Command T

Software Design

Yejin Shin
yejin.shin@stonybrook.edu

Zhanarbek Osmonaliev
zhanarbek.osmonaliev@stonybrook.edu

Hasung Jun
hasung.jun@stonybrook.edu

Fabio Calero
fabio.calero@stonybrook.edu

Table of Contents

Table of Contents **2**

1. Introduction **4**

1.1 Product Description 4

1.2 Scope 4

1.3 Users 5

1.4 User Feedback 5

1.5 Existing Alternatives 6

1.6 Definitions 7

1.7 References 7

2. Requirements **8**

2.1 Functional Requirements 8

2.2 Use Cases 9

2.2.1 Use Case: Add a bookmark in Command T Homepage 9

2.2.2 Use Case: Customize layout of the objects 9

2.2.3 Use Case: Add a note using Notes widget 10

2.2.4 Use Case: Add a task using Todo List widget 10

2.2.5 Use Case: Move a bookmark into another folder. 11

2.2.6 Use Case: Delete Objects 11

2.2.7 Use Case: Add bookmark using the extension 12

2.2.8 Use Case: Change the background image 12

2.3 Non-functional Requirements 13

3. System Architecture **13**

3.1 Overview 13

3.1.1. Front-end 13

3.1.2. Back-end 13

3.1.3. Database 14

3.2 UML Class Diagrams 14

3.3 UML Sequence Diagrams 15

3.3.1. Use Case #1 : Add Bookmark 15

3.3.2. Use Case #2 : Add Note 16

3.4 API Design 16

3.5 Deployment 16

3.6 Code Conventions 17

3.6.1. File Structure 17

3.6.2. Naming 17

3.6.3. Styling 17
3.6.4. Gatsby/React Conventions 18

4. Schedule **19**
4.1 Milestone 1 19
4.2 Milestone 2 19
4.3 Milestone 3 20
4.4 Milestone 4 20

5. Contributions **20**

1. Introduction

1.1 Product Description

Using modern-day web browsers to save and organize access to web links has become a rather messy and cumbersome task. Safari, Chrome, and Firefox have very poor options for managing and organizing bookmarks. When users add bookmarks, they usually appear unorganized and spread without any meaningful layout. We propose a web application that will serve as a replacement for the Safari, Chrome, and Firefox homepages and allow users to customize their startup page, organize their bookmarks, make todo lists, and add customizable widgets, and take notes all in one place.

Our web app, CommandT, aims to create a webpage that serves as a startup page of any internet browser. Our targets are students and professionals from all fields who want to customize their startup page. CommandT will allow users to manage their bookmarks by creating folders, divide sections, categorize, and more. It will also contain widgets for notes, todo lists, and weather. Users will need to sign up for CommandT to keep their settings next time whenever they access the application again.

Main Features:

1. Managing bookmarks with folders.
2. Allow users to customize order of bookmarks and widgets (todo list, weather, and notes).
3. Allow users to customize the application's background image.
4. Use built-in Notes and Todo-list widgets.
5. Make extensions for Safari, Chrome, and Mozilla Firefox that allow users to save bookmarks when they are on different websites.

Stretch Goals:

1. Sidebar for quick actions such as chat for teams.
2. Calendar widgets.
3. Users will be able to share their todo lists with other users.

1.2 Scope

This project intends to provide an effective and quick way for people to organize their bookmarks on modern browsers. The focus is on allowing the user to customize their startup pages by using advanced options that are not available by default on most browsers. It is not our project's purpose to substitute the in-built startup pages on modern browsers but rather to allow our users to control the content they see by providing an alternative. Furthermore, it is not our project's goal to track our user's daily activity nor to automatically add bookmarks for our users. Instead, each user will opt to save their bookmarks by manually entering them through our web application or use an extension.

The product will be implemented as a responsive web application that will be usable on most modern desktop computer web browsers, specifically Safari, Chrome, and Mozilla Firefox. Our application will not support mobile browsers.

1.3 Users

Our project's primary users are university students who want to increase their productivity and efficiently organize their online class links and adult professionals from all fields whose work might involve visiting several websites regularly. We expect both of these user groups to be somewhat knowledgeable on the basic use of a browser and be somewhat familiar with technology in general.

Although our project primarily focuses on the above-mentioned audience, it will be available to a general audience as well.

1.4 User Feedback

Our goal was to target individuals initially. However, when we presented our mockup for Command T to a friend working at ZeroClassLab, a SUNY Korea startup company, he said that Command T can be used in teams as well; if and only if some shared features are combined. This feedback confirmed that Command T does not have enough functionality to dramatically enhance their team's productivity due to limited widgets. Thus it should be focused solely on individual uses. However, as this project proceeded, we got another feedback that adding a team feature would be really helpful and increase Command T's perfection. Our team decided to implement the calendar feature, which was one of our stretch goals, for this reason.

We also received design feedback. They asked how we will handle a situation when the text is invisible from setting a background image that has similar color to text color. We decided to handle the situation by changing the text color to a color that contrasts with the color of the background image.

After the interdisciplinary project pitch, a few professors and students were willing to use our Figma prototype and give feedback as alpha-testers. We will gather about 4~6 more testers to receive feedback. We plan to have the testing via Zoom. When the users enter the zoom session, they will get a link to our Figma prototype and interact with it. They will be asked to give feedback right after they have fully experienced our prototype. These participants will be informed about our requirements and initial design as well at this time.

The last feedback will be after the full implementation of Command T. We will release the Beta version of Command T and gather more participants from the last testing. The new participants will be asked to answer several questions including "Do you think Command T is ready to be released?" and "Any more features are needed?". The continuing participants from the last test will be asked "Does our final product match the requirements and design?". Only they can answer this question since they watched the evolution of Command T. Based on their responses, Command T will go through revision or be released.

1.5 Existing Alternatives

Existing alternatives to our product include the built-in bookmark managers from Chrome, Firefox, and Safari, as well as several bookmark manager applications found on the web. We feel that our product brings something new to the table. Therefore, we detail our main competitors and the shortcomings in their product that we think our project addresses.

1. Atavi: a free bookmark manager that allows users to create a customized homepage with bookmarks presented in an elegant and organized fashion. Atavi enables users to organize bookmarks into folders and to sync their bookmarks with any device or browser. It allows the users to customize almost everything from the background picture to the number of bookmarks on the page. Most of the inspiration for our product comes from Atavi. Effectively, we have tried to make a better improved Atavi with additional features to increase ease of use and productivity.

2. Raindrop.io: a visually enticing bookmark manager that allows its users to organize bookmarks into collections which can be displayed in headline format, card format, and mood board format. This application automatically tracks how frequently the user accesses a specific bookmark and groups bookmarks by frequency of use, as well as grouping the broken links in one place for the convenience of its users. Raindrop.io also allows users to save more than just bookmarks on their cloud accounts. It allows users to save pictures or to upload files into their collections. Raindrop.io compares to CommandT in so much as they are both bookmark managers. However, the philosophy behind our product and the design, and the feature set are completely different.

3. Bookmark Ninja: a cross-platform bookmark manager with extensive features, including a tab system for organizing bookmarks and the ability to tag bookmarks and then sort through bookmarks by tag. Booking Ninja is a very well-rounded app, but it differs from our product in that our product offers simplicity, where Bookmark Ninja is complicated. The many tools in Bookmark Ninja make it hard to learn to use for the uninitiated while our product strives to be intuitive and user friendly for all users, an idea that was inspired by Apple's philosophy of simplistic design.

4. Save to Pocket: a very simplistic yet popular web app for managing bookmarks. It allows its users to add bookmarks into lists that can be nested and annotate each bookmark. Its main shortcoming is that it is not customizable and the list can become cluttered very easily, making it hard to manage a large number of bookmarks effectively.

5. Google Bookmarks: a cloud-based bookmark managing application that comes as an extension for google chrome. It allows its users to save bookmarks to their google accounts, add labels and notes to each bookmark. Google Bookmarks doesn't let users put bookmarks

into folders, and the interface is rather unattractive and not designed to be a homepage in the user's browsers. Note: Google Bookmarks is not to be confused with the chrome built-in bookmark manager as it is a separate service.

6. Pinboard: the oldest bookmark manager in this list and most likely the one with the largest user base. It has a rather old look that takes no consideration for the user experience.

On top of the applications listed above, there are various note-taking apps that offer bookmark organizing features such as Evernote, Notion, and One Note. These apps offer a fast clipper tool and allow its users to manage their bookmarks into folders. However, the process is relatively slow due to the number of steps it takes to save bookmarks, it is not customizable, and the applications are not desktop friendly since these applications were originally designed to be mobile and tablet apps.

1.6 Definitions

- **Object**: refers to widgets, folders, and bookmarks.
- **Start up page:** the page displayed when users open up a new tab.
- **Layout**: the distribution of widgets on the screen.
- **Widgets**: a component of the interface that provides a service to the user (e.g. the weather widget: a component that provides weather information)
- **Web App**: short for “web application” meaning an application that is hosted on a web server.
- **Browser**: an application that allows the user to surf the web and display websites (e.g. Google Chrome, Mozilla Firefox, and Apple’s Safari).
- **Bookmark**: an entry or icon that effectively serves as a shortcut to a website the user has previously visited.
- **In-built**: available by default.
- **Todo list**: a list of tasks set to be performed at a later time by the user.
- **Todos**: several tasks belonging to a todo list.
- **Extension**: a small module that can be added to a browser to customize the use of the browser.

1.7 References

- iOS 14 - Apple. (2021). Retrieved from https://www.apple.com/ios/ios-14/
- Atavi. (2021). Retrieved from https://atavi.com/
- Raindrop.io — All-in-one bookmark manager. (2021). Retrieved from https://raindrop.io/
- Bookmark Ninja - Online Bookmark Manager. (2021). Retrieved from https://www.bookmarkninja.com/
- Pocket: How to Save. (2021). Retrieved from https://getpocket.com/
- Google - Bookmarks. (2021). Retrieved from https://www.google.com/bookmarks/
- Pinboard. (2021). Retrieved from https://pinboard.in/

- New Tab Redirect. (2021). Retrieved from https://chrome.google.com/webstore/detail/new-tab-redirect/icpgjfneehieebagbmdbhnlpiopdcmna

2. Requirements

2.1 Functional Requirements

Users can:

1. Create an account by logging in through Google account.
2. Create an account by signing up with email.
3. Log out or delete an account through account settings
4. Create and store bookmarks in folders.
5. Create folders for bookmarks.
6. Create and delete tasks using the Todo List widget.
7. Make, edit, style, and delete notes using the Notes widget.
8. * Add media in the notes.
9. Access weather information through the Weather widget.
10. Change preferred location for weather information in account settings
11. Customize background picture of the application
12. Position bookmarks in preferred grid order
13. Position widgets in preferred list order
14. Install an extension for Safari, Chrome, and Mozilla Firefox.
15. Create bookmarks by using the Safari, Chrome, or Mozilla Firefox extensions.
16. Set Command T as their start up page.
17. * Create an event using Calendar widget
18. * Share an event with the users' team using Calendar widget.
19. * Chat with users' team.
20. Get information about the installation and use of our web application.
21. Highlight and identify bookmarks by color tags.

System can:

1. Automatically save changes when the user uses Notes or Todo List widget.
2. Determine user's login status.
3. Save objects and their respective layout.
4. Permanently delete user's account

2.2 Use Cases

2.2.1 Use Case: Add a bookmark in Command T Homepage

Primary Actor:	User
Priority:	Essential
Precondition:	1. User is logged in. 2. The user has copied the URL to add.
Trigger:	User finds a site he wants to bookmark.
Scenario:	1. User selects to add a bookmark 2. System shows a screen to enter the URL, title, border color and folder location. 3. User pastes the URL, enters the title, chooses border color, and chooses folder. 4. User selects to save. 5. System verifies inputs are valid and displays updated bookmarks.
Extensions:	2-3a. User chooses to cancel editing. 2-3a.1 System removes the screen. 5a. User left URL blank. 5a.1 System alerts to fill the URL field and proceed to Step 3.

2.2.2 Use Case: Customize layout of the objects

Primary Actor:	User
Priority:	Essential
Precondition:	User is logged in.
Trigger:	User wants to change the layout of the objects.
Scenario:	1. User selects the setting. 2. System presents setting options (i.e. edit layout, edit profile, change background image). 3. User selects to edit layout. 4. System makes objects editable. 5. User changes the object's position as he desires. 6. User repeats step 5 as desired. 7. Users choose to cancel edit mode. 8. System disables objects from being editable.
Extensions	2-3a. User cancels to select editing layout. 2a. 1 System removes (or hides) setting options. 5a. User tries to place an object outside of its section.

	5a. 1 The object returns to an original position.

2.2.3 Use Case: Add a note using Notes widget

Primary Actor:	User
Priority:	Secondary
Precondition:	User is logged in.
Trigger:	User wants to make a note.
Scenario:	1. User selects the Notes widget. 2. System displays the Note screen. 3. User selects to add a note. 4. System displays an empty note. 5. User enters a title and a content. 6. User repeats steps 3 to 6 as desired. 7. System automatically saves the note.
Extensions	2-5a. User chooses to quit Notes. 2-5a. 1 System automatically saves the note. 2-5a. 2 System removes the Notes screen. 6a. User did not enter a title and a content. 6a. 1 System does not autosave the note.

2.2.4 Use Case: Add a task using Todo List widget

Primary Actor:	User
Priority:	Secondary
Precondition:	User has logged in.
Trigger:	User wants to add a task to the Todo List.
Scenario:	1. User selects the Todo List widget. 2. System displays the Todo List screen. 3. User enters his or her task to add to a todo list. 4. Users repeat step 3 as desired.
Extensions	3a. User chooses to add empty task 3.a 1 System does not save an empty task. 4a. User is done with adding tasks. 4a. 1 System autosaves the tasks. 4a. 2 System quits the Todo List screen.

2.2.5 Use Case: Move a bookmark into another folder.

Primary Actor:	User
Priority:	Essential
Precondition:	User is logged in and has at least one bookmark in a folder.
Trigger:	User wants to move a bookmark into another folder.
Scenario:	1. User selects settings. 2. System presents the settings options. 3. User selects the “Edit layout” option. 4. System allows objects to be editable. 5. User drags a bookmark on top of a folder he wants to move it to. 6. System moves the location of the bookmark into the folder and shows updated bookmarks.
Extensions	5a. User drags a bookmark on top of the current folder. 5a.1. The bookmark icon will go back to its initial position.

2.2.6 Use Case: Delete Objects

Primary Actor:	User
Priority:	Essential
Precondition:	User has logged in.
Trigger:	User wants to remove an object.
Scenario:	1. User goes to Settings and chooses to change the current layout. 2. System allows objects to be editable. 3. User selects an object he wants to remove. 4. System displays a screen with an alert that tells the user to confirm his action. 5. User confirms to remove the object. 6. System removes the object.
Extensions	5a. User cancels deletion. 5a.1. System closes the screen.

2.2.7 Use Case: Add bookmark using the extension

Primary Actor:	User
Priority:	Essential
Precondition:	1. User is logged in. 2. User installed the necessary extension. 3. User is on a different webpage (other than Command T).
Trigger:	User wants to bookmark the webpage he or she is visiting.
Scenario:	1. User clicks the extension icon. 2. System shows a screen with the current page's URL, title, default folder, and default color (the default folder is the most recent bookmarked folder, and the default color is clear color). 3. User chooses the folder location, color tag, and saves the bookmark. 4. System verifies the URL and title is valid. 5. System saves the bookmark into the folder.
Extensions	4a. Empty URL and title 4a.1 System cancels the adding bookmark process.

2.2.8 Use Case: Change the background image

Primary Actor:	User
Priority:	Essential
Precondition:	User is logged in.
Trigger:	User wants to change the background image.
Scenario:	1. User selects the setting. 2. System presents setting options (i.e. edit layout, edit profile, change background image). 3. User chooses to change the background image. 4. System shows a screen with photos and allows the user to search more on the internet. 5. User selects a photo. 6. System updates the change.
Extensions	3-5a. User cancels to change background image 3-5a.1 System closes the screen with images and shows Command T.

2.3 Non-functional Requirements

- Users can use all functions in Google Chrome, Safari, and Firefox on desktop web browsers.
- Users can use their existing google accounts to sign in without providing their personal information through our website.
- The personal information of the user who signed up through our website will not be visible to other users but only to system administrators.
- Page will be loaded under 500 ms.
- The error in all functions provided on our website is less than 1 percent.

3. System Architecture

3.1 Overview

3.1.1. Front-end

Since we are building an application that must be efficient and load fast, we decided to use GatsbyJS as a front-end framework. It pre-builds the application and generates static pages for optimized loading. It encapsulates ReactJS and has many additional plugins that could make our development process easier. For instance, we use a SEO optimization plugin that injects meta tags for each page generated.

For styling and creating layouts, we use Material UI framework. It has flexible component styling and lets you customize the global theme. Additional icons are imported from React Feather. For the background image, the application makes search query requests to Unsplash API to retrieve image urls.

3.1.2. Back-end

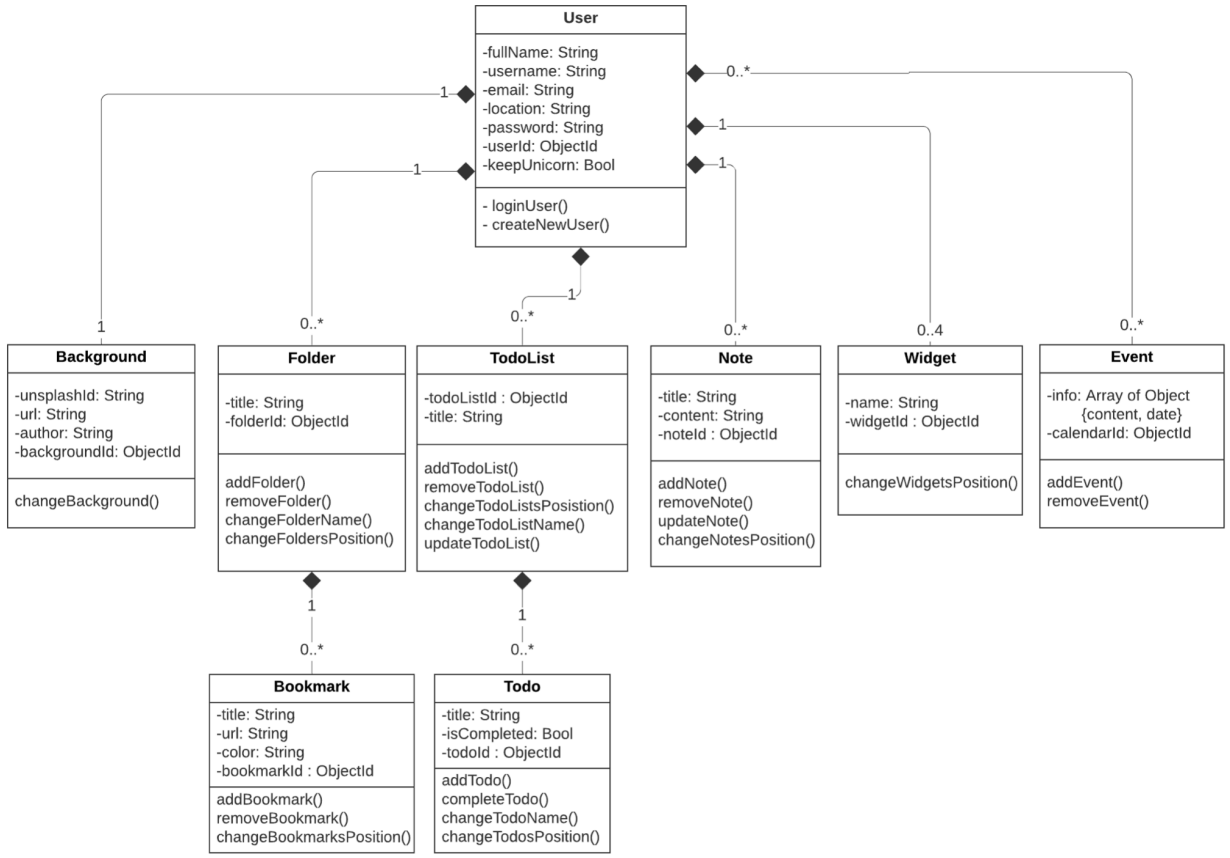
We are going to complete back-end development with Node.js. Within the Node.js runtime environment, a web framework called Express will be hosted. The versions of Node.js and Express are V14 and V4, respectively. We chose Node.js and Express as our back-end technologies because both Gatsby.js and Node.js are javascript languages, which can be more convenient for us to program consistently. Additionally, performance is one of the essential factors to be chosen from users to use our website as a homepage. This problem can be resolved from Node.js, which provides asynchronous processing.

3.1.3. Database

We chose MongoDB version 4.4.3 because a non-relational database is more flexible with modifications than relational databases. MongoDB Atlas will be used as the cloud database for Command T. We will use Google Cloud as our cloud provider and set Tokyo, the closest available region to South Korea, as its region. Since we are making Command T as one of our projects in college and not aiming to make a profit, we chose the free tier option of MongoDB Atlas. This free option offers M0 Sandbox, 2 GB storage, and shared RAM and vCPU.

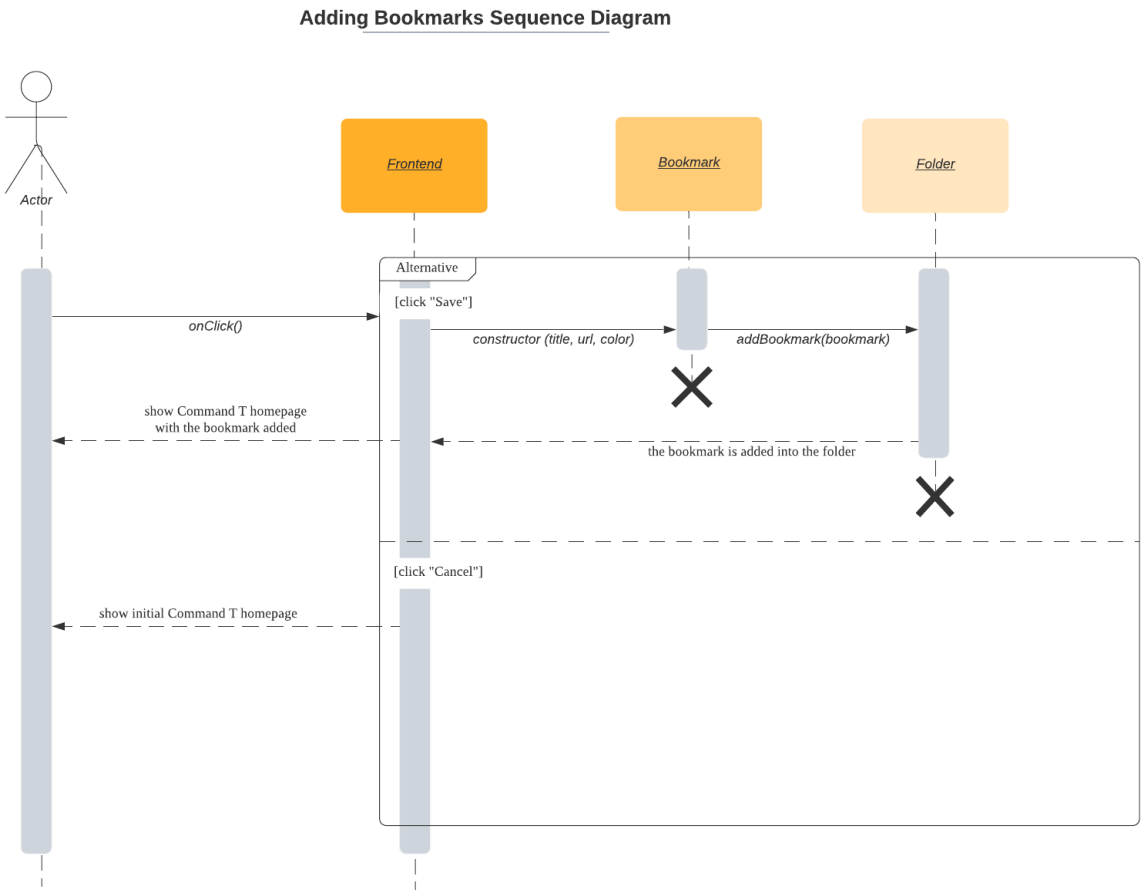
3.2 UML Class Diagrams

To see more in detail:
https://lucid.app/lucidchart/722401bd-5049-451a-b769-90ef73993a3a/view#

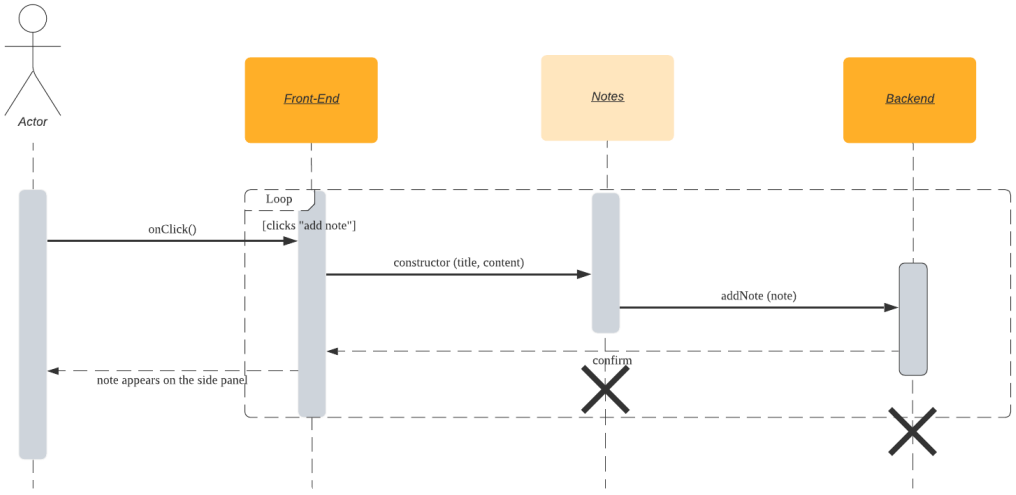


3.3 UML Sequence Diagrams

3.3.1. Use Case #1 : Add Bookmark



3.3.2. Use Case #3 : Add Note



3.4 API Design

https://docs.google.com/spreadsheets/d/1JkZ9CarcvO-EY3gF4CRZ5DRFvtUMn6zfcC2XodCmMKs/edit?usp=sharing

3.5 Deployment

We will adopt a deployment technique known as continuous integration. It is partly implemented on our Github repository, where Github actions automatically build our Gatsby site and publish it to `gh-pages` repository, after which the website becomes live.

We plan to host the front-end of the project on Heroku, a cloud platform as a service (PaaS). We will configure Heroku to fetch and build our `master` branch on each push in its dashboard section. Additionally, to ensure safe deployment, we will configure Heroku to fetch `master` only if all Github build checks have been passed.

Since Gatsby builds an efficient static webpage out of our React application, there is no necessity of having an actively running server for the website. Thus, we are separating our API handling server from the website. We will host the node server that will handle our API requests on Heroku as well. However, it will reside on a subdomain: `api.commandt.com`

Our database will be hosted on MongoDB Atlas, a cloud database service.

Github's student developer pack gives access to Heroku's lightweight Linux containers called Dyno, where we will host our application. Also, the pack includes a $200 credit for MongoDB Atlas and access to MongoDB Compass. We can also obtain a domain with a free SSL certificate for one year on Name.com through Github.

3.6 Code Conventions

3.6.1. File Structure

Below is the folder structure with the most crucial elements. We must add relevant components and files to their respective folders.

- src
    - components - React components
        - homepage - components used on homepage
        - landing - component used on landing page
    - pages - pages that are rendered
- [CHANGELOG.md](<http://changelog.md>) - important changes
- gatsby-config.js - website configurations, exports

- gatsby-browser.js - linking SASS to the website

3.6.2. Naming

Team KGB follows naming conventions written in Google's JavaScript Style Guide, with some exceptions on naming JS/JSX components and CSS classes.

When it comes to file and CSS class naming, we use a spinal-case naming convention, mostly used in AngularJS file naming system, because it is easier to read and navigate on horizontal scroll when there are many items.

Example: `hyphen-separated-lowercase-words.js` or `.bookmark-wrapper`

3.6.3. Styling

React and HTML objects should be styled in the global SASS file. Unless strictly necessary, elements can be styled inline inside components using Material-UI helper methods (`styled`, `withStyles`) and with HTML style methods.

CSS classes should be named according to the structural naming convention to be intuitive and to avoid redundant class names. Our CSS classes follow this modularity described below:

`[element]-[utility]-[component]-[style]`

- `element` is the name of the functional component, to which styles are going to be applied

Example: `folder`

- `utility` is used to add an extension to the element based on utility

Example: `widget-wrapper, button-link`

- `component` is used to refer to child component of the element

Example: `widget-content, widget-title`

- `style` is used to specify the styling

Example: `button-success`

3.6.4. Formatting

Our team uses a formatting extension, called Prettier. It is embedded into the project as a development dependency package. Whenever the code is saved, it is indented and formatted.

3.6.4. Gatsby/React Conventions

Components

Components must be created as constants or declared as functions. This way of creating components is more common in the industry:

```
function Bookmark(props) {
return (<div></div>)
}
```
or
```
const Bookmark = (props) => {
return (<div></div>)
}
```

We do not follow the old EC6 way of creating components:

```
class Bookmark extends React.Component{
render() { <div></div> }
}
```

State

Component state should be managed using new React Hook, that is useState(), introduced in React v16.8:

```
const [background, setBackground] = useState(props.background)
```

We do not follow the old format of state management:

```
constructor(props){
super(props)
this.state = props.background
}
```

4. Schedule

To see more in detail: https://trello.com/b/9p8PsC3W/to-do

4.1 Milestone 1

Task 1. Refactor landing page, Notes window, Todo List window assigned to **Zhanarbek Osmonaliev**.
Task 2. Complete Unsspash API integration assigned to **Zhanarbek Osmonaliev**.
Task 3. Implement change layout function that changes the position of Widgets, bookmarks, and folder movable with drag assigned to **Fabio Calero**.
Task 4. Implement adding a bookmark function without a database assigned to **Yejin Shin**.
Task 5. Implement adding a folder function without a database assigned to **Yejin Shin**.
Task 6. Format local data into JSON format assigned to **Yejin Shin**.

Task 7. Implement add and delete functions for Note Widget assigned to **Yejin Shin**.
Task 8. Implement switching folder function assigned to **Hasung Jun**.
Task 9. Implement animation for changing layout assigned to **Hasung Jun**.
Task 10. Implement add and delete functions for Todo List and Todos assigned to **Hasung Jun**.

4.2 Milestone 2

Task 1. Implement editing note with style for Note widget assigned to **Zhanarbek Osmonaliev**.
Task 2. Implement the Hide/Show Unicorn button assigned to **Zhanarbek Osmonaliev**.
Task 3. Prepare back-end hosting with Heroku assigned to **Fabio Calero**.
Task 4. Connect MongoDB with Back-end assigned to **Fabio Calero and Yejin Shin**.
Task 5. Perform first user testing assigned to **Yejin Shin**.
Task 6. Implement changing position functions for notes, todo list and todos in widget's modal assigned to **Hasung Jun**.
Task 7. Implement Authentication system to login with Google account using OAuth 2.0 and without using Google account (our login system) assigned to **Fabio Calero**.

4.3 Milestone 3

Task 1. Prepare a presentation assigned to **Zhanarbek Osmonaliev, Fabio Calero, Yejin Shin, and Hasung Jun**.
Task 2. Deploy Command T with Heroku assigned to **Zhanarbek Osmonaliev, Fabio Calero, Yejin Shin, and Hasung Jun**.
Task 3. Test users with beta release assigned to **Zhanarbek Osmonaliev, Fabio Calero, Yejin Shin, and Hasung Jun**.
Task 4. Implement Calendar Widget assigned to **Zhanarbek Osmonaliev**, **Hasung Jun (By May 2nd).**

4.4 Milestone 4

Task 1. Polish any inconsistent layout assigned to **Zhanarbek Osmonaliev, Fabio Calero, Yejin Shin, and Hasung Jun**.
Task 2. Fix bugs reported from user testing assigned to **Zhanarbek Osmonaliev, Fabio Calero, Yejin Shin, and Hasung Jun**.
Task 3. Implement Firefox extension assigned to **Fabio Calero**.
Task 4. Implement a Chrome extension assigned to **Yejin Shin**.
Task 5. Implement a Safari extension assigned to **Hasung Jun**.

5. Contributions

Zhanarbek Osmonaliev:

- Complete writing in 3.1.1. Front-end, 3.5. Deployment, 3.6. Code Convention
- Participated in designing 3.4. API Design
- Participating scheduling 4. Schedule
- Participating in the presentation as speaker on April 15th.

Fabio Calero:

- Complete making 3.3.2. Use case #2
- Participating scheduling 4. Schedule
- Participating in the presentation as speaker on April 15th.
- Proofreading and coherence checking of the document.

Yejin Shin:

- Complete writing in 3.1.3. Database.
- Complete making 3.3.1. Use case #1
- Participated in designing 3.4. API Design
- Participating scheduling 4. Schedule
- Review past Software Requirements Specification document.
- Made MongoDB collections and documents based on the UML Diagram.

Hasung Jun:

- Complete writing in 3.1.2. Back-end
- Complete making 3.2. UML Class Diagrams
- Participated in designing 3.4. API Design
- Participating scheduling 4. Schedule
- Review past Software Requirements Specification document.