**Web-Engineering 2**

Book Universe

Maxime Fritzsch, Nick Schroeder

Inhalt

[Introduction 3](#_Toc90996416)

[Business View 4](#_Toc90996417)

[Functional View 4](#_Toc90996418)

[Technical View 5](#_Toc90996419)

[Implementation View 6](#_Toc90996420)

[Security View 7](#_Toc90996421)

[Process View 8](#_Toc90996422)

[Wireframe 9](#_Toc90996423)

[Test View 10](#_Toc90996424)

[Legal View 11](#_Toc90996425)

[Stakeholder Matrix 12](#_Toc90996426)

[Work Breakdown Structure 12](#_Toc90996427)

[Effort Estimation 12](#_Toc90996428)

# Introduction

Dear Mr Amor,

This document includes the documentation for our project “Book Universe”.

For a short installation guide, please refer to our [GitHub Repository](https://github.com/nbs2904/bookUniverse), there you will also find our README file. If you decide to clone the repository on your machine you will need to include following credentials in an “.env” file. Where to exactly put them is explained by our installation guide in more detail. If you rather use our submitted zip-folder then there is no need to add this file.



# Business View

## What are the internal and external drivers?

From our perspective, there is a lack of easily accessible electronic books without the need of an additional reading device or software, which often brings costs. For physical books, there are libraries which allows people to borrow books for a very small amount of money, which makes knowledge easily available to many people. With our application, we want to do the same for electronic books, so that they are both available without an eBook reader and without spending large amounts of money, thus making knowledge available to even more people. With this, we try to fix the one flaw of physical libraries, which are fixed to one location and can’t operate globally, unlike our application. Because of this, we think there is a big market for the application, and it will be used by many people, if executed right, as not everyone has a library close by.

## What business models and processes are there?

There are multiple business models ranging from a single monthly subscription model which enables users to read every book available in the catalogue. Another possibility could be different kinds of subscription models where, depending on the model, a user can just read a limited number of books available. A third model would be to make everything free for users, but also include advertisement to finance development costs.

## Who participates in the business process?

The three parties participating would be first of all the users, with the intention to get easier access to read books online. Secondly, developers to create needed application and lastly websites or institutions that publish eBooks for free, making them available to be included in our book catalogue.

## What are the objectives of the project?

The main objective of the project is to enable people to have an easy way to read books, wherever they are. Of course, to achieve this objective, it would be good thing if we were able to generate some money to improve our application further and reach even more people.

## How is the success of the project measured?

Success of our application is measured by the number of active users. An active user is defined as a person who regularly spends time using the application to read one or multiple books.

# Functional View

## What will the finished solution do?

The finished solution is supposed to enable people to borrow and read books in their browser using our application.

## How is it used and what services does it offer?

The application lets a user choose from a catalogue of books. Once they borrowed one or multiple books they will appear in their own library. There they either can extend the duration of how long they have borrowed the book, return the book, or can read the book directly in the browser, without the need of downloading or the installation of additional software. Once they decide to close a book the current position is saved so they can start right where they stopped the next time they start reading.

## What information will it provide? For whom?

The application will provide many details regarding the books which can be borrowed. Those include crucial details for the user like the title, the language, or the page count, but also information like the ISBN, the author, or the genre(s) of the book. Additionally, once a book was borrowed, the application stores the date to which the book will have to be returned, so that the user always knows how much time is left while reading the book.

## What quality features does the solution have to offer?

It is unclear which features we will be able to include in our application till the deadline, but long term, one logical feature would for example enable the user to filter the catalogue for specific genres or authors. Additionally, it would probably help the user to decide on a book if an algorithm would suggest books based on the reading history of the user or on his/her favourite genres.

# Technical View

## How is the system structured and constructed?

The application consists of two main components, the back- and frontend. The purpose of the backend, an express server, is to connect to the used MongoDB database and handle incoming request from the frontend/user. The frontend displays information in the browser which it received from the backend server. The framework used for the frontend is angular.

## Which interfaces and boundary conditions are there?

One of the boundary conditions set during the development of the first version of the application is that it should just handle books in the epub format instead of handling multiple eBook formats or something else readable like magazines.

Secondly, for the first initial version of the application no algorithm will be implemented to analyse users reading behaviour to customise recommendations.

Lastly, a book rating system is planned to be implemented, but will still be in the pipeline during the first phase of deployment.

## What applications and data are needed?

eBooks in the .epub format are needed to display books and make it possible for user to read directly in the browser. Additionally, a book cover in png-format and meta information of every book will be needed in order to give users a short outline of the book making it easier for them whether to read a book. Furthermore, the user of course needs to provide some data (email, password, …) when creating an account.

## What will the infrastructure look like?

The infrastructure of the application will be based on the MEAN-Stack, using a MongoDB Database to store data, an Express Server will be implemented as an API, Angular as frontend framework will be used and Nodejs as the runtime environment.

## What standards are set?

A predefined code-style will be enforced using ESlint and the available plugin for typescript. The directory structure should not deviate from the standard angular folder structure, to prevent problems during the build process.

## How are the quality requirements achieved?

As mentioned above ESlint will be implemented right at the beginning to enforce code structure and style. To prevent any incorrect code to be published to the used GitHub Repository a hook is to be implemented using the package “husky” to check every staged file for flaws. Additionally, every now and then both teammates will check each other’s work to ensure quality.

# Implementation View

## Which products and components (from which manufacturer) are required for the system?

The main components consist of the open-source framework Angular and open-source runtime environment node.js. Additionally, publicly available eBooks represent an important part.

## How is the system developed and rolled out?

During development the editor Visual-Studio Code will be mainly relied on since it offers an extension called “live-share” to enable developers to code at the same time or share ports of a localhost in one session. Additionally, GitHub is used to keep different versions of the code.

A cloud service will eventually be needed to roll out the created web application, which can be connected to the used GitHub Repository. Therefore, after any fix or update the build will automatically be recreated.

The global rollout will not be immediately, as we will only slowly expand into other regions after offering the application only in Germany after the first rollout. This has to be done because the rights to the books offered are held by different companies in different countries, which makes a lot of negotiations necessary that are time intensive.

## What verification methods are used?

To verify a user, the data entered during login is compared with the data saved in the database. In our case, this data is the email and the password hash. If a user in the database can be found with an identical password hash and email, the verification was successful, and the user is logged in.

## Who pays what?

The application is supposed to be financially self-sufficient as we will try to monetize it by using a subscription model. This subscription model will be structured into several tiers, which determine how much freedom the user has when choosing books to borrow, with the highest tier unlocking the whole catalogue. Besides this subscription model, we are not planning on using additional ways of monetization like showing adds or selling user data, but of course this all depends on how the money earned from the paying users can make up for the resources we need to keep the application running.

# Security View

As security measure in the frontend an angular guard was implemented. This guard checks the localStorage for specific values. Once a user logged in the value “isAuthenticated” is set to true and the userId is stored as well. If one of those values do not exist or is empty, then the user will be routed back to the login page. Additionally, the userId’s length must be of 24 characters.

To implement a higher level of security one could let the guard check the database, if given userId actually exists, every time the user is routed to another site. But due to performance reasons this solution was not implemented.

User passwords are stored as hashes, using the MD5 Hash Algorithm. Therefore, making passwords useless if they will ever be leaked from the database. We chose this algorithm because we don’t need longer hashes for the number of users we expect.

The profile information a user can actually manipulate themselves is checked for being valid before sending a request to the backend and a second time just before pushing any changes to the database to prevent any loss of data or malicious data manipulation.

When it comes to dynamic routes, values, for example a bookId, is checked if it exists before trying to receive data from the database. Therefore, preventing errors either in the backend or preventing user from actively manipulating routes to display something they should not have access to.

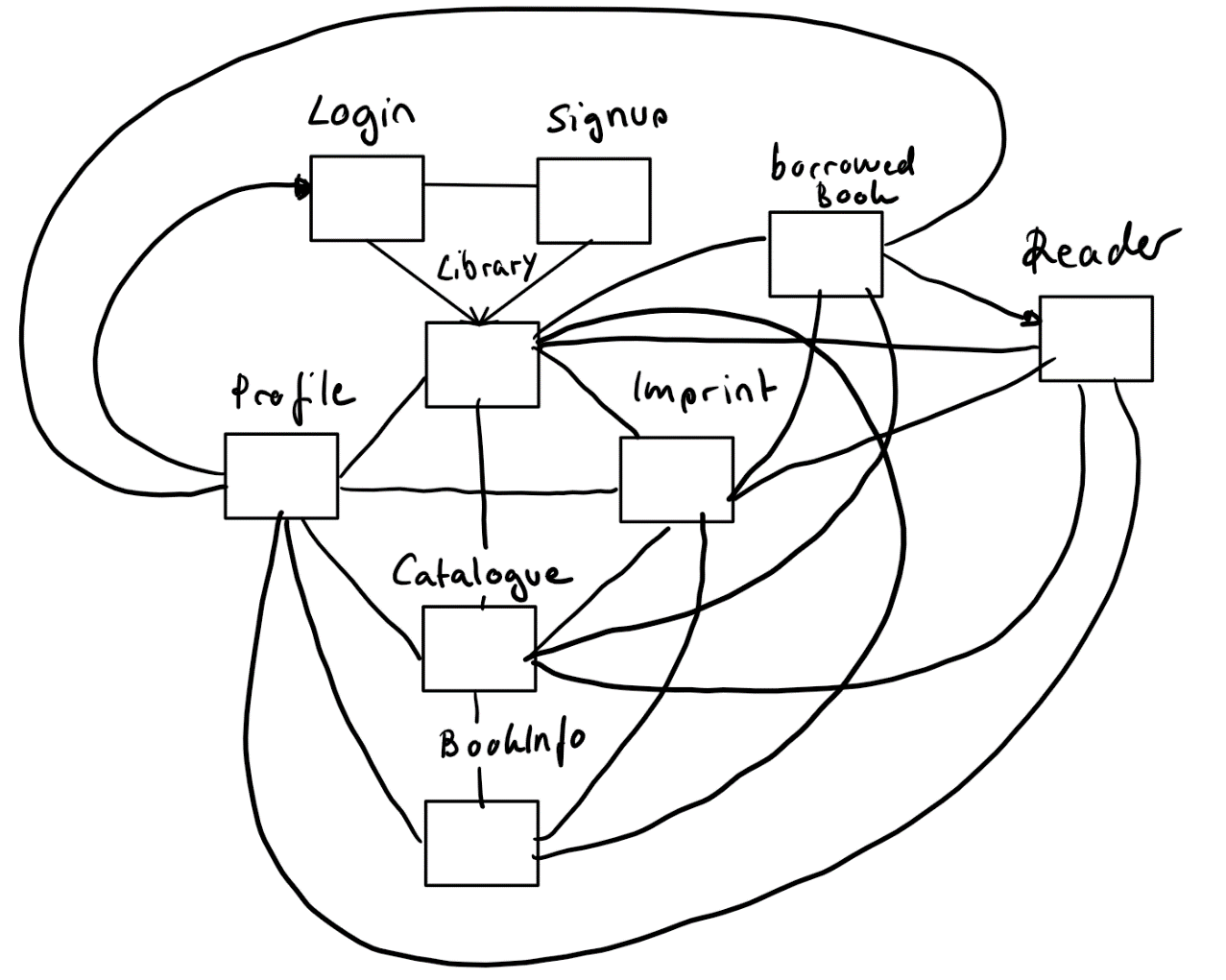
# Process View

|  |  |
| --- | --- |
| Login Process | Borrowing Process |
|  |  |

# Wireframe

|  |  |
| --- | --- |
| Login | User Profile |
|  |  |
| Reader | Signup Form |
|  |  |
| Book Info Page | Borrowed Book Info |
|  |  |
| Catalogue | Imprint |
|  |  |
| Detail view libary |  |
|  |  |

Site View



# Test View

## Which tests should be planned?

Not only unit tests should nearly cover 100% of the written code but e2e tests are also to be implemented in order to test the application from a user perspective.

Testing single methods in order of making sure edge cases are being handled properly. Additionally, functions should be mocked in order to specifically test a single function.

## When are the tests performed?

The tests should be co-developed with each component, reducing the risk of accumulating a larger number of bugs in the code. The tests will be performed every time before building the application and also every time after a sizable new feature was implemented.

## Which data should be entered?

Only data, which fits to a predefined scheme, making sure there will not be any type of unpredicted cases.

# Legal View

In general, one important legal aspect for our application are the rights for the books which can be borrowed. In our first version, which we will use for testing our application, only books which are part of the public domain will be used to not infringe on any copyright.

Long term, it will be necessary for us to negotiate with the publishers and authors connected to the books we want to offer with our application. We must do that so it will be possible for us to grow in user numbers, as the number of books within the public domain is limited. Thus, we need secure the rights for the books, even though we probably need to enter some kind of revenue share system with the owners of the book rights.

Optimally, we would be able to make deals for multiple countries, but unfortunately, book rights are often sold to different publishers in different countries, so it is not unlikely we will have to negotiate several times for the same book to offer it worldwide.

We would of course also need to register Book Universe as a company and secure a domain. This would be possible for the domains we would want to use as most of them are not in use and the rights are available. The exception for this would be bookuniverse.org, but fortunately this is not an important domain for us so that we wouldn’t really need to negotiate with the current owners of the domain.

|  |  |
| --- | --- |
| Domain | Availability |
| book-universe.com | yes |
| bookuniverse.com | yes |
| book-universe.de | yes |
| bookuniverse.de | yes |
| book-universe.org | yes |
| bookuniverse.org | no |

# Stakeholder Matrix

Concern

Influence

developer

user

publisher/  
authors

# Work Breakdown Structure

# Effort Estimation

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Frontend | Time | Backend | Time | Documentation | Time |
| Implement different views | 12 h | Database structure | 45 min | Comment functions | 30 min |
| Configure e2e test | 2 h | Configure unit tests | 3 h | READMe File | 30 min |
| Configure build automation | 1 h | API routes | 1,5 h | API YAML Definition | 1,5 h |
| Implement router | 30 min | Configure controllers | 4 h |  |  |
| Implement guard | 1 h |  |  |  |  |