The Notes Application

System Specifications

[SE1 - 2020]

Group H:

Roman Shevchuk

Kiryl Volkau

Illia Manzhela

Mohd Wafa

M Hasibur Rahman

Instructor: Maciej Bednarz

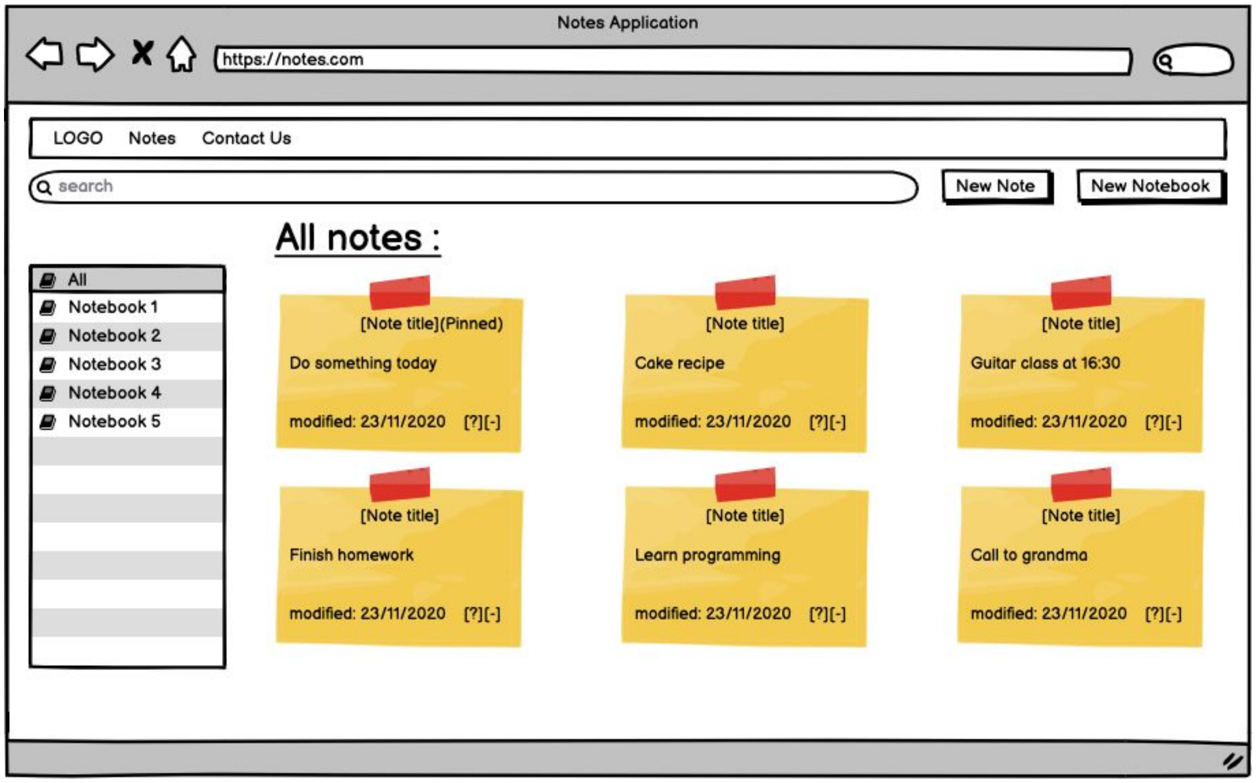
Content

1. General Overview

1.1 Introduction

The main purpose of this document is to present a detailed description of the Notes application that we are going to develop. The paper will explain all the features of system, user requirements as well as the internal, external functionalities and non functional requirements of the system.

A visual representation of the system:



The main idea is to create a flexible system to allow the user to manage his/her notes and to arrange them in a convenient way, so that it can become a very handy tool in one's everyday life. The project is also portable for mobile devices since we are going to develop a mobile application. It will allow the user to easily and quickly access and edit his memos if the PC or a laptop are not available. Some small mobile widget displaying recent notes will also be available and can make some positive impact on different life situations.

1.2 Scope of the project

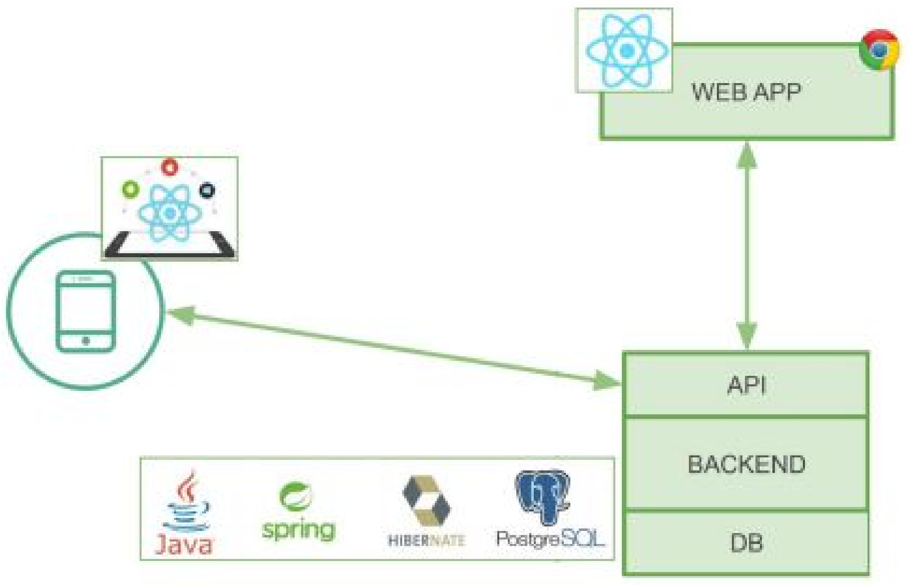
This software system will be a typical note taking application with a target audience ranging from students to average users who would like to store their ideas, memories or notes. This system will be designed to maximize the convenience of creating any of those by providing portable accessibility to the system at all times by using both a mobile and a web application. The system will meet the average user’s needs of a private and instant note storing system with a bunch of additional features to extend the range of possibilities.

More specifically, the application will consist of 3 components:

- Web Application
- Mobile Application
- Backend Server

Web Application which will run on the client side will serve for convenient user experience for those who use browsers. It will communicate with Backend Server through REST APIs and guarantee continuous connection to the source of data and possible actions until the user logs out. Client side application will be written in ReactJS framework. Backend Server will be connected to the database and will handle all the requests for storing data or retrieving it and sending back to the user. We will use the Java framework called Spring for the backend component. Mobile Application is assumed to be for Android devices and the React Native is assumed to be used as the framework for the development. Basically, Mobile and Web versions will share the same functions to: display, filter, add, update and delete notes. Each note will also have a category, so after the creation, it can be added to any of existing notebooks or a new one. There also will be a possibility to pin a note, so that it will always appear at the top of any selections, being an important one. Each note can also be shared with ‘outside world’ by the URL address which can be generated and used to access one specific note without being logged in. In this way we add the possibility for users to share some notes with each other (in this case the accessed note is read only). Users can also export and import content of a note by pressing the button and choosing either the destination folder and name or just specifying the .txt file to get the note body from. The backend server will be exposing APIs so that both web and mobile clients can execute all the necessary operations. All the notes data will be stored in the database and will be used by the server to ensure both applications functioning. This design will allow users to execute desired operations related to notes and have an instant updated system across multiple platforms.

The communication diagram and the technology stack:



2. System Requirements

2.1 System requirement overview

The application will be a permanent, multiple platform based system allowing to create and manage notes. It has three components: Client side application, Backend server and a Mobile application. All of those are used to maintain a continuous real-time updated system which shows the latest modified and stored data on both PC and mobile devices. In order to get all privileges and be able to perform all the possible actions a user should be registered in the system and logged in later on. Otherwise he/she is considered as an anonymous user and is only capable of viewing shared notes.

2.2 Functional requirements

2.2.1 User stories

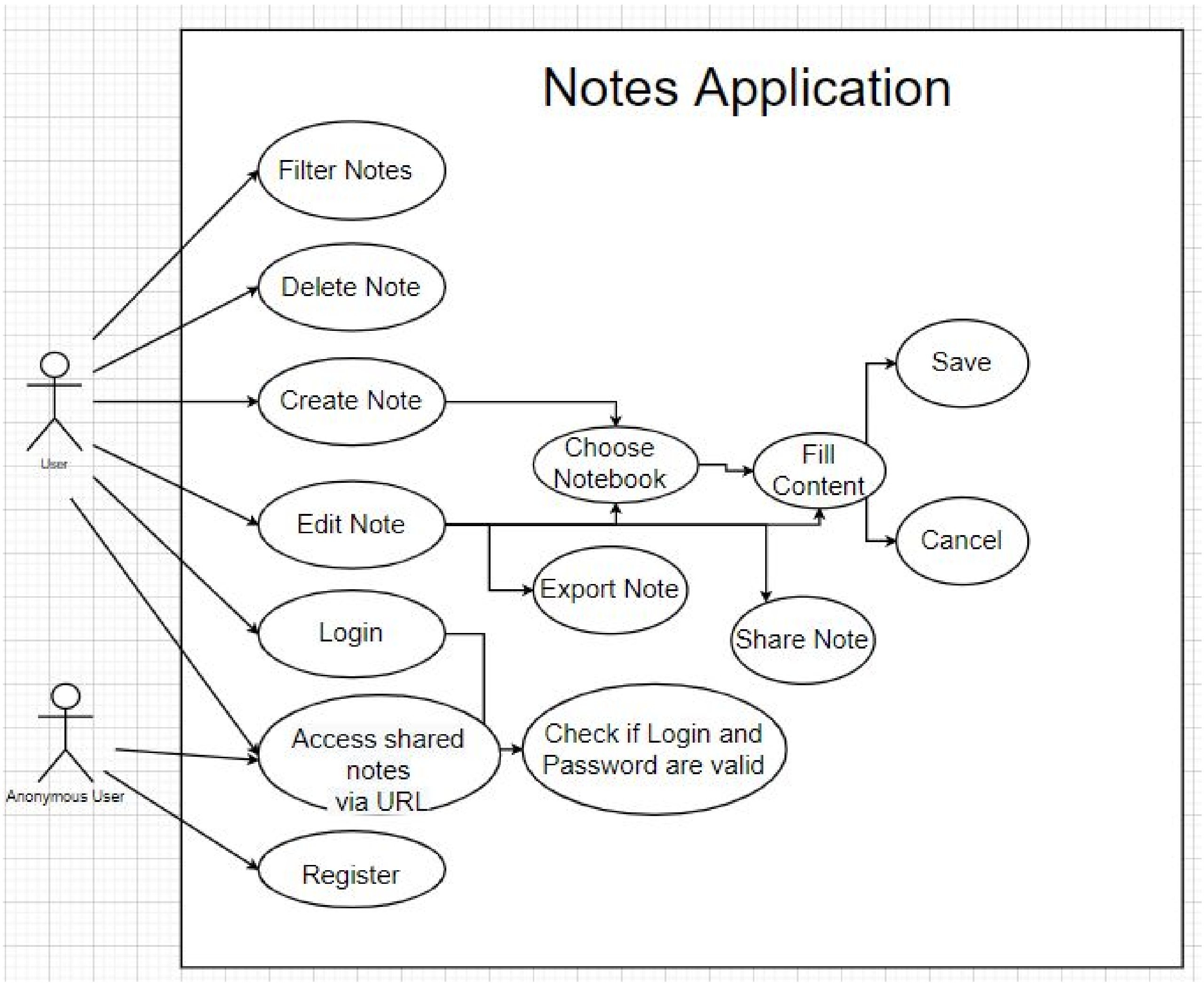
- As a user, I want to organize my notes, so that I can easily access them in less time.
    - Given notes, when a user puts a note in a notebook, he can later access it and all the other notes related to that notebook.
- As a user, I want to edit notes, so that I can update them whenever I want.
    - Given notes, when a user wants to update the content, he is able to change a specific note.
- As a user, I want to filter notes, so that I can view the notes that I am only interested in.
    - Given notes, when the user filters notes with specific identification, then he is able to view only those desired notes.

- As a user, I want to delete notes, which I do not need anymore or which are wrong.
    - Given a not needed note, when the user deletes a note, then it is no longer visible in the notebook.
- As a user, I want to display only the notes from one notebook.
    - Given notes, when the user displays notes of a specific notebook, then only those notes are visible.
- As a user, I want to add notes in an existing notebook, so that I can easier access them later.
    - Given notes, when a user adds a note in a specific notebook, then it can later be found in that notebook.
- As a user, I want to export some notes as a txt file, so that I can have it saved on my computer.
    - Given notes, when the user exports a note, then that note is saved on the computer.
- As a user, I want to import some .txt files, so that I can have it in my notes.
    - Given notes, when the user imports a txt file, then that file gets saved as a note.
- As a user, I want to share some notes, so that some person would have access to it.
    - Given notes, when the user wants to share some note, an URL address gets generated via which the note can be accessed.

2.2.2 Use Case Diagram

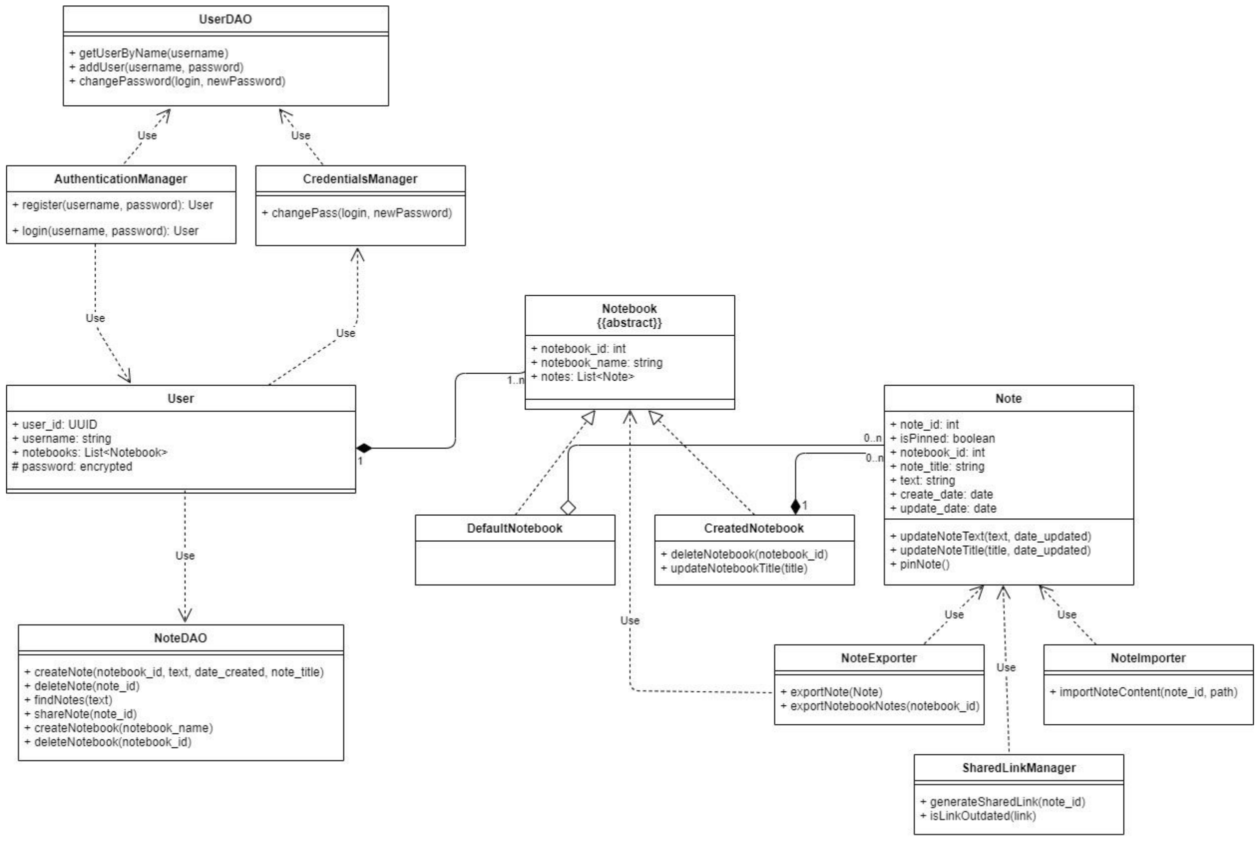
As we can see from the diagram, there are two actors which represent Anonymous User (not logged in) and User (logged in), one system called Notes Application, and many use cases. Let's dive into the cases. Since the app is about note taking, their saving and sharing, User can Create Note specifying one of the stored notebooks, Delete Note, Filter Notes which filters all the notes, Edit Note which contains content edition and a possibility to either export the chosen note or import it's content. The login case means logging in to some existing account, which triggers a password validation function, whereas Register means signing up. Finally, Access Shared Notes via URL represents a function which would generate a link which can be shared in order to get read-only access to some notes without the need to have an account to make the service more delightful.

The user case diagram for our application :



2.2.3 Class Diagram

The user class diagram for our system is as follows :

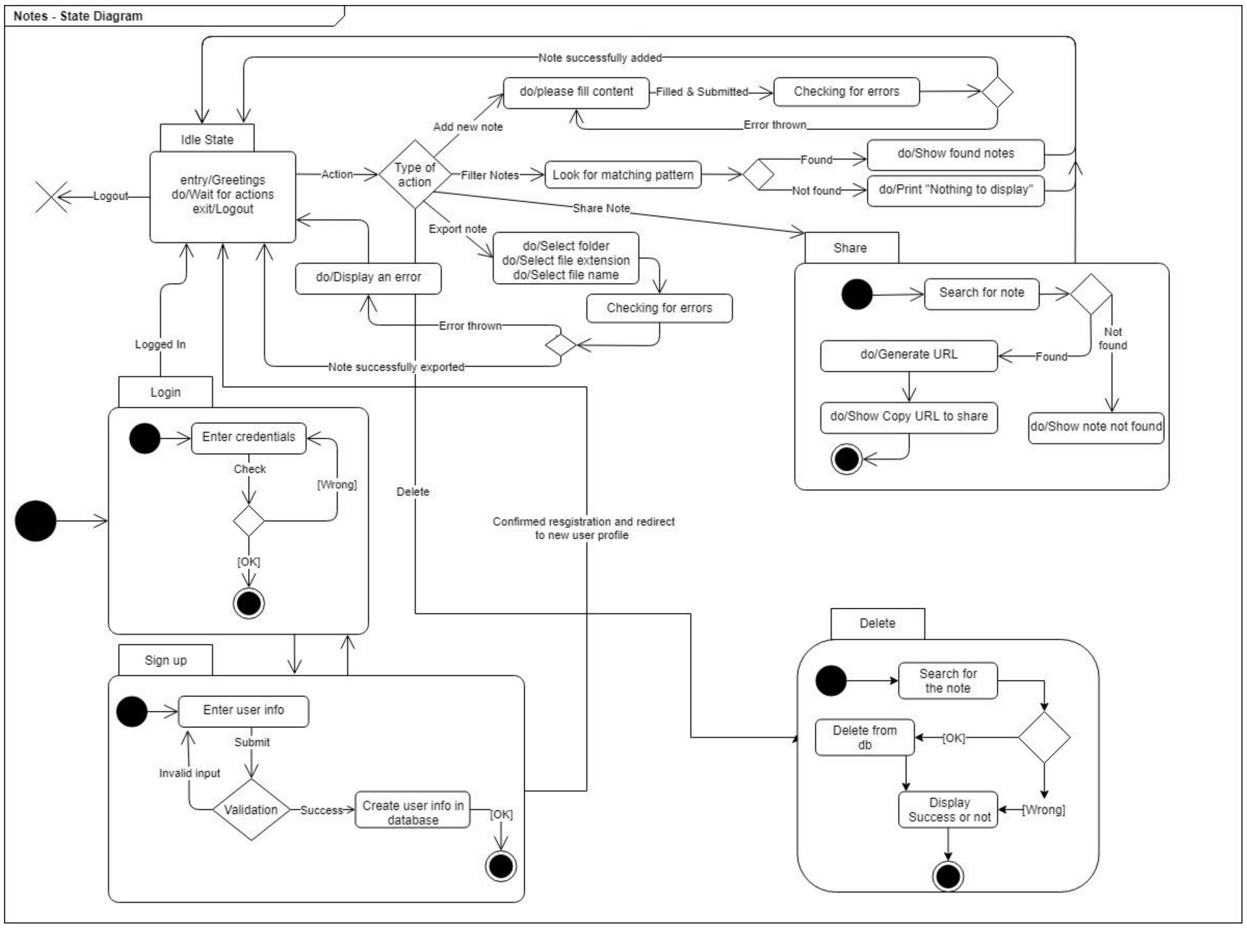

UserDAO
+ getUserByName(username)
+ addUser(username, password)
+ changePassword(login, newPassword)
Use
Use
AuthenticationManager
+ register(username, password): User
+ login(username, password): User
CredentialsManager
+ changePass(login, newPassword)
Use
Use
Notebook
{{abstract}}
+ notebook_id: int
+ notebook_name: string
+ notes: List<Note>
1..n
User
+ user_id: UUID
+ username: string
+ notebooks: List<Notebook>
# password: encrypted
1
Note
+ note_id: int
+ isPinned: boolean
+ notebook_id: int
+ note_title: string
+ text: string
+ create_date: date
+ update_date: date
+ updateNoteText(text, date_updated)
+ updateNoteTitle(title, date_updated)
+ pinNote()
0..n
0..n
1
DefaultNotebook
CreatedNotebook
+ deleteNotebook(notebook_id)
+ updateNotebookTitle(title)
Use
Use
Use
Use
Use
NoteDAO
+ createNote(notebook_id, text, date_created, note_title)
+ deleteNote(note_id)
+ findNotes(text)
+ shareNote(note_id)
+ createNotebook(notebook_name)
+ deleteNotebook(notebook_id)
NoteExporter
+ exportNote(Note)
+ exportNotebookNotes(notebook_id)
NoteImporter
+ importNoteContent(note_id, path)
SharedLinkManager
+ generateSharedLink(note_id)
+ isLinkOutdated(link)

As seen from the diagram above, Notebook is an abstract class. The classes DefaultNotebook and CreatedNotebook have a realization relationship with NoteBook. DefaultNotebook is empty since it defines the notebook which exists by default for every user and thus contains no methods such as deleteNotebook(notebook_id), which is present in a CreatedNotebook. Thus CreatedNotebook and DefaultNotebook act as our subclasses.

Class Note is the most important part of the system and undergoes various operations and thus has many dependencies and relations. The NoteExporter, NoteImporter and SharedLinkManager classes depend on Note, since the methods of these classes will give a successful output only if the note exists in the database. Same principle applies between the NoteExporter and Notebook dependency relation. Note has an aggregation relation with DefaultNotebook and a composition one with CreatedNotebook since Note may or may not be in the default Notebook but a note cannot exist without a Notebook i.e. notes live and die with its associated notebook. Moreover, the same principle applies for the composition relation between Notebook and User that is, a Notebook cannot exist without a user.
Classes defined as DAO are classes which communicate with the Database. All the methods described in the DAO classes are the functionalities that a user is able to execute and thus have a dependency relation such that Users depend on NoteDAO. AuthenticationManager depends on User input to either register or allow the user to login. AuthenticationManager communicates with the UserDAO to perform action according to user input i.e create or log in to existing user accounts. CredentialsManager also communicates with the UserDAO and upon successful changes in the database, User is able to change their credentials.

2.2.4 State diagram

The state diagram for our application is as follows :



In the initial state for our application, the first transition is towards the login state, which asks for credentials and checks whether they are correct, and then it proceeds out of the page. The user also has the option to sign up for the application by entering the user info, and after validation, the user is created in the database.

In the idle state of the application, the user has the options to add a new note, filter notes, share notes, export, and delete notes. They also have the option to logout of the application, and by doing so, terminates the node.
All the actions selected by the user, after completing them successfully, return to the idle state of the application. For adding a new note, the user first fills the content and checks for any errors in the content submitted. To filter notes, the application looks for matching patterns as entered by the user; if such a note is found, it displays them, otherwise it prints the error “ nothing to display “.
To share the note, the application searches for the note that the user wants to share, if the note is found, the application generates a URL and allows the user to copy the URL. If a note is not found, the error is printed, ”Note not found “.
And finally, to delete a note, the state starts by searching for the note, finds it, and deletes from the database. If the note is not found, it displays the error, and ends the state.

2.2.5 Functionalities

- Users can log in to the app and register.
- Users can log out.
- Users can create notes.
- Users can create notebooks.
- Users can move notes from one notebook to another.
- Users can rename and edit notes.
- Users can rename notebooks.
- Users can export and import notes.
- Users can share notes via links.
- Users can delete notes.
- Users can pin notes.

- Users can delete notebooks.
- Users can change their authentication data.

2.3 Non-Functional requirements with FURPS+

- Usability
  - Users are expected to have basic knowledge on how note taking apps work. Additionally, it is assumed that they understand the structure of the file systems (folder-subfolder-file).
  - Users use keyboard and mouse/touchpad as an input device in case of working with the computer and screen in case of working from the mobile app. The output device in both cases is the display.
  - Proper tutorial for completely new users is provided.
- Reliability
  - The app is available from any browser and on the iOS and Android mobile operating systems.
  - The system is a note-taking app for educational purposes, so it may contain bugs and minor errors which can be tolerated.
  - Data is preserved in the database on the remote server and is not lost in case of the unexpected system termination (unless the database has become inaccessible or has been dropped).
  - Notes can be edited only by the owning user.
  -
- Performance
  - The system should hold the amount of users with notes not exceeding the physical limitations of the server.
  - All notes operations should be atomic.
- Supportability
  - The developing team fixes bugs after they have been reported.

- Updates are done by simply updating code on the server, not requiring the user to install something (except for the one-time install of the app on the mobile).

- Implementation requirements
  - Graphic user interface is required.
  - Languages of implementation are limited to those supporting building REST APIs and supporting mobile development.
  - Platform of implementation should be *nix-like (for hosting) and mobile ones, as listed above.
- Physical requirements
  - Should run on all new browsers and mobile phones running under Android.
- Operations
  - The developers’ team are responsible for delivering new versions of the product and extending features, deploying it and fixing bugs.