INTERNATIONAL INFORMATION TECHNOLOGIES UNIVERSITY  
FACULTY OF INFORMATION TECHNOLOGY  
DEPARTMENT OF COMPUTER ENGINEERING AND TELECOMMUNICATIONS

TSP PROJECT

**BookStore**

MAJOR 5B070400 – COMPUTER SYSTEMS AND SOFTWARE ENGINEERING

|  |  |  |  |
| --- | --- | --- | --- |
|  | **STUDENTS:** | **AksakalovBeksultan**  **Abdazimov Begzod**  **Utemisov Ersultan**  (group, student name) | **\_\_\_\_\_\_\_\_\_\_\_\_\_**  (signatures) |
|  | **SUPERVISOR:** | **senior lecturer**  **Bolshibayeva A.K.**  (position, name) | **\_\_\_\_\_\_\_\_\_\_\_\_\_**  (signature) |

Almaty 2018

Contents

Description of the program

Scope

Business objectives and success criteria

Requirements

Functional requirements

Non – functional requirements

Competitive analysis

Development cycle

Launch

Appendix A

Goals

Abilities of each team member

Goal of each cycle

Distribution of roles among team members

Strategy

Main strategy

Alternative strategy

Conceptual design

Risk management

Reuse strategy

Planning

General schedule for project

Gantt chart for general planning

Development planning stage

Design

Design for reuse

Design for usability

Wireframe

Web application architecture

Cycle 1

Planning

Requirements

Functional requirements

Non functional requirements

Design review and Inspections

Design

Code review

Code review prerequisites

Some things to look for in html

Some things to look for in CSS

Browser support

Desktop browser

Testing

System testing

Acceptance testing

Alpha testing

Unit testing

Postmortem

Cycle 2

Planning

Requirements

Functional requirements

Non functional requirements

Design review and Inspections

Design

Code review

Code review prerequisites

Some things to look for in html

Some things to look for in CSS

Browser support

Desktop browser

Testing

System testing

Acceptance testing

Alpha testing

Unit testing

Postmortem

Cycle 3

Planning

Requirements

Functional requirements

Non functional requirements

Design review and Inspections

Design

Code review

Code review prerequisites

Some things to look for in html

Some things to look for in CSS

Browser support

Desktop browser

Testing

Unit testing

Integrated testing

System testing

Usability testing

Postmortem

Heuristics

Appendix B and making commitments

Trello

Reference

# Description of the website

The business-to-consumer aspect of electronic commerce (e-commerce) is the most visible business use of the World Wide Web. The primary goal of an e-commerce site is to sell goods and services online.

This project deals with developing an e-commerce website for Online Book Sale. It provides the user with a catalog of different books available for purchase in the store. In order to facilitate online purchase a shopping cart is provided to the user.

This is a project with the objective to develop a basic website where a consumer is provided with a shopping cart application and also to know about the technologies used to develop such an application.

And in our Lab we used HTML and CSS markup languages to create our work. Lab consists of 8 pages:

* Mainpage
* Contacts
* About us
* Sign up
* FAQ
* New coming
* Best sellers
* Cart

# Scope

## **Business Objectives and Success Criteria:**

* To create a web application that supposed to keep track of project’s tasks.
* To create a web application that allows tracking user’s tasks in one location.
* Provide necessary tools for project’s success in one web application.

## **Requirements**

### Functional Requirements

* User should be able to see their cart/orders
* User should be able to see their order history
* User should be able to see their favorite goods
* User should be able to see new comings
* User should be able to manage their carts:
  1. Should be able to Add an order
  2. Should be able to Delete an order
  3. Should be able to Buy more than 1 item of one type (multiply)
* User should be able to search books by search field
* User should be able to see categories of books
* User should be able to register and login

### Non-Functional Requirements

* Web site should show correctly
* Web site should support all type of media querries:
  + Small devices (Tablets, 768px and up)
  + Medium devices (Desktops, 992px and up)
  + Large devices (Large Desktops, 1200px and up)
* Web site should be color-designed.
* Web siteshould be updated time by time
* Web siteshould give information in English
* Web siteshould support database connection.

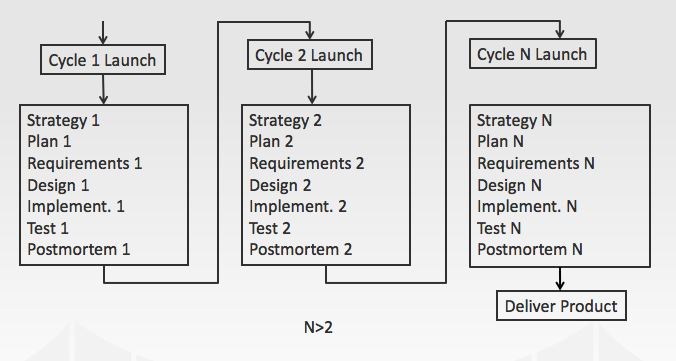
## **Competitive analysis**

There are a lot of book store web sites that offer great usability and suitable functions.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Attributes | Our BookStore | | | Flip.kz | | | Meloman.kz | | |
| Weight | Rating(1-5) | Value | Weight | Rating (1-5) | Value | Weight | Rating (1-5) | Value |
| Market share | 0.2 | 4 | 1.00 | 0.2 | 5 | 0.2 | 0.2 | 5 | 1.00 |
| Market growth | 0.2 | 5 | 1.00 | 0.15 | 4 | 0.6 | 0.2 | 5 | 1.00 |
| Quality | 0.15 | 5 | 0.6 | 0.15 | 4 | 0.6 | 0.15 | 4 | 0.60 |
| Brand equity | 0.15 | 5 | 0.75 | 0.06 | 3 | 0.10 | 0.15 | 4 | 0.60 |
| Distribution system | 0.15 | 4 | 0.60 | 0.15 | 5 | 0.15 | 0.15 | 5 | 0.30 |
| Product cost | 0.06 | 3 | 0.15 | 0.06 | 3 | 0.15 | 0.06 | 4 | 0.15 |
| Promotional effectiveness | 0.06 | 4 | 0.10 | 0.06 | 2 | 0.10 | 0.06 | 3 | 0.15 |
| Productive capacity | 0.06 | 3 | 0.15 | 0.2 | 4 | 0.80 | 0.06 | 3 | 0.10 |
| Total weight | 1.0 |  |  | 1.0 |  |  | 1.0 |  |  |

## **Developments cycles**

* Launch
* Strategy
* Plan
* Requirements
* Design
* Implementation
* Test
* Postmortem



# Launch

Appendix A

**Goals**

**Goal 1**: Develop a quality product

**Measure**: Number of defects that founds during the testing

**Goal 2**: Do disciplined personal work

**Measure 1**: 100% data recorded in TFS

**Measure 2**: 100% of forms completed

**Begzod**: Team Lead, Planning manager, Quality/Process Manager, Test manager.

**Goal3:** Role of each member, create requirements, plan, process and product monitor, task organizer.

**Beksultan**: Development manager, Support manager, Test manager.

**Goal 4**: Team product foreman, technical “scrounger”.

## **Abilities of each team member**:

Team Leader:

* Provide direction to a team for achieving a qualitative result.
* Solve the problems and motivate the team

Development Manager:

* Produce a quantity and a qualitative product
* Use all the team skills on the product

Planning Manager:

* Construct convenient, concrete plan for all team members and all work
* Make a schedule for all work

Quality/Process Manager:

* All team members accurately report data
* All team inspections are properly moderated and reported

Support Manager:

* The team has suitable tools
* No unauthorized changes to baseline
* All risks and issues recorded

**Entry criteria**: some name book

**Exit criteria**: finished web site, which shows books of particular genre and gives opportunity to buy books, delete order cart, to log in to the system.

**Goal of each cycle**

**1-cycle:** Create and implement design of web application "Bookstore". It has to follow main UI/UX standards.

**2-cycle**: To make a layout of the site this means part of the front-end.

**3-cycle**: Back-end. Create all functionalities. Create an admin panel so that he can manage the site. Make overview of project and deploy it.

**Distribution of roles among team members**

Time size – 2 people

**Team lead** – primary contact for team – Begzod

**Development manager** – team product foreman - Beksultan

**Planning** **manager** – task organizer – Begzod

**Quality**/**process** **manager** – process and product monitor - Ersultan

**Support** **manager** – technical “scrounger” - Beksultan

**Test** **manager** – Beksultan, Begzod,Ersultan

# Strategy

**M1, 2, 3, 4, 5, 6, 7, 8, 9 – Meetings**

**C1, 2, 3- Cycles**

## Main Strategy

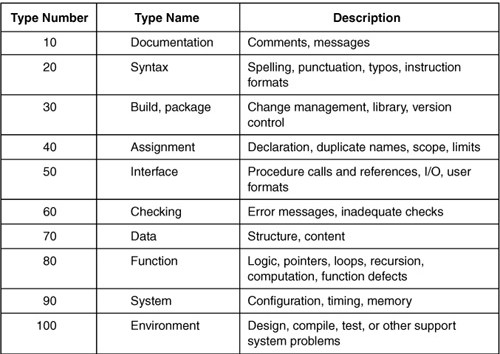
|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Taskname** | **Task owner** | **Start time** |
| M1, M2 | Role of each member, create requirements, plan | Begzod | 16.02 |
|  | Search necessary tools, install | All | 28.02 |
|  | Create main and alternative strategies | Beksultan | 28.03 |
|  | Managing process by trello (To do list) | Begzod | 2.03 |
| C1 | Create design of Web site | All | 10.03 |
|  | Implement the design | Begzod | 12.03 |
| M3, M4 | Discussion of planning of Cycle-2 | All | 16.03 |
| C2 | Front-End | Begzod | 21.04 |
| M6, M7 | Creating Report | All | 24.04 |
| C3 | Implement following functions | Beksultan | 26.04 |
| M8, M9 | Presentation of product | All | 29.04 |
|  | Testing product for defects | Ersultan | 29.04 |
|  | Unplanned tasks | All | 29.04 |
|  |  |  |  |

## **Conceptual Design**

Bookstore program's going to be implemented using markup languages for part of front-end: **Html, JavaScript, CSS**. And, for back-end: **PHP** language. Books in our web store can view by price. Bookstore is color-coded for example, color of book that ordered before will mark with another color. The next goal of the project is improving the functionality of web site. These functionalities are a: searching a book, add several books in cart.

Components of Web Design

* **Colors:**  
  The site’s colors should convey your company’s personality or brand.  Colors have a big emotional factor so it’s important to choose colors your audience will love
* **Fonts:**  
  The site’s fonts should also be based on the company’s brand and the feeling you want to convey to customers.  The fonts need to be easy to read.
* **Pictures & Graphics:**  
  Choose pictures and graphics that display your company and products in the best light possible.  Hire a professional photographer if possible.  Images speak volumes about your company and play a key role in your website’s look and feel.
* **Complexity:**  
  Strive for simplicity in your design.  Keep it simple, yet not so simple that it lacks appeal.
* **Usability:**  
  Make sure the website’s design is user-friendly.  The navigation needs to be straightforward and the design needs to be created based on how users will interact with the site.
* **Clarity:**  
  Make sure all images and graphics on the site are sharp.
* **Consistency:**  
  Keep the design of the site (colors, fonts, button styles, heading sizes, etc.) consistent throughout the site.
* **Defect standards**.



**Table-7.1.** The PSP Defect Type Standard

## **Risk Management**

### internet connection problem

Probability: High

Impact: High

User need internet connection to use your application, if there are some problems with internet, user won't use bookstore.

Correction: Executing process group (without analysts) have to introduce new function to keep last update before receiving internet connection in the 1st stage.

### Components or products aren’t maintainable

Probability: High

Impact: Low

Technology components, tools or platforms that is difficult to maintain (e.g. rare skills, complex or experimental).

Correction: Choose framework which is widespread, which big community and solutions for common problems.

### Project team misunderstand requirements

Probability: Low

Impact: Medium

When requirements are misinterpreted by the project team a gap develops between expectations, requirements and work packages.

Correction: Giving right explanation of requirements to each team member.

### Resource shortfalls

Probability: Low

Impact: High

Inability to secure sufficient resources for the project.

Correction: Support Manager should afford materials and control each version of project cycle.

### PERSONAL

Probability: Low

Impact: High

There is may be problems with human’s health, problem with personal life, with family, etc.

Correction: Provide person who can replace them.

## **Reuse Strategy**

While getting ready to begin a project, the team realizes that there are existing components that can be reused.

**Reuse strategy for requirements:**

* Create main requirements that refer to another requirement of cycle.
* A team strategically designs components so that they'll be reusable in future projects.

**Reuse strategy for coding process:**

* During the coding reuse one method to overwrite
* The code contains a reference to reused code, and thus they have distinct life cycles and can have distinct versions.
* The code should contain a local or private copy of the reused code, and thus need to share a single life cycle and a single version.

**Types of reuse strategies that used in application:** software libraries, design patterns, frameworks.

**TSPi Strategy Form: Form STRAT**

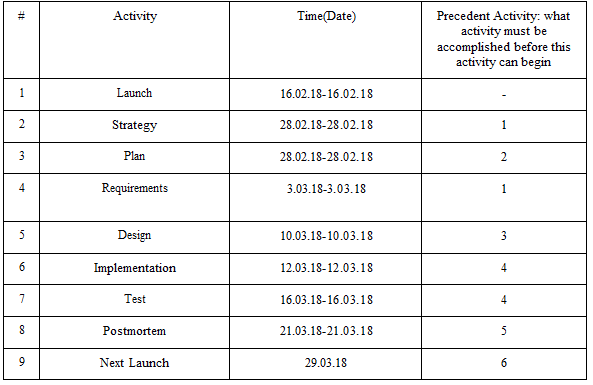
**Name** Aksakalov Beksultan, Abdazimov Begzod,Utemisov Ersultan

Cycle 1: here I was doing the design part. In this part no lines of code, because it is just we did it drawing mock up and prototype.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Functions | Cycle LOC | | | | Cycle Hours | | |
| 1 | 2 | | 3 | 1 | 2 | 3 |
| Front-end (10 pages) |  | 5000 |  | |  | 100hours |  |
| Login page (back-end) |  | 500 | 200 | |  | 25hours | 10hours |
| Registration page (back-end) |  | 400 | 100 | |  | 20hours | 5hours |
| Search (back-end) |  | 200 | 300 | |  | 10 hours | 12hours |
| Categories and select by price (back –end) |  | 250 | 100 | |  | 9hours | 7hours |
| Add books to cart(back -end) |  | 469 | 50 | |  | 20 hours | 2hours |
| Admin panel (back end) |  | 1000 | 500 | |  | 30hours | 12hours |
|  |  | | | |  | | |

# Planning

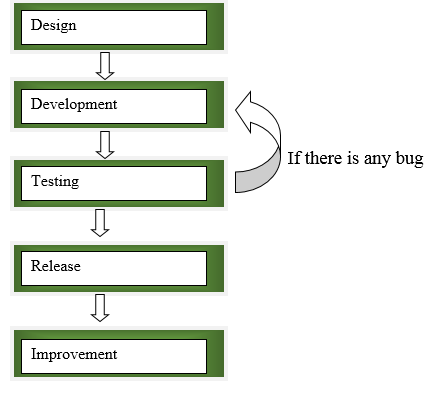
## **General Schedule for project**

****

## **Gantt chart for general planning**

****

## **Development planning stage**

****

# Design

## Designing for Reuse

Before the creating of design of web application.

* Use the technology tools that have been made in previous projects.
* Use this site for drawing <https://moqups.com/> moqups.
* Use this site [Mockflow.com](https://vk.com/away.php?to=http%3A%2F%2FMockflow.com&cc_key=) for wireframe.
* Use components of Web Design

## Designing for Usability

One way to make products usable is to produce scenarios for every key user function. Then analyze these scenarios and make sure that they reflect the kind of system you believe the users will want. When you are not sure how any function should work, either review the relevant scenarios with an application expert or build and demonstrate a simple prototype. And we decided create wireframe.

## Wireframe

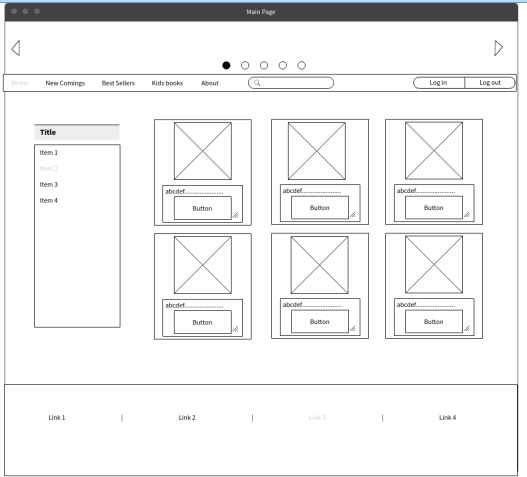
****

Figure 1.0 Main page

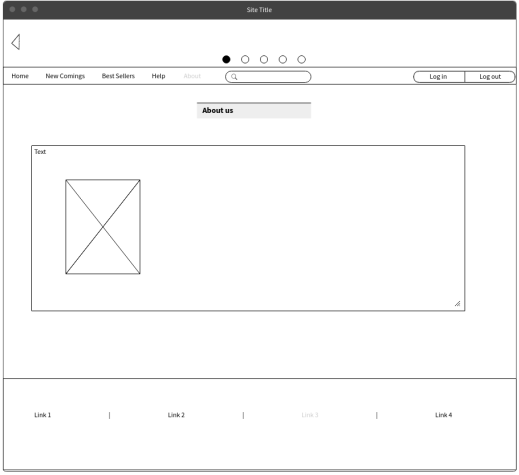


Figure 1.1 About us

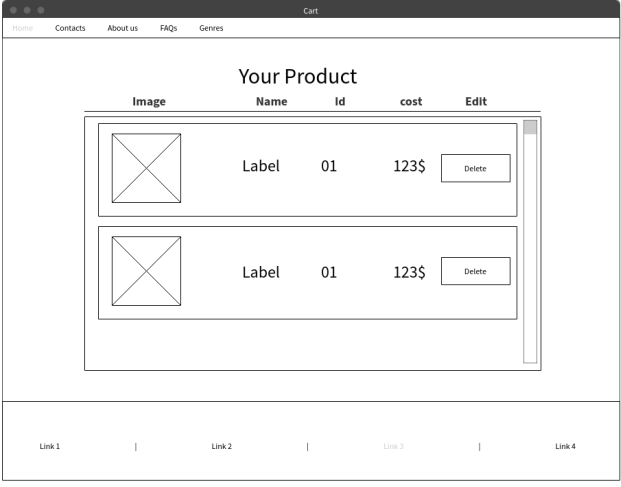


Figure 1.2 Your Product

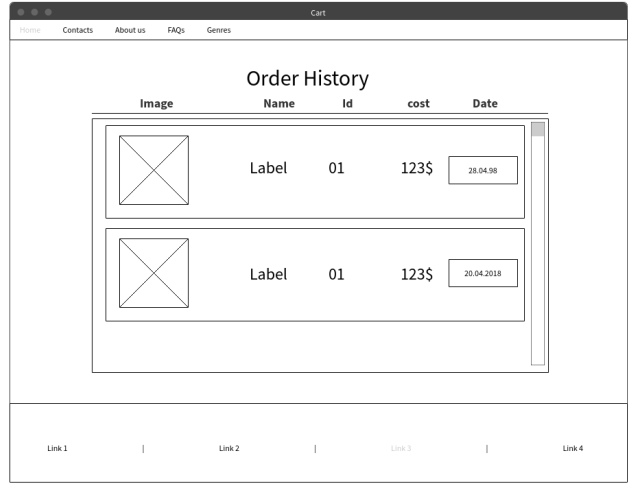


Figure 1.3 Order history

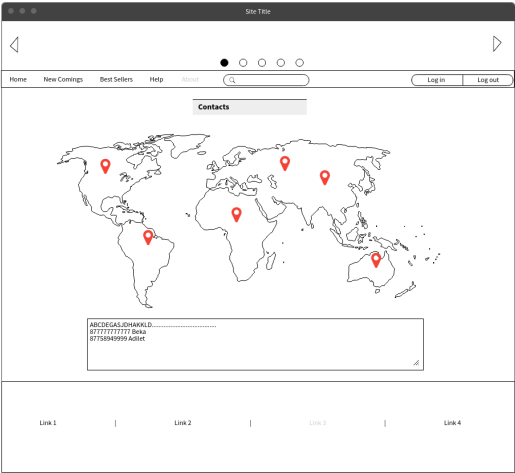


Figure 1.4 Contacts

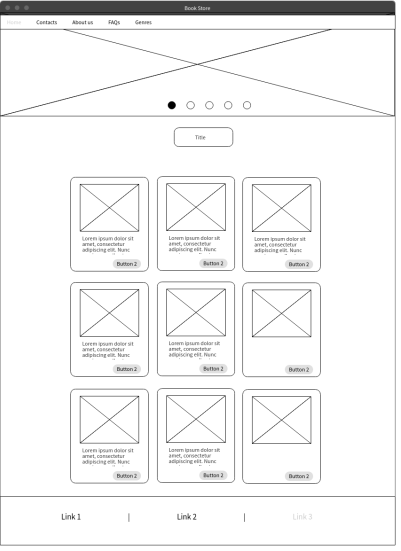


Figure 1.5 Best sellers

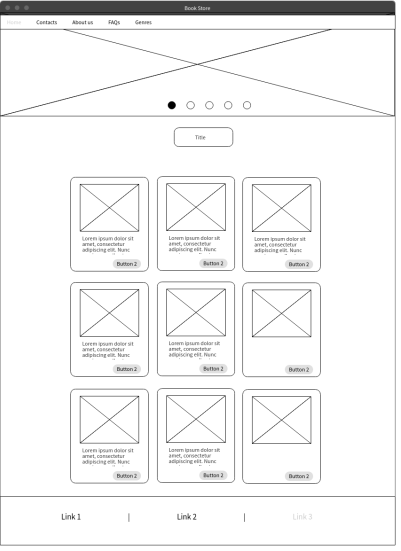
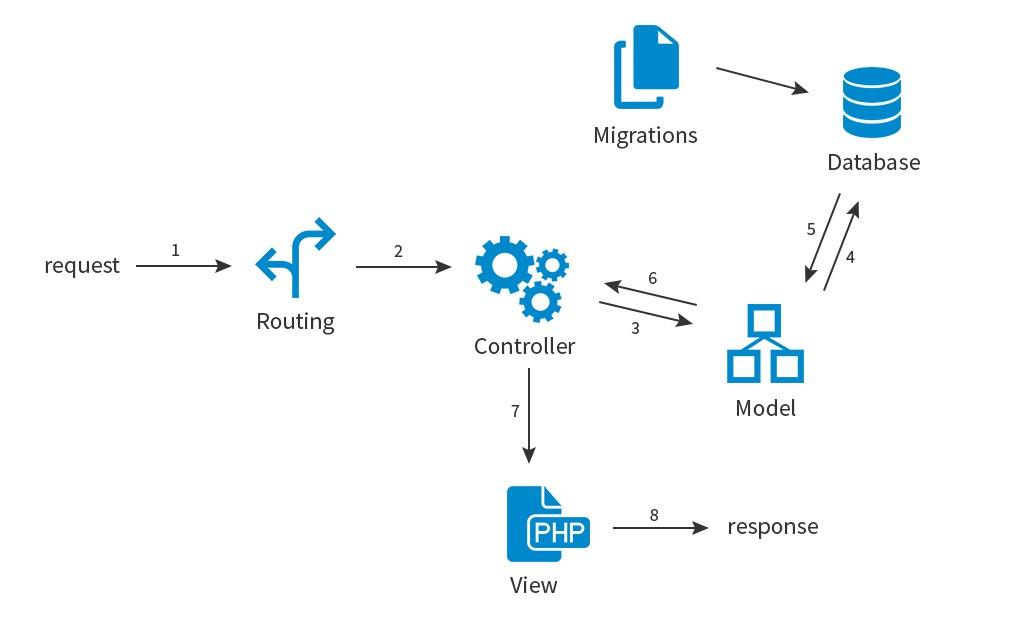


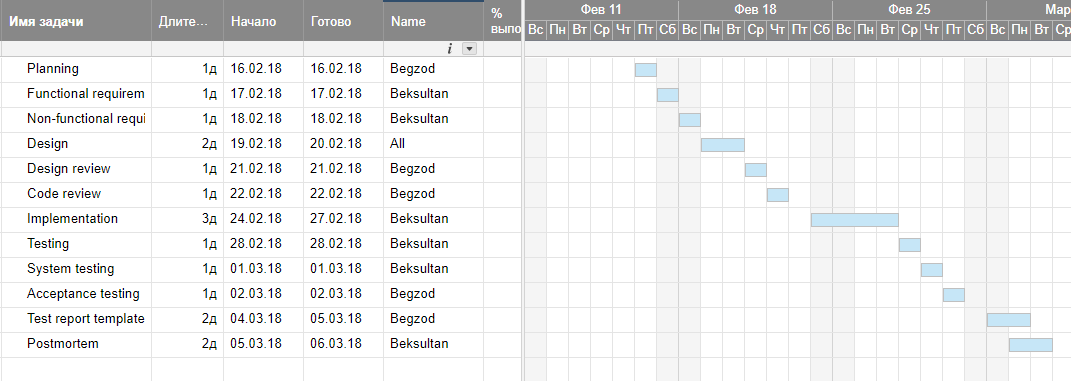
Figure 1.6 New coming

## **Web application Architecture**



# Cycle #1

## **Planning**

****

## **Requirements**

### Functional requirements

* User should be to see all pages
* User should be to see all buttons
* User should be to see all categories of books

### Non-Functional Requirements

* Web application should support all type of platforms (Mac Os, Windows, Linux, etc.).
* Web application should be color-coded.

## **Design Review and Inspections**

Reviewers: Begzod

Date: 06.04.2018

|  |  |
| --- | --- |
| Is the design feasible from a technology, cost, and schedule standpoint? | + |
| Does the design support proceed to the next development step? | + |
| Does the design use standard techniques and avoid exotic, hard-to-understand elements? | + |
| Is the design unjustifiably complex? | - |
| Is the design lean (i.e., are all of its parts strictly necessary)? | + |
| Will the design be easy to port to another environment if appropriate? | + |
| Does the design allow for scalability? | + |
| Does the design address all issues from the requirements of cycle-1-2-3? | + |
| Does the design follow all standards necessary for the system? (i.e., datestandards) | + |

## **Design**

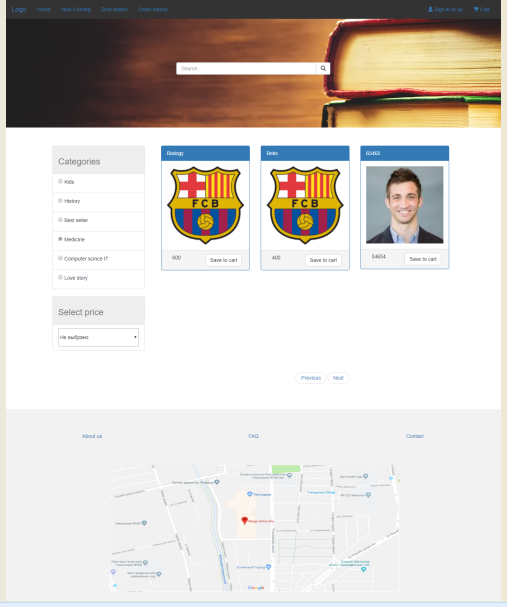
****

Figure 1.7 Main

## **Code review**

Date: 06.04.2018

Presenters: Beksultan, Begzod

### Code review prerequisites

* Thorough and meaningful code reviews take time. We need to scope for code review just like any other task. For that to happen, everyone must believe in its purpose and importance — that means management and product managers must actively encourage code reviews and give developers the necessary time.
* The team has agreed on naming convention, styling, and coding standards, whether it is **HTML, CSS, or JS**. Styling standards give the code a consistent and professional look. A naming convention makes it easier to understand, search, build, and maintain. Coding standards make the code cleaner and easier to read, build, maintain, and debug. There are many popular standards out there. It is less important which one you choose, but it is crucial that the entire team follows one standard. If a rule is controversial within the team, discuss and decide, then change the standard and apply the new rule across the entire code base. Consistency is key.
* The team has agreed on what to review — from spaces and camel Case, to where business logic should be, to whether the new code is violating established architecture.

### Some things to look for in HTML

* Does the code follow agreed-upon standards and conventions?
* Does the markup use the same element ID name more than once? If so, change it to a class because each ID should be unique. If a reusable module contains an element ID, the ID is no longer unique once the module is reused in the same page. Remember that getting an element by ID returns only the first element that matches, which may cause issues when you have multiple elements with the same ID name.
* Are there unnecessary wrapper elements? If a<div>does not have a class, or the<div>has a class but the class has no associated styles, chances are that it is not needed. Remove it and check the layout. If the layout does not change, you do not need that wrapper.
* If the app uses a template language that allows complicated logic, are you over-using it, thus making the markup hard to read? Consider applying the logic to the view model BEFORE the view model is passed to the template.
* Does the markup contain the proper [ARIA](https://en.wikipedia.org/wiki/WAI-ARIA) attributes for accessibility?

### Some things to look for in CSS

* Does the code follow agreed-upon standards and conventions?
* If you are using a CSS precompiler, such as [LESS](http://lesscss.org/) or [SASS](http://sass-lang.com/), are you nesting classes (or even worse, nesting IDs) because it looks more organized? Unnecessary nesting makes overriding CSS difficult; it also hurts performance and maintainability.
* Is CSS in modules namespaced to avoid name clashing? Consider a top-level wrapper class with the same name as the module. If you don’t want class nesting, consider using [BEM convention](https://css-tricks.com/bem-101/).
* Are some class names so generic that name clashes are likely? This can happen with either other modules in the same app or with other libraries that you may use.
* Are regularly used CSS rules (or blocks of rules) repeated so often that they should become mixins?
* If the app uses a CSS library, such as [Bootstrap](http://getbootstrap.com/) or [Skin](https://ebay.github.io/skin), are you maximizing its use? Does your app redefine mixins available in the library, or worse, does your app have blocks of CSS where using a single class from the library would do?
* Arepaddingandmarginused correctly? Ispaddingused whenmarginshould have been, or vice versa?
* Does usingpercentageinstead ofpixelmake sense? How aboutremvs.px?
* Are there duplicate rules for the same selector? Remove the duplicates.
* Are the same rules found for different selectors in the same module? Merge them.
* Are the same rules found across different modules? Put them in a shared CSS file.
* Are there too many nested classes? Are they necessary for override? Try removing some nesting. If the layout doesn’t change, they are not needed.
* Are IDs used for CSS? IDs are HIGHLY [specific](https://specificity.keegan.st/) and need other IDs to override them. And if you must insist on using IDs for CSS, DO NOT nest IDs. First, nested IDs require the same number of nested IDs or more to override, which makes your CSS longer and more complicated. Second, and this relates to fundamental CSS principle — IDs are by definition unique. If an ID needs to be nested to work, it’s not unique. It should not be an ID. Make it a class instead.
* If the code uses media queries, see the “Media queries” section in this [post](https://www.ebayinc.com/stories/blogs/tech/mobile-first-a-retrospective/).

### Browser support

One difficulty in front-end development is getting a web app to work well across devices, operating systems, and browsers. With the large number of device, operating system, and browser combinations in the wild, including the old and the native, working well everywhere is an impossible task for developers. The reasonable approach is to get your app’s traffic breakdown and test the most popular combinations during development.

To review layout and browser quirks, an in-person code review is the most effective as the reviewer can play with the feature/fix while reviewing the code. (A very simple exercise, which is also very effective in finding layout bugs, is resizing the browser window.) If an in-person review is not feasible, attach screenshots of the app to the PR, showing the layouts in different forms:

* Desktop browser
* Tablet browser (both portrait and landscape mode)
* Phone browser (both portrait and landscape mode)

## **Testing**

### System testing

JavaScript

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Name | Status | Type | Initiator | Size | Time |
| Jquery.min.js | 200 | script | Index.php | From memory cache | 0ms |
| Bootstrap.min.js | 200 | script | Index.php | From memory cache | 0ms |
| Common.js | 200 | script | Index.php | From memory cache | 0ms |
| Util.js | 200 | script |  | From disk cache | 4ms |
| Stats.js | 200 | script |  | From disk cache | 9ms |

CSS

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Name | Status | Type | Initiator | Size | Time |
| Bootstrap.min.css | 200 | stylesheet | Index.php | From disk cache | 14ms |
| style.css | 200 | stylesheet | Index.php | From disk cache | 13ms |

Status:

**200** it is good(Ok)

**304** not modified

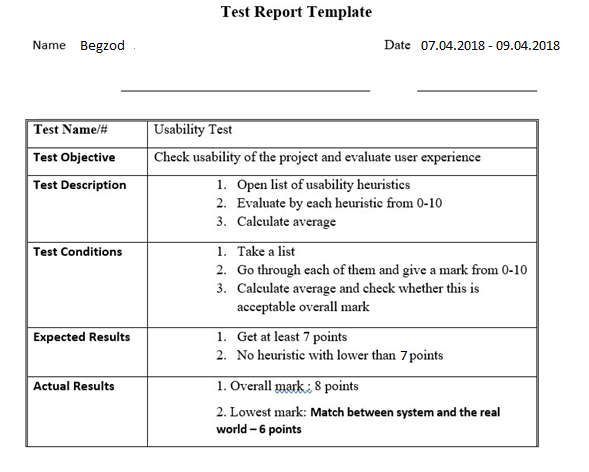
**400** bad requests

**404** not found

### Acceptance testing

During this testing we conducted Alpha testing

### Alpha testing

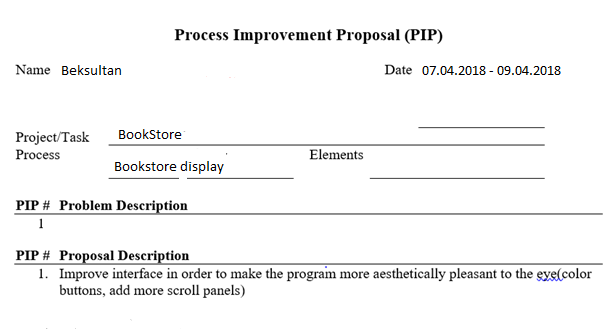
****

### Unit Testing

We did Unit testing for 1-cycle of our web application to be sure that our functions ae working correctly. We found all our bugs and corrected. It was useful when we updated our web application, it means when we included new functionalities. These testing like JavaScript Unit testing, server-side unit testing.

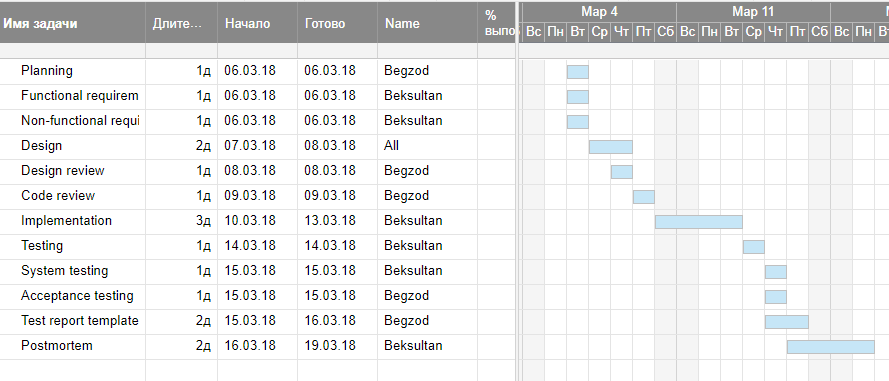
## Postmortem

### Process Improvement Proposal (PIP)



# Cycle#2

## Planning

****

## **Requirements**

### Functional requirements

* User should be to see all pages
* User should be to see all buttons
* User should be to see all categories of books
* User should be to see all functionalities

### NON-Functional requirements

* Web application should support all type of platforms (Mac Os, Windows, Linux, etc.).

## **Design Review and Inspections**

Reviewers: Begzod

Date: 12.04.2017

|  |  |
| --- | --- |
| Is the design feasible from a technology, cost, and schedule standpoint? | + |
| Does the design support proceeding to the next development step? | + |
| Does the design use standard techniques and avoid exotic, hard-to-understand elements? | + |
| Is the design unjustifiably complex? | - |
| Is the design lean (i.e., are all of its parts strictly necessary)? | + |
| Will the design be easy to port to another environment if appropriate? | + |
| Does the design allow for scalability? | + |
| Does the design address all issues from the requirements of cycle-1? | + |
| Does the design follow all standards necessary for the system? (i.e., datestandards) | + |

## Design

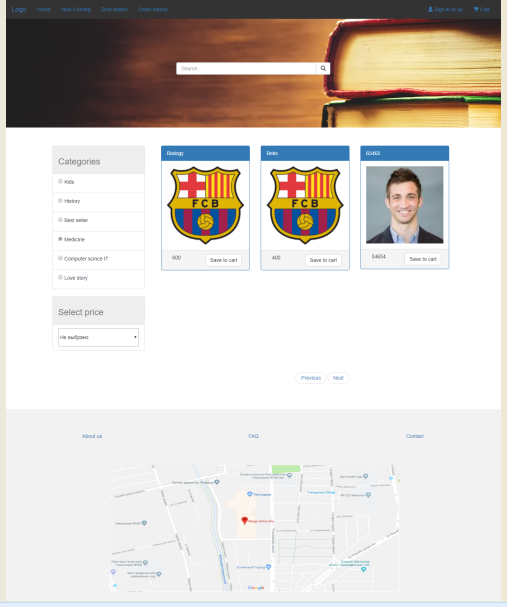
****

Figure 1.7.1 Main

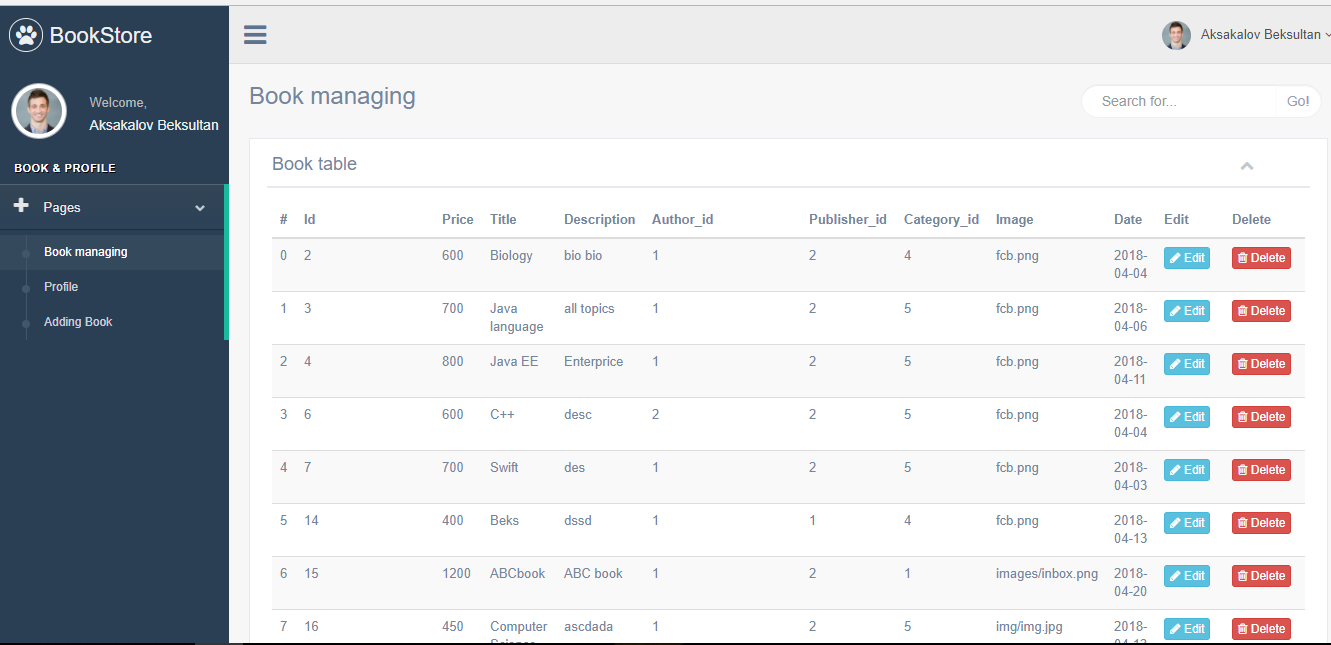


Figure 1.8 Admin panel.

## Code review

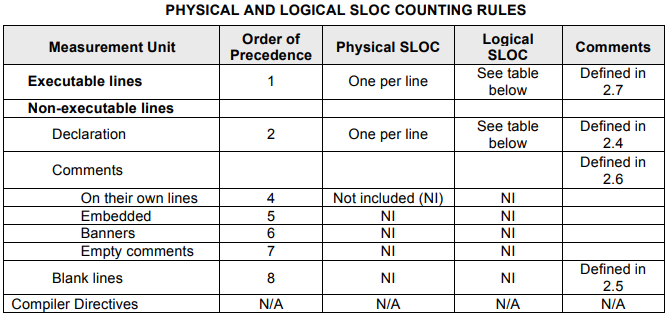
Date: 13.04.2017

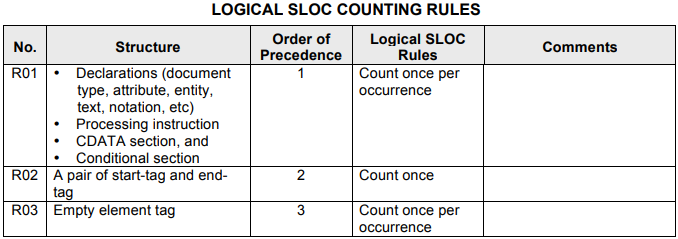
Presenters: Beksultan, Begzod

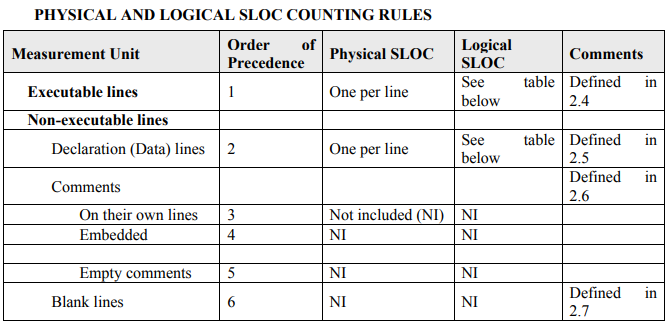
* Code works without errors
* The code is clean and understandable
* Code follows to coding standards and best practices
* Code is not duplicated
* Necessary lines of code are commented
* Code is divided into small functions for better understanding
* Function and variable names explain what they do

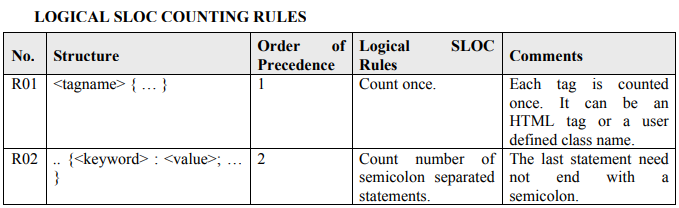
## Implementation Standards

Counting standards: Html, CSS









Coding Standards

Html

|  |  |
| --- | --- |
| **Lowercase names** | |
| Element and attribute names must be in all lower case: | <!-- Correct -->  <inputname="name"type="text" />  <!-- Wrong -->  <inputname="name"TYPE="text" /> |

|  |  |
| --- | --- |
| **Closing tags** | |
| Non-empty elements must have corresponding closing tags. | <h1>My title</h1>  <p>Some text</p> |
| Empty elements must be followed by a corresponding closing tag: | <span></span> |
| Elements with a single tag, such as HR, BR, INPUT, IMG must end with >: | <br>  <hr>  <imgsrc="john.jpg"alt="John Doe"width="200"height="100"> |

|  |  |
| --- | --- |
| **Encoding and charset** | |
| Set encoding of HTML document and its charset to UTF-8 Normalization Form C (NFC): | <metacharset="utf-8" /> |

|  |  |
| --- | --- |
| **HTML anchors** | |
| When you need to link to the section inside a HTML document use ID attribute: | <ahref="#section">link</a>  <divid="section"></div> |

CSS Coding Standards

|  |
| --- |
| **Encoding of CSS files** |
| Encoding of CSS files should be set to UTF-8. |

|  |  |
| --- | --- |
| **Naming Conventions** | |
| Always use hyphens in class names. Do not use underscores or CamelCase notation. | /\* Correct \*/  .sec-nav  /\* Wrong \*/  .sec\_nav  .SecNav |

|  |  |
| --- | --- |
| **Values** | |
| Always define generic font families like sans-serif or serif. | /\* Correct \*/  font-family: "ff-din-web-1", Arial, Helvetica, sans-serif;  /\* Wrong \*/  font-family: "ff-din-web-1"; |
| Shorten hexidecimal color values to 3 digits when possible: | background: #fff; |
| If you use 0 as a value, do not add a unit (px, em, etc.) after it. | /\* Correct \*/  .nava {  padding: 5px05px2px;  }  /\* Wrong \*/  .nava {  padding: 5px0px5px2px;  } |

### Code review prerequisites

* Thorough and meaningful code reviews take time. We need to scope for code review just like any other task. For that to happen, everyone must believe in its purpose and importance — that means management and product managers must actively encourage code reviews and give developers the necessary time.
* The team has agreed on naming convention, styling, and coding standards, whether it is **HTML, CSS, or JS**. Styling standards give the code a consistent and professional look. A naming convention makes it easier to understand, search, build, and maintain. Coding standards make the code cleaner and easier to read, build, maintain, and debug. There are many popular standards out there. It is less important which one you choose, but it is crucial that the entire team follows one standard. If a rule is controversial within the team, discuss and decide, then change the standard and apply the new rule across the entire code base. Consistency is key.
* The team has agreed on what to review — from spaces and camel Case, to where business logic should be, to whether the new code is violating established architecture.

### Some things to look for in HTML

* Does the code follow agreed-upon standards and conventions?
* Does the markup use the same element ID name more than once? If so, change it to a class because each ID should be unique. If a reusable module contains an element ID, the ID is no longer unique once the module is reused in the same page. Remember that getting an element by ID returns only the first element that matches, which may cause issues when you have multiple elements with the same ID name.
* Are there unnecessary wrapper elements? If a<div>does not have a class, or the<div>has a class but the class has no associated styles, chances are that it is not needed. Remove it and check the layout. If the layout does not change, you do not need that wrapper.
* If the app uses a template language that allows complicated logic, are you over-using it, thus making the markup hard to read? Consider applying the logic to the view model BEFORE the view model is passed to the template.
* Does the markup contain the proper [ARIA](https://en.wikipedia.org/wiki/WAI-ARIA) attributes for accessibility?

### Some things to look for in CSS

* Does the code follow agreed-upon standards and conventions?
* If you are using a CSS precompiler, such as [LESS](http://lesscss.org/) or [SASS](http://sass-lang.com/), are you nesting classes (or even worse, nesting IDs) because it looks more organized? Unnecessary nesting makes overriding CSS difficult; it also hurts performance and maintainability.
* Is CSS in modules namespaced to avoid name clashing? Consider a top-level wrapper class with the same name as the module. If you don’t want class nesting, consider using [BEM convention](https://css-tricks.com/bem-101/).
* Are some class names so generic that name clashes are likely? This can happen with either other modules in the same app or with other libraries that you may use.
* Are regularly used CSS rules (or blocks of rules) repeated so often that they should become mixins?
* If the app uses a CSS library, such as [Bootstrap](http://getbootstrap.com/) or [Skin](https://ebay.github.io/skin), are you maximizing its use? Does your app redefine mixins available in the library, or worse, does your app have blocks of CSS where using a single class from the library would do?
* Arepaddingandmarginused correctly? Ispaddingused whenmarginshould have been, or vice versa?
* Does usingpercentageinstead ofpixelmake sense? How aboutremvs.px?
* Are there duplicate rules for the same selector? Remove the duplicates.
* Are the same rules found for different selectors in the same module? Merge them.
* Are the same rules found across different modules? Put them in a shared CSS file.
* Are there too many nested classes? Are they necessary for override? Try removing some nesting. If the layout doesn’t change, they are not needed.
* Are IDs used for CSS? IDs are HIGHLY [specific](https://specificity.keegan.st/) and need other IDs to override them. And if you must insist on using IDs for CSS, DO NOT nest IDs. First, nested IDs require the same number of nested IDs or more to override, which makes your CSS longer and more complicated. Second, and this relates to fundamental CSS principle — IDs are by definition unique. If an ID needs to be nested to work, it’s not unique. It should not be an ID. Make it a class instead.
* If the code uses media queries, see the “Media queries” section in this [post](https://www.ebayinc.com/stories/blogs/tech/mobile-first-a-retrospective/).

### Some things to look for in JavaScript

* Does the code follow agreed-upon standards and conventions?
* Is there [JSDoc](http://usejsdoc.org/) or similar documentation, especially for exposed methods?
* If something is to be fixed/enhanced later, is there a TODO with a developer’s name next to it?
* Is a solution either too clever or too complicated? Is there a slightly longer but simpler and more maintainable solution? If you have to read the code (or its comments) several times to understand what it is trying to do, it is probably too complicated and should be simplified.
* Does each method do one specific task only? If not, break it up into smaller, focused tasks. It will help in reusability and unit testing.
* Is a method particular to one use case, with hard-coded values and hard-coded CSS selectors, and so on? Or is it generic and reusable, and takes parameters? Generic methods are scalable and easier to maintain, AND they are much easier to unit test.
* Does a block of code look familiar from elsewhere? If an existing util method does the same, use it. If there is a util method that is related but slightly different, update it. If the util does not exist, create it. Then, use that util and remove identical and similar blocks globally.
* If an app already uses a utility library such as [Underscore](http://underscorejs.org/) or [Lodash](https://lodash.com/), is the PR introducing another similar library? Discuss with your team which library best suits your app.
* Is the app loading a heavy library when you use only a few methods in it? Is there a lighter alternative? Can you pull out just what you need?
* Does a function take too many parameters? If there are more than 4 (or some number set by your team), use an object to avoid the hassle of parameter ordering and null values.
* Are values and data cached? Any string, object, etc., should be cached if it is used more than once.
* Are variable and function names so short that they are not meaningful at a glance? Longer names (be reasonable, of course) are more likely to be meaningful and self-documenting. Furthermore, searching globally for a longer name will return fewer results, which makes code-digging easier.
* Is there code that looks like the following? Use Lodash’s(or similar method from another library) to get an object property down a path. You can avoid long chains (it’s easy to miss a check) and undefined exceptions when the response is not what you expect.
* **If(a && b){}**
* If using third-party libraries, is the app using deprecated methods? Use the latest APIs as recommended by the libraries.
* Are all console logs and debuggers removed?
* Are listeners removed before being re-added? Does the handler run multiple times on the same event on the same element? One way to check is by printing logs in the handler.
* Pay attention to elements (usually inputs) that are listening to multiple events, such **keydown** and **keypress** and **keyup**. When one user action causes all events to fire, such that the handler is run once for each fired event, is the app still behaving as expected? Even if it is behaving correctly, does it make the app less responsive?
* If the app architecture is based on the modular pattern, is one module referencing or modifying DOM elements of an unrelated module, thus breaking the pattern?

### If the app uses jQuery

* When no element is found for a selector and your code operates on this “missing” selector, jQuery fails silently, which is both GOOD and BAD at the same time. It is good because no error is thrown at run time; it is bad because the app may fail in other ways. A common reason is that the selector was changed and it was not updated globally. Cache the selector to minimize that risk.
* Is the code using jQuery event delegation to minimize number of listeners?
* How are you listening to events on elements that can be re-rendered? Via jQuery event delegation? That way, you don’t have to set up listeners again every time the element is re-rendered. It is easy to miss the re-add, especially when there are multiple render paths.
* Are you using deprecated methods? For example, use **on()** instead of **Bind()/.live().**

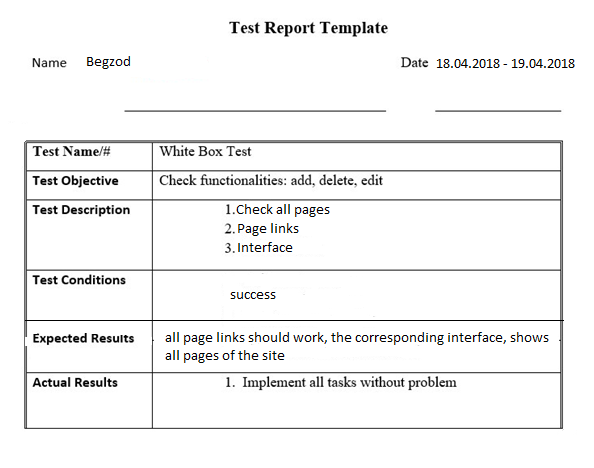
### Browser support

One difficulty in front-end development is getting a web app to work well across devices, operating systems, and browsers. With the large number of device, operating system, and browser combinations in the wild, including the old and the native, working well everywhere is an impossible task for developers.

To review layout and browser quirks, an in-person code review is the most effective as the reviewer can play with the feature/fix while reviewing the code. (A very simple exercise, which is also very effective in finding layout bugs, is resizing the browser window.) If an in-person review is not feasible, attach screenshots of the app to the PR, showing the layouts in different forms:

* Desktop browser
* Tablet browser (both portrait and landscape mode)
* Phone browser (both portrait and landscape mode

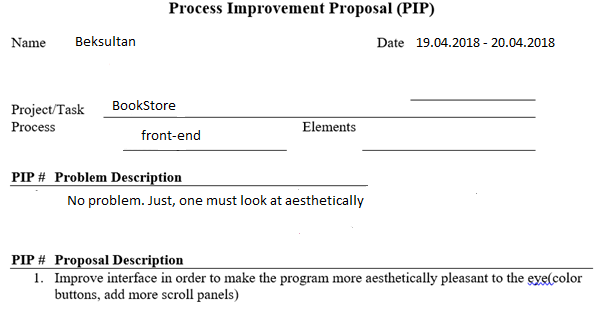
## Testing

****

### Unit Testing

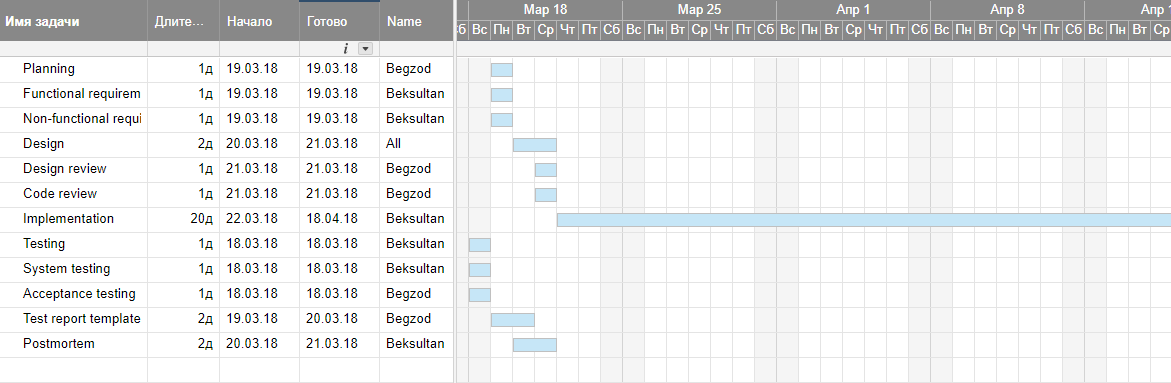
We did Unit testing for 2-cycle of our web application to be sure that our functions is working correctly.

## Postmortem

****

# Cycle #3

## Planning

****

## Requirements

### Functional requirements

* User should be able to register and login to the application.
* User should be able to add books to cart
* User should be able to delete books from cart
* User should be able to search book
* User should be able to order by price
* User should be able to see all order histories

### NON-Functional requirements

* Web application should able to give notification before delete books from cart.
* Application should give information English language.
* Application should support database connection.

## Design Review

Reviewers: Begzod

Date: 23.04.2017

|  |  |
| --- | --- |
| Is the design feasible from a technology, cost, and schedule standpoint? | + |
| Does the design support proceeding to the next development step? | + |
| Does the design use standard techniques and avoid exotic, hard-to-understand elements? | + |
| Is the design unjustifiably complex? | - |
| Is the design lean (i.e., are all of its parts strictly necessary)? | + |
| Will the design be easy to port to another environment if appropriate? | + |
| Does the design allow for scalability? | + |
| Does the design address all issues from the requirements of cycle-1? | + |
| Does the design follow all standards necessary for the system? (i.e., datestandards) | + |

## Design

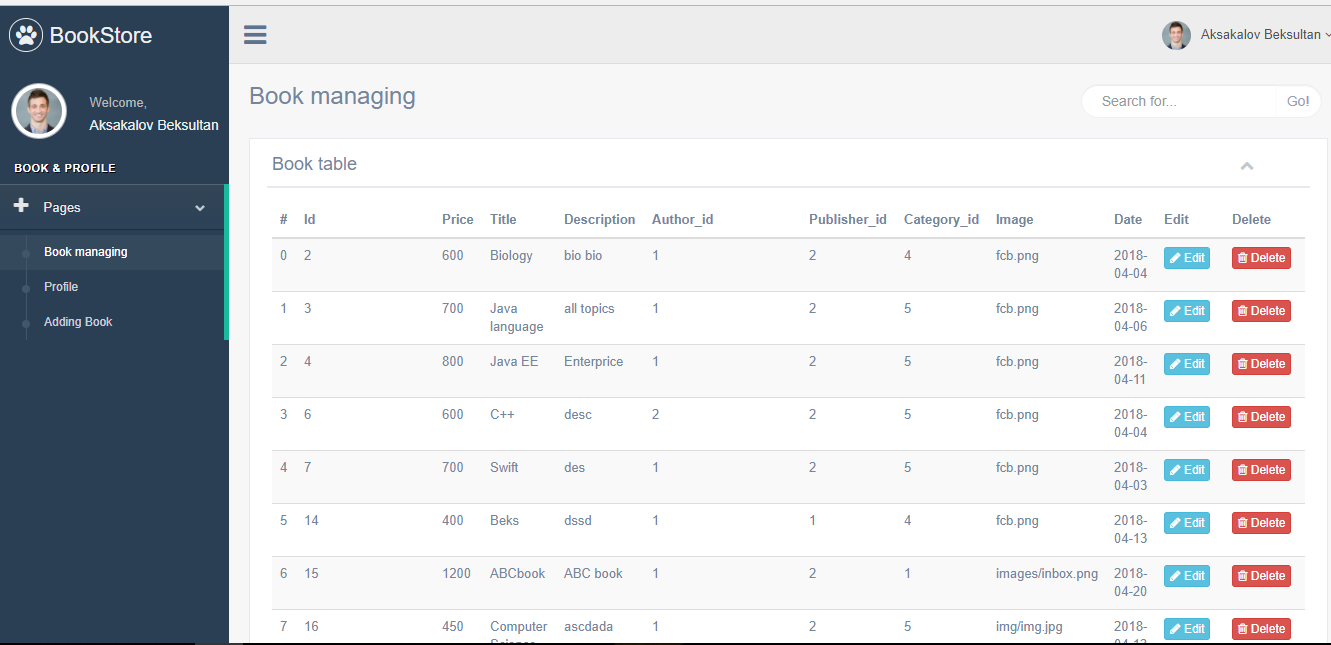
****

Figure 1.9 Admin panel

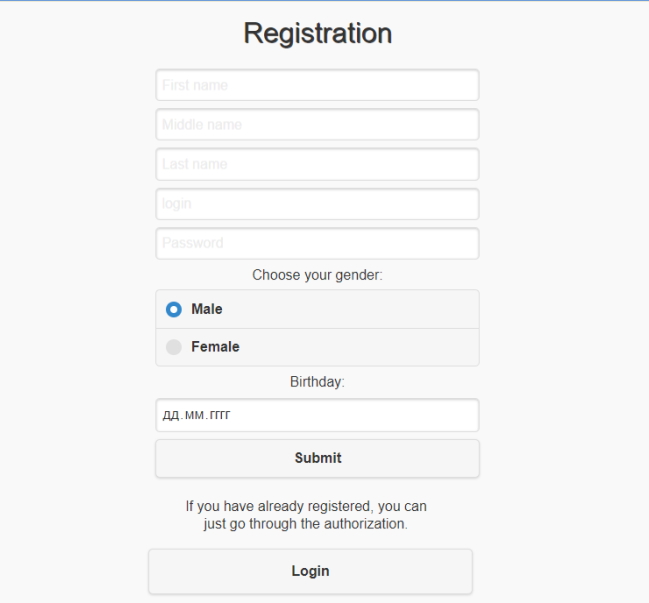


Figure 2 Registration form

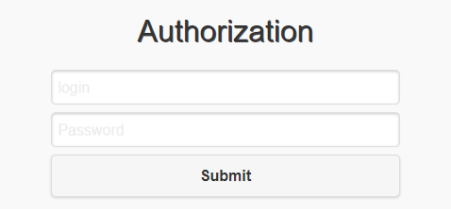


Figure 2.0 Authorization

## Code review

Date: 20.04.2017

Presenters: Rizabek, Nazir, Bakhytzhan

* Code works without errors
* The code is clean and understandable
* Code follows to coding standards and best practices
* Code is not duplicated
* Necessary lines of code are commented
* Code is divided into small functions for better understanding
* Function and variable names explain what they do

## **Testing**

Test report template

Name: Beksultan date: 25.04.2018 – 26.04.2018

Test objective:

Check all functionalities: log in, out, add book to store, delete books from cart, searching books, show all orders.

Admin panel can maintaining books such as delete, edit, add books.

Test description:

* 1. Check all functionalities(user)
  2. Check all functionalities (admin)

Expected results:

* + 1. Admin: able to delete, edit, edit books
    2. User: able to add, delete, search, see all orders books

Actual results:

* 1. Good performance.

### Unit Testing

We did Unit testing for 3-cycle of our web application to be sure that our functions ae working correctly. We found all our bugs and corrected. It was useful when we updated our web application, it means when we included new functionalities. These testing like JavaScript Unit testing, server side unit testing.

### Integrated testing

Including new functions like add, delete, edit we did integration testing. Combined each components are integrated and tested. And we provide integration testing to make sure that our server was connected correctly. During the testing we obtained the following errors and corrected it:

Check links:

* Test all links on a page for correct operation
* Test Web forms on the page:
* Check that password fields do not show the password contents
* Validate the response to a form submit
* Session Management Testing:
* Check application log-in sessions
* Check that session cookies reset between browser sessions
* Validate Cascading Style Sheet (CSS) tags:
* Identify CSS tags that return 404 or other CSS load errors
* Identify HTML id, class, and name attributes that do not match any CSS tags
* Validate JavaScript tags:
* Identify Script tags that return 404 or other load errors
* Identify id, name, on... attributes that do not match any Script tag
* Dynamic content testing (Database testing):
* Check data consistency in database-driven Web forms
* Check create, edit, delete, update (CRUD) tasks
* Verify data retrieval delivers the correct data
* Identify database connectivity and query errors

### System testing

-Checked functional completeness of the system

-Application is running on various browsers and with devices with different configurations

-Tested the fully integrated application including external peripherals by checking how components interact with each other and with a system as a whole.

-Verified through testing of every input in the application to check desired outputs

-Tested user experience with the application

### Usability Testing

**Test Navigation:**

-Check for the user to have apparent and easy controls to move from page to page.  
-Test the flow of a Web application by observing how the user accomplishes their goals  
-Check that the user can find instructions should they not intuitively know how to operate a function  
-Test that common navigation objects appear on every page consistently  
-Test search functions for proper application functions

## Postmortem

Process Improvement Proposal (PIP)

Name: Beksultan date: 26.04.2018 - 27.08.2018

Project: BookStore

Process BookStore display and functionality

Problem Description

**No**

Proposal Description

* 1. Improve interface in order to make the program more aesthetically pleasant to the eye (color buttons, add more scroll panels)
  2. Add ability to access for users adding own products.

# Heuristics

|  |  |  |
| --- | --- | --- |
| Heuristic name | Does the application follow to this heuristic? | Reason |
| Match between system and the real world | + | We have created an online book purchase site that allows users sitting at home to book books through an online website |
| Help and Documentation | + | Application has web page that consists of information that helps to users to solve problems that may occur. We specially created a page FAQ. |
| User control and freed | + | If the user accidentally pressed to add the book to the cart, he can safely remove from the shopping cart |
| Consistency and standards | + | Application uses best practices from web design and use appropriate and understandable words |
| Aesthetic and minimalist design | + | Design clean, pure. No irrelevant information. |
| Help users recognize, diagnose, and recover from errors | + | User sees error messages when he does something wrong. Forexample, incorrect email |
| Visibility of system status | + | App shows what happens in screen. |
| Error prevention | + | User can safely remove order from cart. |
| Recognition rather than recall | + | Program shows all necessary information in one page. User doesn’t have to remember something |
| Flexibility and efficiency of use | - | We didn’t set some hotkeys or shortcuts for the app. |

# APPENDIX B AND MAKING cOMMITMENTS

Meeting #1

Location: Canteen IITU.  
Date: 15 February 11:00

Topic: roles of each member, discussion of requirements  
next meeting topic: discussion of design

Meeting #2

Location: 502 room IITU.  
Date: 20 February 15:00  
Front-end: Begzod

Discussed: Created wireframe of design, main functionalities of application, discussion of mistakes, risks  
next meeting topic: implementation

Meeting #3

Location: IITU library.  
Date: 1 March 14:00  
Writes: Beksultan and testing together.

Discussed: adding new functionalities for application, doing some testing for cycle-1, creating a planning for cycle-2, discussion of risks, creating schedule.  
Next meeting topic: include database for application

Meeting #4

Location: Canteen IITU.  
Date: 5 March 11:00

Back-end: Beksultan

Testing:Beksultan and Begzod.

Discussed: adding new functionalities for application, doing some testing for cycle-2, create planning for cycle-3, discussion of risks.  
Next meeting topic: Include login form

Meeting #5

Location: 502 room IITU.  
Date: 15 March15:00

Back-end: Beksultan

Testing:Beksultan and Begzod.

discussed: adding new functionalities for application, doing some testing for cycle-2, create new planning for cycle-3, discussion of risks, connect with database system, registration, include login form  
next meeting topic: adding new function

Meeting #6

Location: Library at IITU.  
Date: 25 March 14:00  
Back-end:Beksultan

discussed: adding new functionalities for application, doing some testing for cycle-2, create new planning for cycle-3, discussion of risks, connect with database system, registration, include login form  
next meeting topic: improving design of project

Meeting #7

Location: Canteen at IITU   
date: 1 April 11:00  
Back-end: Beksultan

Discussed: adding new functionalities for application, create admin panel, improving design of project, discussion of risks  
next meeting topic: doing testing, identify defects

Meeting #8

Location: 502 room IITU   
date: 18 April 15:00  
Front-end and designer: Begzod

Back-end: Beksultan

discussed: adding new functionalities for application, improving design of project, discussion of new risks, discussion result of testing  
next meeting topic: finishing the project

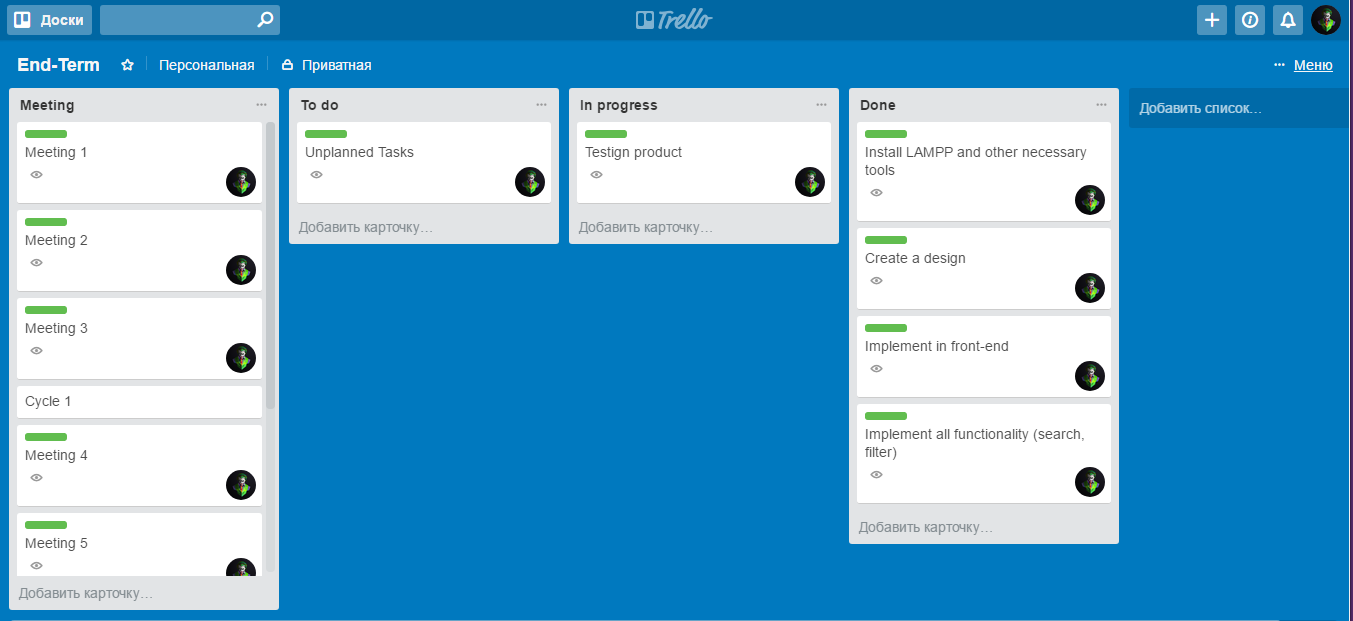
Meeting #9

Location: Library at IITU   
date: 20 April 14:00

Testing : Beksultan and Begzod  
Making presentation: Beksultan and Begzod

Discussed: discussion of final product, doing usability testing, does the presentation for product.

## **Trello**



# REFERENCES

<https://trello.com/b/XsBM5ETE/project070809>

[Brown] Rita Mae Brown, 1988. Starting from Scratch: A Different Kind of Writer’s Manual. New York: Bantam Books.