements - however, without being quite sure what the requirements for the system is, I don't want to rush ahead with something major before I have implemented the basics thoroghly. I realise that my json file is very transitory, as there will be xml in the works, but still, it bugged me having a messy function.

Task 2

The difficulty with writing a vision is that I still have no clear idea of how complex this system can be during these few weeks, or whether to see as a library system for a full-fledged library, as a small scale version mostly used at home, or as a system for a smallish bookshop. They all have different demands. I went with the real library-angle, since I'm guessing that's what the basics of the system was intended for, and it may be difficult to work against what the course idea is. In reality it appeals more to build a system for my home or my school, sicne we don't really have one at the moment, but generally speaking it's better to avoid doing things you actually need in university assignments - they need to meet the requirements of the course first, and your own needs second.

Task 3

The project planning was mostly just a nice was of doing something with a bit more structure than I probably would have otherwise. Planning has been a crucial part of my life for many years now, juggling a complex work, studies and home life. The risk analysis was one part that I wouldn't have put into words had this not been part of the course literature, but it did help me focus on what the real risks are. This of course being a smallish project where I work alone rather than a large project with many people involved, it is a radically more simple plan than the ones I do work with for a living. Time wise I did overestimate the time it would take to do some parts of the first assignment, but the stimates were based on me usually being interrupted while I work, and this last two days I've had perfect calm to focus on this course!

#Assignment 2

##Task 1 ###Subtask A I have encountered UML-diagrams previously, and I do find them very useful in the planning stage for development. Use cases are of course very similar to ways that I've worked in Interaction Design, since I have a lot more experience from that field than programming, and so they do feel quite natural to have here as well. In general, I think the difficult part here is to decide on the scope of the planning stages of this project, knowing that a full scope library system would need more functionality than we can really make during a course like this.

Subtask B

Robustness diagrams in this very case seem a bit overkill for documentation and planning, but I can definitely see the usefulness of it to provide a link between the slightly abstract use cases to the actual implementation of the functionality. Especially for a larger sclae product where the people implementing and the ones designing the system may not be the same people or even in the same location, this seems to provide a good basis for clear technical communication of how the different steps should work.

Subtask C

Sequence diagrams, again feel quite useful. But like with subtask B I'm starting to wonder how many different diagrams are really useful for this fairly limited project. For this kind of project, I do feel that I would do use cases and either sequence diagrams or robustness diagrams, but to add more would start to feel a bit redundant. Again though, I can see that the story is different for larger scale projects. After doing the sequence diagrams I feel a have a better grasp of the system architecture.

##Task 2 This way of doing things does really force me to think through thigs before I do the coding. Of course, it also means I can't test and see if I'm mistaken in some assumption of how things work - but then if I do find out in the end that I made a planning mistake I guess I can go back and redo the sequence diagram. It probably reduces the amount of errors in the end anyways. It is akin to what I tell my students about essay writing (my profession is language teaching), that is is a process that is more about planning than actual writing if you want the end result to be of good quality and reduce the number of errors and unnecessary work you do.

Task 3

Ok so this one took a lot longer than I thought, possibly pasrtly because I started it while I was at the students' LAN party at work, and was quite tired and unfocused, but I really got tangled in how to transform the xml to json. It really should have been ten or twenty minutes work, but it took me two hours to figure that part out. The rest was really quite easy. Since the only real tangle here was the xml, I don't really feel that there was a huge difference between the designed and undesigned approach to implementing a function - the delete function was so much faster and easier to implement, because I had figured out the xml and understood the flow between different functions. I think the lessons I take away from this assignment is:

- What takes time is rarely what you think will take time - it's true about a lot of things in life.
- Converting information between formats is important, but can be time consuming (my partner who works as a bioinfomatician says that this is mostly what working as a scientist is about).
- Planning the flow between functions and what information travels is important, but planning the overall architecture is more important than the details.

Assignment 3

Task 1

OK, so I'm not completely done, but I think this will work better for me if I come back to the plan and fill in some more details after I've done the first two tests and have a clearer view of what I should test. I also think I'll come back and replan the API-tests after I'm done with the Unit tests, instead of planning every test before I start.

## Task 2

Wrote two fairly simple manual tests. I first thought of these as real user tests, but I realised after reading about it that this is not the case - coming from the field of linguistics, pedagogy, UX and behavioural sciences testing does mean something a bit different for me than it does in the fireld of development 😃. Anyway, I do think that manual testing is what comes most natural for me, since it tends to be the an integral part of my natural work flow, though I'm not very thorough in documenting it properly. I see the importance of doing more of these, and it feels a bit wrong to stop at just these two - but I'm as always pressed for time.

Task 3

This was really good - I have been meaning to start looking more closely at unit testing and testing frameworks, and mostly this was not too tricky, although I know I have done very simple tests thus far. This is definitely something I want to dig into further, for my own progression, but also because I know that this is one of the weaker areas at our school.

Task 4

The trickiest part of this was that I got a false positive until I figured out how to use the 'done' functionality when testing code with callbacks. This also made me realise I had a bug in one of the previously written tests. Good learning 😃 Other than this not much more to say about testing - it's kind of a vital part of any development, but that feels lika a no-brainer (in theory, I know it's often lacking in practice).

Assignment 4

Task 1

My main profession is working with people rather than code, but it seems to me to be at least to some degree a similar principle - the plan you make never holds longer than when you start working by that plan. Of course, in the world of software development the plans you make will probably hold better and better the more experience you have, and the better you know the capabilities of your team. However, your plans will probably be continously revised as you discover that you need something that you missed in the planning stage, or that you probably should switch the order of some parts of development.

So at this stage - do I feel that the implementation was easier because of all this planning, designing, writing unit tests and diagrams? Slightly yes. Is it worth the time invested? Not by any means! The time invested in these iterations seem to me to be a bit much for the simple system being built, and I guess that's why I have some trouble taking some parts of the assignments seriously. I get that it has to be this way, though, and I am very much trying to be serious about my work.

Task 2

This iteration the tests really did their work! I haven't had much use for the diagrams yet in this course, but the tests are definitely helpful, and I must say that I think that at least to some degree, I'm going to move towards TDD and BDD. When it comes to diagrams, I'll look into it further at some point, but for this particular project, they haven't been very useful. They don't really seem meant for a small solitary project.

Task 3

Edit book should take a JSON-object as input and look for a book entry with the corresponding id, and if input has a title of a string of at least length one, change the details of the book in the database to the details of the incoming object and save the new book to file. If no title, function should not update.

GetBooksResource should take a search term as input and filter the boks in the database by that search term as author or title, and return the filtered list even if that list is empty.

Time Log

A comment about the time logs here: Since I work 115% and study full time, the time is not easily measured. I manage to work and study in paralell mainly by utilising little times in between, and spending a lot of time thinking about things while doing other work that doesn't require much of my cognition. Thus, the times below says very little about the time actually spent, but it is the only time I can measure. Some of the times will be longer than actual work as well, being done at work while interrupted by colleagues and pupils that need help - or because I had some trouble with Vagrant starting March 10th that took some time more than once.

Assignment/Task	**Expected time**	**Start**	**Finish**	**Actual time**
Assignment 1				
Task 1				
Subtask A	45 minutes	2017-02-03 17.23	2017-02-03 18.16	53 minutes
Subtask B	15 minutes	2018-02-03 18.20	2018-02-03 18.30	10 minutes
Subtask C	60 minutes	2018-02-03 19.15	2018-02-03 19.55	40 minutes
Task 2	45 minutes	2018-02-04 14.22	2018-02-04 14.57	35 minutes
Task 3	75 minutes	2018-02-04 15.43	2018-02-04 16.37	54 minutes
Assignment 2				
Task 1				
Subtask A	120 minutes	2018-02-08 19.50	2018-02-08 21.46	116 minutes
Subtask B	45 minutes	2018-02-08 21.55	2018-02-08 22.30	35 minutes
Subtask C	60 minutes	2018-02-09 20.50	2018-02-09 21.29	39 minutes
Task 2	30 minutes	2018-02-09 21.51	2018-02-09 22.15	24 minutes
Task 3	120 minutes	2018-02-09 22.15	2018-02-09 23.22	
		2018-02-10 10.00	2018-02-1011.20	

		2018-02-11 09.53	2018-02-11 11.15	229 minutes
Assignment 3				
Task 1	60 minutes	2018-02-22 20.51	2018-02-22 22.01	70 minutes
Task 2	60 minutes	2018-02-25 10.09	2018-02-25 10.37	28 minutes
Task 3	150 minutes	2018-02-25 10.40	2018-02-25 12.14	94 minutes
Task 4	120 minutes	2018-02-26 20.25	2018-02-26 21.45	80 minutes
Assignment 4				
Task 1	90 minutes	2018-03-10 16.00	2018-03-10 17.40	100 minutes
Task 2, diagrams	75 minutes	2018-03-10 19.15	2018-03-10 20.35	80 minutes
Task 2, design	30 minutes	2018-03-10 20.35	2018-03-10 20.50	15 minutes
Task 2, test plan	30 minutes	2018-03-11 17.50	2018-03-11 18.10	20 minutes
Task 2, manual tests	30 minutes	2018-03-11 18.10	2018-03-11 18.30	20 minutes
Task 2, unit tests	60 minutes	2018-03-11 20.15	2018-03-11 21.00	45 minutes
Task 2, code	60 minutes	2018-03-11 21.00	2018-03-11 21.20	20 minutes
Task 2, test	15 minutes	2018-03-11 21.20	2018-03-11 21.30	10 minutes
Task 2, document	60 minutes	2018-03-11 21.30	2018-03-11 21.55	25 minutes
Task 3.1, diagrams	30 minutes	2018-03-11 19.45	2018-03-11 20.14	29 minutes
		2018-	2018-	

Task 3.1, design	30 minutes	03-11 22.00	03-11 22.15	15 minutes
Task 3.1, test plan	30 minutes	2018-03-11 22.15	2018-03-11 22.45	30 minutes
Task 3.1, manual tests	30 minutes	2018-03-11 22.45	2018-03-11 23.10	25 minutes
Task 3.1, unit tests	60 minutes	2018-03-12 8.20	2018-03-12 10.05	105 minutes
Task 3.1, code	60 minutes	2018-03-12 18.45	2018-03-12 19.48	63 minutes
Task 3.1, test	15 minutes	2018-13-12 19.50	2018-03-12 20.00	10 minutes
Task 3.1, fix found bugs		2018-13-12 20.00	2018-13-12 20.50	50 minutes
Task 3.1, document	60 minutes	2018-03-14 15.10	2018-03-14 16.00	50 minutes
Task 3.2, diagrams	75 minutes	2018-03-14 16.00	2018-03-14 16.39	39 minutes
Task 3.2, design	30 minutes	2018-03-14 16.40	2018-03-14 17.00	20 minutes
Task 3.2, test plan	30 minutes	2018-03-14 17.00	2018-03-14 17.20	20 minutes
Task 3.2, manual tests	30 minutes	2018-03-14 20.00	2018-03-14 20.20	20 minutes
Task 3.2, unit tests	60 minutes	2018-03-14 20.20	2018-03-14 20.45	25 minutes
Task 3.2, code	60 minutes	2018-03-14 20.50	2018-03-14 22.00	70 minutes
Task 3.2, test	15 minutes	2018-03-14 22.00	2018-03-14 22.10	10 minutes
Task 3.2, document	60 minutes	2018-03-17 11.00	2018-03-17 12.15	75 minutes

Time log falls apart a bit here, due to illness and anxiety attacks (I live with

PTSD) - I honestly have no idea what time I spent on what anymore

Diagrams, use cases, manual tests etc.		2018-03-15 16.00	2018-03-15 17.30	90 minutes
Diagrams, use cases, manual tests etc.		2018-03-15 20.00	2018-03-15 21.30	90 minutes
Diagrams, use cases, manual tests etc.		2018-03-16 10.00	2018-03-16 11.00	60 minutes
Diagrams, use cases, manual tests etc.		2018-03-16 15.00	2018-03-16 16.30	90 minutes
Diagrams, use cases, manual tests etc.		2018-03-16 18.30	2018-03-16 19.30	60 minutes
Api tests	120 minutes	2018-03-17 15.10	2018-03-17 17.05	115 minutes

Use cases

List of use cases.

Use cases for the scope of this project:

1. User wants to list all the books in the library
2. User wants to add new book
3. User wants to find book(s) with a certain title
4. User wants to find all books by a certain author
5. User wants to delete book
6. User wants to change details of book

Use Case 1

User wants to list all the books in the library

Precondition

User is on main library page, without any books listed

Postcondition

User is on books page with books listed

1.1 Main Scenario

1. User clicks on Books
2. Page changes yo books page, and displays book list from database

Secondary Scenarios

1.2 There are no books in database

1. User clicks on books
2. Page changes to books page, but no books are shown

Use Case 2

User wants to add new book

Precondition

User is on main page

Postcondition

User is on books page, books listed including the one added

2.1 Main Scenario

1. User clicks on books
2. User clicks on New Book
3. User fills in form
4. User clicks on Save
5. System displays books page, with newly added book in list

Secondary Scenarios

2.2 User doesn't fill in title

1. User clicks on books
2. User clicks on New Book
3. User fills in form, but not title field
4. User clicks on Save
5. System displays books page, without any changes

Use Case 3

User wants to find book(s) with a certain title

Precondition

User is on main page

Postcondition

User is on books page, wich lists the books that has the sought-after title

3.1 Main Scenario

1. User clicks in search field
2. User types title
3. User clicks Submit
4. System displays book page, with list of books with that title

Secondary Scenarios

3.2 No books with that title are in the database

1. User clicks in search field
2. User types title
3. User clicks Submit
4. System displays book page, without any books listed

Use Case 4

User wants to find book(s) by a certain author

Precondition

User is on library main page

Postcondition

List of Neil Gaiman novels is displayed on the page

4.1 Main Scenario

1. The person types in "Neil Gaiman" into the search field
2. A list of Neil Gaiman novels appears
3. User clicks on titles and reads about them
4. User finds a book to read

Secondary Scenarios

4.2 User spells search words wrong

1. System returns empty list.
2. User corrects spelling and searches again

4.3 System contains no books by chosen author

1. System returns empty list.

Use Case 5

User wants to delete book

Precondition

User is on main page

Postcondition

User is on books page, which lists all books, which does not contain deleted book

5.1 Main Scenario

1. User clicks in search field
2. User types title
3. User clicks Submit
4. System returns list of books with matching title
5. User clicks on book
6. User clicks on Delete
7. System displays Books page, without deleted book

Secondary Scenarios

Use Case 6

User wants to change details of book

Precondition

User is on main page

Postcondition

User is on book page with chosen book changed.

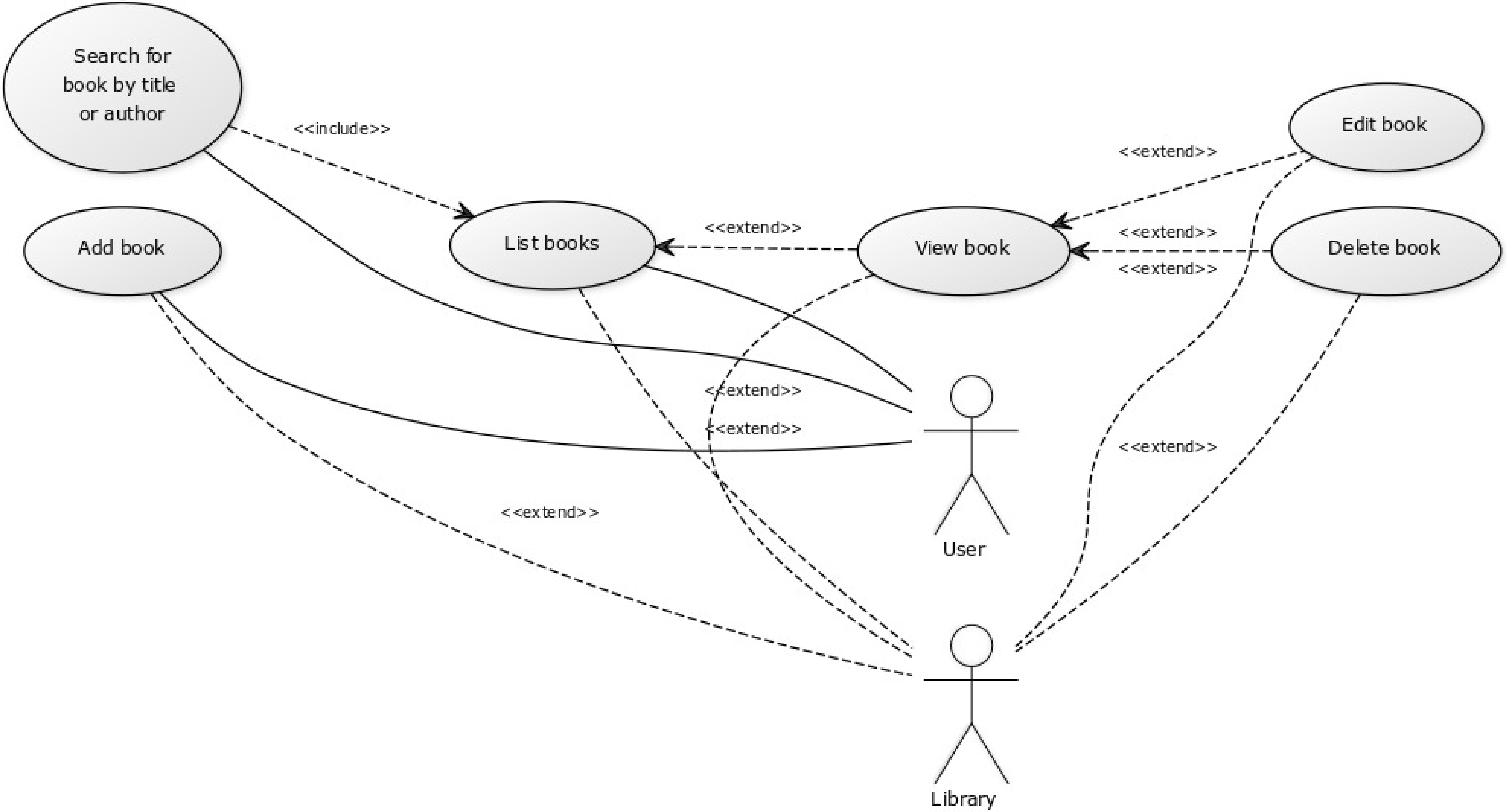
6.1 Main Scenario

1. User clicks in search field
2. User types title
3. User clicks Submit
4. System returns list of books with matching title
5. User clicks on book
6. User makes changes in fields
7. User clicks save
8. System displays book page with changes

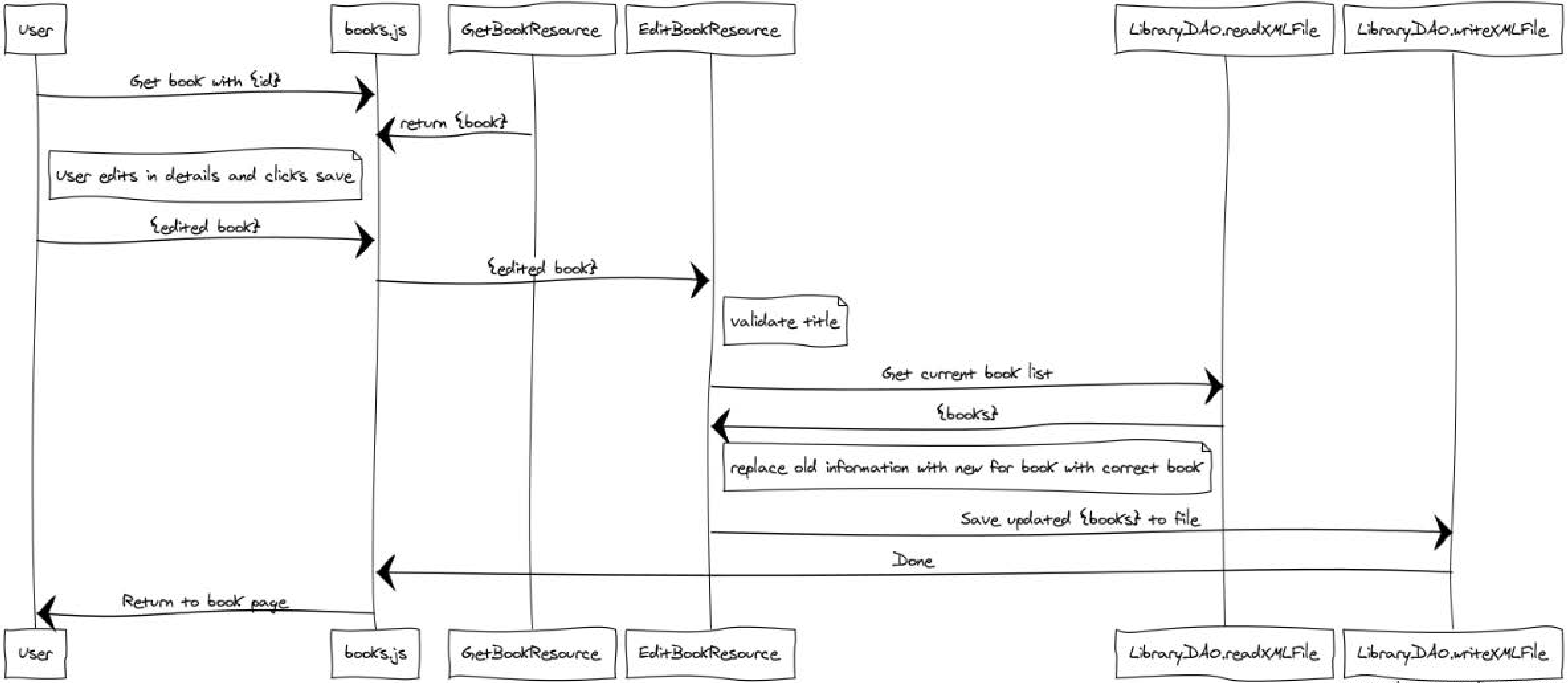
Secondary Scenarios

6.2 User removes title

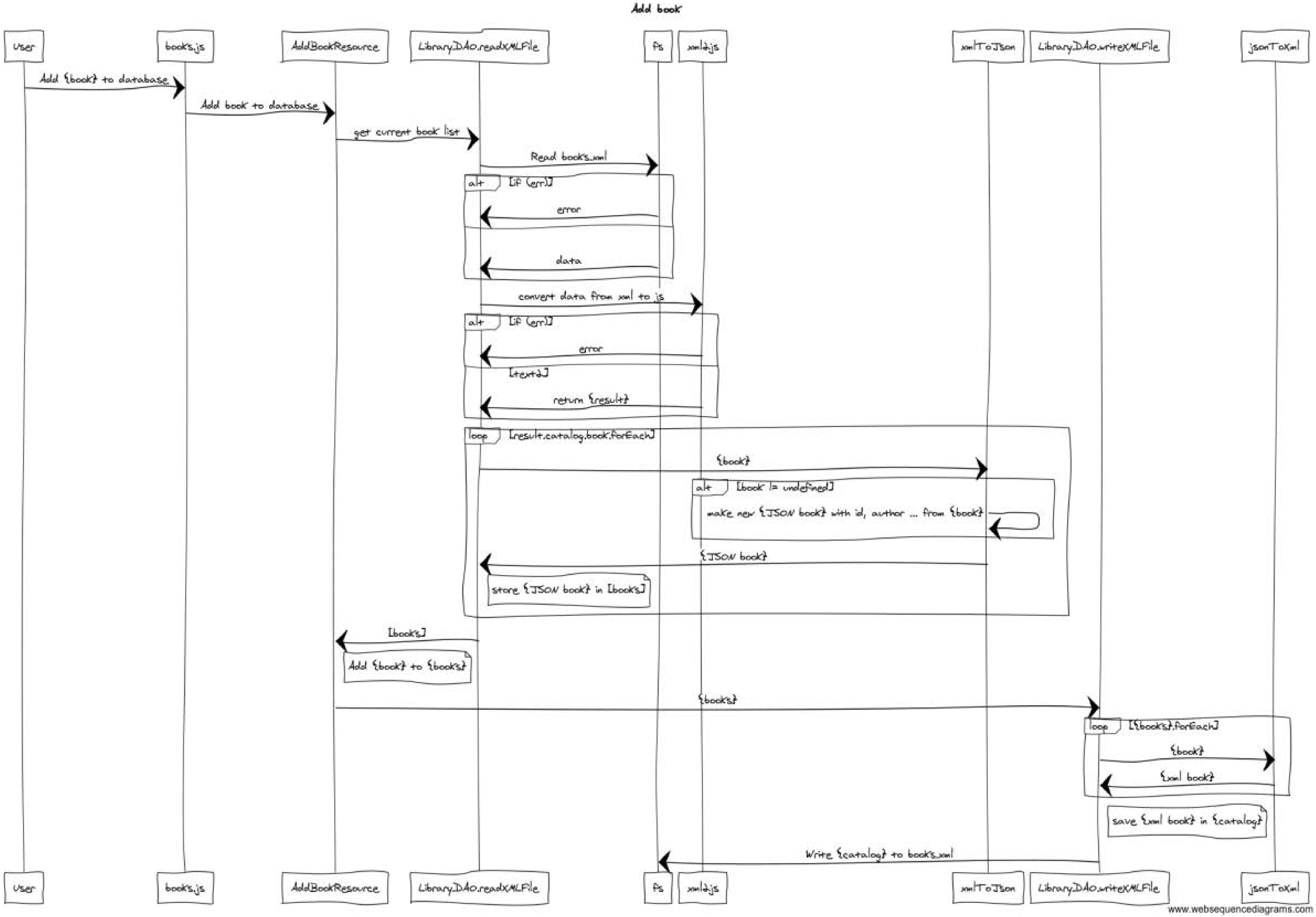
1. User clicks in search field
2. User types title
3. User clicks Submit
4. System returns list of books with matching title
5. User clicks on book
6. User removes title
7. User clicks save
8. System displays book page without changes

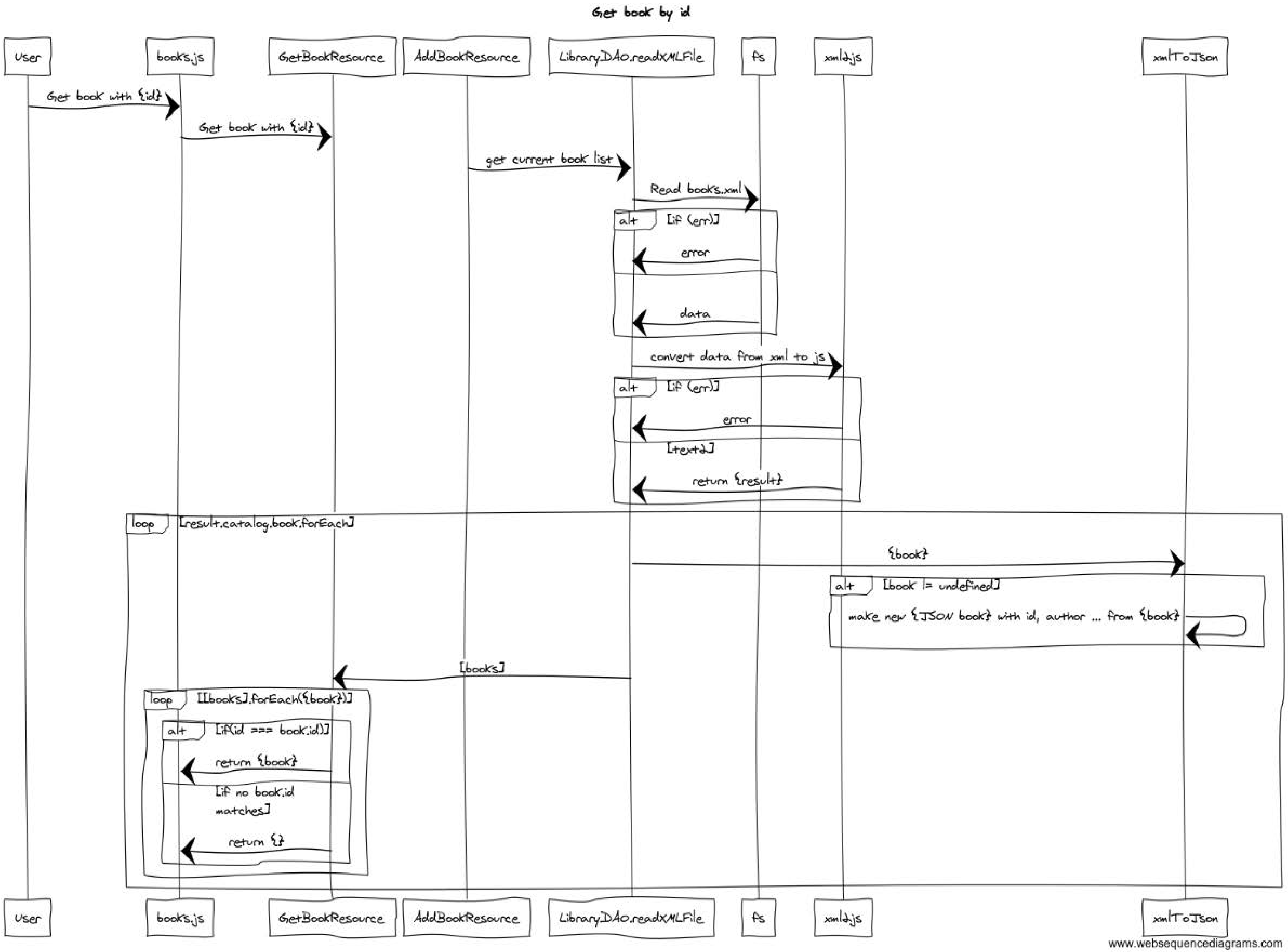

Search for book by title or author
<<include>>
Add book
List books
<<extend>>
View book
Edit book
<<extend>>
<<extend>>
Delete book
<<extend>>
<<extend>>
<<extend>>
<<extend>>
<<extend>>
User
Library

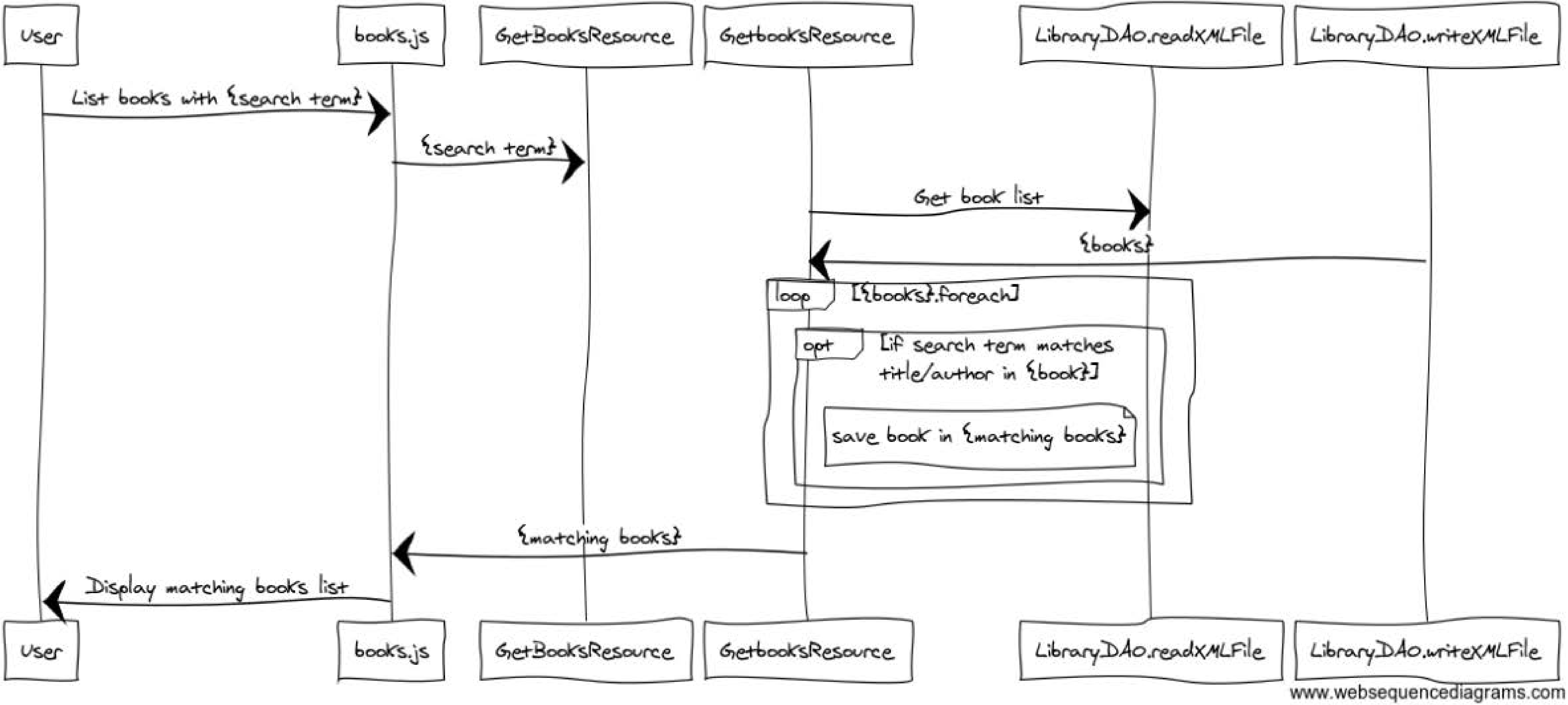
Edit book (Get book with id, see corresponding diagram for details)

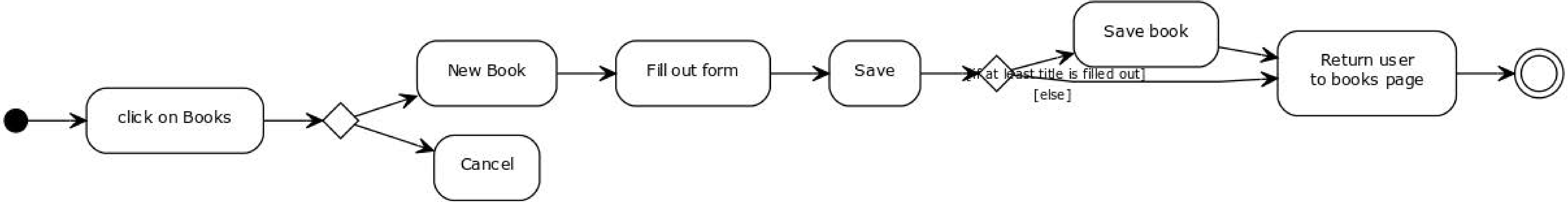


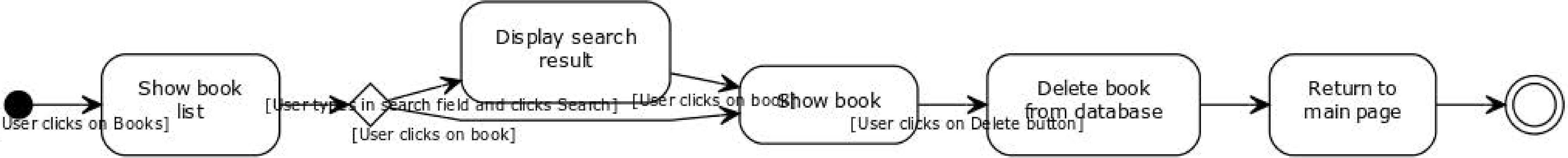
www.websequencediagrams.com

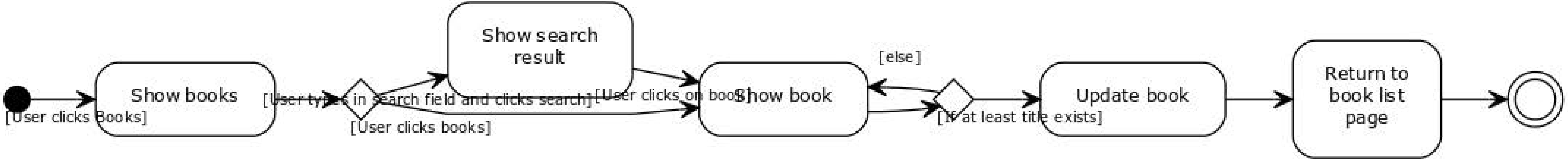

Add book
User
books.js
AddBookResource
LibraryDAO.readXMLFile
fs
xml2js
xmlToJson
LibraryDAO.writeXMLFile
jsonToXml
Add {book} to database
Add book to database
get current book list
Read books.xml
alt
[if (err)]
error
data
convert data from xml to js
alt
[if (err)]
error
[text2]
return {result}
loop
[result.catalog.book.forEach]
{book}
alt
[book != undefined]
make new {JSON book} with id, author ... from {book}
{JSON book}
store {JSON book} in [books]
[books]
Add {book} to {books}
{books}
loop
[{books}.forEach]
{book}
{xml book}
save {xml book} in {catalog}
Write {catalog} to books.xml
User
books.js
AddBookResource
LibraryDAO.readXMLFile
fs
xml2js
xmlToJson
LibraryDAO.writeXMLFile
jsonToXml
www.websequencediagrams.com

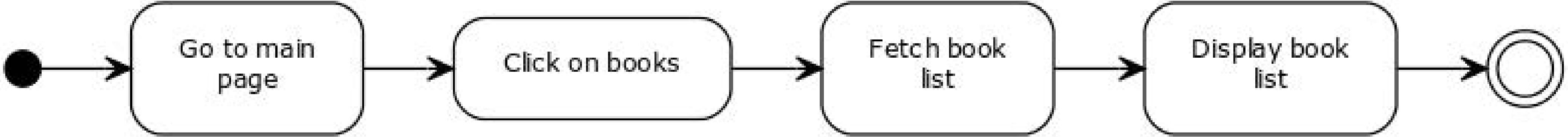

Get book by id
User
books.js
GetBookResource
AddBookResource
LibraryDAO.readXMLFile
fs
xml2js
xmlToJson
Get book with {id}
Get book with {id}
get current book list
Read books.xml
alt
[if (err)]
error
data
convert data from xml to js
alt
[if (err)]
error
[text2]
return {result}
loop
[result.catalog.book.forEach]
{book}
alt
[book != undefined]
make new {JSON book} with id, author ... from {book}
[books]
loop
[[books].forEach({book})]
alt
[if(id === book.id)]
return {book}
[if no book.id matches]
return {}
User
books.js
GetBookResource
AddBookResource
LibraryDAO.readXMLFile
fs
xml2js
xmlToJson
www.websequencediagrams.com

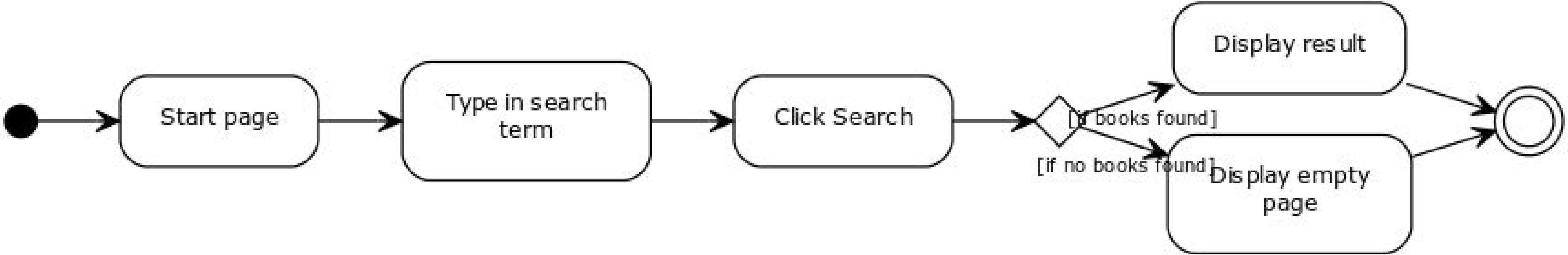

User
books.js
GetBooksResource
GetbooksResource
LibraryDAO.readXMLFile
LibraryDAO.writeXMLFile
List books with {search term}
{search term}
Get book list
{books}
loop
[{books}.foreach]
opt
[if search term matches title/author in {book}]
save book in {matching books}
{matching books}
Display matching books list
User
books.js
GetBooksResource
GetbooksResource
LibraryDAO.readXMLFile
LibraryDAO.writeXMLFile
www.websequencediagrams.com


click on Books
New Book
Cancel
Fill out form
Save
[if at least title is filled out]
[else]
Save book
Return user to books page


Show book list
[User clicks on Books]
Display search result
[User types in search field and clicks Search]
[User clicks on book]
[User clicks on book]
Show book
[User clicks on Delete button]
Delete book from database
Return to main page


[User clicks Books]
Show books
[User types in search field and clicks search]
[User clicks books]
Show search result
[User clicks on book]
Show book
[else]
[If at least title exists]
Update book
Return to book list page


Go to main page
Click on books
Fetch book list
Display book list


Start page
Type in search term
Click Search
[if books found]
[if no books found]
Display result
Display empty page

Manual test 2.1: Add New Book

Testing use case: User adds new book.

Description

This manual test is meant to test the Add new book functionality of the library system, this being one of the primary use cases for the system and a prerequisite of most other uses of the system.

Prerequisites

Prerequisites: Server up and running; user on main page.

Test steps

1. Click on Book
2. Click on New Book
3. Write in “The Graveyard Book” in Title field
4. Write “Neil Gaiman” in Author field
5. Click on Save

Expected result

Book should be added to database, and user should be returned to main library page – where the new book should show.

Tested: 2018-03-13

Result

☐ Test succeeded

☒ Test failed

Comments by tester: Book did not appear on book page

Manual test 2.1: Add New Book

Testing use case: User adds new book.

Description

This manual test is meant to test the Add new book functionality of the library system, this being one of the primary use cases for the system and a prerequisite of most other uses of the system.

Prerequisites

Prerequisites: Server up and running; user on main page.

Test steps

1. Click on Book
2. Click on New Book
3. Write in “The Graveyard Book” in Title field
4. Write “Neil Gaiman” in Author field
5. Click on Save

Expected result

Book should be added to database, and user should be returned to main library page.

Tested: 2018-03-14

Result

☒ Test succeeded

☐ Test failed

Comments by tester: Worked as expected

Manual test 2.2: Add New Book

Testing use case: User adds new book

Description

This manual test is to make sure that the library system can handle an erroneous book entry gracefully, that the book is not added into the system and system does not crash in the procedure.

Prerequisites

Prerequisites: Server up and running; user on main page.

Test steps

1. Click on Book
2. Click on New Book
3. Write in “A description of the world heritage sites in Brasil” in Description field
4. Write “10” in Price field
5. Click on Save

Expected result

Book should not be added to database. User should be returned to main library page.

Tested: 2018-03-14

Result

☒ Test succeeded

☐ Test failed

Comments by tester: Worked perfectly now, book did not appear, nor was it in books.xml

Manual test 2.2: Add New Book

Testing use case: User adds new book

Description

This manual test is to make sure that the library system can handle an erroneous book entry gracefully, that the book is not added into the system and system does not crash in the procedure.

Prerequisites

Prerequisites: Server up and running; user on main page.

Test steps

1. Click on Book
2. Click on New Book
3. Write in "A description of the world heritage sites in Brasil" in Description field
4. Write "10" in Price field
5. Click on Save

Expected result

Book should not be added to database. User should be returned to main library page.

Tested: 2018-03-13

Result

☐ Test succeeded

☒ Test failed

Comments by tester: Test failed, book was added despite no title

____

____

Manual test 3.1.1: Find Specific Book by title

Testing use case 3: Find specific book

Description

Tests if the system can serve up a book that is searched for by title

Prerequisites

Server running.

User on main page.

“The Graveyard Book” in database

Test steps

1. Click in search field
2. Type “The Graveyard Book”
3. Click Submit

Expected result

User should land on Books page, with The Graveyard Book by Neil Gaiman listed

Result

☐ Test succeeded

☐ Test failed

Comments by tester:____

____

____

Manual test 3.1.2: Find by author

Testing use case 3.1

Description

Tests if the system can handle a search by author, and list all the books by that author

Prerequisites

Server running.

User on main page.

Books in database by Isaac Asimov

Test steps

1. Click in serach field
2. Type “Isaac Asimov”
3. Click Submit

Expected result

User should land on Books page, with a list of books by Isaac Asimov, and no other books.

Result

☐ Test succeeded

☐ Test failed

Comments by tester:____

____

____

Manual test 3.2.1: Bad search

Testing use case 3.2

Description

Tests that system can handle a bad search gracefully, where the user inputs terms that do not match anything in the library

Prerequisites

Server running.

User on main page.

“När änglar dör” not in database

Test steps

1. Click in search field
2. Type “När änglar dör”
3. Click Submit

Expected result

User should land on books page, with no books listed

Result

☐ Test succeeded

☐ Test failed

Comments by tester:____

____

____

Manual test 6.1: Edit Book

Testing use case: Edit book

Description

Test where the user clicks on one book, edits the details of that book, saves and then checks if the changes are saved.

Prerequisites

Server running.

User on main page.

“The Graveyard Book” in book database

Test steps

1. Click on books
2. Click on “The Graveyard Book”
3. Change Published date to “2008-09-30”
4. Click Save
5. Click “The Graveyard Book” again and check that published date is changed

Expected result

Details of book changed and saved to database.

Tested: 2018-03-14

Result

☒ Test succeeded

☐ Test failed

Comments by tester: Test worked as expected ____
____
____
____

Manual test 6.2: Edit Book

Testing use case: Edit book, no title

Description

Test to check if system rejects changes made to a book if those changes includes removing the title completely.

Prerequisites

Server running.

User on main page

“The Graveyard Book” in book database

Test steps

1. Click on books
2. Click on “The Graveyard Book”
3. Remove all input from title field
4. Click Save
5. Check that “The Graveyard Book” is still in the list, and its details unchanged,

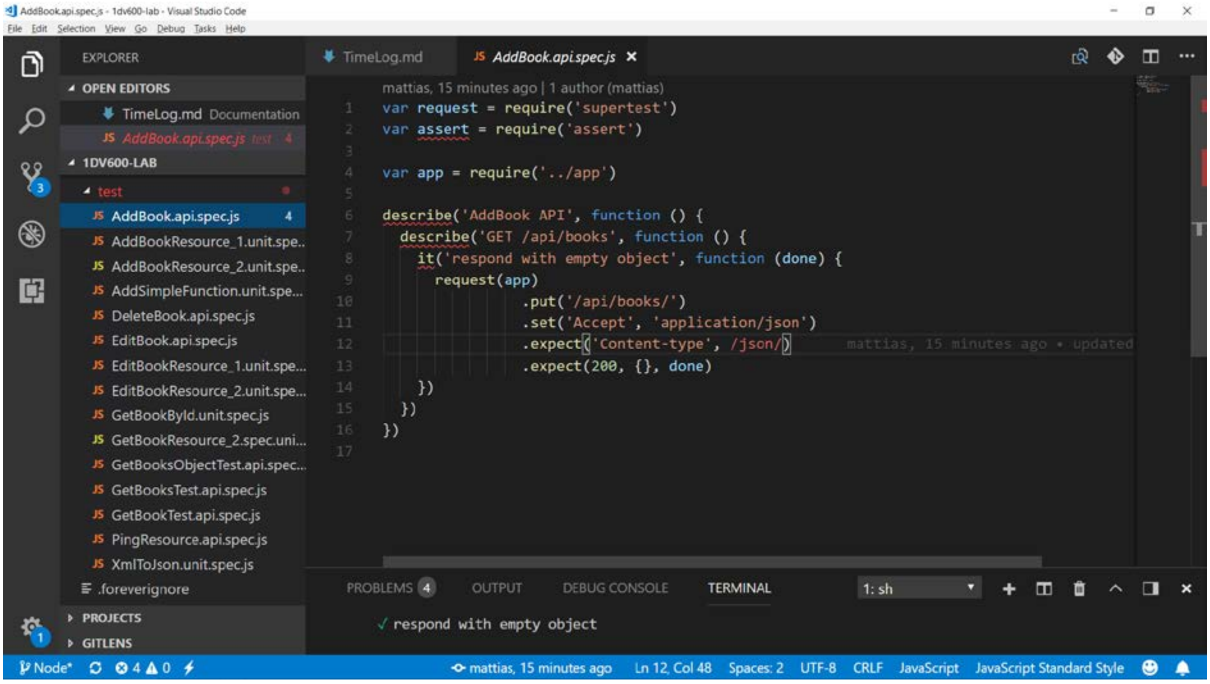
Expected result

Result

☒ Test succeeded

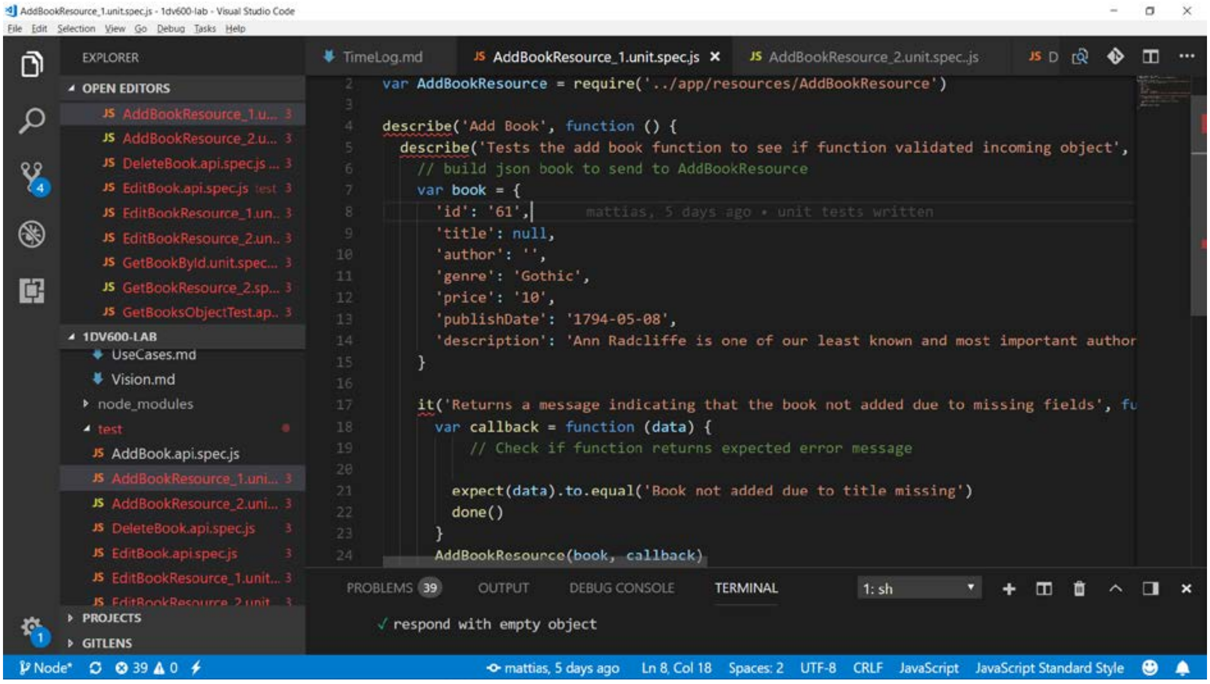
☐ Test failed

Comments by tester: Test worked as expected


AddBook.api.spec.js - 1dv600-lab - Visual Studio Code
File Edit Selection View Go Debug Tasks Help
EXPLORER
OPEN EDITORS
TimeLog.md Documentation
AddBook.api.spec.js test 4
1DV600-LAB
test
AddBook.api.spec.js 4
AddBookResource_1.unit.spe...
AddBookResource_2.unit.spe...
AddSimpleFunction.unit.spe...
DeleteBook.api.spec.js
EditBook.api.spec.js
EditBookResource_1.unit.spe...
EditBookResource_2.unit.spe...
GetBookById.unit.spec.js
GetBookResource_2.spec.uni...
GetBooksObjectTest.api.spec...
GetBooksTest.api.spec.js
GetBookTest.api.spec.js
PingResource.api.spec.js
XmlToJson.unit.spec.js
.foreverignore
PROJECTS
GITLENS
TimeLog.md
AddBook.api.spec.js
mattias, 15 minutes ago | 1 author (mattias)
mattias, 15 minutes ago • updated
PROBLEMS 4
OUTPUT
DEBUG CONSOLE
TERMINAL
1: sh
Node*
4 0
mattias, 15 minutes ago
Ln 12, Col 48
Spaces: 2
UTF-8
CRLF
JavaScript
JavaScript Standard Style

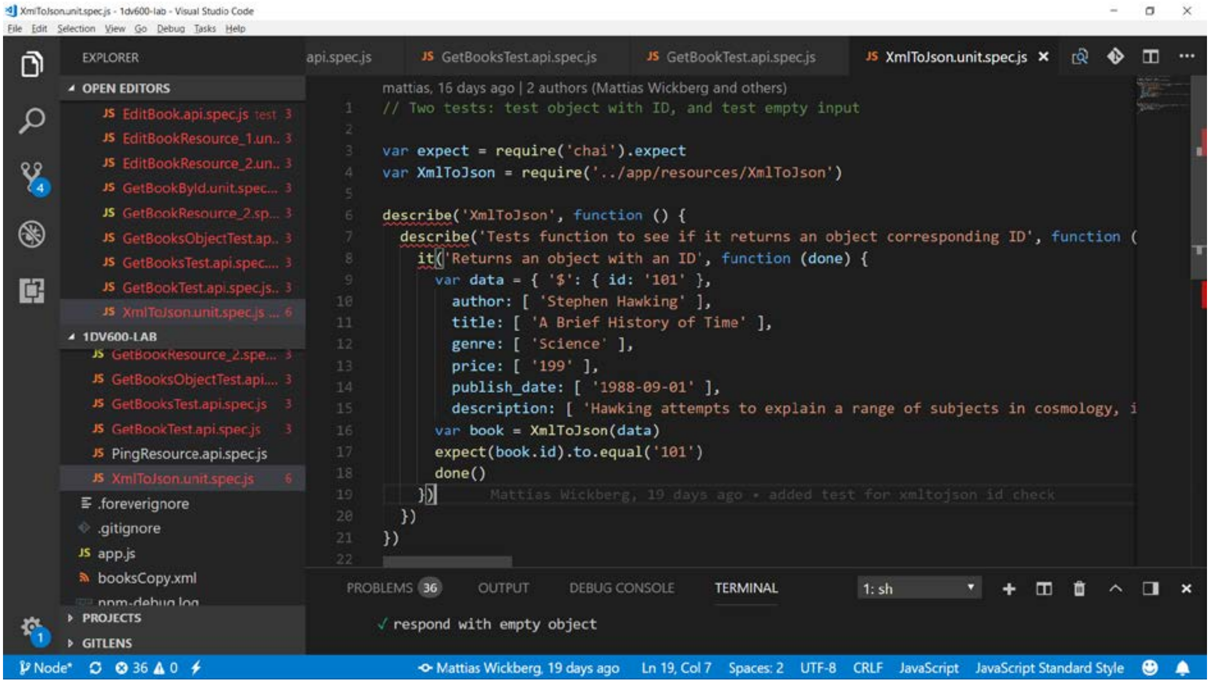
```
var request = require('supertest')
var assert = require('assert')

var app = require('../app')

describe('AddBook API', function () {
  describe('GET /api/books', function () {
    it('respond with empty object', function (done) {
      request(app)
              .put('/api/books/')
              .set('Accept', 'application/json')
              .expect('Content-type', /json/)
              .expect(200, {}, done)
    })
  })
})
```

```
✓ respond with empty object
```


AddBookResource_1.unit.spec.js - 1dv600-lab - Visual Studio Code
File Edit Selection View Go Debug Tasks Help
EXPLORER
OPEN EDITORS
AddBookResource_1.u... 3
AddBookResource_2.u... 3
DeleteBook.api.spec.js ... 3
EditBook.api.spec.js test 3
EditBookResource_1.un... 3
EditBookResource_2.un... 3
GetBookById.unit.spec... 3
GetBookResource_2.sp... 3
GetBooksObjectTest.ap... 3
1DV600-LAB
UseCases.md
Vision.md
node_modules
test
AddBook.api.spec.js
AddBookResource_1.uni... 3
AddBookResource_2.uni... 3
DeleteBook.api.spec.js 3
EditBook.api.spec.js 3
EditBookResource_1.unit... 3
EditBookResource_2.unit... 3
PROJECTS
GITLENS
TimeLog.md
AddBookResource_1.unit.spec.js
AddBookResource_2.unit.spec..js
D
mattias, 5 days ago • unit tests written
PROBLEMS 39
OUTPUT
DEBUG CONSOLE
TERMINAL
1: sh
Node*
39 0
mattias, 5 days ago
Ln 8, Col 18
Spaces: 2
UTF-8
CRLF
JavaScript
JavaScript Standard Style

```
var AddBookResource = require('../app/resources/AddBookResource')

describe('Add Book', function () {
  describe('Tests the add book function to see if function validated incoming object',
    // build json book to send to AddBookResource
    var book = {
      'id': '61',
      'title': null,
      'author': '',
      'genre': 'Gothic',
      'price': '10',
      'publishDate': '1794-05-08',
      'description': 'Ann Radcliffe is one of our least known and most important author
    }

    it('Returns a message indicating that the book not added due to missing fields', fu
      var callback = function (data) {
          // Check if function returns expected error message

        expect(data).to.equal('Book not added due to title missing')
        done()
      }
      AddBookResource(book, callback)
```

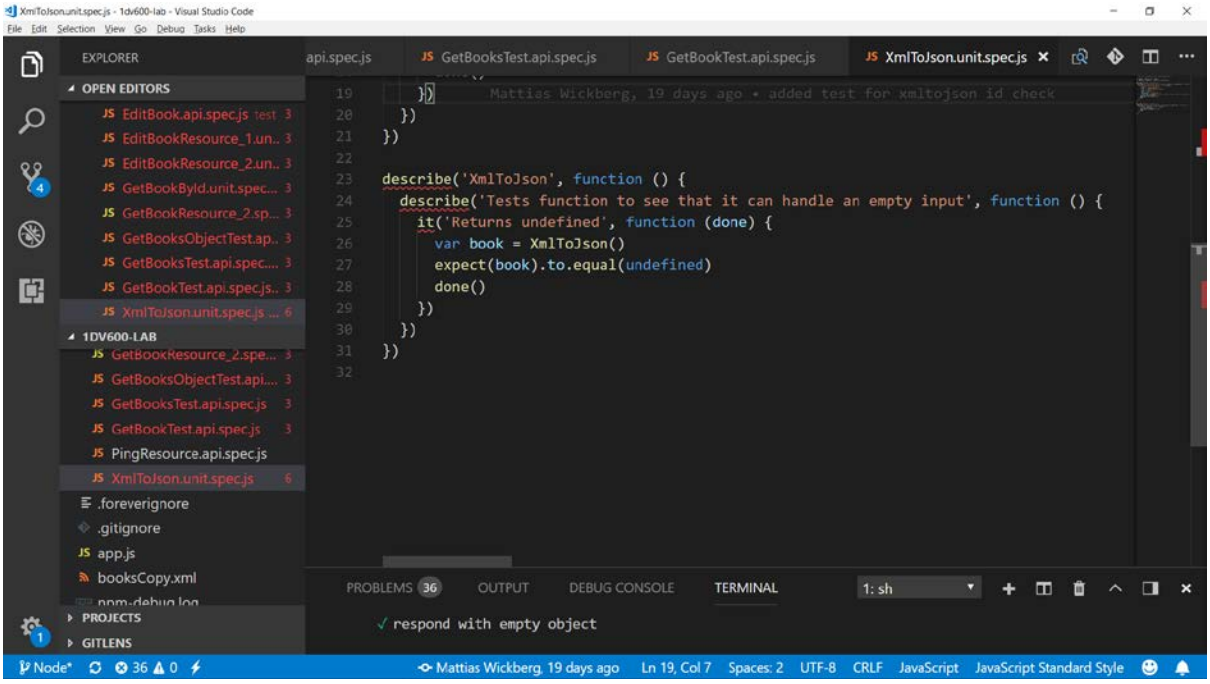
```
✓ respond with empty object
```


XmlToJson.unit.spec.js - 1dv600-lab - Visual Studio Code
File Edit Selection View Go Debug Tasks Help
EXPLORER
OPEN EDITORS
EditBook.api.spec.js test 3
EditBookResource_1.un... 3
EditBookResource_2.un... 3
GetBookById.unit.spec... 3
GetBookResource_2.sp... 3
GetBooksObjectTest.ap... 3
GetBooksTest.api.spec.... 3
GetBookTest.api.spec.js... 3
XmlToJson.unit.spec.js ... 6
1DV600-LAB
GetBookResource_2.spe... 3
GetBooksObjectTest.api.... 3
GetBooksTest.api.spec.js 3
GetBookTest.api.spec.js 3
PingResource.api.spec.js
XmlToJson.unit.spec.js 6
.foreverignore
.gitignore
app.js
booksCopy.xml
npm-debug.log
PROJECTS
GITLENS
api.spec.js
GetBooksTest.api.spec.js
GetBookTest.api.spec.js
XmlToJson.unit.spec.js
PROBLEMS 36
OUTPUT
DEBUG CONSOLE
TERMINAL
1: sh
Node* 36 0
Mattias Wickberg, 19 days ago
Ln 19, Col 7
Spaces: 2
UTF-8
CRLF
JavaScript
JavaScript Standard Style

mattias, 16 days ago | 2 authors (Mattias Wickberg and others)

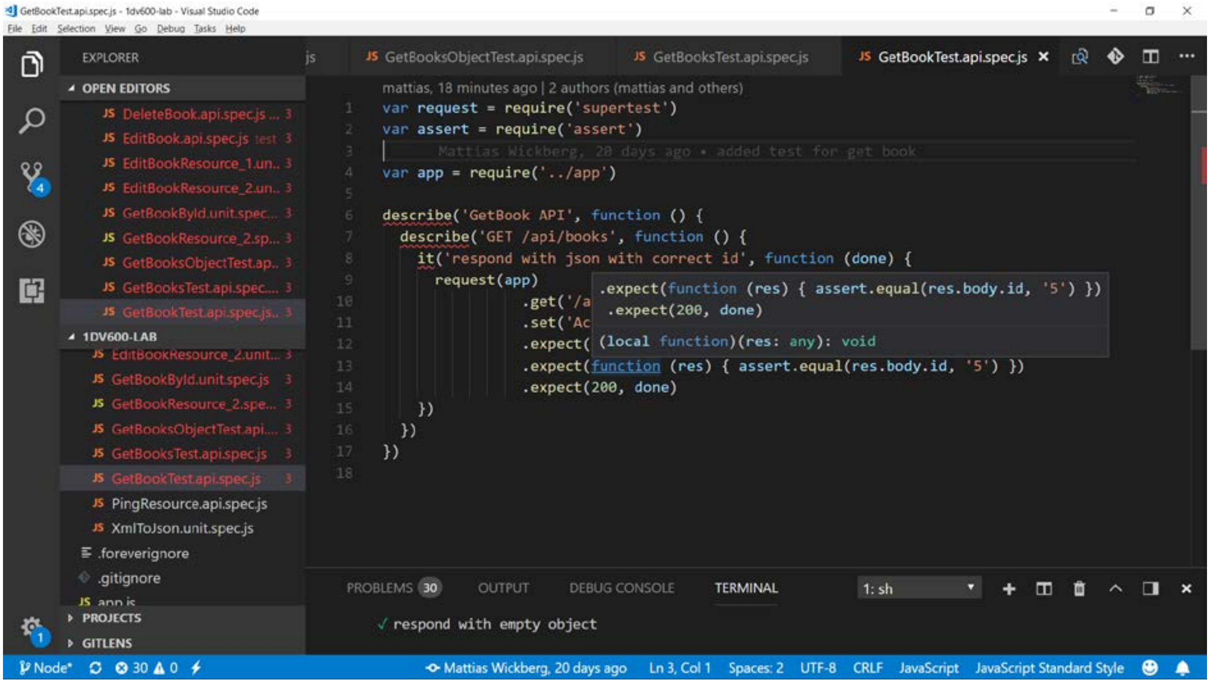
```
// Two tests: test object with ID, and test empty input

var expect = require('chai').expect
var XmlToJson = require('../app/resources/XmlToJson')

describe('XmlToJson', function () {
  describe('Tests function to see if it returns an object corresponding ID', function (
    it('Returns an object with an ID', function (done) {
      var data = { '$': { id: '101' },
        author: [ 'Stephen Hawking' ],
        title: [ 'A Brief History of Time' ],
        genre: [ 'Science' ],
        price: [ '199' ],
        publish_date: [ '1988-09-01' ],
        description: [ 'Hawking attempts to explain a range of subjects in cosmology, i
      var book = XmlToJson(data)
      expect(book.id).to.equal('101')
      done()
    })      Mattias Wickberg, 19 days ago • added test for xmltojson id check
  })
})
```

```
✓ respond with empty object
```


XmlToJson.unit.spec.js - 1dv600-lab - Visual Studio Code
File Edit Selection View Go Debug Tasks Help
EXPLORER
OPEN EDITORS
EditBook.api.spec.js test 3
EditBookResource_1.un... 3
EditBookResource_2.un... 3
GetBookById.unit.spec... 3
GetBookResource_2.sp... 3
GetBooksObjectTest.ap... 3
GetBooksTest.api.spec.... 3
GetBookTest.api.spec.js... 3
XmlToJson.unit.spec.js ... 6
1DV600-LAB
GetBookResource_2.spe... 3
GetBooksObjectTest.api.... 3
GetBooksTest.api.spec.js 3
GetBookTest.api.spec.js 3
PingResource.api.spec.js
XmlToJson.unit.spec.js 6
.foreverignore
.gitignore
app.js
booksCopy.xml
npm-debug.log
PROJECTS
GITLENS
api.spec.js
GetBooksTest.api.spec.js
GetBookTest.api.spec.js
XmlToJson.unit.spec.js
PROBLEMS 36
OUTPUT
DEBUG CONSOLE
TERMINAL
1: sh
Node* 36 0
Mattias Wickberg, 19 days ago
Ln 19, Col 7
Spaces: 2
UTF-8
CRLF
JavaScript
JavaScript Standard Style

```
    })      Mattias Wickberg, 19 days ago • added test for xmltojson id check
  })
})

describe('XmlToJson', function () {
  describe('Tests function to see that it can handle an empty input', function () {
    it('Returns undefined', function (done) {
      var book = XmlToJson()
      expect(book).to.equal(undefined)
      done()
    })
  })
})
```

```
✓ respond with empty object
```

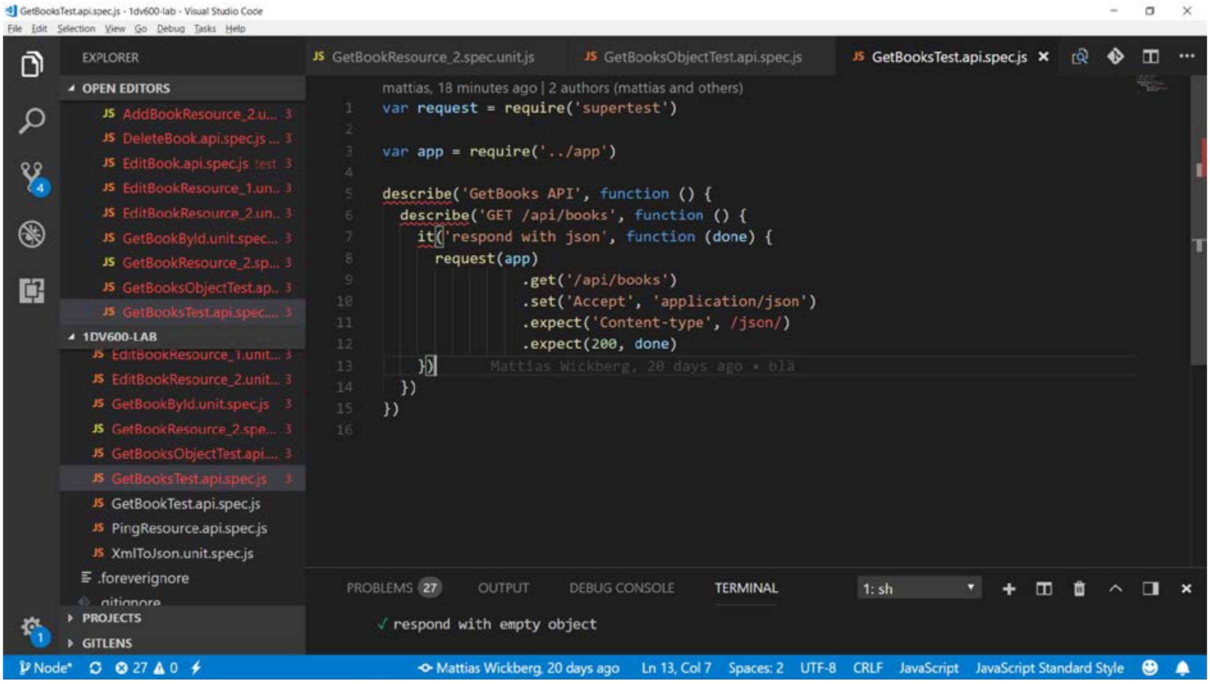


GetBookTest.api.spec.js - 1dv600-lab - Visual Studio Code
File Edit Selection View Go Debug Tasks Help
EXPLORER
OPEN EDITORS
DeleteBook.api.spec.js ... 3
EditBook.api.spec.js test 3
EditBookResource_1.un... 3
EditBookResource_2.un... 3
GetBookById.unit.spec... 3
GetBookResource_2.sp... 3
GetBooksObjectTest.ap... 3
GetBooksTest.api.spec... 3
GetBookTest.api.spec.js... 3
1DV600-LAB
EditBookResource_2.unit... 3
GetBookById.unit.spec.js 3
GetBookResource_2.spe... 3
GetBooksObjectTest.api.... 3
GetBooksTest.api.spec.js 3
GetBookTest.api.spec.js 3
PingResource.api.spec.js
XmlToJson.unit.spec.js
.foreverignore
.gitignore
app.js
PROJECTS
GITLENS
js
GetBooksObjectTest.api.spec.js
GetBooksTest.api.spec.js
GetBookTest.api.spec.js
mattias, 18 minutes ago | 2 authors (mattias and others)
PROBLEMS 30
OUTPUT
DEBUG CONSOLE
TERMINAL
1: sh
Node*
30 0
Mattias Wickberg, 20 days ago
Ln 3, Col 1
Spaces: 2
UTF-8
CRLF
JavaScript
JavaScript Standard Style

```
var request = require('supertest')
var assert = require('assert')
        Mattias Wickberg, 20 days ago • added test for get book
var app = require('../app')

describe('GetBook API', function () {
  describe('GET /api/books', function () {
    it('respond with json with correct id', function (done) {
      request(app)
              .get('/a
              .set('Ac
              .expect(
              .expect(function (res) { assert.equal(res.body.id, '5') })
              .expect(200, done)
    })
  })
})
```

```
.expect(function (res) { assert.equal(res.body.id, '5') })
 .expect(200, done)
(local function)(res: any): void
```

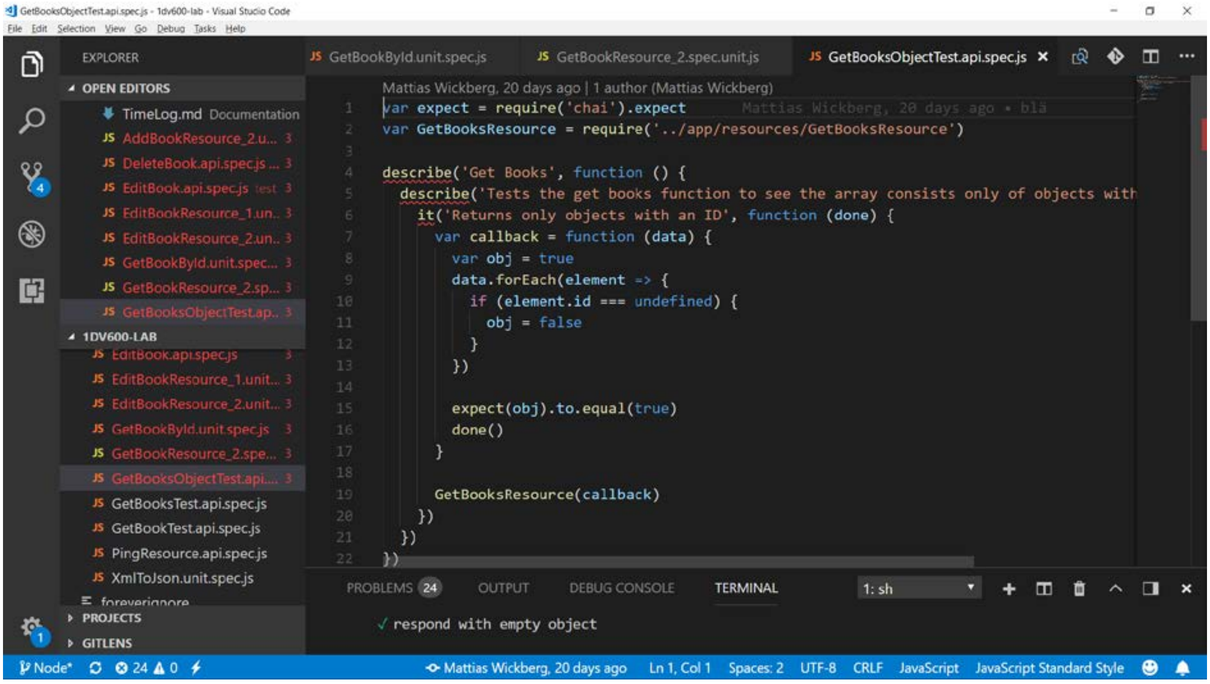
```
✓ respond with empty object
```



GetBooksTest.api.spec.js - 1dv600-lab - Visual Studio Code
File Edit Selection View Go Debug Tasks Help
EXPLORER
OPEN EDITORS
AddBookResource_2.u... 3
DeleteBook.api.spec.js ... 3
EditBook.api.spec.js test 3
EditBookResource_1.un... 3
EditBookResource_2.un... 3
GetBookById.unit.spec... 3
GetBookResource_2.sp... 3
GetBooksObjectTest.ap... 3
GetBooksTest.api.spec.... 3
1DV600-LAB
EditBookResource_1.unit... 3
EditBookResource_2.unit... 3
GetBookById.unit.spec.js 3
GetBookResource_2.spe... 3
GetBooksObjectTest.api.... 3
GetBooksTest.api.spec.js 3
GetBookTest.api.spec.js
PingResource.api.spec.js
XmlToJson.unit.spec.js
.foreverignore
.gitignore
PROJECTS
GITLENS
GetBookResource_2.spec.unit.js
GetBooksObjectTest.api.spec.js
GetBooksTest.api.spec.js
mattias, 18 minutes ago | 2 authors (mattias and others)
PROBLEMS 27
OUTPUT
DEBUG CONSOLE
TERMINAL
1: sh
Node*
27 0
Mattias Wickberg, 20 days ago
Ln 13, Col 7
Spaces: 2
UTF-8
CRLF
JavaScript
JavaScript Standard Style

```
var request = require('supertest')

var app = require('../app')

describe('GetBooks API', function () {
  describe('GET /api/books', function () {
    it('respond with json', function (done) {
      request(app)
              .get('/api/books')
              .set('Accept', 'application/json')
              .expect('Content-type', /json/)
              .expect(200, done)
    })        Mattias Wickberg, 20 days ago • blä
  })
})
```

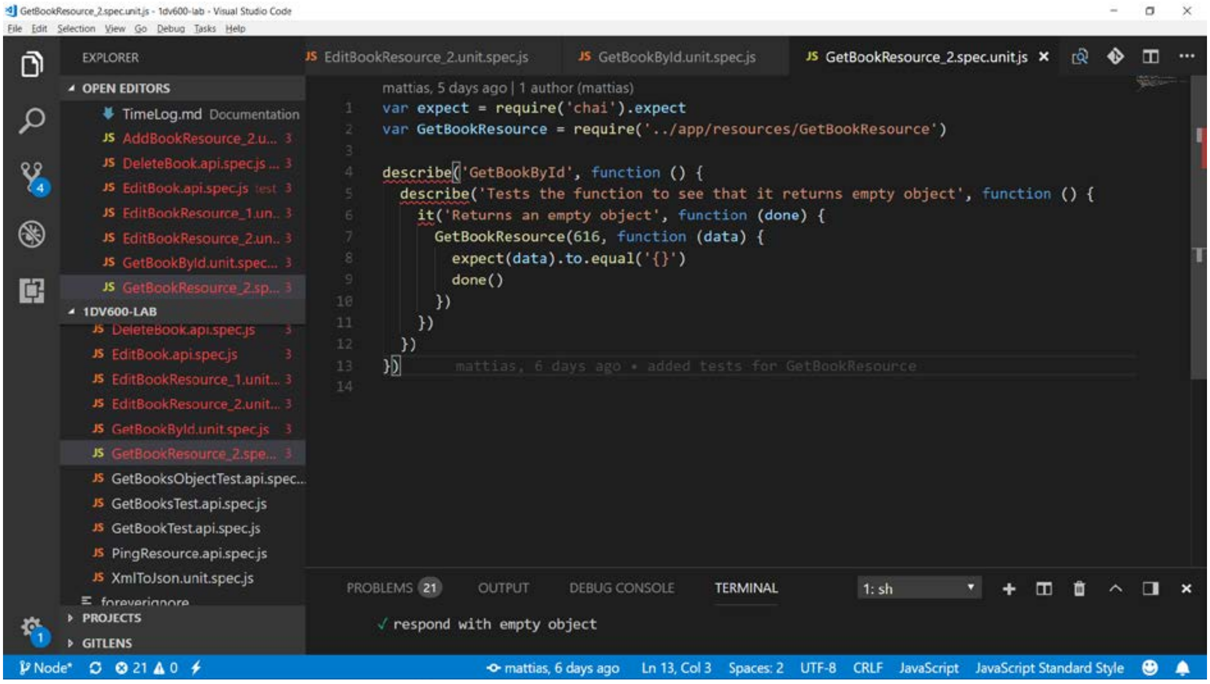
```
✓ respond with empty object
```



GetBooksObjectTest.api.spec.js - 1dv600-lab - Visual Studio Code
File Edit Selection View Go Debug Tasks Help
EXPLORER
OPEN EDITORS
TimeLog.md Documentation
AddBookResource_2.u... 3
DeleteBook.api.spec.js ... 3
EditBook.api.spec.js test 3
EditBookResource_1.un... 3
EditBookResource_2.un... 3
GetBookById.unit.spec... 3
GetBookResource_2.sp... 3
GetBooksObjectTest.ap... 3
1DV600-LAB
EditBook.api.spec.js 3
EditBookResource_1.unit... 3
EditBookResource_2.unit... 3
GetBookById.unit.spec.js 3
GetBookResource_2.spe... 3
GetBooksObjectTest.api.... 3
GetBooksTest.api.spec.js
GetBookTest.api.spec.js
PingResource.api.spec.js
XmlToJson.unit.spec.js
foreverignore
PROJECTS
GITLENS
GetBookById.unit.spec.js
GetBookResource_2.spec.unit.js
GetBooksObjectTest.api.spec.js
Mattias Wickberg, 20 days ago | 1 author (Mattias Wickberg)
Mattias Wickberg, 20 days ago • blä
PROBLEMS 24
OUTPUT
DEBUG CONSOLE
TERMINAL
1: sh
Node* 24 0
Mattias Wickberg, 20 days ago Ln 1, Col 1 Spaces: 2 UTF-8 CRLF JavaScript JavaScript Standard Style

```
var expect = require('chai').expect
var GetBooksResource = require('../app/resources/GetBooksResource')

describe('Get Books', function () {
  describe('Tests the get books function to see the array consists only of objects with
    it('Returns only objects with an ID', function (done) {
      var callback = function (data) {
        var obj = true
        data.forEach(element => {
          if (element.id === undefined) {
            obj = false
          }
        })

        expect(obj).to.equal(true)
        done()
      }

      GetBooksResource(callback)
    })
  })
})
```

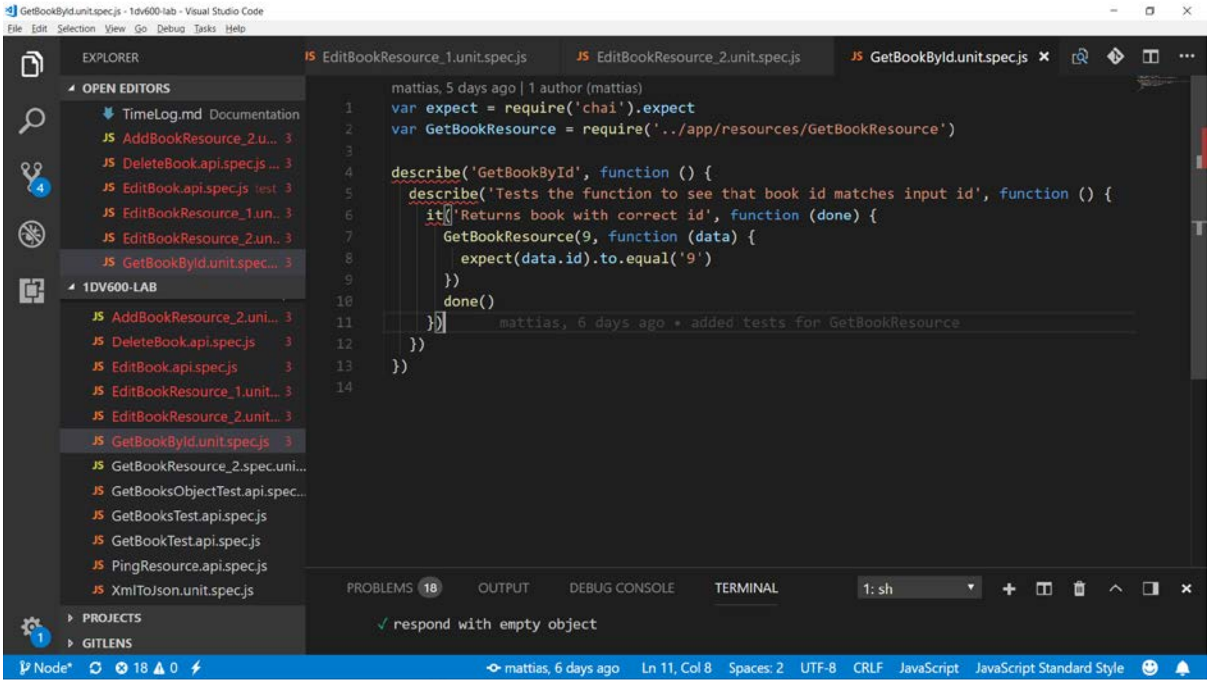
```
✓ respond with empty object
```



GetBookResource_2.spec.unit.js - 1dv600-lab - Visual Studio Code
File Edit Selection View Go Debug Tasks Help
EXPLORER
OPEN EDITORS
TimeLog.md Documentation
AddBookResource_2.u... 3
DeleteBook.api.spec.js ... 3
EditBook.api.spec.js test 3
EditBookResource_1.un... 3
EditBookResource_2.un... 3
GetBookById.unit.spec... 3
GetBookResource_2.sp... 3
1DV600-LAB
DeleteBook.api.spec.js 3
EditBook.api.spec.js 3
EditBookResource_1.unit... 3
EditBookResource_2.unit... 3
GetBookById.unit.spec.js 3
GetBookResource_2.spe... 3
GetBooksObjectTest.api.spec...
GetBooksTest.api.spec.js
GetBookTest.api.spec.js
PingResource.api.spec.js
XmlToJson.unit.spec.js
foreverignore
PROJECTS
GITLENS
EditBookResource_2.unit.spec.js
GetBookById.unit.spec.js
GetBookResource_2.spec.unit.js
mattias, 5 days ago | 1 author (mattias)
mattias, 6 days ago • added tests for GetBookResource
PROBLEMS 21
OUTPUT
DEBUG CONSOLE
TERMINAL
1: sh
Node* 21 0
mattias, 6 days ago Ln 13, Col 3 Spaces: 2 UTF-8 CRLF JavaScript JavaScript Standard Style

```
var expect = require('chai').expect
var GetBookResource = require('../app/resources/GetBookResource')

describe('GetBookById', function () {
  describe('Tests the function to see that it returns empty object', function () {
    it('Returns an empty object', function (done) {
      GetBookResource(616, function (data) {
        expect(data).to.equal('{}')
        done()
      })
    })
  })
})
```

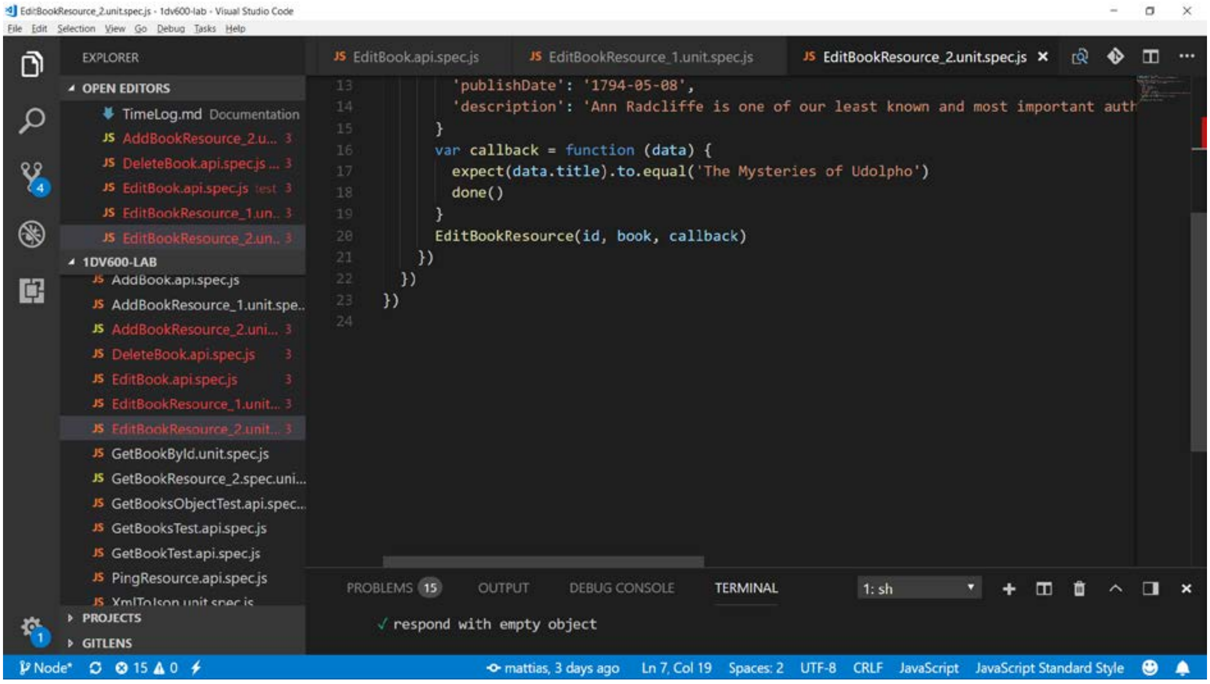
```
✓ respond with empty object
```



mattias, 5 days ago | 1 author (mattias)

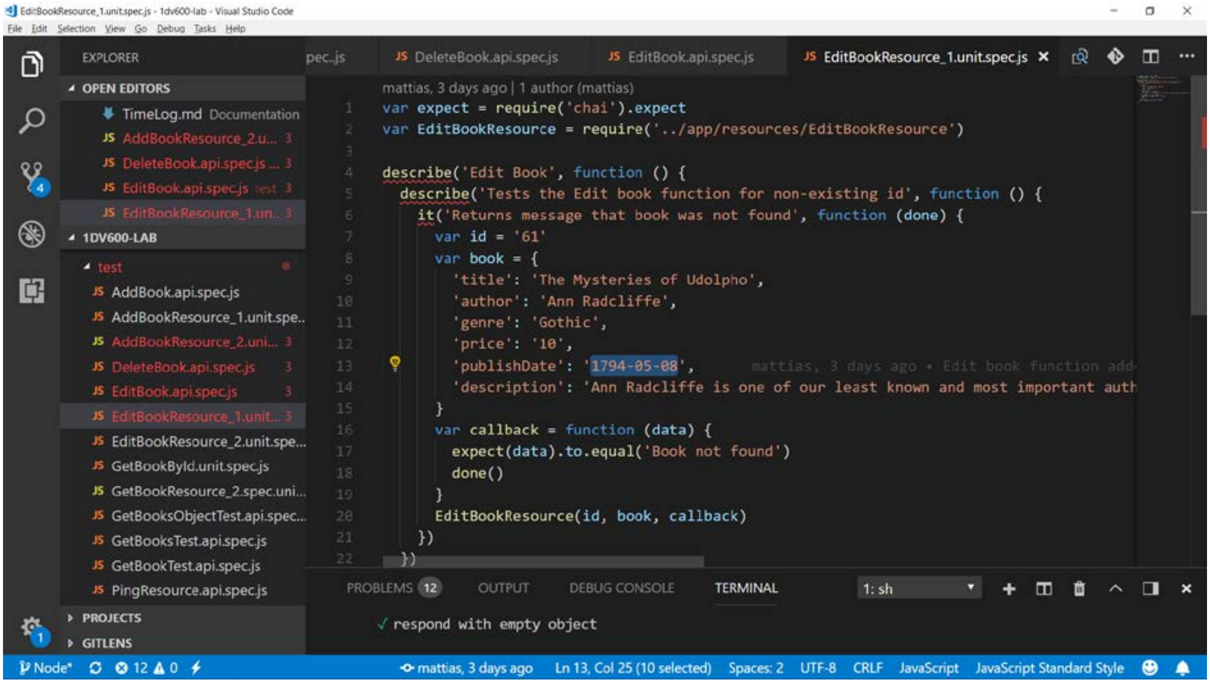
```javascript
var expect = require('chai').expect
var GetBookResource = require('../app/resources/GetBookResource')

describe('GetBookById', function () {
  describe('Tests the function to see that book id matches input id', function () {
    it('Returns book with correct id', function (done) {
      GetBookResource(9, function (data) {
        expect(data.id).to.equal('9')
      })
      done()
    })        mattias, 6 days ago • added tests for GetBookResource
  })
})
```

```
✓ respond with empty object
```



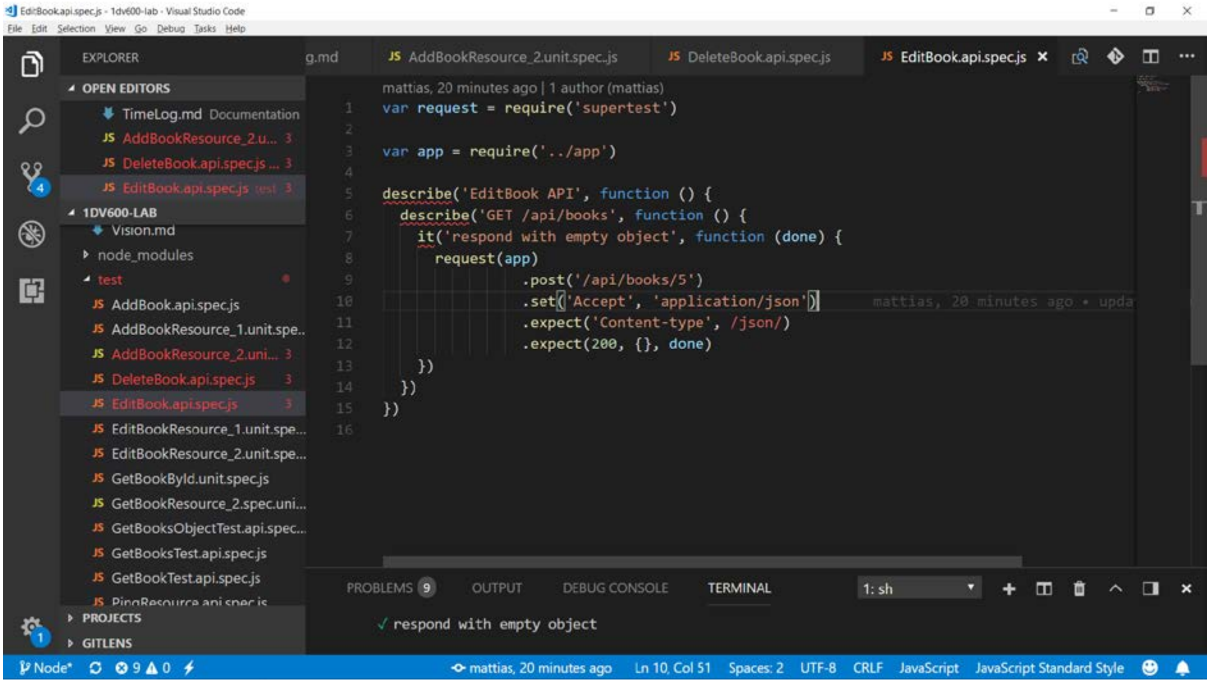
```javascript
          'publishDate': '1794-05-08',
          'description': 'Ann Radcliffe is one of our least known and most important auth
        }
        var callback = function (data) {
          expect(data.title).to.equal('The Mysteries of Udolpho')
          done()
        }
        EditBookResource(id, book, callback)
      })
    })
  })
```

```
✓ respond with empty object
```


EditBookResource_1.unit.spec.js - 1dv600-lab - Visual Studio Code
File Edit Selection View Go Debug Tasks Help
EXPLORER
OPEN EDITORS
TimeLog.md Documentation
AddBookResource_2.u... 3
DeleteBook.api.spec.js ... 3
EditBook.api.spec.js test 3
EditBookResource_1.un... 3
1DV600-LAB
test
AddBook.api.spec.js
AddBookResource_1.unit.spe...
AddBookResource_2.uni... 3
DeleteBook.api.spec.js 3
EditBook.api.spec.js 3
EditBookResource_1.unit... 3
EditBookResource_2.unit.spe...
GetBookById.unit.spec.js
GetBookResource_2.spec.uni...
GetBooksObjectTest.api.spec...
GetBooksTest.api.spec.js
GetBookTest.api.spec.js
PingResource.api.spec.js
PROJECTS
GITLENS
pec.js
DeleteBook.api.spec.js
EditBook.api.spec.js
EditBookResource_1.unit.spec.js
mattias, 3 days ago | 1 author (mattias)
PROBLEMS 12
OUTPUT
DEBUG CONSOLE
TERMINAL
1: sh
Node*
12
0
mattias, 3 days ago
Ln 13, Col 25 (10 selected)
Spaces: 2
UTF-8
CRLF
JavaScript
JavaScript Standard Style

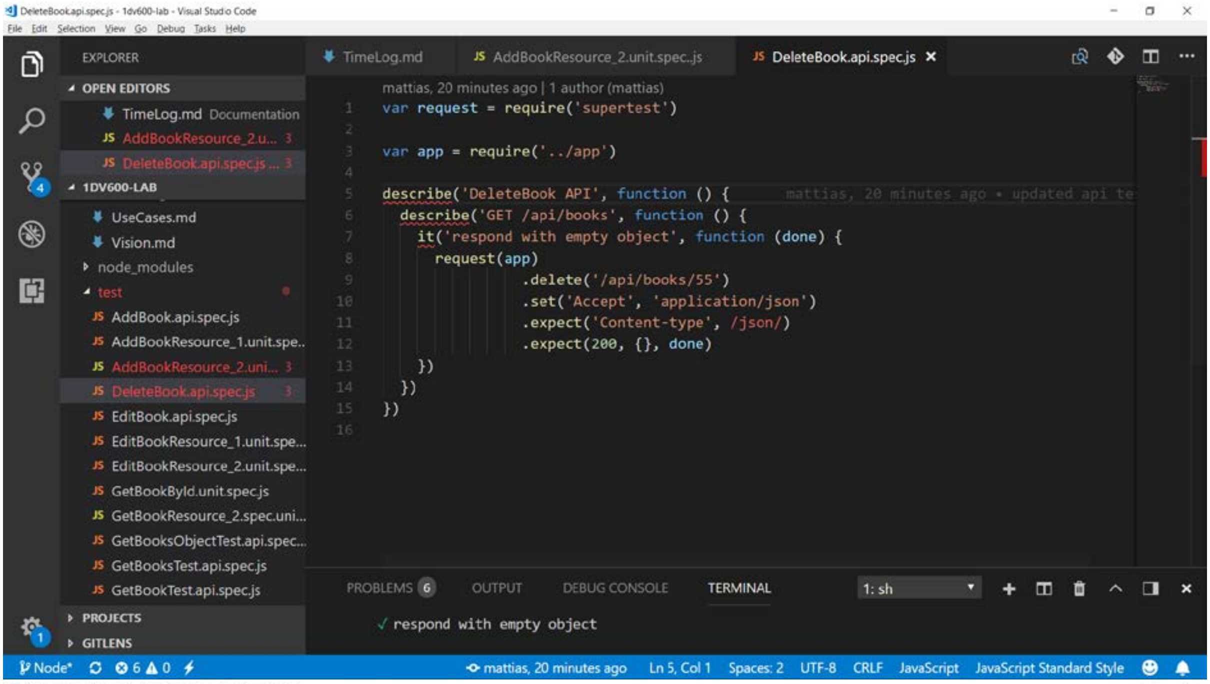
```
var expect = require('chai').expect
var EditBookResource = require('../app/resources/EditBookResource')

describe('Edit Book', function () {
  describe('Tests the Edit book function for non-existing id', function () {
    it('Returns message that book was not found', function (done) {
      var id = '61'
      var book = {
        'title': 'The Mysteries of Udolpho',
        'author': 'Ann Radcliffe',
        'genre': 'Gothic',
        'price': '10',
        'publishDate': '1794-05-08',      mattias, 3 days ago • Edit book function add
        'description': 'Ann Radcliffe is one of our least known and most important auth
      }
      var callback = function (data) {
        expect(data).to.equal('Book not found')
        done()
      }
      EditBookResource(id, book, callback)
    })
  })
```

```
✓ respond with empty object
```



```
var request = require('supertest')

var app = require('../app')

describe('EditBook API', function () {
  describe('GET /api/books', function () {
    it('respond with empty object', function (done) {
      request(app)
                .post('/api/books/5')
                .set('Accept', 'application/json')      mattias, 20 minutes ago • upda
                .expect('Content-type', /json/)
                .expect(200, {}, done)
    })
  })
})
```

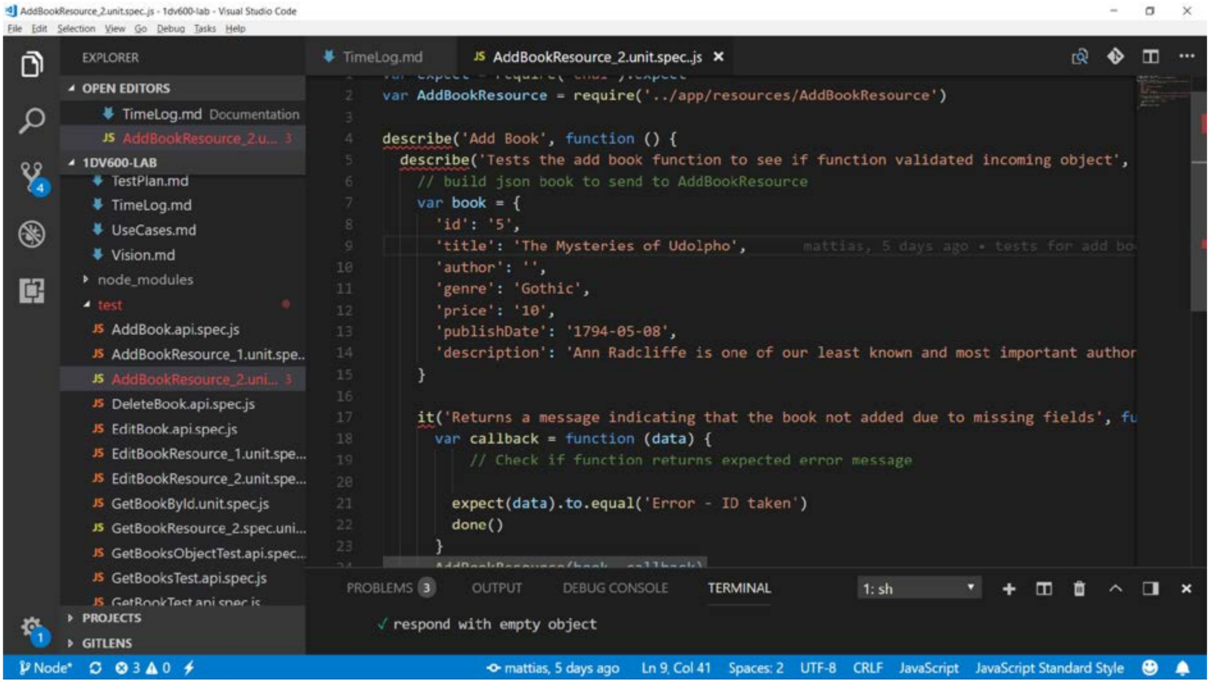
```
✓ respond with empty object
```



mattias, 20 minutes ago | 1 author (mattias)

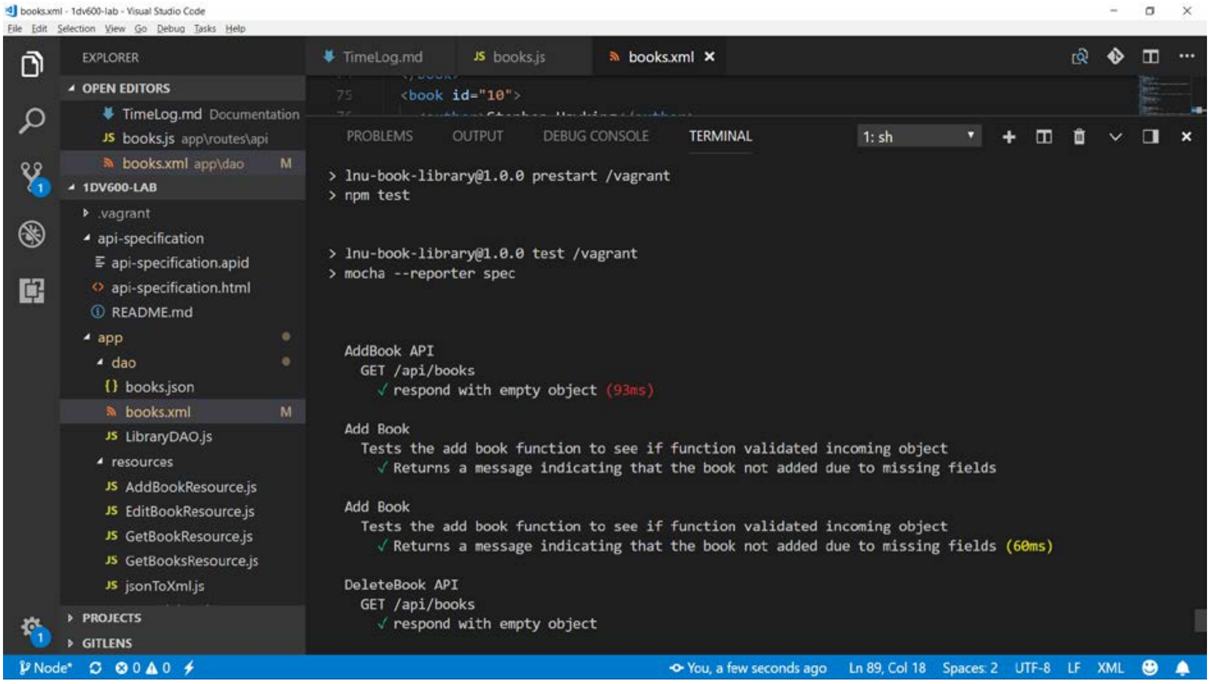
```
var request = require('supertest')

var app = require('../app')

describe('DeleteBook API', function () {       mattias, 20 minutes ago • updated api te
  describe('GET /api/books', function () {
    it('respond with empty object', function (done) {
      request(app)
                .delete('/api/books/55')
                .set('Accept', 'application/json')
                .expect('Content-type', /json/)
                .expect(200, {}, done)
    })
  })
})
```

```
✓ respond with empty object
```



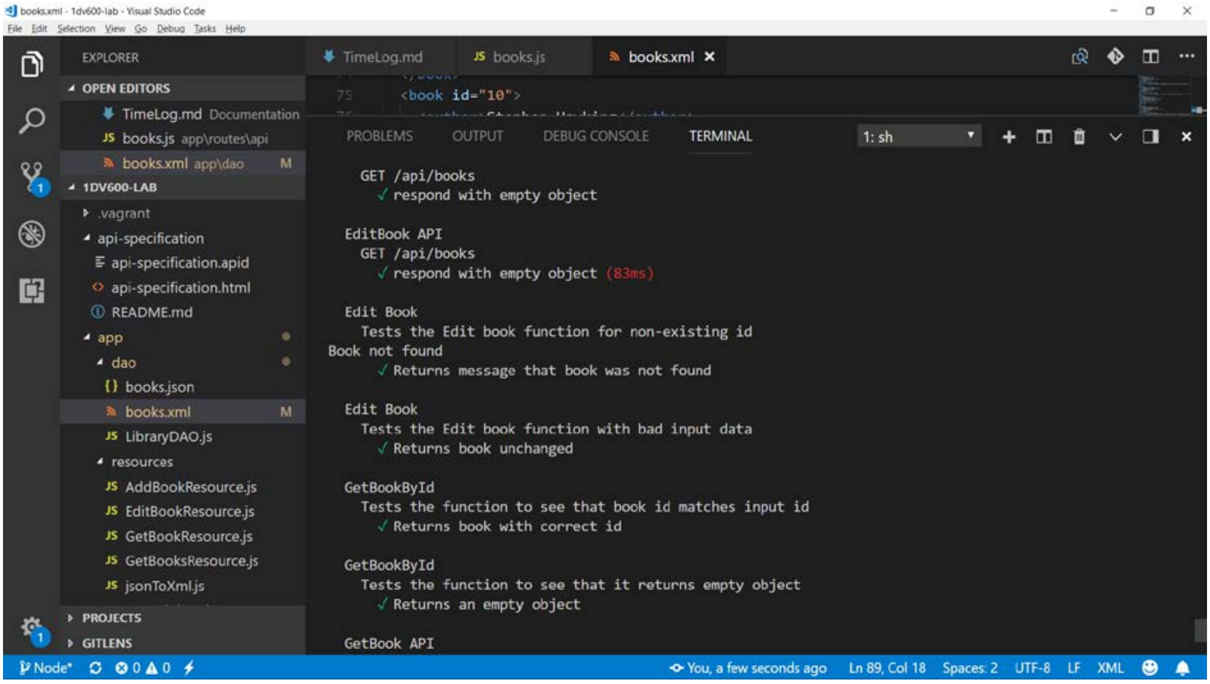
```
var expect = require('chai').expect
var AddBookResource = require('../app/resources/AddBookResource')

describe('Add Book', function () {
  describe('Tests the add book function to see if function validated incoming object',
    // build json book to send to AddBookResource
    var book = {
      'id': '5',
      'title': 'The Mysteries of Udolpho',       mattias, 5 days ago • tests for add bo
      'author': '',
      'genre': 'Gothic',
      'price': '10',
      'publishDate': '1794-05-08',
      'description': 'Ann Radcliffe is one of our least known and most important author
    }

    it('Returns a message indicating that the book not added due to missing fields', fu
      var callback = function (data) {
        // Check if function returns expected error message

        expect(data).to.equal('Error - ID taken')
        done()
      }
      AddBookResource(book, callback)
```

```
✓ respond with empty object
```



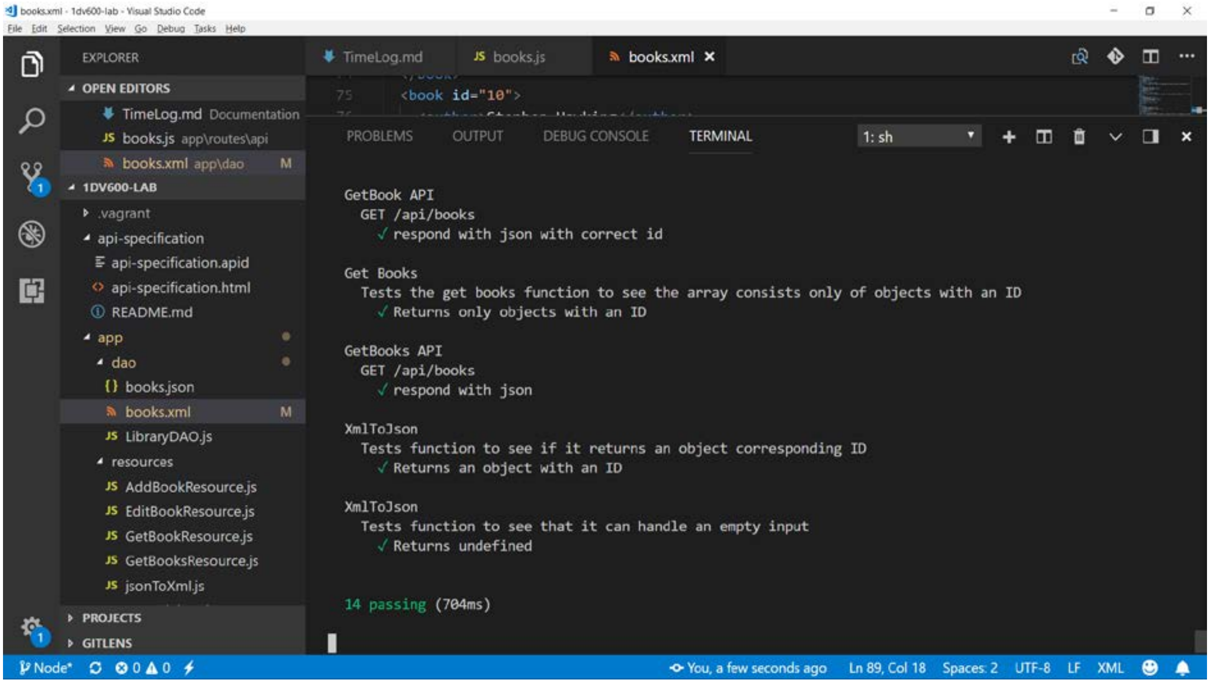
```
75      <book id="10">
```

```
> lnu-book-library@1.0.0 prestart /vagrant
> npm test


> lnu-book-library@1.0.0 test /vagrant
> mocha --reporter spec



  AddBook API
    GET /api/books
      ✓ respond with empty object (93ms)

  Add Book
    Tests the add book function to see if function validated incoming object
      ✓ Returns a message indicating that the book not added due to missing fields

  Add Book
    Tests the add book function to see if function validated incoming object
      ✓ Returns a message indicating that the book not added due to missing fields (60ms)

  DeleteBook API
    GET /api/books
      ✓ respond with empty object
```



```
75      <book id="10">
```

```
    GET /api/books
      ✓ respond with empty object

  EditBook API
    GET /api/books
      ✓ respond with empty object (83ms)

  Edit Book
    Tests the Edit book function for non-existing id
Book not found
      ✓ Returns message that book was not found

  Edit Book
    Tests the Edit book function with bad input data
      ✓ Returns book unchanged

  GetBookById
    Tests the function to see that book id matches input id
      ✓ Returns book with correct id

  GetBookById
    Tests the function to see that it returns empty object
      ✓ Returns an empty object

  GetBook API
```



```
<book id="10">
```

```
GetBook API
  GET /api/books
    ✓ respond with json with correct id

Get Books
  Tests the get books function to see the array consists only of objects with an ID
    ✓ Returns only objects with an ID

GetBooks API
  GET /api/books
    ✓ respond with json

XmlToJson
  Tests function to see if it returns an object corresponding ID
    ✓ Returns an object with an ID

XmlToJson
  Tests function to see that it can handle an empty input
    ✓ Returns undefined


14 passing (704ms)
```