

ONLINE NOTE TAKING SYSTEM

BY

Janta Roy Antor
ID: 142-15-3806

AND

Md Shawon Sarkar
ID: 142-15-4003

This Report Presented in Partial Fulfillment of the Requirements for the
Degree of Bachelor of Science in Computer Science and Engineering.

Supervised By

Ms. Nusrat Jahan
Lecturer
Department of CSE
Daffodil International University



DAFFODIL INTERNATIONAL UNIVERSITY

DHAKA, BANGLADESH

MAY 2018

APPROVAL

This Project titled “**Online Note Taking System**”, submitted by Janta Roy Antor, ID No: 142-15-3806 and Md Shawon Sarkar, ID No: 142-15-4003 to the Department of Computer Science and Engineering, Daffodil International University, has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of B.Sc. in Computer Science and Engineering (BSc) and approved as to its style and contents. The presentation has been held on 6th May 2018.

BOARD OF EXAMINERS

Dr. Syed Akhter Hossain
Professor and Head

Department of Computer Science and Engineering
Faculty of Science & Information Technology
Daffodil International University

Chairman

Dr. Sheak Rashed Haider Noori

Associate Professor and Associate Head

Department of Computer Science and Engineering
Faculty of Science & Information Technology
Daffodil International University

Internal Examiner

Md. Zahid Hasan

Assistant Professor

Department of Computer Science Engineering
Faculty of Science & Information Technology
Daffodil International University

Internal Examiner

Dr. Mohammad Shorif Uddin

Professor

Department of Computer Science and Engineering
Jahangirnagar University

External Examiner

DECLARATION

We hereby declare that, this project has been done by us under the supervision of **Ms. Nusrat Jahan, Lecturer**, Department of Computer Science and Engineering, Daffodil International University. We also declare that neither this project nor any part of this project has been submitted elsewhere for award of any degree or diploma.

Supervised by:

Ms. Nusrat Jahan
Lecturer
Department of Computer Science and Engineering
Daffodil International University

Submitted by:

Janta Roy Antor
ID: 142-15-3806
Department of Computer Science and Engineering
Daffodil International University

Md Shawon Sarkar
ID: 142-15-4003
Department of Computer Science and Engineering
Daffodil International University

ACKNOWLEDGEMENT

At our hearts, first of all, we would like to express our gratefulness to our Creator; it's the genuine and divine blessings from Him that made us uphold and complete our project.

We are really grateful and wish our gratitude to our respectful supervisor mam, **Ms. Nusrat Jahan, Lecturer**, Department of CSE, Daffodil International University. With her profound cognition and interest in our 'Web-Based' OCR oriented project, she continuously encouraged us to carry on with our project. Her righteous supervision, endless patience on checking the inferior drafts we were creating, constructing our thoughts and concepts more precisely about our project not only helped us to create our project but also shape our outlook about our misconceptions and be able to correct them.

We also like to convey our heartiest gratefulness to **Dr. Syed Akhter Hossain, Professor and Head**, Department of CSE and our respectful teachers at CSE department who has taught and inspired us in many ways.

And we feel the gratefulness to our family and parents, who have blessed us with their support and patience.

Finally, our amazing friends at Daffodil International University, who listened to our babblings and discussed and motivated us along the way to make our journey to this far.

ABSTRACT

The web is delivering application content in an enormous way regardless of platform and environment. And our aim is to build an application, which will run on the web and help users to manage their schedules, thinking a little better. Our intention is to build a note-taking application with OCR facility that can get the texts from user's image snippet that contains text. Initially, we are enabling support for Bangla and English language. And later as our project grows we will add other languages that can recognize and saved as text. Also attaching tags on a note will help to search faster. Image to text extraction is currently done by using Tesseract OCR engine which is probably a better tool to use for recognizing characters. To say more, it will help people to get better time maintenance as they properly keep track of their chores. And the results after using the Tesseract is quite good on images having less background noise. Our application is more about alignment as a user may have a lot of things to do in parallel, this will particularly help them to better manage their time. And also the text to image facility will be helpful to extract the text automatically than typing.

Table of Contents

Contents	Page No
Approval	i
Declaration	ii
Acknowledgement	iii
Abstract	iv
Table of Contents	v-vii
List of Tables	viii
List of Figures	ix-x
Chapter 1: Introduction	1-3
1.1 Introduction	1
1.2 Motivation	1-2
1.3 Objectives	2
1.4 Expected Outcome	2
1.5 Report Layout	3
Chapter 2: Background	4-10
2.1 Introduction	4
2.2 Related Works	4-5
2.3 Comparative Studies	6-9
2.4 Scope of the Problem	9-10
2.5 Challenges	10

Chapter 3: Requirement Specification	11-22
3.1 Business Process Modeling	11-12
3.2 Requirement Collection and Analysis	12-14
3.3 Use Case Modeling and Description	14-19
3.4 Data Flow Diagram	20-21
3.5 Design Requirements	22
Chapter 4: Design Specification	23-32
4.1 Front-end Design	23-28
4.2 Back-end Design	28-31
4.3 Interaction Design and UX	31
4.4 Implementation Requirements	32
Chapter 5: Implementation and Testing	33-39
5.1 Implementation of Database	33-36
5.2 Implementation of Front-end Design	36
5.3 Implementation of Interactions	37
5.4 Testing Implementation	37-38
5.5 Test Results and Reports	38-39
Chapter 6: Conclusion and Future Scope	40
6.1 Discussion and Conclusion	40
6.2 Scope for Further Developments	40
References	41-42
Appendix	43-44

List of Tables

Tables	Page No
Table 2.1: Comparison of Different Note-taking Application	8-9
Table 3.1: Use Case Description Sign Up	15
Table 3.2: Use Case Description Sign In	15
Table 3.3: Use Case Description Authentication	16
Table 3.4: Use Case Description of TakeNote	16
Table 3.5: Use Case Description of EditNote	17
Table 3.6: Use Case Description of DeleteNote	17
Table 3.7: Use Case Description of UploadPicture	18
Table 3.8: Use Case Description of ExtractText	18
Table 3.9: Use Case Description of SystemAction	18
Table 3.10: Use Case Description of DeleteUser	19
Table 3.11: Use Case Description of DeleteContent	19
Table 5.1: Integration Testing Result	38
Table 5.2: Validation Test	39

List of Figures

Figures	Page No
Figure 3.1: Business Process Modelling	11
Figure 3.2: Use Case Modelling	14
Figure 3.3: Data Flow Diagram of the application	20
Figure 4.1: Homepage	23
Figure 4.2: Sign Up Page	24
Figure 4.3: Sign In Page	24
Figure 4.4: Main Application	25
Figure 4.5: Create Note Form	25
Figure 4.6: Create Tag Form	26
Figure 4.7: All Tags	26
Figure 4.8: Image Upload form	27
Figure 4.9: Text from Image	27
Figure 4.10: About Page	28
Figure 4.11: Entity Relationship Diagram	30
Figure 5.1: Database Tables	34
Figure 5.2: Users table	34
Figure 5.3: Admins table	34
Figure 5.4: Tags table	35
Figure 5.5: Notes Table	35
Figure 5.6: Empty Submitted Table	37

Figure A1: Tesseract System Architecture	43
Figure A2: Tesseract Word Recognizer	43
Figure A3: MVC Architecture	44
Figure P1: Plagiarism Report	45

CHAPTER 1

INTRODUCTION

1.1 Introduction

From ancient time to our own modern time note-taking or the persistence of recording personal deeds and interests was present. This approach was followed nearly all the decades through human history to commemorate scientific development. Now instead of using the old paper-ink process, note taking has deduced to the software. It is recognized as a self-disciplinary act in cognitive science. There are many aspects and formations of organizing this complex human behavior that underlines the sole aspect of note-taking and organizing- processing knowledge and information. Its functionalities, approach, purpose, and the medium has changed with the flow of time and technology. But timidly it's divided into two methods: Linear and Non-linear note taking.

From this time on, we will focus to build and document kind of mixed of these two methods in our project to develop a simple yet powerful tool for reorganizing notes and related methodologies.

There are six Chapters in this document containing Introduction, Background, Requirement studies, Design Specification, Implementations and Testing and Future Scopes of this project. And the Introduction part is based on the motivation, project objectives, the immediate expected outcome of the project and a brief container for report items layout.

1.2 Motivation:

Reading, researching, scientific work through, taking tracks of personal, academic or professional activities all require to take notes; either short, brief or large amount of information.

And these processes regarding note-taking, often require a lot of reading; books, journals, research papers, blog posts and even forum or community posts. In case of learning computer programming, it is obvious to check and check the basics, keep track of the work through, practices, certain websites, bookmark portions of a book or useful repositories; so that one can always have a glimpse of his yesterday's work or even past months or years. So, in that way, it is necessary to combine all the notes categorically with one's interest and choice in one place. It's more important

to students to keep notes of their ongoing and upcoming reading, works or anything that will be needed later.

Suppose, one has read a book thousands page long, and a few months later you are in need of that book for some work, but the problem is he or she hasn't taken any notes. So, it's basically another span of time has to be spent to find what his/her need for that book. If the book were noted properly, he/she might be able to save their time by looking only in the notes, not the entire book. So, our primary motivation is to build a note organizer application that can help people to take linear notes on their useful information from books, classes, work places, webpages etc. The list would go on.

1.3 Objectives:

So, our primary observation is to build a note-taking web application, that will help people especially students, teachers, professionals on the busy schedule to summarize their tasks on shorthand notes or in-details notes. The user can customize his own way of categorizing the notes- alphabetically, by added time, by tag names and set reminders for time being.

We also tend to create new options like user can add text from a snipping picture of text. The app will generate a search button to search notes, items user have saved in it by tags, categories and note titles.

1.4 Expected Outcome:

The main thing for ourselves here is to build an application with a uniform UI so that its users can feel much ease and minimal complexity. There will options like add notes and under that, there will be some sub-sections like note title, description, category, tag; user also can add pictures of texts that will be converted to texts on note description if the user wants. Or he can save the sniping picture as a note as well. The search option will help to find any notes. User has to log in to the app and all the activities, notes will be saved to his account.

We will now consider only for the web application, which will be cross-platform and later we will generate a mobile application for our app.

1.5 Report Layout:

The following points are for the overall layout of this report.

- ✓ In the first chapter “Introduction”, project’s basic introduction, the motivate factors behind it and what we expect to be done is stated.
- ✓ In the second chapter titled “Background”, we have tried to understand various related works, compared them, searched for the scopes and challenges in our’s.
- ✓ In the third chapter “Requirement Specification”, we have looked at the functional and nonfunctional requirements of our project, deriving Use Cases, Data Flow Diagram to better understand the project’s structure.
- ✓ In the fourth chapter “Design Specification”, we have provided a details talk on designing Front-End, Back-End, Interactive designs and the implementation requirements. We have attached various design images of our system.
- ✓ In the chapter fifth chapter “Implementation and Testing”, we have included our implementations of Front-End and Back-End and testing various functionalities. Also, stating Testing results.
- ✓ In the sixth and last chapter “Conclusion and Future Scope”, we have figured out the major future developments of our project and what the current situation will work for the user accomplishments.

CHAPTER 2

BACKGROUND

2.1 Introduction:

This part of the document contains the theoretical dependencies of the project. The next few paragraphs will define how our project is related, inspired or has adherence to other applications of note taking; and a brief comparative study among different existing systems, scopes of our project and the complexities and challenges we are going to face to cover the project.

2.2 Related Works:

Note taking has been key prospect for people of all spheres. Now-a-days it is mostly valuable to almost everyone who needs to keep track of their tasks or schedules. Low level Artificial Intelligence programs will take this place of note taking automation one day, but for today for the mass generations of people need to take note anyway.

We have mentioned earlier, how note-taking approach has been a vital part of human timeline development and scientific improvement. The Ancient Greeks developed hypomnema, personal records on important subjects [1]. In the Renaissance and early modern period commonplace books which served as a similar artifact became popular. Philosopher John Locke once developed an indexing method, which rolled as a structure for commonplace books; like, it inspired another book, Bell's Common-Place Book, 'Formed generally upon the Principles Recommended and Practised by Mr Locke' [2] nearly a century later.

We will mention a few well developed software's of today and their basic needs:

Evernote:

Evernote is probably most popular and used application in this field. It has services for all major os and mobile platforms. It was developed by Evernote Corporation and has free and premium service for note saving. It is free for basic, but one has to pay for the premium packages for more storage. Bookmark, clippings can also be saved with the browser extension.

OneNote:

Microsoft's OneNote comes by default with the Windows 10. OneNote allows user to draw and save this as a note with the general note-taking feature. It is also free and has versions for different platforms.

Google Keep:

Google introduced this product as simple and minimal functionalities of note-taking that works just fine. It can be integrated with a browser to save notes while browsing internet.

AllMyNotes Organizer:

A Windows-based application that only takes notes in outline bulleting with indent. A user can import various file formats to save as notes. Has various features along as task scheduler (alarm, reminder etc.) and passwords saving on tree folders.

Ipython Notebook/Jupyter:

This notebook is not actually a note taking system, but it is for python programming and notes along side by side. A user can combine programming code, output, and annotations in a single interactive shell.

Basket Notepads:

A Unix-like(KDE) platform note taking software that enables a user to take basic notes, organized by tree / tags.

All of the approaches have quite similar features and plans. We will discuss more about them in the comparative studies. And also, the note taking approaches in general form.

As the system we tend to develop will be featuring an OCR (Optical Character Recognition) approach to save notes as texts from images. And some relational functionalities of modern day note taking software will be implemented.

2.3 Comparative Studies:

This portion of the chapter is focused on basic, advanced and other objectives among the existing software systems and also an overview of note taking in general and the forms, norms, speculations of it.

2.3.1 Note taking approaches:

Besides of what note-taking techniques, it can be broadly divided by two methods:

Linear Note-taking:

Outlines processing on linear note taking tend to proceed down a page, using headings and bullets to organize content. A Common pattern consists of headings that use Roman numerals, letters of the alphabet, and Arabic numerals at different levels.

An example would be:

I. Note1

A. Sub-section

1. Sub1

2. Sub2

II. Note2

A. Another Sub-section

This sort of approach has drawbacks too. In written notes, its cringe is to go back and editing piece of a section is messy.

Non-linear note-taking:

There are different types of non-linear indexing or note-taking techniques [4], including, Concept mapping, Clustering, Cornell system, Instant replays, Idea mapping, Knowledge maps, Learning maps, Mind mapping, Model maps, Semantic networks etc.

The following are details about a few.

Skeleton Prose:

Skeleton prose is the most common aspect among the note-taking formats. The notes are structured as numbered points and proses or paragraphs with indentations and headings. It can be thought of an essay plan, and most cases it has its own drawbacks. Like sequencing/amending the slowly building arguments from a book/article can be even difficult to add or amend. It can or can't indicate the connection or relation between the parts of the document. [5]

Cornell Note-taking system:

This note-taking approach was advised for Cornell University students and has publicized in the book 'Study Skills for University Students' by Walter Pauk [6]. Its purpose is to take structured, organized notes so that one can take an active look at the notes when revising. This format is necessary for students who have to engage in the summary of ideas for something. Can be used for taking notes during lectures or seminars, organizes knowledge bases for further reorganization and revision. [7]

2.3.2 Comparison based on functionalities:

There are many formats, techniques, approaches out there for notes, but these two are the most common of them all. And our project is about actively taking clear notes and having an OCR (Optical Character Recognition) approach to suit the image texts with the notes. Now in the following table we will attach some popular systems to have a comparative look over their functionalities and basic features:

Table 2.1 Comparison of Different Note-taking Application [8]

Applications	License	Organizing Principle	Search	File import/export/save formation	Platform
AllMyNotes	Proprietary, Freemium.	Tree	Yes	Proprietary, encrypted; import: plain text, RTF, CSV, images, HTML; exports: plain text, HTML, RTF, images	Windows
Basket Notes	Open Source (GPL)	Tree, Tags	Yes	Rich text; export as HTML; import text files.	KDE (Unix-based)
Day One	Proprietary, Commercial.	Chronological, Tags	Yes	Markdown (in GUI), XML (data file); Export as: pdf, txt, md.	iOS, MAC OS X.
Evernote	Proprietary, Freemium, Business Pricing.	Tags, notebooks, stacks.	Yes	Notes stored as XML; both free and paid versions allow notes of any file type	All major platforms.
Gnote	Open Source (GPL)	Notebooks	Yes	NoteXmlFormat , HTML, PDF	Linux

Google Keep	Proprietary, Free.	Tags, Colors.		export to Google Doc and thence to PDF, Word, ODT etc.	Android, Chrome OS, Web- based.
Ipython Notebook	BSD	Interactive Shell scripts.	N/A	Html files can be generated.	Cross Platform
Keynote NF	Open Source (NPL)	Notebooks, notes, tree.	Yes	Internal: combination of TXT and RTF.	Windows

2.4 Scope of the Problem:

This project is supposedly for building scalable and general purpose note-taking software that will include the basic functionalities of a modern note-taking software. And also, for the starting point, the advanced feature is to deal with the retrieving of texts from an image. Now, the part below will be for the scopes of our project we've observed so far:

2.4.1 Improve on basic functionalities:

First of all, for taking notes it needs to be organized properly. Maximum applications allow a user to structure them in tags/categories. We are considering to make it as per the user's customization. A user will be able to manage the settings for organizing type and also the interface setups.

2.4.2 Advanced formatting scopes:

Drawing over the texts on an image/snippet, marked text on notes, exporting notes to email/messaging, searching notes from an archive, auto saves, priority scheduler, link to other notes etc.

2.4.3 OCR Approach:

So far we have observed and seen different applications both with basic purpose and various other applications. Few applications have the file attachment features. So, we are experimenting with the images that contain text from pdf or any document or just an image; extract/ retrieve texts from it and save it as the note or save the image in a note/as a note.

So, the ‘Optical Character Recognition’ (OCR) on a note-taking app will do the task more easily for user to apprehend.

2.5 Challenges:

2.5.1 Data Manipulation:

Notes and back-end text dependencies, attachments will scale up-to huge data to manipulate, based on the number of account/people are using it. Primarily it is not a great deal, but in the long run, the data organization and availability in real time will be challenging to acquire.

2.5.2 Character Clarity:

Retrieving texts from an image will be an acute position to ensure clarity, because the numbers, letters, symbols will be difficult to retrieve for time to time as the image has low-resolution results to lower pixels and blurry image. And for any other language than English, the OCR tools and techniques are not solely available.

2.5.3 Storage Management:

Applications of large-scale data manipulations have certain drawbacks like storage capability. The greater the data synchronization, the greater the storage is. So, note-taking services will require dedicated or remote data centers to manage the services.

2.5.4 Scaling/Leveraging:

Now, if we consider all the load efficiency, data enlargement tackling and other performance issues, then scaling the application up-to performance standard is an important task. Scaling works up to many levels- data storage, retrieving, transaction measurement as per time, user access to real-time service and consistent design of the application to provide all the leveraging the app needs. And in broader cases, database scalability, network, hardware efficiency, storage leveraging are important factors in maintaining the capability of a system upon growing amount of work.

2.5.5 Security:

User authentication and managing user’s data and other records in terms of security are main factors. The system and its nodes should be able to detect fraud authentication or bot access. And the data presented on the user's hand have to be precise and perfect so that system doesn’t appear vulnerable. Plus the contents users are saving on the applications should pass through content filters to detect harmful contents.

CHAPTER 3

REQUIREMENT SPECIFICATION

3.1 Business Process Modeling:

Business process modeling (BPM) in business process management and systems engineering is the representation of processes of an enterprise, with the chance to the current process may be analyzed, re-organized and automated [9].

The following figure is representing the business process modelling of our application. The user has to register for further actions in the application.

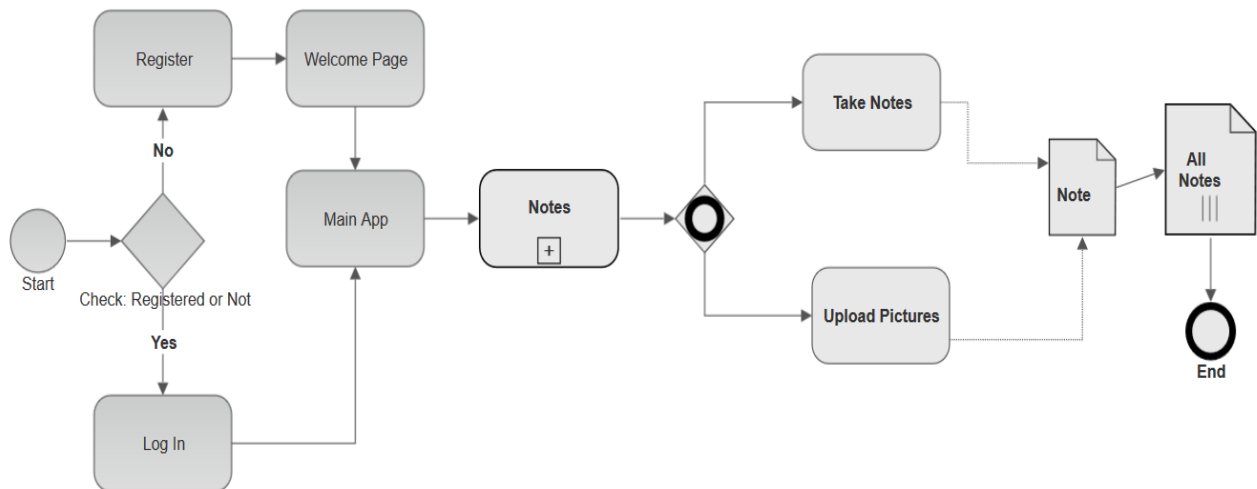


Figure 3.1 Business Process Modelling

❖ Business Process Modelling notations:

➤ Events:

Start: We start off as a new or registered user.

End: After all the required process is done, the notations are halted by an end event.

➤ **Flows:**

Sequence flow with conditions:

Denoted by an arrow front and diamond on the back from one Flow Object to another, via a Sequence Flow link, but is flowed under condition or checking. Basically, set as a flow of sequence between activities via gateways with condition [10].

Sequence Flow:

Flows proceed from one Flow Object to another, via a Sequence Flow link without any conditions. . Shown as the sequence of flows without any conditional dependencies.

➤ **Gateways:**

Exor:

The first gateway is an exclusive or gateway. Like one of the events are true the process will follow, not both but at least one. This gateway is a data-based exclusive pathways for decision and merging. Each user has to create an account to log in or registered, then the user will be redirected to the main application.

Inclusive Or:

The collapsed sub-process inside the application will merge the linear notes and the uploaded picture's text as notes. Which will be flown to the notes.

➤ **Sub-process:**

Collapsed sub-process will be expanded by the linear notes and text notes acquired from images.

3.2 Requirement Collection and Analysis:

Software Requirement Specification or SRS documentation is a part of requirements engineering in which it helps to understand the problem ones trying to solve [11]. Requirements analysis greatly depends on the type of the problem and the scope its having, plus description of resources components it's going to need to accomplish the task.

For our titled application, we have discussed the functional and nonfunctional requirements of the system, and then the hardware and software requirements.

3.2.1 Functional Requirements:

- ✓ Sign Up: If the user is new, than has to sign up to the site.
- ✓ Sign In: If already registered, than user can log in with valid credentials.
- ✓ Taking Notes: The user can take notes and save it for later use.
- ✓ Modifying note-contents: User can update or delete the created note at any time.
- ✓ Uploading Photos for notes: User can upload pictures to save as note.
- ✓ Text Extraction: User can extract text from the pictures and save it altogether.

3.2.2 Non-functional requirements:

❖ Site Usability:

- ✓ The systems primary goal is to provide a uniform look interface and feel between all the web pages.
- ✓ The system will provide use of icons and toolbars.
- ✓ The recent notes will go up.
- ✓

❖ Ease at use:

- ✓ The system tends to provide the user easy access and better control of everything.
- ✓ Initially the webpage will be in English, but Bangla will come afterwards.

❖ Security:

- ✓ The System should be able to delete inappropriate content and user.
- ✓ Users credentials should be saved in a secure database e.g. SSL encryption.

3.2.3 Hardware Specification:

- ✓ RAM: Nothing particular as it is a web application. But should be able to handle the browser operation space.
- ✓ Connection: Internet connection required.

3.2.4 Software Specification:

- ✓ Operating System: Any working OS.
- ✓ Browser: Any working browser, but Google Chrome or Mozilla Firefox preferred.

3.3 Use Case Modeling and Description:

A use case is a description of how a user will use the planned system to accomplish business goals [12]. It can be sketchy or descriptive. Both has importance on requirement engineering.

Below there is the use case diagram of our developing project, based on the requirements of our specification. We will start as brief descriptions on the use cases and then at the end the specifications will be covered with details description.

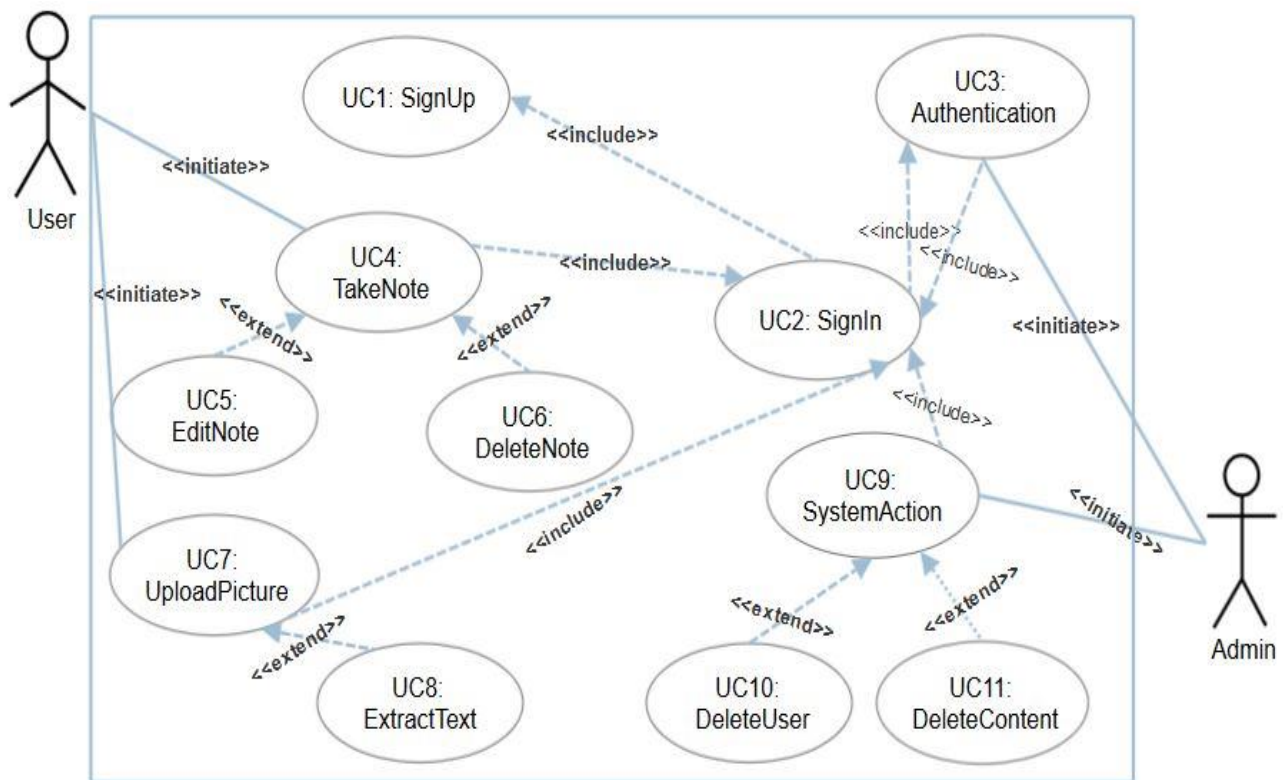


Figure 3.2 Use Case Modeling.

Table 3.1 describes the Sign Up process of our application. It includes four fields to complete and various exceptions.

Table 3.1 Use Case Description Sign Up

Use case name:	Sign Up
Actor:	Admin, User
Pre-condition:	None
Primary Path:	<ol style="list-style-type: none"> 1. Enter user name 2. Enter password 3. Confirm password 4. Enter email 5. Click “Sign Up” Button
Exceptional Path:	<ol style="list-style-type: none"> 1.1 user name is taken 1.2 name length is smaller than required 2.1 Password length is not valid, min 8 characters. 3.1 Password didn’t match 4.1 Email format is not valid

Table 3.2 describes the Sign In process of our application. It includes two fields to complete and various exceptions. User needs to input valid email and password, given in Sign Up.

Table 3.2 Use Case Description Sign In

Use case name:	Sign In
Actor:	Admin, User
Pre-condition:	Sign Up
Primary Path:	<ol style="list-style-type: none"> 1. Enter email 2. Enter password
Exceptional Path:	<ol style="list-style-type: none"> 1.1 User email didn’t match 2.1 User Password is invalid

Table 3.3 describes the Sign In process of our application. It gives Admin the functionality to check user authentication which is elaborated on Admins other Use Case's.

Table 3.3 Use Case Description Authentication

Use case name:	Authentication
Actor:	Admin
Pre-condition:	Sign In
Primary Path:	1. Check user authentication
Exceptional Path:	1.1 Admin is not Signed In

Table 3.4 describes the process of taking notes by user. A user has to input note's title, select tags, write the body or description and then click 'Save' button to save it in his account. Exceptions are title text length exceeded, any field is empty.

Table 3.4 Use Case Description of TakeNote

Use case name:	TakeNote
Actor:	User
Pre-condition:	Sign In
Primary Path:	1. Enter title 2. Select Tags 3. Short description 4. Click "Save"
Exceptional Path:	1.1 Limit Exceeded 2.1 Field Empty 3.1 Empty Field

Table 3.5 describes the process of editing notes by user. A user has to input note's new title, select tags, edit the body or description and then click 'Update' button to save it . Exceptions are title text length exceeded, any field is empty.

Table 3.5: Use Case Description of EditNote

Use case name:	EditNote
Actor:	User
Pre-condition:	Sign In
Primary Path:	<ol style="list-style-type: none"> 1.Edit title 2. Select Tags 3. Edit description 4. Click "Update"
Exceptional Path:	<ol style="list-style-type: none"> 1.1 Limit Exceeded 2.1 Not Selected 3.1 Empty Field

Table 3.6 describes the process of deleting notes by user. A user can delete a note by selecting 'Delete' button and then confirm it.

Table 3.6: Use Case Description of DeleteNote

Use case name:	DeleteNote
Actor:	User
Pre-condition:	Sign In
Primary Path:	<ol style="list-style-type: none"> 1. Delete Note
Exceptional Path:	<ol style="list-style-type: none"> 1.1 Already deleted or doesn't exist.

Table 3.7 describes the process of uploading images by a user. It can be JPG,PNG,BMP, TIF format. Otherwise it will throw an exception.

Table 3.7: Use Case Description of UploadPicture

Use case name:	UploadPicture
Actor:	User
Pre-condition:	Sign In
Primary Path:	1. Upload Picture
Exceptional Path:	1.1 Image is too large or not in the proper format. 1.1 Image background is noisy.

Table 3.8 describes the process of extracting text from the uploaded image by a user. The image is passed on Tesseract OCR to get the text.

Table 3.8: Use Case Description of ExtractText

Use case name:	ExtractText
Actor:	User
Pre-condition:	Sign In, UploadPicture
Primary Path:	1. Click “Save” after uploading picture.
Exceptional Path:	1.1 Image is noisy 1.2 Image is too large in pixels 1.3 Image language is not Bangla or English.

Table 3.9 describes the process of system or admin can initiate some action against infringement.

Table 3.9: Use Case Description of SystemAction

Use case name:	SystemAction
Actor:	Admin
Pre-condition:	Sign In
Primary Path:	1. Initiate action based on infringement.
Exceptional Path:	None

Table 3.10 describes the process to delete a user from system, permanently or temporarily.

Table 3.10: Use Case Description of DeleteUser

Use case name:	DeleteUser
Actor:	Admin
Pre-condition:	Sign In
Primary Path:	1. Delete scam user.
Exceptional Path:	None.

Table 3.11 describes the process to delete inappropriate contents from system, permanently or temporarily.

Table 3.11: Use Case Description of DeleteContent

Use case name:	DeleteContent
Actor:	Admin
Pre-condition:	Sign In
Primary Path:	1. Delete inappropriate contents added by user. And giving warning.
Exceptional Path:	None.

3.4 Data Flow Diagram:

A data flow diagram (DFD) illustrates how a system maintains the flow of data in its functionalities, with its inputs and outputs. The name indication is the clue here, that, it imagines the flow of information, as where data comes from, where it is stored and processed [13].

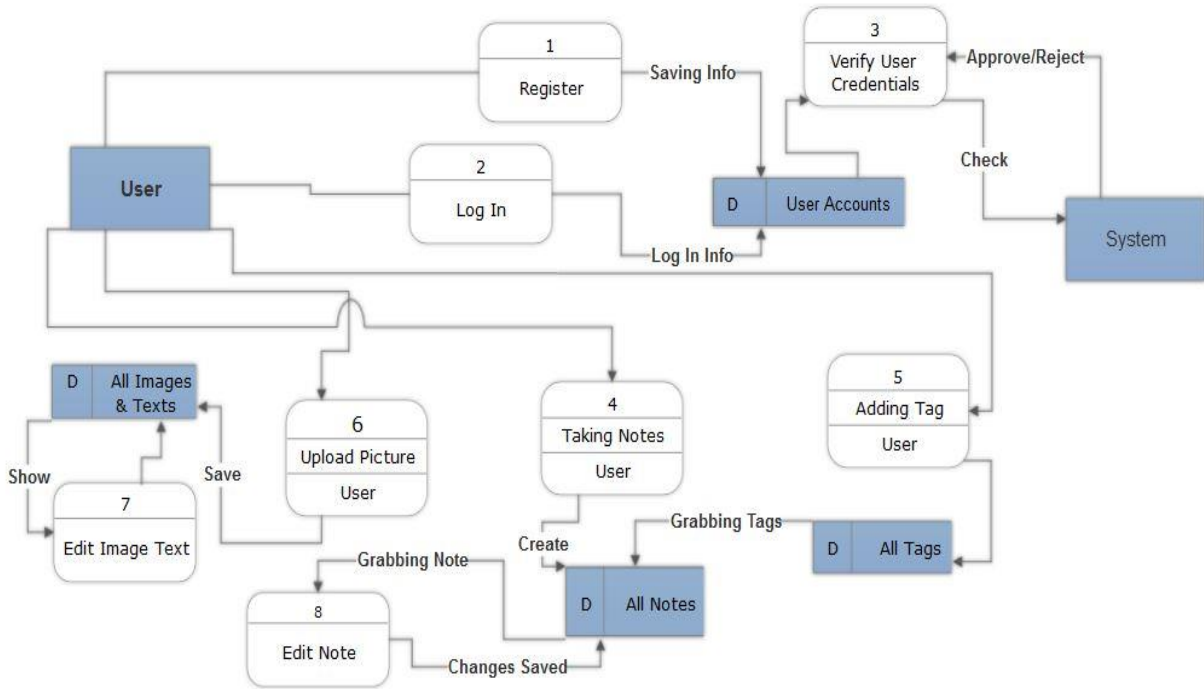


Figure 3.3: Data Flow Diagram of the application

There are various processes for representing data flow diagrams (Yourdon & Coad, Gane & Sarson). In our document we have used Gane and Sarson, which is used for visualizing information systems of a system. We have discussed various process notations and outside entities in the system used in the dfd, below:

Process Notations:

A process notation is something that visualizes the incoming data flow into outgoing data flow. In other words, process notations represent the processes going on the system. Here we have eight process notations:

1. **Register:** User registers in the application with valid credentials.
2. **Log In:** To use the app after registration it is obvious to log in the system.
3. **Verify User Credentials:** The system will make sure that the user credentials are alright.
4. **Taking Notes:** User can add notes and the data will be stored in the All Notes datastore.
5. **Adding Tag:** User can add tag to use it on notes.
6. **Edit Note:** User can get the note from datastore and save it back.
7. **Upload Picture:** User can upload images.
8. **Edit Image Text:** Text extraction from image can be imperfect based upon image's quality, so user can edit the text.

Datastore Notations:

Datastore represents the data storage of the application or system. It is also referred as files. In our diagram they are addressed as 'D' on the start. Here, we have four data store notations,

1. **User Accounts:** All user account data is stored here.
2. **All Notes:** User's taken note is stored.
3. **All Tags:** Data of the tags are stored.
4. **All Images & Texts:** Uploaded images and corresponding texts are saved here.

Dataflow Notations: Dataflow notations are used to show the flow of data between the process and entities. It is represented as an arrow from the incoming data flow to the outgoing flow.

External Entities:

External entities are the ones who interacts with the process notations. In our case, there are two external entities, User and System.

1. **User:** User interacts with the app directly and uses it functionalities.
2. **System:** The system also interacts with the functionalities, but it verifies that everything is going fine.

3.5 Design Requirements:

1. Home page for guest/visitor.
2. Different homepage for the logged in user.
3. Sign in page
4. Sign out page.
5. User should be able to take notes on the main applications by signing in.
6. No functionalities will be allowed unless user is logged in.
7. Uniform UI and responsive based on screen length.
8. User should be able to add tags then attach it to a note.
9. User have the options to upload picture and then extract texts from it.
10. The design will be as per ease usage for the users/system admin.
11. User can add, edit and delete contents anytime, when logged in.
12. Admin dashboard will allow admins to delete or check on anything unusual or infringes with the general terms of the system.

CHAPTER 4

DESIGN SPECIFICATION

4.1 Front-end Design:

Front end design is the view of an application or system. It is something that an end user will use and interact with. It is the GUI or Graphical User Interface that the user watch while communicating with the system. So, it plays an important rule to gain user. As simple, usable reliable and uniform UI is required to make a good application.

Now we will show and discuss our applications UI's a bit:

4.1.1 Home Page:

The home page contains the 'Sign In' and 'Sign Up' buttons. It's the page which the users/guest will see upon opening the application. It also contains little overview of our system. And additional carousel pictures and 'Pricing', 'About' button.

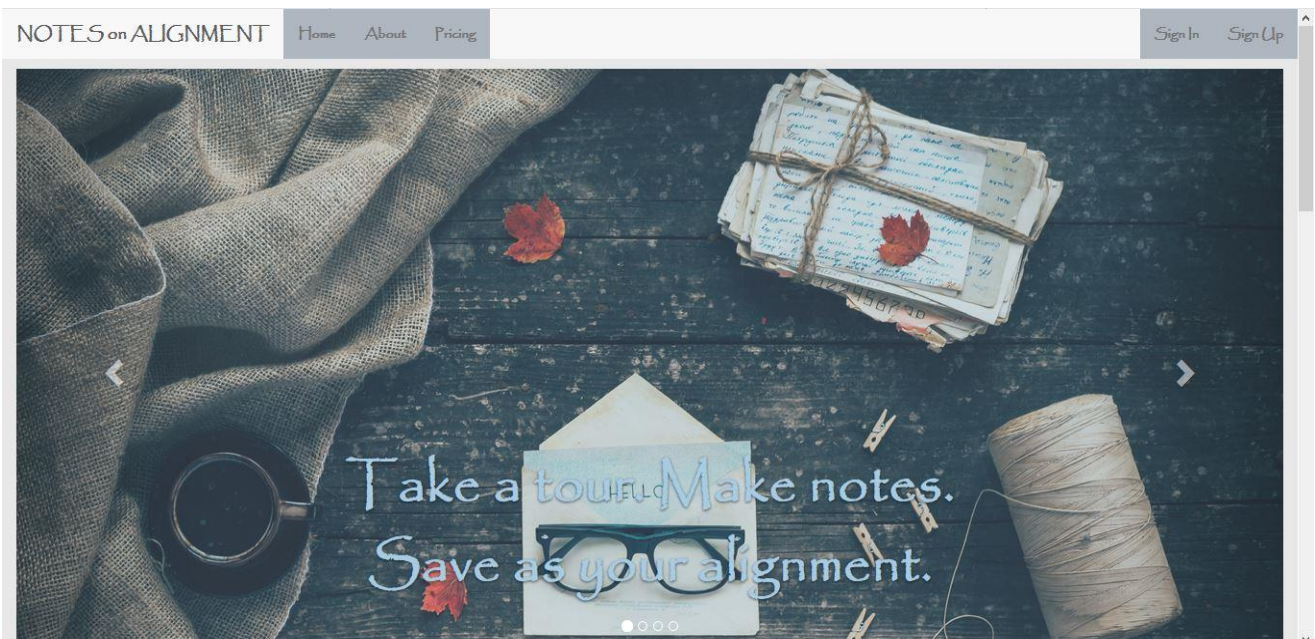
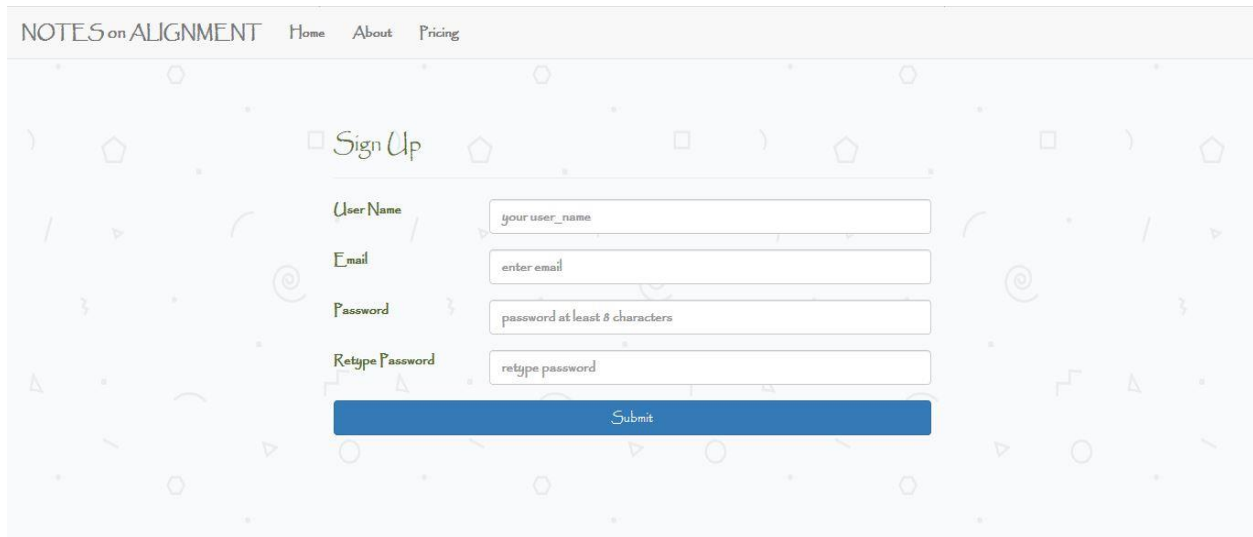


Figure 4.1: Homepage

4.1.2 Sign Up:

Sign up page is for user registration for the system. It contains a registration form for user. The user can register with valid credentials. The user credentials will be as per the rules of the system, which will be on the FAQ page of our application. The interface is like:

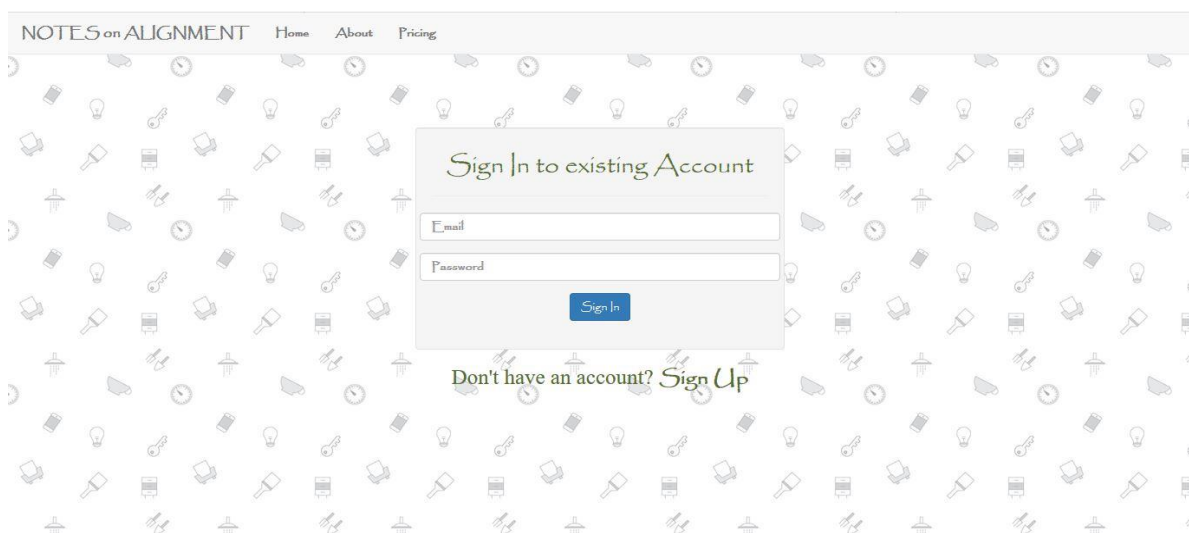


The screenshot shows a web page titled "NOTES on ALIGNMENT" with navigation links for "Home", "About", and "Pricing". The main content area features a "Sign Up" form. The form includes four input fields: "User Name" (placeholder: "your user_name"), "Email" (placeholder: "enter email"), "Password" (placeholder: "password at least 8 characters"), and "Retype Password" (placeholder: "retype password"). A blue "Submit" button is positioned below the fields. The background of the page is decorated with a pattern of small, light-colored geometric shapes.

Figure 4.2: Sign Up Page

4.1.3 Sign In:

After user have successfully signed up or registered. They can sign in with their password and email.



The screenshot shows a web page titled "NOTES on ALIGNMENT" with navigation links for "Home", "About", and "Pricing". The main content area features a "Sign In to existing Account" form. The form includes two input fields: "Email" and "Password". A blue "Sign In" button is positioned below the fields. Below the form, there is a link that says "Don't have an account? Sign Up". The background of the page is decorated with a pattern of small, light-colored icons representing various objects like a lightbulb, a key, a clock, and a pencil.

Figure 4.3: Sign In Page

4.1.4 Main Application:

After the user has successfully logged in or registered, they will be redirected to the homepage of the main application. Where the main magic happens. User can create, update, delete notes, to-do lists and tags, upload images and extract text from it. Also here the user notes are shown from latest.

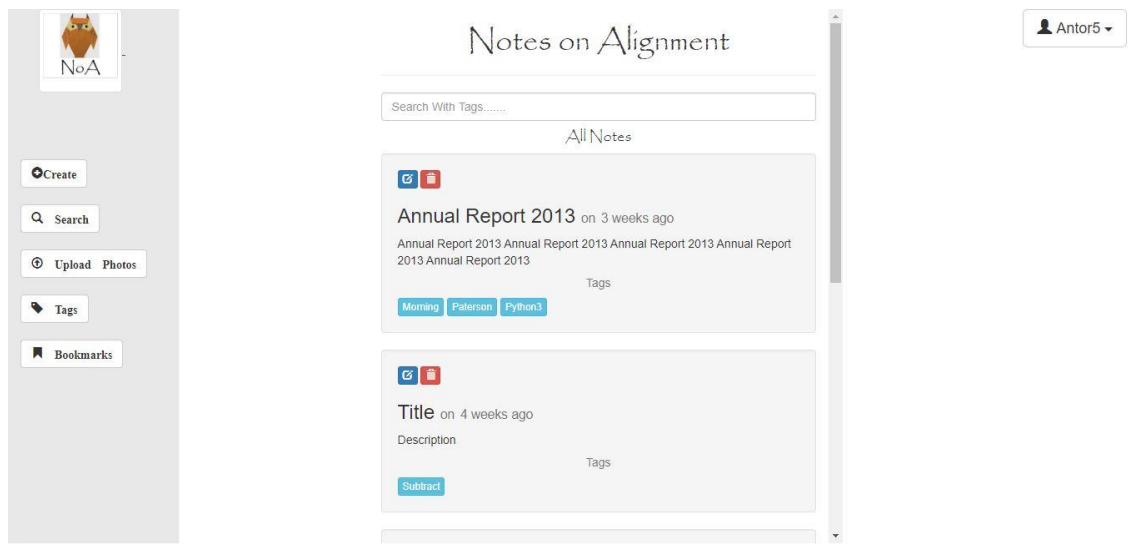


Figure 4.4: Main Application

4.1.5 Create Notes:

The sub-section of the main application. User can fill up the details about the notes and save it.



Figure 4.5: Create Note Form

4.1.6 Add Tags:

The add tag part is to create a tag for the notes. The user can attach the tags to the notes. But, they have to create the tag from here first. Just the title and pressing 'Save' will do the saving.

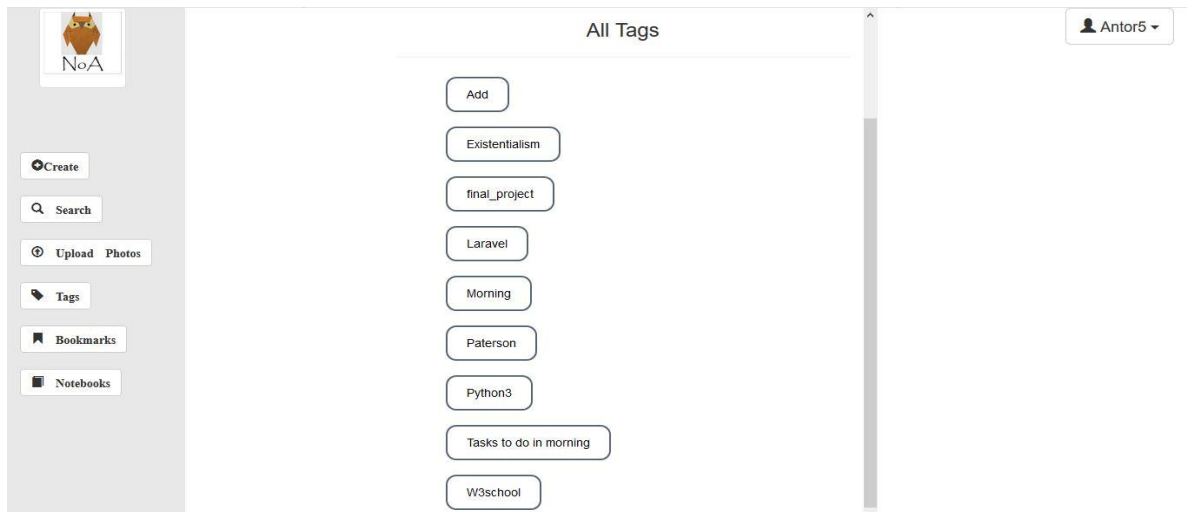


The screenshot shows a web interface for 'Notes on Alignment'. On the left is a sidebar with a user profile 'NoA' and navigation buttons: Create, Search, Upload Photos, Tags, Bookmarks, and Notebooks. The main content area has the title 'Notes on Alignment' and a user dropdown 'Antor5'. Below the title are two tabs: 'Add Tags' (active) and 'Tags'. The 'Add Tags' tab contains a text input field with the placeholder 'Give it a name' and a 'Save' button.

Figure 4.6: Create Tag Form

4.1.7 View Tags:

The added tags via the 'Add Tag' form will be displayed alphabetically on this section.



The screenshot shows the 'All Tags' page. The sidebar is identical to Figure 4.6. The main content area has the title 'All Tags' and a user dropdown 'Antor5'. Below the title is a vertical list of tags, each in a rounded rectangular button: Add, Existentialism, final_project, Laravel, Morning, Paterson, Python3, Tasks to do in morning, and W3school. A vertical scrollbar is visible on the right side of the list.

Figure 4.7: All Tags

4.1.8 Image Uploads:

The main functionality of our application is to extract texts from an image. So, this section will be for uploading images and save it to database. User has to upload only image files (jpeg, png, tif, bitmap). Otherwise it will not be uploaded.

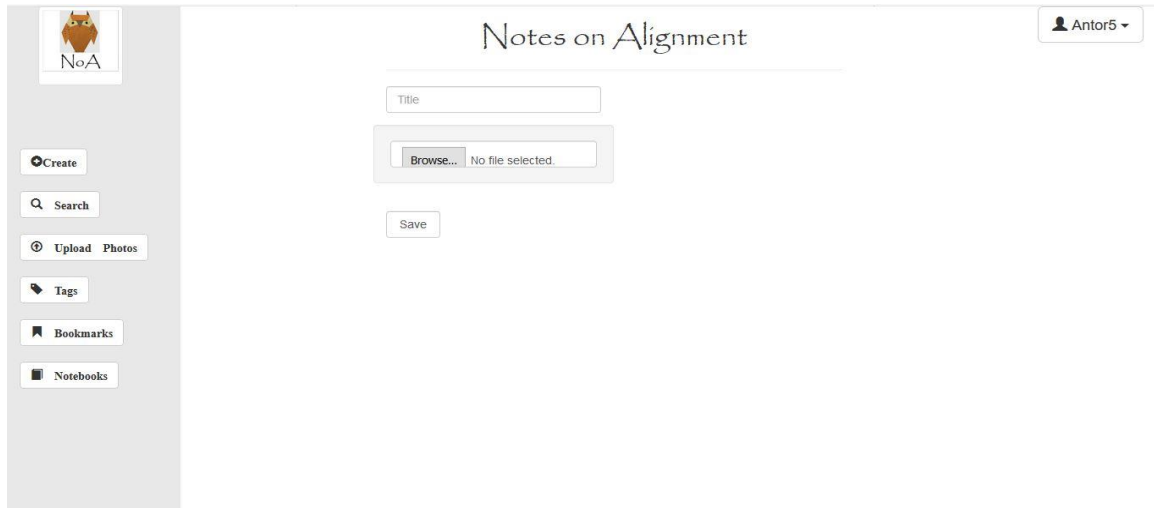


Figure 4.8: Image Upload form

4.1.9 See Text and Uploaded Image:

To extract image texts it will be passed on Tesseract OCR engine to generate the result.

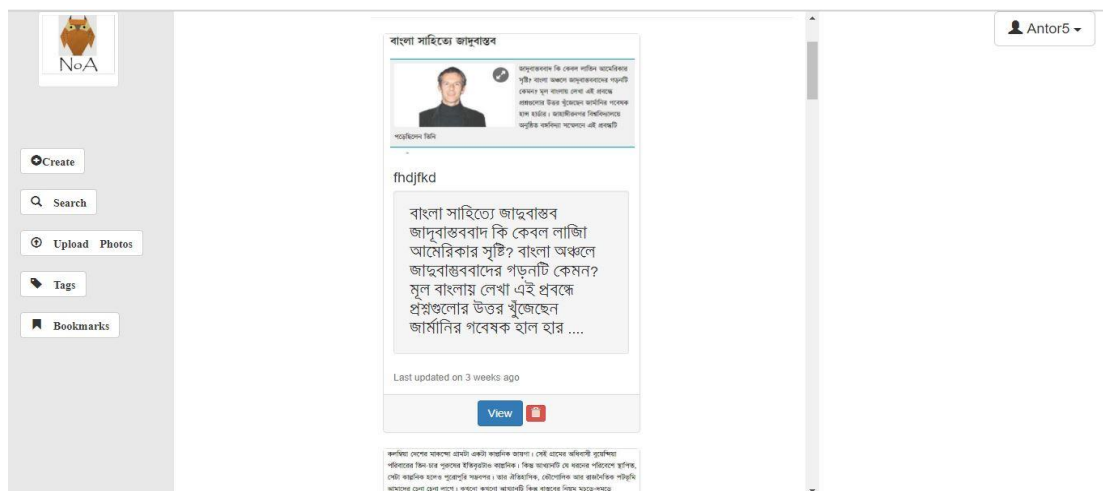


Figure 4.9: Text from Image

4.1.10 About Page:

The about page will contains who we are and what is we up to.



Figure 4.10: About Section

4.2 Back-end Design:

Back-end of an application means the rendering and storing of data the user never sees. It is the operations with database, server and content delivery networks that altogether serves user in the user end or front-end. Now that we have talked about the front-end, and what it will be look like, let's take a look at our back-end for a while.

4.2.1 Database:

The powerful application's data model is the most critical part of it. Database integration is the thing that gives dynamicity of an application or system. Now we will explore our E-R Diagram (Entity Relationship Diagram).

Entity Relationship Diagram:

The entity relationship data model or E-R model is used to advance the database design with specified enterprise scheme which lays the structure of the logical structure of the database [14]. It is an important notation of an application as it maps the main interactions and collective data

handling of the system. E-R model has three basic concepts behind it: Entity sets, Relationship sets and Attributes. The following is the description of our E-R diagram in brief.

Entity Sets:

An entity is like a thing or object in the real world, which can be distinguished from another thing or object. For example, each person in a family is an entity. An entity has many attributes, and values for some set of properties can genuinely identify an entity [15].

In our application, there are six total entity sets. For each entity except Admin there's a foreign key, as user_id, to uniquely identify a user's content.

- ✓ User: The entity set user includes user_name, email, password and uses ID as primary key. User's account credentials will be saved as these attributes.
- ✓ Admin: Admin entity set contains admin_name, email, and password and uses ID as primary key.
- ✓ Note: Notes entity includes a user_id as foreign key, which is from the User entity, and also other attributes title, description, tags, and an optional reminder. Each note will be saved and retrieve using these attributes.
- ✓ Tag: Tag entity has attributes title, primary key ID a user_id. Tags will be updated and used to notes using these attributes.
- ✓ Image: Image corresponds to the user uploaded image. It has an ID as primary key, the image, edited_image, title and image_text.
- ✓ Todo_list: To-do list maintains user's list of doing things. It has ID as primary and user_id for identification, plus list of contents.

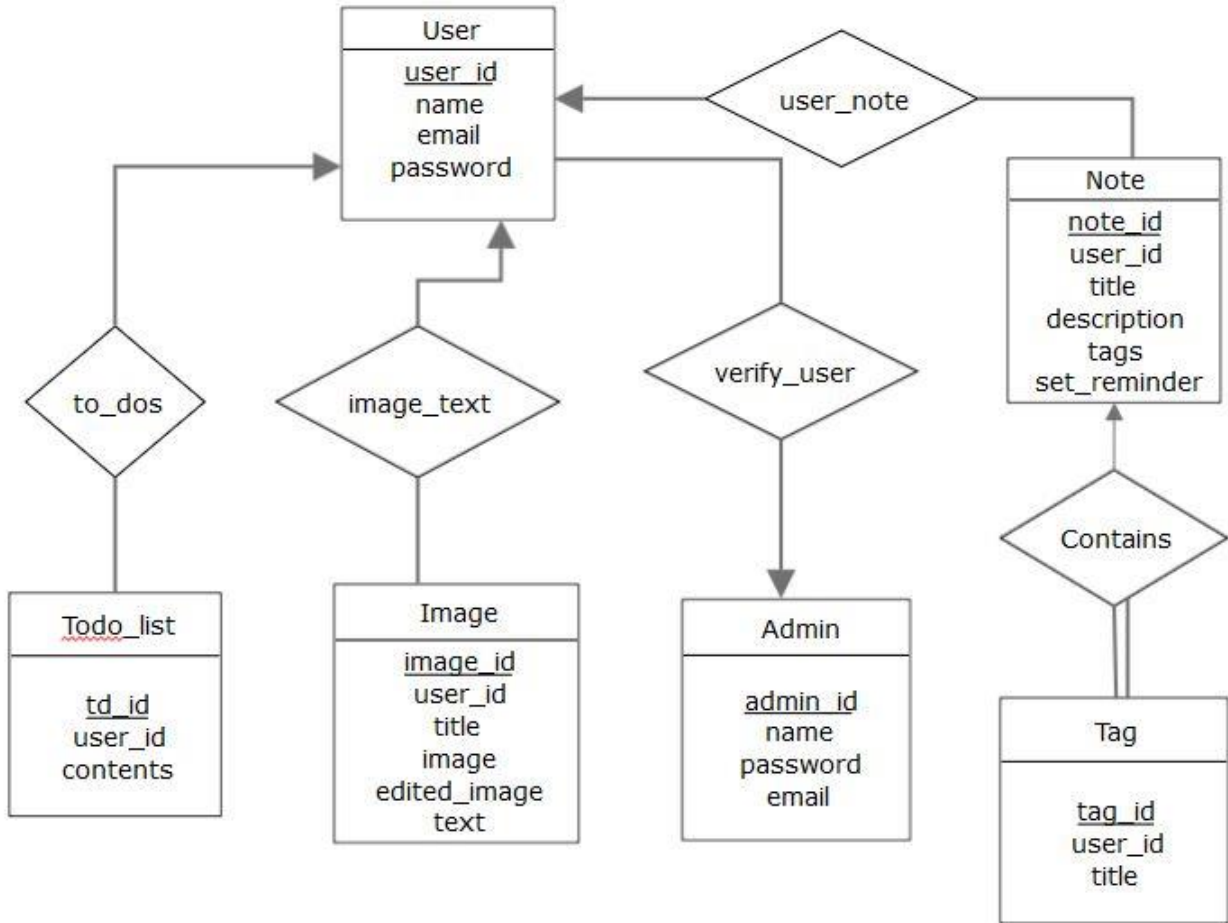


Figure 4.11: Entity Relationship Diagram

Relationship Sets:

A relationship is an association among the entity sets. In our E-R design, we have five relationship sets:

- ✓ **user_note:** relating users with notes they have taken.
- ✓ **contains:** relates tags with notes.
- ✓ **verify_user:** checks user credentials before using the system via Admin.
- ✓ **image_text:** relates users with uploaded image and its texts.
- ✓ **to_dos:** Relates to-do lists to a particular user.

Attributes:

Entities with attributes:

- ✓ **User:** with attributes (**user_id**, name, email, and password).
- ✓ **Note:** with attributes (**note_id**, user_id, title, description, tags, set_reminder).
- ✓ **Tag:** with attributes (**tag_id**, user_id, title).
- ✓ **Admin:** with attributes (**admin_id**, name, password, email).
- ✓ **Image:** with attributes (**image_id**, user_id, image, edited_image, title, text).
- ✓ **Todo_list:** with attributes (**td_id**, user_id, contents).

4.3 Interaction Design and UX

An applications purpose is to help its end user for particular functionalities. So, end-user is the main entity of an application. So, what the user wants is smooth interactive system with minimal difficulties. Interaction design includes:

4.3.1 Structuring Contents:

Structuring application content as such that it is easier to find the functionalities by a user. We have tried to do this as much in our project to keep the UI simple.

4.3.2 Usability:

Usability of an application depends on the user's accomplishment of a certain task in the application with ease and without much difficulty. We have used minimal design patterns to keep the app ease to use.

4.3.3 Human Computer Interaction:

Human Computer Interaction (HCI) represents the design and evaluation of interactive computing systems that human use and the study surrounding that phenomena [16].

So, a system architect has to be able to understand the users/stakeholders mindset and thinking while developing the system. It is tough and we have to keep in mind the reason why end user will be interested in the application.

4.4 Implementation Requirements

Here is the full stack of our application we have used:

- ✓ Developing Tools/IDE: Sublime Text 3, PhpStorm
- ✓ Server: Apache
- ✓ Database: MySQL
- ✓ Languages: PHP (Back-end), HTML, CSS and JavaScript (Front-end).
- ✓ Frameworks: Laravel 5.5 (Back-end), Bootstrap (Front-end)
- ✓ JavaScript Libraries: jQuery, select2.
- ✓ OCR Engine: Tesseract (see more at Appendix A)

CHAPTER 5

IMPLEMENTATION AND TESTING

5.1 Implementation of Database:

As we mentioned in last chapter, that we are using Laravel as our backend tasks. Now, in here we will talk about how the contents are aligned at user level. (See Appendix A. for more about MVC and Laravel's lifecycle.)

Here's what happens when a user requests for a page:

- i. The responsible router of that http request triggers the Controller function for that data.
- ii. Controller renders the view with that data, if it requires the data from the database, it requests to the Model for that data.
- iii. Model than acquires the data and passes to the controller to render it to the view.
- iv. If user submit something to the system via http post request, the controller validates the submitted data and then send it to the database through Model, based on proper scheme.

In previous chapter we have stated that we will use MySQL as our database management system. Along with PHP and its Laravel framework. Based on the entities we have designed, we have made relational database table upon the design. Here's what we have implemented:

Database tables:

- ✓ users
- ✓ admins
- ✓ tags
- ✓ notes
- ✓ migrations
- ✓ password_restes

Table	Action	Rows	Type	Collation	Size	Overhead
<input type="checkbox"/> admins	★ Browse Structure Search Insert Empty Drop	0	InnoDB	latin1_swedish_ci	16 KiB	-
<input type="checkbox"/> images	★ Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16 KiB	-
<input type="checkbox"/> migrations	★ Browse Structure Search Insert Empty Drop	5	InnoDB	utf8mb4_unicode_ci	16 KiB	-
<input type="checkbox"/> notes	★ Browse Structure Search Insert Empty Drop	3	InnoDB	utf8mb4_unicode_ci	16 KiB	-
<input type="checkbox"/> password_resets	★ Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16 KiB	-
<input type="checkbox"/> tags	★ Browse Structure Search Insert Empty Drop	9	InnoDB	utf8mb4_unicode_ci	32 KiB	-
<input type="checkbox"/> users	★ Browse Structure Search Insert Empty Drop	2	InnoDB	utf8mb4_unicode_ci	32 KiB	-
7 tables	Sum	19	InnoDB	latin1_swedish_ci	144 KiB	0 B

Figure 5.1: Database Tables

5.1.1 Table Structures:

- ✓ users table: columns (**id**, name, email, password, remember_token, created_at, updated_at)

#	Name	Type	Collation	Attributes	Null	Default	Co
<input type="checkbox"/> 1	id 🔑	int(10)		UNSIGNED	No	None	
<input type="checkbox"/> 2	name	varchar(191)	utf8mb4_unicode_ci		No	None	
<input type="checkbox"/> 3	email 📧	varchar(180)	utf8mb4_unicode_ci		No	None	
<input checked="" type="checkbox"/> 4	password	varchar(191)	utf8mb4_unicode_ci		No	None	
<input type="checkbox"/> 5	remember_token	varchar(100)	utf8mb4_unicode_ci		Yes	NULL	
<input type="checkbox"/> 6	created_at	timestamp			Yes	NULL	
<input type="checkbox"/> 7	updated_at	timestamp			Yes	NULL	

Figure 5.2: Users table

- ✓ admins table: columns (**id**, name, email, password, remember_token, created_at, updated_at)

#	Name	Type	Collation	Attributes	Null	Default
<input type="checkbox"/> 1	id 🔑	int(11)			No	None
<input type="checkbox"/> 2	name	varchar(100)	latin1_swedish_ci		No	None
<input type="checkbox"/> 3	password	varchar(100)	latin1_swedish_ci		No	None
<input type="checkbox"/> 4	email	varchar(100)	latin1_swedish_ci		No	None
<input type="checkbox"/> 5	remember_token	varchar(100)	latin1_swedish_ci		No	None
<input type="checkbox"/> 6	created_at	timestamp		on update CURRENT_TIMESTAMP	No	CURRENT_TIMESTAMP
<input type="checkbox"/> 7	updated_at	timestamp			No	0000-00-00 00:00:00

Figure 5.3: Admins Table

- ✓ tags table: columns (**id**, user_id, name, created_at, updated_at)

#	Name	Type	Collation	Attributes	Null	Default
<input type="checkbox"/> 1	id	int(10)		UNSIGNED	No	None
<input type="checkbox"/> 2	user_id	int(11)			No	None
<input type="checkbox"/> 3	name	varchar(191)	utf8mb4_unicode_ci		No	None
<input type="checkbox"/> 4	created_at	timestamp			Yes	NULL
<input type="checkbox"/> 5	updated_at	timestamp			Yes	NULL

Figure 5.4: Tags Table

- ✓ notes table: columns (**id**, user_id, title, tags, body, created_at, updated_at)

#	Name	Type	Collation	Attributes	Null	Default
<input type="checkbox"/> 1	id	int(10)		UNSIGNED	No	None
<input type="checkbox"/> 2	user_id	int(11)			No	None
<input type="checkbox"/> 3	title	varchar(191)	utf8mb4_unicode_ci		No	None
<input type="checkbox"/> 4	tags	varchar(191)	utf8mb4_unicode_ci		No	None
<input type="checkbox"/> 5	body	text	utf8mb4_unicode_ci		No	None
<input type="checkbox"/> 6	created_at	timestamp			Yes	NULL
<input type="checkbox"/> 7	updated_at	timestamp			Yes	NULL

Figure 5.5: Notes Table

- ✓ migrations table: It contains all the migrations done from the beginning of the database creation.
- ✓ password_resets table: It is generated by Laravel's Auth Façade that contains the token and email field.

5.1.2 Models:

From implementation requirements, we have stated that we will use PHP and its Laravel web framework. Laravel is an MVC (Model-View-Controller) architecture framework. But its architecture is much more complex, it also has routing and middleware that restricts certain user to visit a page and vice-versa.

In our application, the Models are the representations of our main entities. In here, the Mapping cardinalities (relationships) of our database are coded. Let's explore our relationships on the Models:

One-to-Many:

In 'User' model, we have used the ElequentORM relationship keys. A user may have as many notes, tags and images. So, in User model we defined this using **hasMany** method of Eloquent.

BelongsTo:

One note, tag, image should be belong to a user. In Image, Tag and Note models, we defined this using ElequentORM's **belongsTo** method.

5.2 Implementation of Front-end Design:

In the front end design part we have stated various parts of our projects that will be in the application. And we have attached various design images of our design implementations. Now we will discuss the implementations phases of our design.

Markup:

At first we have used HTML to mark the elements on our application pages. Then we have used CSS properties to handle the style properties of our markup pages.

Grid:

We've used Bootstrap 3 for our application gridding. In Bootstrap grid, there are 12 columns to work with and rows can be as any number.

Blade Template:

After the markup and arrangement done, we integrated the .html and design files into the Laravel framework's view. Laravel requires to make these files into a blade extension file. Blade's modified syntax is useful working with the template.

Front-end as View:

To finally view the template, we have to set the routes with proper http request methods. And each route corresponds to a function of the responsible controller. I.e. the Image controller will be for image upload and operations.

5.3 Implementation of Interactions:

5.3.1 User Interactions:

Sign In: User has to sign with the valid credentials required to sign in to the system. Empty or wrong credentials will generate errors.

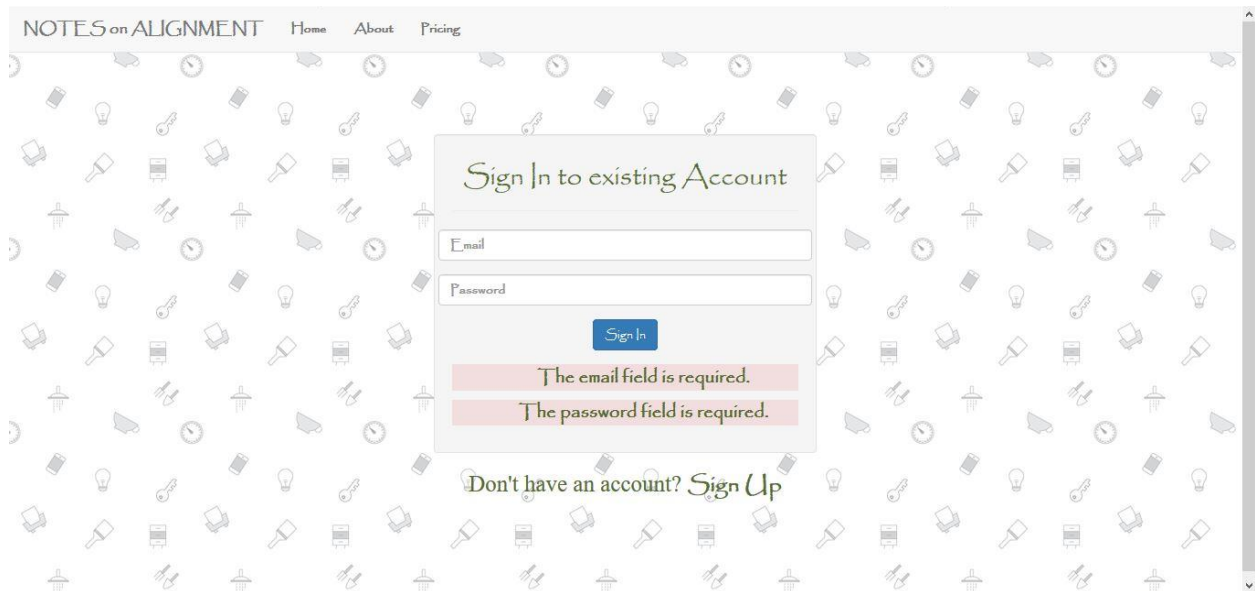


Figure 5.6: Empty submission result

Log Out: User can sign out easily by selecting 'Log Out' under user name toggle.

Take, Save and Modify Note: User can save, edit and delete notes easily.

Tags: User can save, edit, and delete tags as well.

Images: User can upload, extract texts and edit and delete the image and text if it's necessary.

5.4 Testing Implementation:

Unit Testing:

Unit testing is a software testing technique that are used to test different units of a software. We can use it to test our individual modules and render the results to check the activities.

Implementation issues:

✓ **Integration:**

System integration testing is a testing process that checks a software system's coexistence with others. In a multiple environment, required that each has passed system testing, Integration Testing tries to test the required interactions. Our application will be run on web. So, a web browser with almost all latest JavaScript binding will work fine [17].

✓ **Usability:**

Usability is something that can be tested with end users, on how they will accomplish their tasks through the system. More easily, smoothly users can interact is required for a good system. We will check the user interactions as well. And in testing results the result will be updated [18].

✓ **Validation:**

Validation testing is a software testing to check if the system meets the fulfillments of its specifications and intended purpose. It is also named as Software Quality control. In next section, we will represent the results and report of our testing.

5.5 Test Results and Reports

Integration Testing:

Table 5.1 Integration testing result

Test Case	Expected Result	Observed Result	Test Result
Server can accept request and reply accordingly.	Can do both.	Can do both.	Pass
User can load the application properly.	Site working properly.	Site working properly.	Pass

Validation Test:

Table 5.2 Validation Test

Test Case	Expected Result	Observed Result	Test Result
User Sign Up	Can do with required and valid credentials.	Can do.	Pass.
User Sign In	Successful with valid credentials.	Successful	Pass.
Adding Note with tags	Should be able to save with required info.	Can do.	Pass.
Updating/Deleting note	Can be done Successfully.	Success.	Pass.
Add tags	Yes	Yes, can do.	Pass.
Update/Delete Tags	Can do.	Can do	Pass.
Image Upload/Text Extraction	Successful with system defined rules.	Success.	Pass
Admin Sign In	Can do after successful Sign Up	Can do.	Pass.

Chapter 6

Conclusion and Future Scope

6.1 Discussion and Conclusion:

'Notes on Alignment' focuses on good user experiences. As they use and work on the system, it allows them to add notes, tags around them, image to text extraction, to-do list. As we use Google's Tesseract OCR engine to read and extract text from the images. It can recognize a wide range of languages. But in our project we will focus only on Bangla and English language recognition. In that way it will take less time to process the data. Some features and benefits of our applications are:

- The application can be used as personal daily to-do organizer
- So, that a user can plan a better daily routines to work with
- It allows to extract texts from an image, which will save a lot of time rather than typing the whole text on notes.
- Laravel's default CSRF token matching will help to eliminate any cross site forgery.

6.2 Scope for Further Developments

- Applying Deep Learning to recognize handwritten characters from image. Right now we are using Tesseract OCR which cannot recognize handwritten characters.
- Sharing user's contents on other media or via email.
- Using scripted bots to check user contents before saving to database.
- Introducing business plan on storage of contents.

Our project has the potential to integrate various techniques and features, and make it more engaging to the users. It will only benefit the user to organize and systematize their life in their own way.

References

- [1] Note-taking, available at << <https://en.wikipedia.org/wiki/Note-taking/> >>, last accessed on 01-07-2017 at 9.30pm.
- [2] Note-taking: History, available at << <https://en.wikipedia.org/wiki/Note-taking#History/>>>, last accessed on 01-07-2017 at 9.40pm.
- [3] Note-taking: Linear note-taking, available at << https://en.wikipedia.org/wiki/Note-taking#Linear_note-taking/>>, last accessed on 01-07-2017 at 10.00pm.
- [4] Note-taking: Non-linear note-taking, available at <<https://en.wikipedia.org/wiki/Note-taking#Non-linear_note-taking/>>, last accessed on 01-07-2017 at 10.30pm.
- [5] General Study Skills: Effective Study Skills, available at <<http://www.studyinglanguages.ac.uk/during_your_study/study_skills_general/>>, last accessed on 02-07-2017 at 6.00pm.
- [6] Cornell Notes, available at <<https://en.wikipedia.org/wiki/Cornell_Notes/>>, last accessed on 04-04-2018 at 12pm.
- [7] The Cornell Note-taking System, available at <<<http://lsc.cornell.edu/notes.html/>>>, last accessed on April 04-04-2018 at 12.15pm.
- [8] Comparison of notetaking software: General Information, Basic Features, Advanced formatting and content, available at <<https://en.wikipedia.org/wiki/Comparison_of_notetaking_software/>>, last accessed on 01-07-2017 at 10.30pm.
- [9] Business process management, available at <<https://en.wikipedia.org/wiki/Business_process_management/>>, last accessed on 20-03-2018 at 7.30pm.
- [10] Why & How: Business Process Management, available at <<<http://www.modernanalyst.com/Resources/Articles/tabid/115/ID/864/Why-How-Business-Process-Modelling.aspx/>>>, last accessed on 20-03-2018 at 8.10pm.

- [11] Ivan Marsic, “Software Engineering”, Rutgers University, New Brunswick, New Jersey, ch-2, section 2.2, page 68, Available: <<<http://www.ece.rutgers.edu/~marsic/books/SE/>>>, September 10, 2012.
- [12] Ivan Marsic, “Software Engineering”, Rutgers University, New Brunswick, New Jersey, ch-2, section 2.4, page 88, Available: <<<http://www.ece.rutgers.edu/~marsic/books/SE/>>>, September 10, 2012.
- [13] Data Flow Diagram, available at <<<https://www.smartdraw.com/data-flow-diagram/>>>, last accessed on 22-03-2018 at 8.00am.
- [14] Abraham Silberschatz, Henry F. Korth, S. Sudarshan, “Database System Concepts”, McGrawhill, 6th ed., ch-7, sec. 7.1, page 260, 2011.
- [15] Abraham Silberschatz, Henry F. Korth, S. Sudarshan, “Database System Concepts”, McGrawhill, 6th ed., ch-7, sec. 7.2, page 262, 2011.
- [16] Human Computer Interaction, available at <<https://en.wikipedia.org/wiki/Human%E2%80%93computer_interaction/>>, last accessed on 24-03-2018 at 9.05am.
- [17] Integration Testing, available at <<https://en.wikipedia.org/wiki/Integration_testing/>>, last accessed on March at 24, 2018 at 9.30am.
- [18] What is usability testing, available at <<<https://www.experienceux.co.uk/faqs/what-is-usability-testing/>>>, last accessed on 24-03-2018 at 10.00am.
- [19] Ray Smith, “Architecture and Data Structures”, available at << <https://github.com/tesseract-ocr/tesseract/wiki/Technical-Documentation/>>, last accessed on 10-08-2017 at 11.30pm.
- [20] Ray Smith, “Architecture and Data Structures”, available at << <https://github.com/tesseract-ocr/tesseract/wiki/Technical-Documentation/>>>, last accessed on 10-08-2017 at 11.30pm.

Appendix

Appendix A: Related Diagrams:

We have repeatedly said in our documentation that we have used Tesseract OCR engine. Now let's see the system architecture below, nominally a pipeline:

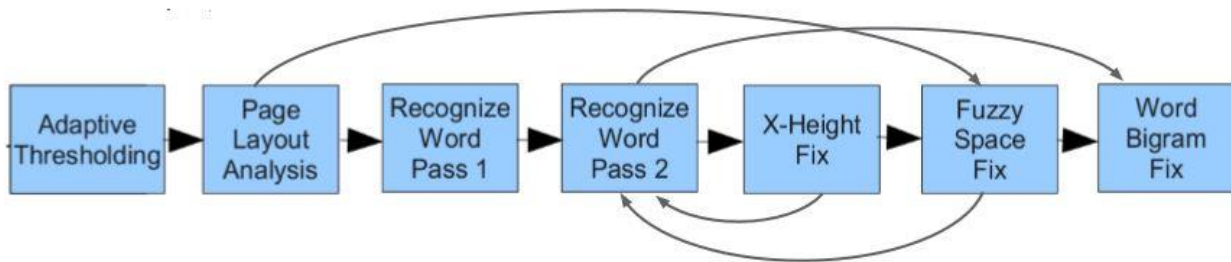


Figure A1: Tesseract System Architecture [19]

And here's a diagram of how Tesseract word recognition works:

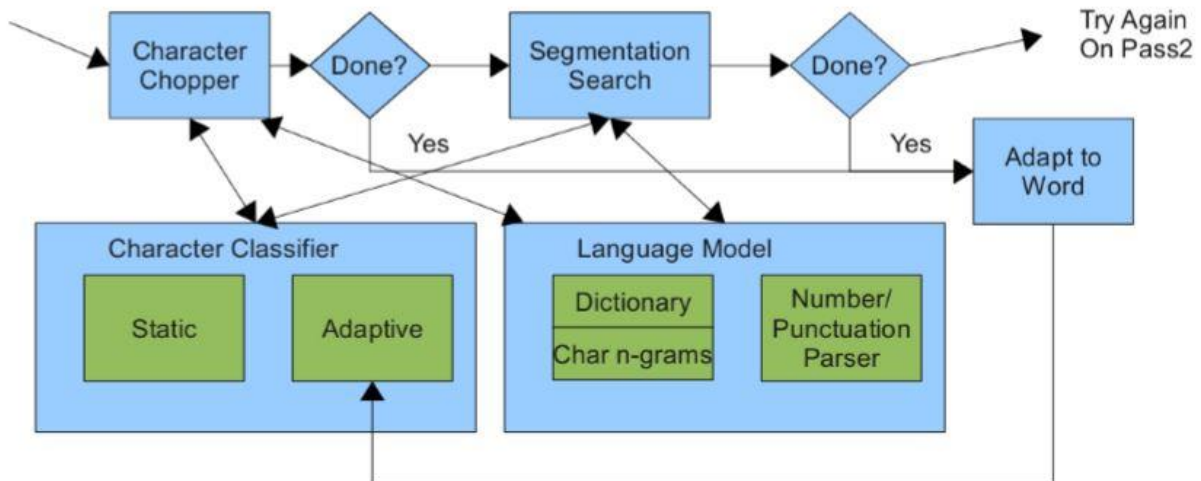


Figure A2: Tesseract Word Recognizer [20]

Appendix A: Related Diagrams:

Although Laravel is not technically an MVC framework, but it do have the M as Model and shares similarity with MVC design. Below is an image of MVC architecture.

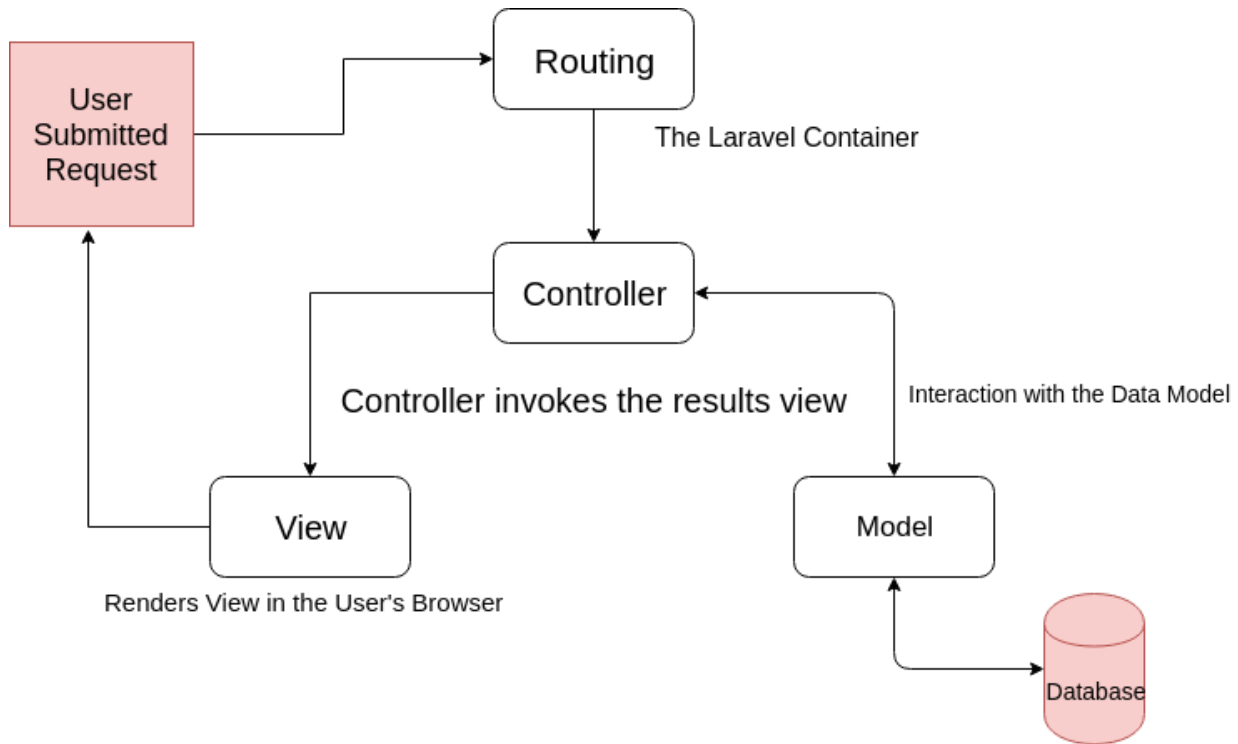


Figure A3: MVC Architecture

And apart from that we can view Laravel's cycle of request handling as follows:

Request > Services > Routing > Logic > Response.

Plagiarism Report

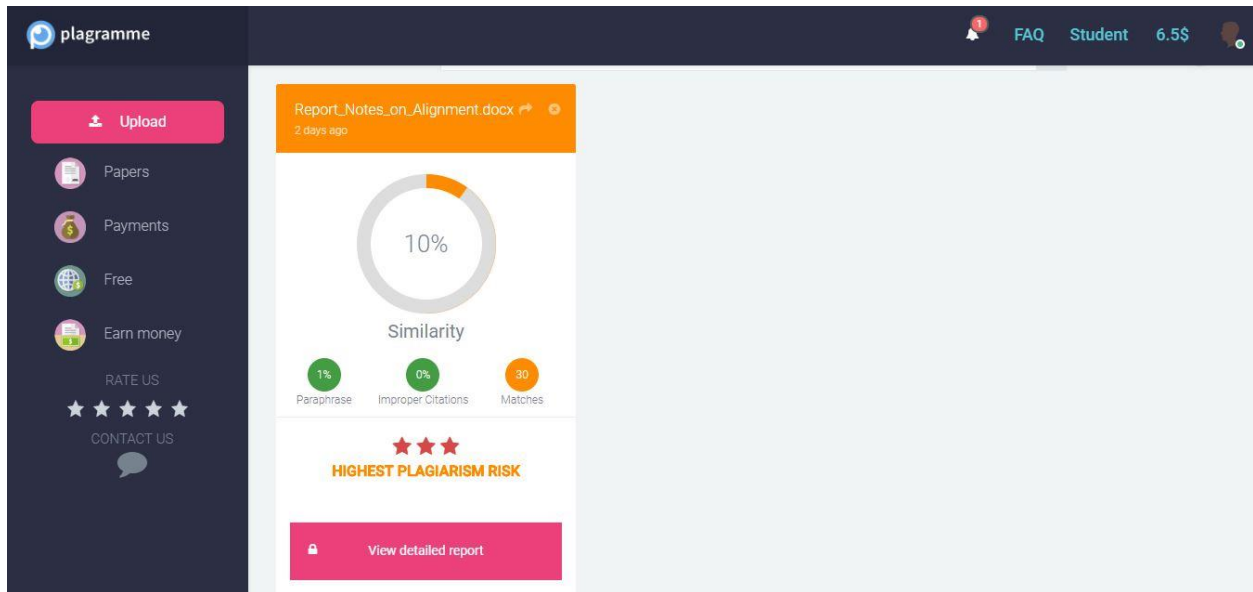


Figure P1: Plagiarism Report