

EasyNotes

A Minor Project Report
Submitted in Partial fulfillment for the award of
Bachelor of Engineering in Computer Science & Engineering

Submitted to
**RAJIV GANDHI PROUDYOGIKI VISHWA VIDYALAYA
BHOPAL (M.P)**



MINOR PROJECT REPORT

Submitted by
Kunal Jain [0103CS181076]

Under the supervision of
Lilesh Pathe



Department of Computer Science & Engineering
Lakshmi Narain College of Technology, Bhopal (M.P.)
Session 2019-2020



LAKSHMI NARAIN COLLEGE OF TECHNOLOGY, BHOPAL

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

This is to certify that the work embodied in this project work entitled “**EasyNotes**” has been satisfactorily completed by the **Kunal Jain** (0103CS181076). It is a bonafide piece of work, carried out under the guidance in **Department of Computer Science & Engineering, Lakshmi Narain College of Technology, Bhopal** for the partial fulfillment of the **Bachelor of Engineering** during the academic year 2020-2021.

Guided By

Lilesh Pathe

Approved By

Dr. Bhupesh Gour

**Prof. & Head of
Department of Computer Science & Engineering**



LAKSHMI NARAIN COLLEGE OF TECHNOLOGY, BHOPAL

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

ACKNOWLEDGEMENT

We express our deep sense of gratitude to Lilesh Pathe, Department of Computer Science & Engineering L.N.C.T., Bhopal. Whose kindness valuable guidance and timely help encouraged me to complete this project.

A special thank goes to Dr. Bhupesh Gour (HOD) who helped me in completing this project work. He exchanged her interesting ideas & thoughts which made this project work successful.

We would also thank our institution and all the faculty members without whom this project work would have been a distant reality.

Kunal Jain [0103CS181076]

| S.NO. | TOPICS | PAGE NO. |
|--------------|--|-----------------|
| 1. | Problem Domain | 5 |
| 2. | Literature Survey | 6-12 |
| 3. | Major objective & scope of project. | 13-15 |
| 4. | Problem Analysis and requirement specification | 16-18 |
| 5. | Detailed Design (Modeling and UCD/DFD) | 19-21 |
| 6. | Hardware/Software platform environment | 22-24 |
| 7. | Snapshots of Input & Output | 25-29 |
| 8. | Coding | 30-79 |
| 9. | References | 80 |

CHAPTER-1

PROBLEM DOMAIN

Note-taking has been an important part of human history and scientific development. The Ancient Greeks developed *hypomnema*, personal records on important subjects. In the Renaissance and early modern period, students learned to take notes in schools, academies and universities, often producing beautiful volumes that served as reference works after they finished their studies. In pre-digital times, people used many kinds of notebooks, including accounting waste books, marginalia, and commonplace books, as this world are walking on digital path and almost everything are available online, it is worth saying that: “A study without it’s notes is like a body without a soul. Note-taking is a central aspect of a complex human behavior related to information management involving a range of underlying mental processes and their interactions with other cognitive functions. The person taking notes must acquire and filter the incoming sources, organize and restructure existing knowledge structures, comprehend and write down their explanation of the information, and ultimately store and integrate the freshly processed material. The result is a knowledge representation, and a memory storage. Studies comparing the performance of students who took handwritten notes to students who typed their notes found that students who took handwritten notes performed better on examinations, hypothetically due to the deeper processing of learned material through selective rephrasing instead of word-for-word transcription which is common when typing notes, but it is not possible for a person either he is student or any employee to making notes of the material while learning it, either he can learn that material properly or end up with making perfect notes of it, he can’t do both work perfectly, Studies comparing the knowledge of topic learnt by students who only focus on the learning is found more than to students who do work of making notes of it side-by-side, because they can’t able to focus on that topic properly. The study is started from the review of the problem faced by students that they can’t make notes of topic they learnt just because they can’t focus to done with both tasks perfectly at same time.

To solve this problem (or at least to make it easier to cope with) EasyNotes provide a solution which will be further explained.

2.1 Survey

2.1.1 Why is note-making important?

Note taking is an underrated skill. It is not just important for college students, it's a valuable life skill, if you are able to make good notes, you will be more effective in pursuing your goals whether you are a student, writer, or entrepreneur. There's a correlation between taking notes and creative thinking. Creativity isn't a mythical power; it's about having lots of ideas and melding the best ones together. You need notes to record and flesh-out these ideas. Both Richard Branson and Bill Gates, notable creative thinkers, are avid note takers. Branson offers a clear instruction, *"It doesn't matter how you record your notes — as long as you do."* Effective note taking helps you to remember information and aids your understanding of that information. Once created, your notes then act as a record of your thinking and they also provide the source material for your next creative or business project. It is a specialist form of writing.

2.1.2 How notes boost your understanding

You will only truly understand concepts when you have taken the time to process them, when you take notes you are turning a passive activity – reading or listening – into an active process. Scanning your eyes across a page or listening to someone speak is too easy. You need to do something with the information you receive. That's why note making is best process to utilise that information you got, until you don't make notes of that you temporarily remember that thing after some span of time you will forget what you learnt. Studies shows students who making notes and revise them time-to-time is do more smart work than those who not make notes and do hard work to pass their examinations. According to the Collins English dictionary revision means

“to read things again”. This implies that you have read and understood your study material at least once prior to starting revision. The importance of revision is twofold. Firstly, it helps you to remember facts, figures, topics and methodologies that you have covered some time ago. Secondly, if done correctly it will help increase your confidence and reduce anxiety – you will be well prepared for your examination. When you make your own notes, you are sorting, selecting, and combining new information with your existing knowledge. Finding where new ideas fit in with what you already know is how to synthesize unfamiliar concepts into your working knowledge. Studies have shown that comprehension and recall are improved when people make their own notes rather than rely on materials provided by others. This is called *“The generation effect”*, by making notes, you force yourself to construct a conceptual mental representation of the ideas you are grappling with. It is this process that helps you comprehend the subject matter you are working with. It’s the process of writing notes, and not the notes themselves, that helps you gain command of new concepts.

2.1.3 Notes aid your memory

The *“spacing effect”* is the idea that you need to revisit ideas in order to make them stick in your mind, you don’t necessarily need to re-read an entire book or re-watch a whole lecture to remember ideas over the long-term. Instead, you can revisit your notes to refresh the ideas that you’ve encountered, when you read over notes a day after making them, then a week later, and then a month later, you are far more likely to remember the concepts that you initially encountered. You can’t note each and every thing or just as the same you learn from material, filter out the fluff and only make the efforts to remember things that matter to you. Studies shown that by experiment of taking three boxes: The first box contains note cards that you consult once a day; you consult the second box once a week; and you consult the third box once a month. When you are familiar with a card, you move it to the next box. When unfamiliar, you bring the flashcard to a more frequently consulted box, why note-making and time-to-time revision of it is important.

2.1.4 Notes are repository of your thinking

Even if you don't make the effort to commit everything to your long-term memory, your notes are still useful as your personal reference library, an external memory aid, when you need to recall something that you came across a while ago, you can go over the main ideas by looking over your notes. This is why good, comprehensive notes are so useful; they are a catalogue of the best ideas you've encountered, and what you thought about them. Your repository of ideas is especially useful when you keep digital notes. Digitised notes can be grouped and tagged in a variety of ways, as well as searched via text queries.

2.1.4 Notes are resource of your writing

When you have taken good, comprehensive notes and included your own reaction to ideas as well as succinct summaries of key ideas, these notes are invaluable to your future self. Your notes become stand-alone artifacts that can prompt ideas and solutions to your current projects and tasks, by reading over a series of notes grouped by a particular topic or theme, you can start to see how a creative project might take shape. In effect, you can use your notes to visualise your thinking on a topic. How has your prior reading (or watching, or listening) addressed this topic.

2.2 Different ways for note making

2.2.1 Outline method

This method is used for simplicity and is one of the easiest methods of taking notes. Anyone can pick up this method and use it with no issues. When using this method, the idea is to select four or five key points that are going to be covered in a specific lesson. Under those key points, you write more in-depth sub-points based on what is being discussed on those topics. The idea with this form of note taking is so it doesn't overwhelm you. But you'll pay attention in a different manner. In the case of this approach, if you know what's being discussed, you'll focus on the important aspects of that topic rather than wonder what's coming up next.

Use this method in cases where:

- You want your notes to be organized from the start.
- To see the relationships between both topics and subtopics.
- You want to convert the points into questions to quiz yourself on later.

2.2.2 Cornell method

As by name it is developed in the 1950s by Cornell University, this is the most common note taking method around. In fact, the outline method is likely inspired by this method as there are similarities to it. In this method, you are still using key points, but this method goes deeper into the organizing method. For one, the page is broken into three sections:

- a narrow column called the “cue”
- a wider column for your actual notes
- a summary at the bottom

The cue section is the section where you fill out main points, people, potential test questions and more. This section is devoted to helping you recall larger topics and ideas. The note section is devoted to expanding and explaining those cue points. You still want to summarize them to an extent using headings. When getting into specifics, you want to indent them and use a numbering system, either roman numerals, numbers, or letters. The summary section is the section you write up at the end summarizing all of the information in a clear sentence or two. You want both the summary and the cue to be simple seeing as your notes are where you want all of the details.

This method is great if you:

- Want notes to be organized even further and easier to review.
- Want to pull out major ideas and concepts quickly.

2.2.3 Mind mapping method

Mind mapping is a method that works for subjects that have interlocking topics or complex and abstract ideas. Chemistry, history, and philosophy are examples where this method shines. The use of the map is to serve as a visual aid for how every topic is related to one

another. It also allows you to go into detail on particular ideas or topics. An example of this at work is looking at the French Revolution. First, you'd start with that concept at the center and then begin branching off that led to events, and people that sparked the French Revolution. You can start off with broad general ideas and during the course or when you are reviewing, you can add in sub-concepts to those branches. Things like dates, support facts, concepts that you see between people and events. That being said, this method doesn't apply to only those kinds of topics. Any kind of topic that you can break into various points can also help as well. Another example can be talking about different forms of learning and using the nodes to discuss each method and what each one is like.

This type of method for note taking is great for:

- Visual learners who struggle with studying via notes.
- For people who need to remember and connect relationships, and events with topics.

2.2.4 Flow notes method

Discussed in a post in College Info Geek, this method is for those who want to maximize active learning in the classroom and save time in reviewing. The idea of flow notes is to treat yourself as a student rather than transcribing word for word. In this method, you'll jot down topics, then start drawing arrows, make doodles, diagrams and graphs to get a general idea out there. This method also helps in drawing other bridges and form connections in various fields or within the subject. If some information reminds you of another piece of information or technique, make a note and jot it down.

2.2.5 Sentence method

Another simple method and is a lesser version of flow notes. The idea with this is a simple note-taking. You're jotting down everything that's being said to the best of your ability. It's genuine transcription at its finest. The problem with this method is that it can be tough to keep up with everything else that's happening. If you're writing notes by hand, you will

definitely be missing key points and ideas. On a computer, you may be able to keep up, however, you may face challenges still. Despite those problems, there are still advantages to this method. Compared to every other method, this provides the most details and information for review:

- You can still be brief by covering the main points.
- Your notes are already simplified for you to study and review them immediately.

2.2.6 Charting method

Charting notes take the Cornell method and divide a sheet into three columns. Similar to the mind mapping method, this helps you in connecting relationships and facts together between topics. This method is a lazier method than the other ones mentioned above but works for the people who want to highlight key pieces of information on various topics and want to organize facts for easy review.

2.2.7 Digital method

By using that method we can use out all created notes and manage them online, without using pen and paper, it lead to reduce the use of paper so effectively less number of trees are cut off, indirectly by using this we are saving environment, it happened many times that notes made using papers are lost or misplaced so we have to start everything written on it from scratch, as now every thing is relying on and transferred to internet, so risk of misplacing will be reduced and you can access and manage your notes anytime anywhere.

And in terms to making perfect understandable notes you can use any of above-mentioned methods from 2.2.1 to 2.2.6 depending on method you want to follow.

2.3 Solution

A web-based database system resides on an Internet server. The database can be accessed through a web browser. A distributed system is a system consisting of a collection of autonomous machines connected by communication networks and equipped with software systems designed to produce an integrated and consistent computing environment. Distributed systems are helpful in letting the users to co-operate all the activities in a more effective and efficient manner. The key purpose of the distributed systems is represented by resource sharing, openness, concurrency, scalability, fault tolerance and transparency. Web-services provide a standard means of interoperating between different software applications running on a variety of platforms and/or frameworks. Web applications use web documents written in a standard format such as HTML and JavaScript, which are supported by a variety of web browsers. Advantages of web-based distributed databases are easy maintenance and updating, reusability and modularity, distribution of data update and security. The architecture used for the web-based distributed database is the Client/Server model. EasyNotes is a web application where user can create and manage their all notes in one place, the main point of solution about all problem mentioned in above sections that by using that web application one can fully concentrate on learning and the work of making notes leave it to that application because you don't have to put your mind to make notes you learning by using smart technology of speech-to-text it can make notes of all stuff it hear. You can manage and correct it by clearing mistakes done by it and by providing idea about what you want to say in place of word it heard mistakenly. As in progress with smart world it's become necessary to run along with time, it can be possible when you can put your mind on learning with 100% efficiency.

CHAPTER-3

MAJOR OBJECTIVE AND SCOPE OF THE PROJECT

The main objective of *EasyNotes* is to reduce burden on students of doing multiple tasks at one time especially while studying that's why they can't concentrate on topics they are learning with full of their mind, somewhere being a multitasker is very good skill but in field of study it is just opposite you have to try at fullest to concentrate on studying and only studying while you study. But on other side making notes of it is keeping same importance along with learning for students, because without notes study is like empty vessel which is of no use. It's about students but not only students need to make notes everyone of any field have to make notes, beside of its other objective is to save environment by running along with digital world.

This application provides user friendly interface to manage and create your notes on internet, also it has many features which makes it different like you can color-code your notes, Study shown that the use of bright colors captures your attention and is especially stimulating for visual learners. However, the colors are more than just eye-catching: The significance you give to each hue serves as a form of mental shorthand, giving context to the material you mark and helping you to process it more meaningfully and efficiently. All features are mentioned in ahead sections.

3.2 Scope

3.2.1 *Managing notes*

In this subsection, we describe the functional features offered by our proposed system in detail. They are as follows:

1. *Registration and a successful login of the Customer*

This module allows one to register themselves to the site in order to establish a username and password for future logins. Once a customer logs in he/she can then continue on to the other features offered by the system. The registration page includes a name field, a username to be

specified, to confirm and reconfirm the customer's chosen password, the customer's phone number. Once the customer registers, he/she is now allowed to log in.

2. *Updating Profile*

The registered customer can view and update his/her profile here. The profile includes the information that the customer provided during registration such as name, email, phone number, username, profession and password. Once the customer views the profile, he/she can click on the back button to view the home page.

3. *Adding a new note*

The registered customer can add a new note, in term to create a one he/she has to click on create button on dashboard they redirected after successful login, in this they can choose the color code of note, it's title and content of it, both field are can't left empty you have to write on either field, if you only write it's content automatically it's title is updated with same content, as title can't be left empty, in term to create a new note you can use voice typing feature to type the content just by dictating the content of note in your voice or you can also use speakers as it can detect the foreign voices. In future I'll add feature in this so it can type your notes by using internal audio of system.

4. *Updating a note*

You can update your created notes just as you create a note, you can update it's color-code, it's title, content by either typing or by dictating its content.

5. *Sorted notes*

All notes are sorted in order to latest to oldest, currently it shows all notes from current time to time when user registered on application, In future updates I'll add another feature of search so user can search between specified dates.

6. *Pinning a note*

Some of our notes are important and keeping value prior from other notes so to keep our eye on it to remember this feature can become handy, by using this user can pin our notes to top of the page.

7. *Deleting a note*

After work specified in notes or they are intended for is over they become useless and we don't need them anymore, so user can delete note from their account easily.

8. *Logout*

After user done with application, he/she can end their login session by logging out him/her out.

CHAPTER-4

REQUIREMENT SPECIFICATION

From conclusion of all surveys and discussion with students' various requirements have arrived at EasyNotes:

4.2.1 Interface Requirements

The system users are –

1. EasyNotes admin i.e. me as the system administrators.
2. Registered persons as the normal users.

4.2.2.1 System Administrators

The administrator logs on to the system by inserting administrator user name and password. Administrator can do any transaction as well as editing all details inside the database such as adding, editing and deleting a new user or adding, editing and deleting items.

4.2.2.2 System Users

The users have to enter the user name and password and click on 'Login' button. If user makes any mistake the system will ask for the correct username and password until he enters the correct one. When user wants to make any changes about creating/updating the notes, all features are available at user dashboard.

4.2.2.3 Hardware Interfaces

1. At client-side user's PCs/Smartphone with a JavaScript enabled web browser.
2. At server-side a system with Python, Django installed.

4.2.3 Performance Requirements

1. The response time for menu changes will be not be more than 3 seconds.
2. The time for create a new note will not be more than 3 seconds.
3. The time taken to update the note or get information of the note will not be more than 2 seconds.
4. The time taken to prompt message boxes will not more than 2 seconds.

4.2.3 Design Constraints

1. The system is based on menu driven interfaces.
2. Menu selection will be done by using the mouse and the key board keys.
3. Confirmation messages on taken actions, input acceptance and error conditions will be displayed after each input.
4. Error messages will be displayed at the time of detection of input errors and the system errors.

4.2.4 Attributes

1. Qualities of the product Reliability –

The system is thoroughly tested at the time of delivery so that computational errors are minimized.

2. Maintainability –

The website is maintained by me.

3. Security –

Only the User will have the authority to edit details in Users and Notes. No one can enter the system without a username and a password. Normal system users cannot access the User login. All deleting actions are notified by a message box asking to confirm deletion.

5.1 The Workflow of proposed solution

As the result of the survey, the best solution that deals with the problem domain is to create a digital environment for users where they can create and manage all notes they created and using speech-to-text mechanism to reduce work of user to making notes by themselves. there are two layers of the model of the project:

1. Browser with JavaScript enabled
2. Server with installed tools (Python, Django) and other software.

5.1.1 Layer 1 (Browser Layer)

Various devices on which JavaScript enabled browser can be run act as the main device which will interact with the user directly. Through this layer after successful registration and login of the user. User can browse and create/update any new note or existing notes in his/her account, as this layer is responsible of interacting the user to the Application that layer play a vital role to build a model of the project.

5.1.2 Layer 2 (Server)

All the data user entered at client side in Layer 1 will be stored in database through Django and Python which is must be installed at server-system with their supportive and essential software tools. The scope of Layer 1 is limited to one session so data entered there is erased from that after one session so with purpose of saving all data entered by user in particular session in database on server side.

5.2 Use Case Diagram

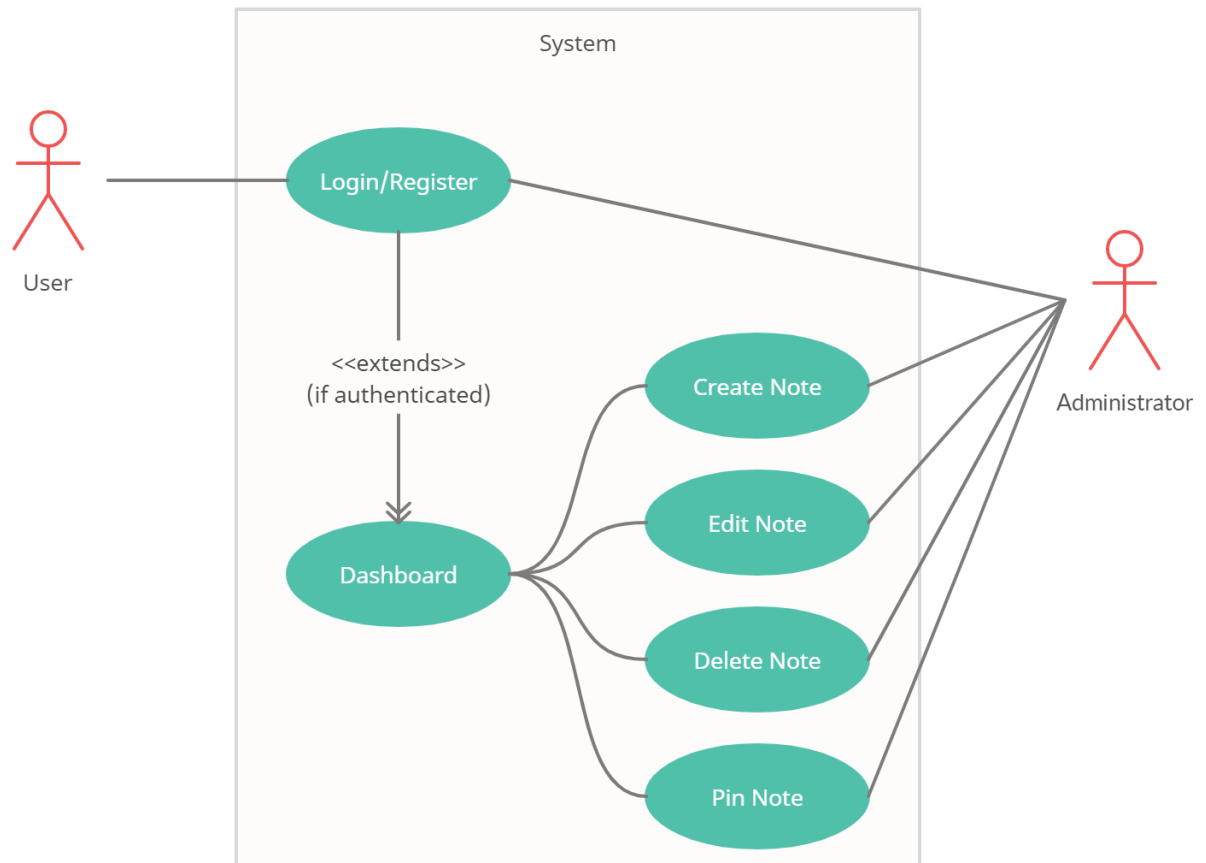
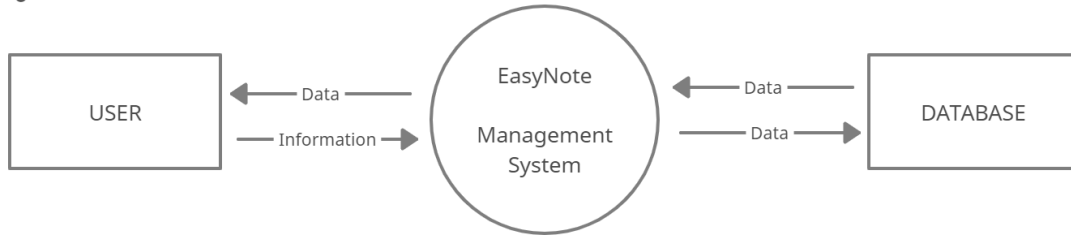


Fig 5.1 – UCD of project

5.3 Data Flow Diagrams (DFDs)

Level - 0



Level - 1

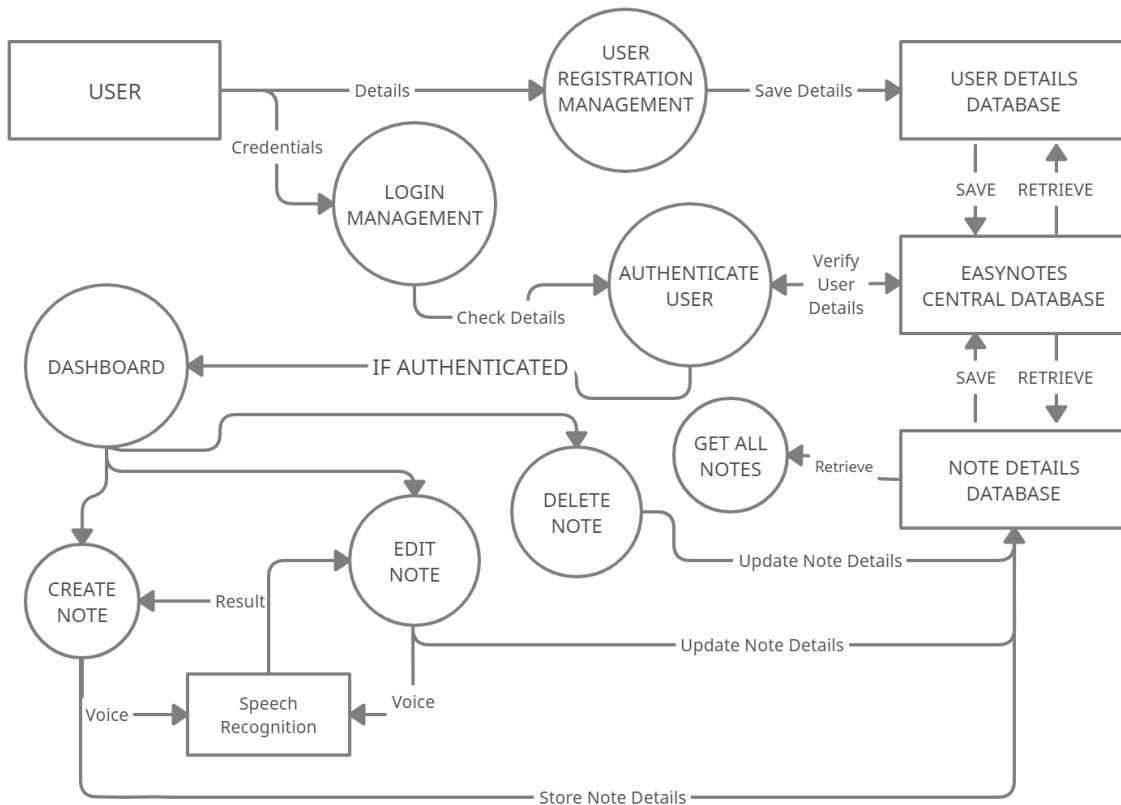


Fig 5.2 – Data Flow Level 0 and Level 1 Diagram

CHAPTER-6

HARDWARE AND SOFTWARE PLATFORM ENVIRONMENT

6.1 Hardware Specifications

The hardware used in our final proposed model of project for a web-based application are mentioned below:

1. Personal Computer (PC) with installed any of operating systems that support browser with JavaScript enabled and a working internet connection.
2. Any Smartphone with installed any of operating systems that support browser with JavaScript enabled and a working internet connection.
3. A server machine which provides support to Python, Django and Apache Virtual host using WSGI to host that web application.

6.2 Software Specification

The software specification that should be present on the devices are discussed below, along with the software(s) used by different servers.

6.2.1 Operating System of devices

- a. In smartphones, PCs including desktop, laptop, all operating systems of all versions are supported, if JavaScript enabled browser is installed in it.

6.2.2 Software used as tools for development

a. Visual Studio Code (v1.42.1)

It is a streamlined source-code editor, which provides a friendly interface to code, we used that software with its Live Server Plugin to get live-time updates in our pages as soon we edit its source code.

And it's used to write a source code of all Hyper Text Markup Language (HTML) [for provide structure to front-end], Cascading Style Sheets (CSS) [for designing of a front-end], JavaScript pages, jQuery codes. It is mainly used to build a front-end of our project.

b. PyCharm (v2021.1.1)

PyCharm is an integrated development environment (IDE) used in computer programming, specifically for the Python language. It is developed by the Czech company JetBrains. It provides code analysis, a graphical debugger, an integrated unit tester, integration with version control systems (VCSes), and supports web development with Django.

c. argoUML

This software is used to design Entity Relationship Diagrams (ERDs) and Data Flow Diagrams (DFDs) in a convenient way by using its utilities, it's used to form all UML diagrams related to our project.

6.2.3 Server-side tools and software

The server is responsible for receiving data entered from devices through different form created related to different purposes in different webpages, and then process that received data and store at their respective places of models created in database. It's responsible for receiving and returning of data. The server uses following software installations to function properly.

a. Python 3

It is an interpreted high-level general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

b. Django

One of the famous libraries of Python, used for back-end programming, Django's primary goal is to ease the creation of complex, database-driven websites. The framework emphasizes reusability and "pluggability" of components, less code, low coupling, rapid development, and the principle of don't repeat yourself. Python is used throughout, even for settings, files, and data models. Django also provides an optional administrative create, read, update and delete interface that is generated dynamically through introspection and configured via admin models.

6.2.4 Client-side tools and Software

This is a web-based application so on client side, only a web-browser needed with JavaScript enabled in it.

CHAPTER-7

SNAPSHOTS OF INPUT & OUTPUT

Following are snapshots of data displayed on JavaScript enabled browser at client-side –

1. Login / Register page

The figure displays two screenshots of the EasyNotes application interface, both featuring a dark theme and a central form area.

Top Screenshot (Login Page):

- Header:** The EasyNotes logo is located in the top-left corner.
- Title:** The word "LOGIN" is centered at the top of the form area.
- Inputs:** There are two input fields: "Username" and "Password", both with orange borders.
- Link:** A "Forgot password" link is positioned below the password field.
- Button:** A "LOGIN" button with an orange border is centered below the inputs.
- Footer:** At the bottom, it says "New here?" followed by a "Sign up" button with a green border.

Bottom Screenshot (Register Page):

- Header:** The EasyNotes logo is located in the top-left corner.
- Title:** The word "REGISTER" is centered at the top of the form area.
- Inputs:** There are four input fields: "Name", "Email id", "Username", and "Password", all with orange borders.
- Button:** A "REGISTER" button with an orange border is centered below the inputs.
- Footer:** At the bottom, it says "Have an account?" followed by a "Sign in" button with a green border.

Fig 7.1 – Login and Register pages

2. Dashboard page

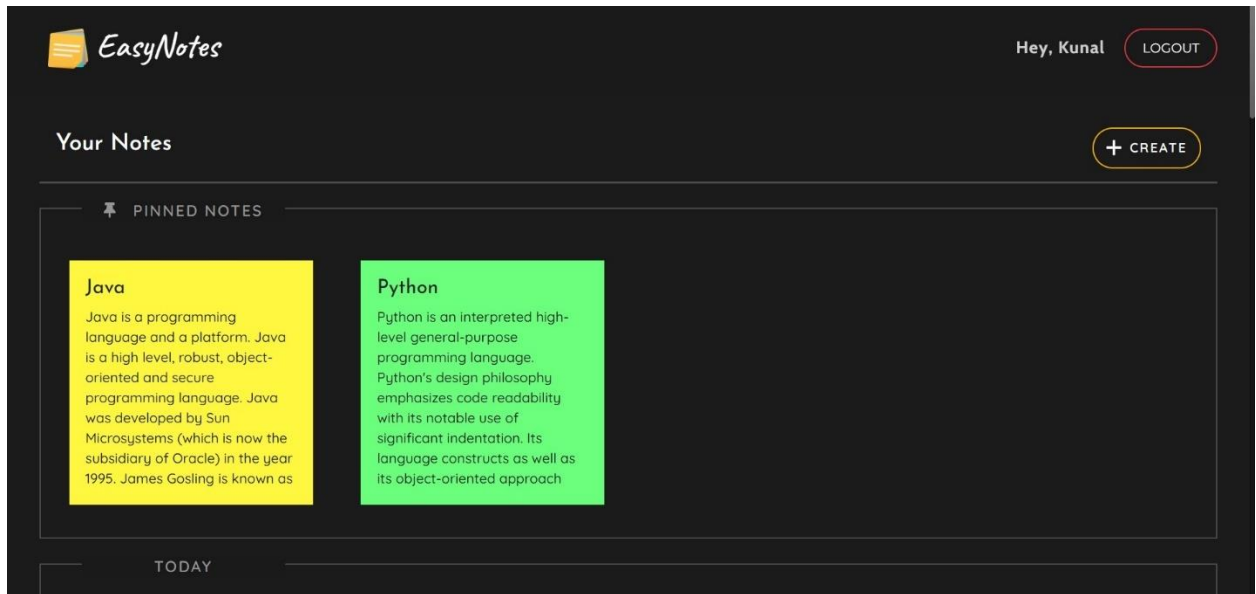


Fig 7.3 – Main Dashboard

a. Menu options

i. Create a new note –

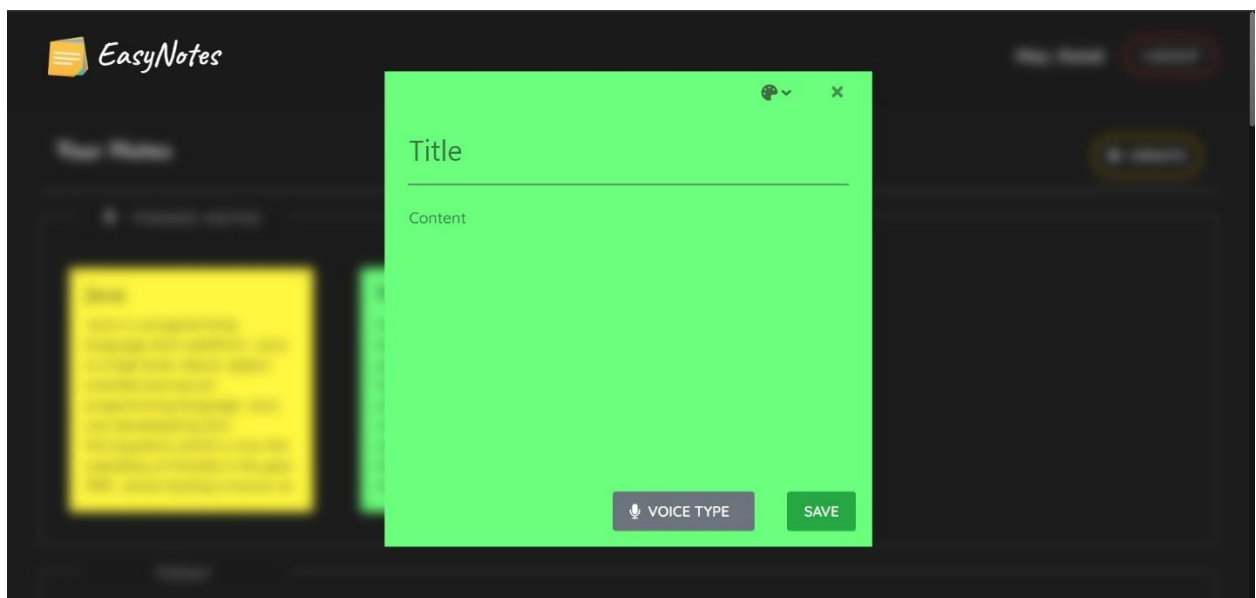


Fig 7.4 – Create Note box

- ii. Edit an existing note –

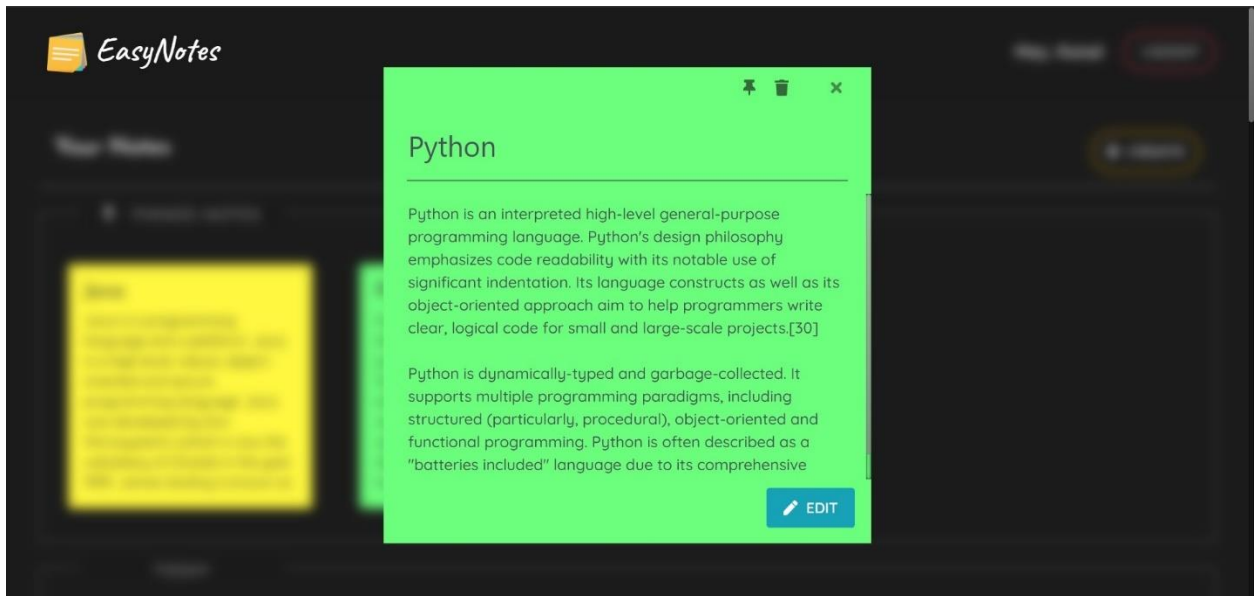


Fig 7.5 – Edit Note box

- iii. Logout – on proceed for logout page confirms once by popping alert with message “Are you sure for Logout?”, on confirming user’s session will end and he/she redirect to login page.

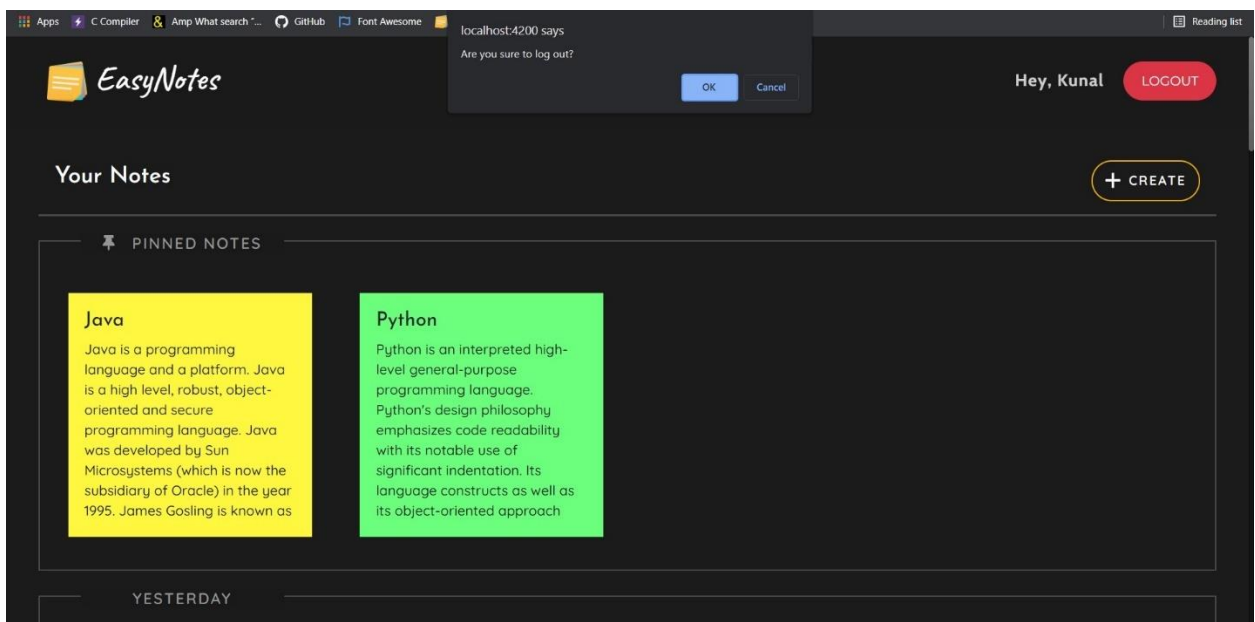


Fig 7.11 – Logout confirmation alert


Following are snapshots of description of data present in django at server side –

```
api > accounts > serializers.py > ...
1  from rest_framework import serializers
2  from django.contrib.auth.models import User
3
4  # User Serializer
5  class UserSerializer(serializers.ModelSerializer):
6      class Meta:
7          model = User
8          fields = ('id','first_name','last_name','username', 'email')
9
10 # Register Serializer
11 class RegisterSerializer(serializers.ModelSerializer):
12     class Meta:
13         model = User
14         fields = ('id','first_name','last_name','email','username','password')
15         extra_kwargs = {'password': {'write_only': True}}
16
17     def create(self, validated_data):
18         user = User.objects.create_user(validated_data['username'],email = validated_data['email'],password = validated_data['password'],fi
19         return user
```

Fig 7.12 – Description of Authentication model


```
api > dashboard > models.py > ...
1  from django.db import models
2  from django.contrib.auth.models import User
3  from django.db.models.deletion import CASCADE
4
5
6  class Note(models.Model):
7      title = models.CharField(max_length=255)
8      uname = models.ForeignKey(User,to_field="username",on_delete=models.CASCADE)
9      color = models.CharField(max_length=20,default='#6cfff7d')
10     content = models.TextField(blank=True)
11     created_on = models.DateTimeField(auto_now_add=True)
12     pinned = models.BooleanField(default=False)
13
```

Fig 7.13 – Description of Note model

api > accounts >  urls.py > ...

```
1 from django.urls import path,include
2 from knox import views as knox_views
3 from .views import RegisterAPI,LoginAPI,getUserByUsername
4
5 urlpatterns = [
6     path('add',RegisterAPI.as_view(), name='register'),
7     path('login', LoginAPI.as_view(), name='login'),
8     path('uname/<str:username>', getUserByUsername.as_view(), name = 'uname'),
9     path('logout', knox_views.LogoutView.as_view(), name='logout'),
10    path('logoutall', knox_views.LogoutAllView.as_view(), name='logoutall'),
11 ]
```

Fig 7.14 – Account URLs

api > dashboard >  urls.py > ...

```
1 from .views import createNote, getNote, getNotesBetween, removeNote, setPin, updateNote
2 from django.urls import path,include
3
4 urlpatterns = [
5     path('<str:uname>/add/<str:color>',createNote.as_view(),name="addnote"),
6     path('get/<int:nid>',getNote.as_view(),name="getnote"),
7     path('<str:uname>/getall/<str:from_>/<str:to_>',getNotesBetween.as_view(),name="getnotes"),
8     path('update/<str:uname>/<int:nid>',updateNote.as_view(),name="updatenote"),
9     path('remove/<int:nid>',removeNote.as_view(),name="removeNote"),
10    path('setpin/<str:uname>/<int:nid>',setPin.as_view(),name="setPin")
11 ]
```

Fig 7.14 – Dashboard URLs

1. Front-end (HTML, CSS, JSP, JavaScript Pages)

- index.html

```
<!doctype html>
<html lang="en">

<head>
  <meta charset="utf-8">
  <title>EasyNotes</title>
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="assets/logo.png">
  <link rel="stylesheet" href="assets/bootstrap/bootstrap.min.css">
  <script src="assets/bootstrap/bootstrap.min.js"></script>
  <link rel="stylesheet" href="assets/fontawesome-pro-
master/web/css/all.css">
  <link rel="preconnect" href="https://fonts.gstatic.com">
  <link
    href="https://fonts.googleapis.com/css2?family=Josefin+Sans:ital,wght@0,
100;0,200;0,300;0,400;0,500;0,600;0,700;1,100;1,200;1,300;1,400;1,500;1,600;
1,700&display=swap"
    rel="stylesheet">
  <link rel="preconnect" href="https://fonts.gstatic.com">
  <link
    href="https://fonts.googleapis.com/css2?family=Montserrat:ital,wght@0,10
0;0,200;0,300;0,400;0,500;0,600;0,700;0,800;0,900;1,100;1,200;1,300;1,400;1,
500;1,600;1,700;1,800;1,900&display=swap"
    rel="stylesheet">
```

```

<link rel="preconnect" href="https://fonts.gstatic.com">
<link
  href="https://fonts.googleapis.com/css2?family=Titillium+Web:ital,wght@0
,200;0,300;0,400;0,600;0,700;0,900;1,200;1,300;1,400;1,600;1,700&display=swa
p"
  rel="stylesheet">
<link rel="preconnect" href="https://fonts.gstatic.com">
<link href="https://fonts.googleapis.com/css2?family=Caveat:wght@400;500;6
00;700&display=swap" rel="stylesheet">
<link rel="preconnect" href="https://fonts.gstatic.com">
<link href="https://fonts.googleapis.com/css2?family=Quicksand:wght@300;40
0;500;600;700&display=swap"
  rel="stylesheet">
<link rel="preconnect" href="https://fonts.gstatic.com">
<link href="https://fonts.googleapis.com/css2?family=Cabin:wght@700&display=
swap" rel="stylesheet">
<script src="assets/fontawesome-pro-master/web/js/all.min.js"></script>
</head>

<body>
  <app-root></app-root>
</body>

</html>

```

- authentication.component.html

```

<div class="box container-fluid">
  <div class="container">
    <div class="login">
      <span class="lmsg" *ngIf="lmsg">{{lmsg}}</span>
      <form #userLog="ngForm" (ngSubmit)="signIn(userLog.form)">
        <h2>Login</h2>

```

```

        <div class="form-group">
            <input type="text" name="username" id="uname" #luname="ngModel"
class="form-control"
                [(ngModel)]="username" [class.notvalid]="luname.invalid &&
luname.touched"
                placeholder="Username" required>
            <small class="text-danger" [class.d-
none]="luname.valid || luname.untouched">Username is
                required</small>
        </div>
        <div class="form-group">
            <input type="password" name="password" id="pwd" #lpwd="ngModel"
class="form-control mt-20"
                [(ngModel)]="password" [class.notvalid]="lpwd.invalid && lp
wd.touched" placeholder="Password"
                required>
            <small class="text-danger" [class.d-
none]="lpwd.valid || lpwd.untouched">Password is
                required</small>
        </div>
        <div class="form-group">
            <a href="#">Forgot password</a>
        </div>
        <span class="text-danger" style="font-family: 'Montserrat'; font-
size: 14px;"
            *ngIf="invalidCredentials"><i style="font-size: 16px;"
                class="mr-1 far fa-exclamation-
circle"></i>Invalid Credentials</span>
        <div class="form-group mt-3">
            <input type="submit" value="Login">
        </div>
    </form>
    <div class="panel">
        <h5>New here?</h5>
        <button (click)="userLog.reset()">Sign up</button>
    </div>
</div>
<div class="register">
    <form #userReg="ngForm" (ngSubmit)="signUp(userReg.form)">
        <h2>Register</h2>
        <div class="form-group">
            <input type="text" name="name" id="name" class="form-
control" pattern="[A-z' ']{1,20}"

```



```

        #rname="ngModel" [(ngModel)]="u.name" [class.notvalid]="rname.invalid && rname.touched"
        placeholder="Name" required>
        <div class="errbox" *ngIf="rname.errors && (rname.valid || rname.touched)">
            <small class="text-danger" *ngIf="rname.errors.required">Name is required</small>
            <small class="text-danger" *ngIf="rname.errors.pattern">Invalid name</small>
        </div>
    </div>
    <div class="form-group">
        <input type="email" name="email" id="email" #mail="ngModel" class="form-control"
            [(ngModel)]="u.email" [class.notvalid]="mail.invalid && mail.touched" placeholder="Email Id"
            pattern="[A-z\d\._-]+@[A-z0-9-]+\.[A-z]{2,5}(\.[A-z]{2,5})?" required>
        <div class="errbox" *ngIf="mail.errors && (mail.valid || mail.touched)">
            <small class="text-danger" *ngIf="mail.errors.required">Email ID is required</small>
            <small class="text-danger" *ngIf="mail.errors.pattern">Invalid email address</small>
        </div>
    </div>
    <div class="form-group">
        <input type="text" name="username" id="runame" #runame="ngModel" class="form-control"
            [(ngModel)]="u.uname" [class.notvalid]="runame.invalid && runame.touched" placeholder="Username"
            pattern="[a-z0-9\._-]{1,20}" required>
        <div class="errbox" *ngIf="runame.errors && (runame.valid || runame.touched)">
            <small class="text-danger" *ngIf="runame.errors.required">Username is required</small>
            <small class="text-danger" *ngIf="runame.errors.pattern">Invalid username : It should only
                contain alphabetic and numeric characters and not any special characters except '.' and
                '_'</small>
        </div>
    </div>
    <div class="form-group">

```

```

        <input type="password" name="password" id="rpwd" pattern="(?=.*
[%$+@=#!?_*-])(?=.*[A-Z])(?=.*[a-z])(?=.*[0-9])[a-zA-Z0-9?*@!_\.#%&-
=+]{8,}" #rpwd="ngModel" class="form-control"
        [(ngModel)]="u.pass" [class.notvalid]="rpwd.invalid && rpwd
.touched" placeholder="Password"
        required>
        <div class="errbox" *ngIf="rpwd.errors && (rpwd.valid || rpwd.t
ouched)">
            <small class="text-
danger" *ngIf="rpwd.errors.required">Password is required</small>
            <small class="text-
danger" *ngIf="rpwd.errors.pattern">Invalid password - It should of minimum 8
            characters and a combination of uppercase letter(A-
Z), lowercase letter(a-z), digits(0-9) &
            special characters(!@#$%...)</small>
        </div>
    </div>
    <span class="text-danger" style="font-family: 'Montserrat'; font-
size: 14px;" *ngIf="msg && error"><i
        style="font-size: 16px;" class="mr-1 far fa-exclamation-
circle"></i>{{msg}}</span>
    <div class="form-group">
        <input type="submit" [class.disabled]="userReg.invalid" value="
Register">
    </div>
</form>
<div class="panel">
    <h5>Have an account?</h5>
    <button (click)="userReg.reset()">Sign in</button>
</div>
</div>
</div>
</div>

```

- dashboard.component.html

```

<div class="nav-buttons" *ngIf="us.isUserLoggedIn()">
    <span class="mr-4" *ngIf="fname">Hey, {{fname}}</span>
    <button (click)="logout()" class="logout">
        Logout
    </button>
</div>
<div class="container add">

```

```

<div class="controls">
  <div class="cc"><span><i class="fas fa-
palette"></i></span><span class="choose"><i class="fas fa-angle-
down"></i></span></div>
  <div class="colors">
    <span><i class="fas fa-check green"></i></span>
    <span><i class="fas fa-check yellow d-none"></i></span>
    <span><i class="fas fa-check blue d-none"></i></span>
    <span><i class="fas fa-check skin d-none"></i></span>
    <span><i class="fas fa-check red d-none"></i></span></div>
    <button (click)="create.reset()" class="close"><span><i class="fas fa-
times"></i></span></button>
  </div>
</div>
<form #create="ngForm" (ngSubmit)="save()">
  <div class="form-group">
    <input type="text" placeholder="Title" #title="ngModel" [(ngModel)]="cn.tit
le" name="aheading" id="heading" autocomplete="off" class="form-control">
  </div><hr>
  <div class="form-group">
    <textarea name="content" id="acontent" placeholder="Content" [(ngModel)]="c
n.content" autocomplete="off" class="form-control"></textarea>
  </div>
  <div class="form-group" class="mcontrol">
    <span class="text-danger emsg">{{e}}</span>
    <div class="">
      <button type="submit" class="btn btn-success">Save</button>
      <button type="button" id="av-type" (click)="type()" class="btn btn-
secondary"><span><i class="fas fa-
microphone"></i></span><span class="vt">Voice type</span></button>
    </div>
  </div>
</form>
</div>
<div class="container edit">
  <div class="controls">
    <div class="cc d-none"><span><i class="fas fa-
palette"></i></span><span class="choose"><i class="fas fa-angle-
down"></i></span></div>
    <div class="colors">
      <span><i class="fas fa-check green"></i></span>
      <span><i class="fas fa-check yellow d-none"></i></span>
      <span><i class="fas fa-check blue d-none"></i></span>
      <span><i class="fas fa-check skin d-none"></i></span>
      <span><i class="fas fa-check red d-none"></i></span></div>

```

```

        <button class="additional pin" (click)="pin()"><span class="thumb"><i class
        ="far fa-thumbtack"></i></span></button>
        <button class="additional unpin" (click)="pin()"><span class="thumb"><i cla
        ss="fas fa-thumbtack"></i></span></button>
        <button class="additional" (click)="remove()"><span><i class="fas fa-
        trash"></i></span></button>
        <button class="close"><span><i class="fas fa-times"></i></span></button>
    </div>
<form #edit="ngForm" (ngSubmit)="update()">
    <input type="text" class="d-none noteid">
    <div class="form-group">
        <input type="text" placeholder="Title" [(ngModel)]="un.title" name="heading
        " id="ehheading" class="form-control ro" readonly required>
    </div><hr>
    <div class="form-group">
        <textarea name="content" id="econtent" [(ngModel)]="un.content" placeholder
        ="Content" class="form-control ro" readonly required></textarea>
    </div>
    <button type="button" class="upd btn btn-info"><span><i class="fas fa-
    pen"></i></span>Edit</button>
    <div class="form-group sub d-none m-control">
        <span class="text-danger emsg">{{e}}</span>
        <div class="">
            <button type="submit" class="btn btn-success">Update</button>
            <button type="button" id="ev-type" (click)="etype()" class="btn btn-
            secondary"><span><i class="fas fa-
            microphone"></i></span><span class="vt">Voice type</span></button>
        </div>
    </div>
</form>
</div>
<div class="container-fluid main">
    <div class="notes">
        <div class="controls">
            <h3>Your Notes</h3>
            <div class="edu">
                <button class="create" (click)="cr()" *ngIf="notes_avl.length > 0">
                <span><i class="far fa-
                plus" style="transform: rotate(180deg);"></i></span> Create</button>
            </div>
        </div>
        <hr>
        <div class="not-created container-fluid" *ngIf="notes_avl.length == 0">
            

```


- login.component.scss

```
$sub: #131313;
$headfont: 'Josefin Sans', sans-serif;
$base: #fc5404;
$dalt: #f98404;
$alt: #f9b208;
$mint: #1eae77;

@mixin flex {
  display: flex;
  align-items: center;
  justify-content: center;
}

.lmsg{
  text-align: center;
  color: white;
  opacity: 0.4;
  position: absolute;
  font-family: 'Quicksand';
  font-size: 14px;
  top: 5px;
}

.box{
  .container{
    width: 400px;
    height: 90vh;
    background-color: $sub;
    margin: 0;
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%,-50%);
    border-radius: 10px;
    overflow: hidden;
    @include flex();
    h2{
      text-transform: uppercase;
      color: white;
      font-family: $headfont;
      font-weight: 600;
      font-size: 2em;
      margin: 1em;
    }
    small{
```

```

        // border: 1px solid white;
        position: absolute;
        // width: inherit;
        margin-top: -3px;
        font-size: 13px;
        padding: 0 30px;
    }
    .form-group{
        margin-bottom: 30px;
    }
    .notvalid{
        border-color: #dc3545;
    }
    .panel{
        display: flex;
        flex-direction: row;
        justify-content: center;
        h5{
            color: white;
            letter-spacing: 1px;
            font-family: 'Montserrat';
            font-size: 14px;
            margin: 8px 10px;
            margin-left: 20px;
        }
        button{
            background: transparent;
            border: 2px solid $mint;
            color: white;
            font-family: 'Montserrat';
            padding: 5px 20px;
            height: auto;
            font-size: 15px;
            border-radius: 20px;
            outline: none;
            transition: .25s;
            &:hover{
                background: $mint;
            }
        }
    }
}
input{
    background: transparent;
    padding: 10px 30px;
    text-align: center;

```

```

border: 2px solid $alt;
letter-spacing: 2px;
outline: none;
border-radius: 20px;
font-family: 'Montserrat';
box-shadow: none;
margin: 5px;
color: white;
font-size: 14px;
transition: 0.25s;

&:focus{
    border-color: $dalt;
}
&[type=submit]{
    border: 2px solid $base;
    text-transform: uppercase;
    padding: 10px 30px ;
    font-weight: 500;
    &:hover{
        background: $base;
    }
    margin-bottom: 40px;
}
}
}
.login,.register{
    background: transparent;
    width: 100%;
    transform: translateX(calc(50%));
    transition: .5s;
    // margin: 0;
    padding: 3em;
    form{
        @include flex();
        flex-direction: column;
        padding: 0 50px;
    }
    a{
        color: rgb(59, 59, 59);
        text-decoration: none;
        font-family: 'Montserrat';
        font-size: 14px;
        transition: .25s;
        &:hover{
            color: white;

```



```

    }
  }
}
.register{
  transform: translateX(100%);
  opacity: 0;
  margin: 0;
  padding: 0;
  input{
    font-size: 14px;
    width: 250px;
    font-family: 'Quicksand';
    padding: 10px;
    &[type=submit]{
      margin-top: 10px;
      font-family: 'Montserrat';
      width: auto;
      padding: 10px 30px;
      margin-bottom: 10px;
    }
    &:last-child{
      margin-bottom: 10px;
    }
  }
}
.panel{
  margin-top: 20px;
}
.form-group{
  margin-bottom: 15px;
}
.errbox{
  text-align: center;
}
small{
  position: sticky;
  transform: translateX(-50%);
  left: 50%;
  width: 100%;
  padding: 0;
}
}
}
}

```

```

@media all and (max-width: 500px){

```

```

    .box .container{
      top: 12%;
      left: 0;
      transform: translateY(0);
      width: 100%;
      padding: 0;
    }
  }
}

```

- dashboard.component.scss

```

$sub: #131313;
$headfont: 'Josefin Sans', sans-serif;
$base: #fc5404;
$dalt: #f98404;
$alt: #f9b208;
$mint: #1eae77;
$nred: #ff4356;
$nskin : #feff9c;
$nbblue: #7afcff;
$nyellow: #fff740;
$ngreen: #6cff7d;
.blur{
  filter: blur(7px);
}
.nav-buttons{
  position: absolute;
  right: 0;
  padding: 30px 40px;
  top: 0;
  display: flex;
  align-items: center;
  span{
    font-size: 1.3em;
    font-family: 'Cabin';
    font-weight: 600;
    letter-spacing: 1.3px;
    color: white;
    opacity: 0.8;
  }
  button{
    padding: 10px 20px;
    background: transparent;
    border: 2px solid #dc3545;
  }
}

```

```

        background: transparent;
        color: white;
        outline: none;
        font-size: 16px;
        font-family: 'Montserrat';
        text-transform: uppercase;
        // font-weight: 600;
        border-radius: 60px;
        transition: .25s;
        &.logout:hover{
            background: #dc3545;
        }
    }
}
.add{
    position: fixed;
    width: 600px;
    height: 80%;
    top: 50%;
    left: 50%;
    padding: 60px 0;
    z-index: 990;
    background: $ngreen;
    transform: translate(-50%,-60%);
    transition: 0.5s;
    visibility: hidden;
    opacity: 0;
    &.active{
        visibility: visible;
        opacity: 1;
        transform: translate(-50%,-50%);
    }
    hr{
        border: 1px solid #3b3b3b65;
        width: 90%;
        background: #3b3b3b65;
    }
    input{
        background: none;
        border: none;
        outline: none;
        box-shadow: none;
        &::placeholder{
            color: rgba(34, 34, 34, 0.637);
        }
    }
}

```

```

    &[type=text]{
      font-size: 35px;
      padding: 10px 30px;
      padding-bottom: 0;
      width: 100%;
    }
  }
  .mcontrol{
    display: flex;
    align-items: center;
    justify-content: space-between;
    .emsg{
      padding: 0 30px;
      font-weight: 500;
    }
  }
  textarea{
    height: 350px;
    resize: none;
    padding: 10px 30px;
    background: none;
    font-size: 19px;
    font-family: 'Quicksand';
    font-weight: 500;
    border: none;
    outline: none;
    box-shadow: none;
    &::placeholder{
      color: rgba(34, 34, 34, 0.637);
    }
  }
  button{
    padding: 10px 20px;
    font-size: 18px;
    margin: -5px 20px;
    font-family: 'Quicksand';
    text-transform: uppercase;
    font-weight: 600;
    float: right;
    span{
      margin-right: 10px;
    }
  }
  .controls{
    margin: 0;

```

```

position: absolute;
top: 10px;
right: 10px;
display: flex;
justify-content: right;
button{
    width: 30px;
    height: 30px;
    background: none;
    color: rgb(29, 29, 29);
    span{
        font-size: 20px;
    }
}
.cc{
    transform: translateX(-10px);
    span{
        width: 20px;
        height: 20px;
        color: #292929ad;
        font-size: 20px;
        cursor: pointer;
        &.choose{
            margin-left: 5px;
            font-size: 18px;
            transition: 0.5s;
            &:hover{
                color: #292929;
            }
            &.active{

                transform: rotate(180deg) translate(9px,-5px);
            }
        }
    }
}

.colors{
    padding: 5px;
    display: flex;
    position: absolute;
    // top: 50px;
    left: -50px;
    opacity: 0;
    transform: translateY(0);

```

```

        justify-content: space-between;
        background: #1a1a1a49;
        transition: 0.5s;
        z-index: -99;
        &.active{
            transform: translateY(35px);
            opacity: 1;
        }

        span{
            height: 20px;
            width: 20px;
            margin: 0 5px;
            display: flex;
            color: #292929ad;
            align-items: center;
            justify-content: center;
            font-size: 14px;
            &:first-child{
                background-color: $ngreen;
            }
            &:nth-child(2){
                background-color: $nyellow;
            }
            &:nth-child(3){
                background-color: $nblue;
            }
            &:nth-child(4){
                background-color: $nskin;
            }
            &:nth-child(5){
                background-color: $nred;
            }
        }
    }
    select{
        background: none;
        border: none;
    }
}

.edit{
    @extend .add;
    visibility: hidden;
    opacity: 0;

```

```

.controls{
  button.additional{
    width: 30px;
    padding: 0;
    margin: 0;
    height: 30px;
    margin: 0 5px;
    display: flex;
    align-items: center;
    justify-content: center;
    border: none;
    transition: .25s;
    // display: none;
    span{
      margin: 0;
      font-size: 20px;
      color: rgba(29, 29, 29, 0.664);
      // &.thumb{
      //   font-size: 23px;
      // }
      &:hover{
        color: rgba(29, 29, 29, 0.808);
      }
    }
  }
}
}

.center{
  display: flex;
  align-items: center;
  justify-content: center;
}

button{
  outline: none;
}

.container-fluid{
  margin-top: calc(2.6em + 70px);
  padding: 40px;
  .not-created{
    display: flex;
    justify-content: space-evenly;
    img{
      width: 15em;
      height: 15em;
    }
  }
}

```

```

.txt{
  width: 60%;
  color: white;
  h1{
    font-family: 'Caveat';
    font-size: 50px;
  }
  button{
    border: 2px solid $alt;
    background: none;
    margin-top: 30px;
    font-family: 'Quicksand';
    letter-spacing: 2px;
    font-weight: 600;
    color: white;
    text-transform: uppercase;
    font-size: 16px;
    padding: 5px 15px;
    display: flex;
    align-items: center;
    border-radius: 60px;
    transition: 0.25s;
    &.create:hover{
      background: $alt;
    }
    span{
      margin-right: 10px;
      font-size: 25px;
    }
  }
}
}

.notes{
  .perday{
    border: 2px solid rgb(77, 77, 77);
    .date{
      font-size: 19px;
      color: rgb(151, 151, 151);
      font-family: 'Quicksand';
      font-weight: 600;
      // margin: 30px;
      position: relative;
      left: 30px;
      top: -30px;
      width: 250px;
    }
  }
}

```



```

        text-align: center;
        background: #1a1a1a;
        letter-spacing: 2px;
    }
    margin: 0;
    margin-top: 30px;
    width: 100%;
    padding: 20px;
    .note{
        width: 300px;
        height: 300px;
        word-wrap: break-word;
        background: $mint;
        margin: 20px 0;
        padding: 20px;
        overflow: hidden;
        &::after{
            content: '';
            position: absolute;
            height: 20px;
            background: inherit;
            width: 80%;
            left: 30px;
            bottom: 20px;
            margin: 0;
        }
        h4{
            font-family: 'Josefin Sans';
        }
        p{
            font-family: 'Quicksand';
            font-weight: 500;
            font-size: 17px;
        }
    }
}
.controls{
    display: flex;
    justify-content: space-between;
    align-items: center;
    padding: 0 20px;
    h3{
        color: white;
        font-family: 'Josefin Sans';
    }
}

```

```

        .edu{
            button{
                border: 2px solid $alt;
                background: none;
                font-family: 'Quicksand';
                letter-spacing: 2px;
                font-weight: 600;
                color: white;
                text-transform: uppercase;
                font-size: 16px;
                padding: 5px 15px;
                display: flex;
                align-items: center;
                border-radius: 60px;
                transition: 0.25s;
                &.create:hover{
                    background: $alt;
                }
                span{
                    margin-right: 10px;
                    font-size: 25px;
                }
            }
        }
    }
    hr{
        border: 1px solid rgb(77, 77, 77);
        background: rgb(77, 77, 77);
    }
}

```

- login.component.ts

```

import { Component, OnInit } from '@angular/core';
import { FormGroup } from '@angular/forms';
import { Router } from '@angular/router';
import * as $ from 'jquery';
import { AUTHENTICATED_USER, AuthService, TOKEN, TOKEN_EXPIRY, User, USER_ID
    } from '../service/auth.service';

@Component({
    selector: 'app-login',

```

```

templateUrl: './login.component.html',
styleUrls: ['./login.component.scss']
}))

export class LoginComponent implements OnInit {

  username:string = '';
  lmsg = ""
  password:string = '';
  invalidCredentials:boolean;
  serror:boolean;
  msg = '';

  u:User = new User('', '', '', '');

  constructor(private us:AuthService, private router:Router) {
    this.invalidCredentials = false;
    this.serror = false;
  }

  untouch(form) {
    Object.keys(form.controls).forEach(key => {
      form.get(key).markAsUntouched();
    });
  }

  signIn(form:FormGroup) {
    if(form.valid){
      this.us.authenticate(this.username,this.password).subscribe(
        data => {
          sessionStorage.setItem(TOKEN_EXPIRY, data.expiry);
          sessionStorage.setItem(USER_ID, "" + data.userId);
          sessionStorage.setItem(AUTHENTICATED_USER, this.username);
          sessionStorage.setItem(TOKEN, `Token ${data.token}`);
          this.router.navigate(['dashboard']);
        },
        error => {
          console.log(error);
          this.invalidCredentials = true;
        }
      )
    }
    else{
      form.markAllAsTouched()
      setTimeout(()=>{

```

```

        this.untouch(form);
    },3000)
}
}

signUp(form:FormGroup){
    if(form.valid){
        let user ={}
        let n = this.u.name.split(" ")
        if(n.length == 1){
            user["first_name"] = n[0]
            user["last_name"] = ""
        }
        else if(n.length == 2){
            user["first_name"] = n[0]
            user["last_name"] = n[1]
        }
        else{
            user["first_name"] = n[0]
            let s = ""
            for(var i = 1;i<n.length;i++){
                s += n[i]+" "
            }
            user["last_name"] = s.trim()
        }
        user["email"] = this.u.email
        user["username"] = this.u.uname
        user["password"] = this.u.pass
        this.us.addUser(user).subscribe(
            response => {
                // alert("Your account is created, please login to continue.");
                this.lmsg= "Your account is created, login to continue"
                setTimeout(()=>{this.lmsg = ''},3000)
                form.reset()
                $('.register').css('transform','translateX(calc(100%))').css('opacity',0);
                $('.login').css('transform','translateX(calc(50%))').css('opacity',1);
                // this.router.navigate(['login'])
            },
            error => {
                // console.log(error);
                this.serror = true;
                if(error.error.username){
                    this.msg = "Username already exists"
                }
            }
        )
    }
}

```

```

        setTimeout(()=>{this.serror = false
        this.msg = ''},3000)
    }

    }
)
}
else{
    form.markAllAsTouched()
    setTimeout(()=>{
        this.untouch(form);
    },3000)
}
}

ngOnInit(): void {

    $('.register .panel button').on('click',function(){
        $('.register').css('transform','translateX(calc(100%))').css('opacity',0);
        $('.login').css('transform','translateX(calc(50%))').css('opacity',1);
    })

    $('.login .panel button').on('click',function(){
        $('.login').css('transform','translateX(calc(-100%))').css('opacity',0);
        $('.register').css('transform','translateX(calc(-50%))').css('opacity',1);
    })

}

}

```

- dashboard.component.ts

```

import { DatePipe } from '@angular/common';
import { Component, OnInit } from '@angular/core';
import { FormGroup } from '@angular/forms';
import { Router } from '@angular/router';
import * as $ from 'jquery';
import { cssNumber } from 'jquery';
import { AUTHENTICATED_USER, AuthService, TOKEN, TOKEN_EXPIRY, USER_ID } from '../service/auth.service';

```

```

declare var webkitSpeechRecognition: any;
import { NoteService } from '../service/dash/note.service';
export class Note{
  constructor(
    public title:string,
    public content:string
  ){}
}
@Component({
  selector: 'app-dashboard',
  templateUrl: './dashboard.component.html',
  styleUrls: ['./dashboard.component.scss']
})
export class DashboardComponent implements OnInit {

  fname:string;
  pinlist = []
  pinnotes = []

  constructor(public us:AuthService, private router:Router, private ns:NoteService) {
    this.us.findUserByUsername(sessionStorage.getItem(AUTHENTICATED_USER)).subscribe(
      data=>{
        let n = data.user.first_name
        this.fname = n.toUpperCase()[0] + n.substr(1,n.length)
      }
    )
  }

  // n:Note[] = [new Note("Good Morning","hey to all"),new Note("Good afternoon","hey to all")]

  logout(){
    let c = confirm('Are you sure to log out?')
    if(c){
      this.us.logout().subscribe(
        result =>{
          sessionStorage.removeItem(AUTHENTICATED_USER);
          sessionStorage.removeItem(TOKEN);
          sessionStorage.removeItem(USER_ID);
          sessionStorage.removeItem(TOKEN_EXPIRY);
          this.router.navigate(['login'])
        },
        error=>{

```

```

        console.log(error);
    }
}
}

cn = new Note('', '');
un = new Note('', '')
e = ''

save(){
    if(this.cn.title != '' || this.cn.content != ''){
        if(this.cn.title == ''){
            this.cn.title = ""+$('#acontent').val()
            this.cn.content = ''
        }
        let n = {}
        n["title"] = this.cn.title
        // if(this.cn.content.length)
        // n["content"] = this.cn.content
        // else
        n["content"] = $('#acontent').val()
        let bg = $('#.add').css('background')
        let u = bg.split(" n")[0].trim()
        // console.log(u)

        this.ns.addNote(this.us.getAuthenticatedUser(),n,u).subscribe(
            response => {
                $('#.main').removeClass('blur');
                $('#.nav-buttons').removeClass('blur');
                $('#.add').removeClass('active');
                $('#.add.colors').removeClass('active');
                $('#.add.controls.choose').removeClass('active');
                this.notes_avl = []
                this.pinnotes = []
                this.aspeech.abort()
                this.refreshAll()
                $('#acontent').val('')
                this.cn = new Note('', '')
            },
            error=>{
                this.e = "Some error occured, please try again later :("
                console.log(error)
                setTimeout(()=>{this.e=""},3000)
                this.cn = new Note('', '')
            }
        )
    }
}

```

```

    }
  )
}
else{
  this.e = "Fields shouldn't be empty"
  setTimeout(()=>{this.e=""},3000)
}
}

open(n){
  if(this.pinlist.includes(n)){
    $('.edit .pin').addClass('d-none')
    $('.edit .unpin').removeClass('d-none')
  }
  else{
    $('.edit .pin').removeClass('d-none')
    $('.edit .unpin').addClass('d-none')
  }
  this.ns.get(n).subscribe(
    response=>{
      this.un = new Note(response.user.title,response.user.content)
      $('.edit').css('background',response.user.color);
      $('.edit .noteid').val(response.user.id)
    }
  )
  $('.main').addClass('blur');
  $('.nav-buttons').addClass('blur');
  $('.edit').addClass('active');
}

update(){
  if(this.un.title != '' || this.un.content != ''){
    if(this.un.title == ''){
      this.un.title = ""+$('#econtent').val()
      this.un.content = ''
    }
    let n = {}
    n["title"] = this.un.title
    n["content"] = $('#econtent').val()
    let bg = $('.edit').css('background')
    let u = bg.split(" n")[0].trim()
    n["color"] = u
    // console.log(u)
  }
}

```



```

let nid = $('.edit .noteid').val()

this.ns.updateNote(this.us.getAuthenticatedUser(),nid,n).subscribe(
  response => {
    $('.main').removeClass('blur');
    $('.nav-buttons').removeClass('blur');
    $('.edit').removeClass('active');
    $('.edit .colors').removeClass('active');
    $('.edit .controls .choose').removeClass('active');
    $('.edit .ro').attr('readonly','true');
    $('.edit .sub').addClass('d-none');
    $('.edit .upd').removeClass('d-none');
    $('.edit .cc').addClass('d-none');
    this.notes_avl = []
    this.pinnotes = []
    this.refreshAll()
    this.espeech.abort()
    $('#econtent').val('')
    this.un = new Note('', '')
  },
  error=>{
    this.e = "Some error ocured, please try again later :("
    console.log(error)
    setTimeout(()=>{this.e=""},3000)
    this.cn = new Note('', '')
  }
)
}
else{
  this.e = "Fields shouldn't be empty"
  setTimeout(()=>{this.e=""},3000)
}
}

remove(){
  let nid = $('.edit .noteid').val()
  if(confirm("Are you sure to delete that note, changes are irrevertible.")){
    this.ns.remove(nid).subscribe(
      response => {
        $('.main').removeClass('blur');
        $('.nav-buttons').removeClass('blur');
        $('.edit').removeClass('active');
        $('.edit .colors').removeClass('active');
        $('.edit .controls .choose').removeClass('active');

```

```

        $('.edit .ro').attr('readonly','true');
        $('.edit .sub').addClass('d-none');
        $('.edit .upd').removeClass('d-none');
        $('.edit .cc').addClass('d-none');
        this.notes_avl = []
        this.pinnotes = []
        this.refreshAll()
        this.un = new Note('','')
    },
    error=>{
        this.e = "Some error ocured, please try again later :("
        console.log(error)
        setTimeout(()=>{this.e=""},3000)
        this.cn = new Note('','')
    }
}
)
}
}
}

```

```

pin(){
    let nid = $('.edit .noteid').val()
    this.ns.setPin(this.us.getAuthenticatedUser(),nid).subscribe(
        response=>{
            let found = false
            for(var i =0;i<this.pinlist.length;i++){
                if(nid == this.pinlist[i]){
                    this.pinlist.splice(i,1)
                    found = true
                    break
                }
            }
            if(!found){
                let x = parseInt(nid+"")
                this.pinlist.push(x)
            }
            $('.main').removeClass('blur');
            $('.nav-buttons').removeClass('blur');
            $('.edit').removeClass('active');
            $('.edit .colors').removeClass('active');
            $('.edit .controls .choose').removeClass('active');
            $('.edit .ro').attr('readonly','true');
            $('.edit .sub').addClass('d-none');
            $('.edit .upd').removeClass('d-none');

```

```

        $('.edit .cc').addClass('d-none');
        this.notes_avl = []
        this.pinnotes = []
        this.refreshAll()
        // console.log(this.pinlist)
    }
)
}

cr(){
    $('.main').addClass('blur');
    $('.nav-buttons').addClass('blur');
    $('.add').addClass('active');
}

notes_avl = []

refreshAll(){
    let now = new Date()
    let pipe = new DatePipe('en-us').transform(now, 'yyyy-MM-dd')
    this.ns.getAll(this.us.getAuthenticatedUser(), '0', pipe).subscribe(
        response=>{
            // console.log(response)
            let key:Array<string> = response.keys
            key.forEach(x => {
                let frame = []
                let y = response.notes[x]
                if(x.includes("TODAY")){
                    frame.push("TODAY")
                }
                else if(x.includes("YESTERDAY")){
                    frame.push("YESTERDAY")
                }
                else{
                    frame.push(x)
                }
                frame.push([])
                y.forEach(i => {
                    i["created_on"] = new Date(i["created_on"])
                    if(i["pinned"]){
                        this.pinlist.push(i["id"])
                        this.pinnotes.push(i)
                    }
                    frame[1].push(i)
                });
            });
        }
    );
}

```

```

        frame[1].sort((a,b)=>b.created_on - a.created_on)
        frame.push(new Date(x))
        this.notes_avl.push(frame)
    });
    this.notes_avl.sort((a,b)=> b[2]-a[2])
    // console.log(this.notes_avl)
}
)
}

```

```

espeech = new webkitSpeechRecognition()

```

```

etype(){
    var ebox = $('#econtent')
    var econtentfill = ''
    this.espeech.continuous = true
    this.espeech.lang = 'en'

    this.espeech.onstart = function(e){
        $('#ev-type .vt').text("Listening")
        $('#ev-type').addClass('active')
    }

    this.espeech.onend = function(e){
        $('#ev-type').removeClass('active')
        $('#ev-type .vt').text("Voice type")
    }

    this.espeech.onspeechend = function(e){
        $('#ev-type .vt').text("No activity")
        setTimeout(()=>{
            this.aspeech.abort()
            $('#ev-type').removeClass('active')
            $('#ev-type .vt').text("Voice type")
        },2000)
    }

    this.espeech.onerror = function(e){
        $('#ev-type .vt').text("Try again")
        if(e.error == 'network'){
            alert('No network connection :/')
        }
        setTimeout(()=>{
            this.aspeech.abort()
            $('#ev-type').removeClass('active')
        }
    }
}

```

```

        $('#ev-type .vt').text("Voice type")
    },2000)
}

this.espeech.onresult = function(e){
    var curr = e.resultIndex
    var trans = e.results[curr][0].transcript
    // console.log(e.results)
    econtentfill += trans
    ebox.val(econtentfill)
}

ebox.on('input',function(){
    econtentfill = ''
    econtentfill += $(this).val()
})

if($('#ev-type').hasClass('active')){
    this.espeech.abort()
}
else{
    if(ebox.val())
        econtentfill += ebox.val()+' '
    else
        econtentfill = ''
    this.espeech.start()
}
}

aspeech = new webkitSpeechRecognition()

type(){
    // var aspeech = new SpeechRecognition()
    var abox = $('#acontent')
    var contentfill = ''
    this.aspeech.continuous = true
    this.aspeech.lang = 'en'

    this.aspeech.onstart = function(e){
        $('#av-type .vt').text("Listening")
        $('#av-type').addClass('active')
    }

    this.aspeech.onend = function(e){
        $('#av-type').removeClass('active')
    }
}

```

```

    $('#av-type .vt').text("Voice type")
}

this.aspeech.onspeechend = function(e){
    $('#av-type .vt').text("No activity")
    setTimeout(()=>{
        this.aspeech.abort()
        $('#av-type').removeClass('active')
        $('#av-type .vt').text("Voice type")
    },2000)
}

this.aspeech.onerror = function(e){
    $('#av-type .vt').text("Try again")
    if(e.error == 'network'){
        alert('No network connection :/')
    }
    setTimeout(()=>{
        this.aspeech.abort()
        $('#av-type').removeClass('active')
        $('#av-type .vt').text("Voice type")
    },2000)
}

this.aspeech.onresult = function(e){
    var curr = e.resultIndex
    var trans = e.results[curr][0].transcript
    // console.log(e.results)
    contentfill += trans
    abox.val(contentfill)
}

abox.on('input',function(){
    contentfill = ''
    contentfill += $(this).val()
})

if($('#av-type').hasClass('active')){
    this.aspeech.abort()
}
else{
    if(abox.val())
        contentfill += abox.val()+' '
    else
        contentfill = ''
}

```

```

        this.aspeech.start()
    }
}

ngOnInit(): void {

    var asp = this.aspeech
    this.refreshAll()

    $('.edu .create').on('click',function(){
        $('.main').addClass('blur');
        $('.nav-buttons').addClass('blur');
        $('.add').addClass('active');
    })

    $('.add .controls .close').on('click',function(){
        asp.abort()
        $('.main').removeClass('blur');
        $('.nav-buttons').removeClass('blur');
        $('.add').removeClass('active');
        $('.add .colors').removeClass('active');
        $('.add .controls .choose').removeClass('active');
    })
    $('.edit .upd').on('click',function(){
        $(this).addClass('d-none');
        $('.edit .cc').removeClass('d-none');
        $('.edit .ro').removeAttr('readonly');
        $('.edit .sub').removeClass('d-none');
    })

    $('.edit .controls .pin .fa-thumbtack').on('click',function(){
        alert('kl')
        if($(this).hasClass('far'))
            $(this).removeClass('far').addClass('fas')
        else
            $(this).removeClass('fas').addClass('far')
    })

    $('.edit .controls .close').on('click',function(){
        $('.main').removeClass('blur');
        $('.nav-buttons').removeClass('blur');
        $('.edit').removeClass('active');
        $('.edit .ro').attr('readonly','true');
        $('.edit .sub').addClass('d-none');
        $('.edit .upd').removeClass('d-none');
    })

```

```

    $('.edit .cc').addClass('d-none');
    $('.edit .colors').removeClass('active');
    $('.edit .controls .choose').removeClass('active');
  })
  $('.add .controls .choose').on('click',function(){
    if($(this).hasClass('active')){
      $('.add .colors').removeClass('active');
      $(this).removeClass('active');
    }else{
      $('.add .colors').addClass('active');
      $(this).addClass('active');
    }
  })
  $('.add .colors').children().each(function(x,e){
    $(e).on('click',function(){
      switch(x){
        case 0:$('.add').css('background','#6cff7d');
          $(".green").removeClass('d-
none').parent().siblings().children().addClass('d-none');
          break;
        case 1:$('.add').css('background','#fff740');
          $(".yellow").removeClass('d-
none').parent().siblings().children().addClass('d-none'); break;
        case 2:$('.add').css('background','#7afcff');
          $(".blue").removeClass('d-
none').parent().siblings().children().addClass('d-none'); break;
        case 3:$('.add').css('background','#feff9c');
          $(".skin").removeClass('d-
none').parent().siblings().children().addClass('d-none'); break;
        case 4:$('.add').css('background','#ff4356');
          $(".red").removeClass('d-
none').parent().siblings().children().addClass('d-none'); break;
      }
      $('.add .colors').removeClass('active');
      $('.add .choose').removeClass('active');
    })
  })
  $('.edit .controls .choose').on('click',function(){
    if($(this).hasClass('active')){
      $('.edit .colors').removeClass('active');
      $(this).removeClass('active');
    }else{
      $('.edit .colors').addClass('active');
      $(this).addClass('active');
    }
  })

```



```

    })

    $('.edit .colors').children().each(function(x,e){
        // b = $('.edit').css('background');
        $(e).on('click',function(){
            switch(x){
                case 0:$('.edit').css('background','#6cff7d');
                $(".green").removeClass('d-
none').parent().siblings().children().addClass('d-none');
                break;
                case 1:$('.edit').css('background','#fff740');
                $(".yellow").removeClass('d-
none').parent().siblings().children().addClass('d-none'); break;
                case 2:$('.edit').css('background','#7afcff');
                $(".blue").removeClass('d-
none').parent().siblings().children().addClass('d-none'); break;
                case 3:$('.edit').css('background','#feff9c');
                $(".skin").removeClass('d-
none').parent().siblings().children().addClass('d-none'); break;
                case 4:$('.edit').css('background','#ff4356');
                $(".red").removeClass('d-
none').parent().siblings().children().addClass('d-none'); break;
            }
            $('.edit .colors').removeClass('active');
            $('.edit .choose').removeClass('active');
        })
    })

}

}

```

- note.service.ts

```

import { HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { APP_URL } from 'src/app.constants';

@Injectable({
    providedIn: 'root'
})
export class NoteService {

    constructor(private http:HttpClient) { }

```

```

addNote(uname,n,u){
    return this.http.post(`${APP_URL}/data/${uname}/add/${u}`,n)
}

setPin(uname,nid){
    return this.http.put(`${APP_URL}/data/setpin/${uname}/${nid}`,{})
}

updateNote(uname,nid,n){
    return this.http.put(`${APP_URL}/data/update/${uname}/${nid}`,n)
}

getAll(uname,from,to){
    return this.http.get<any>(`${APP_URL}/data/${uname}/getall/${from}/${to}`)
}

get(nid){
    return this.http.get<any>(`${APP_URL}/data/get/${nid}`)
}

remove(nid){
    return this.http.delete(`${APP_URL}/data/remove/${nid}`)
}
}

```

- auth.service.ts

```

import { HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { APP_URL, AUTH_HEADERS, CORS_HEADERS } from 'src/app.constants';
import { map } from 'rxjs/operators';

export class User{
    constructor(
        public name:string,
        public email:string,
        public uname:string,
        public pass:string
    ){
    }
}

export class model{

```

```

    constructor(
        public id: number,
        public first_name:string,
        public last_name:string,
        public username: string,
        public email: string
    ){
    }
}

export const AUTHENTICATED_USER = 'authenticatedUser';
export const TOKEN = 'token';
export const USER_ID = 'userId';
export const TOKEN_EXPIRY = 'expiry';

export class AuthResponse {
    constructor(public userId: number,public token:string, public expiry:string, public first_name:string) { }
}

@Inject({
    providedIn: 'root'
})
export class AuthService {

    constructor(private http:HttpClient) { }

    findById(uid){
        return this.http.get<User>(`${APP_URL}/user/uid/${uid}`);
    }

    findByUsername(uname){
        return this.http.get<any>(`${APP_URL}/user/uname/${uname}`);
    }

    addUser(u){
        return this.http.post(`${APP_URL}/user/add`,u);
    }

    authenticate(username:string,password:string){
        return this.http.post<AuthResponse>(`${APP_URL}/user/login`, { username, password }).pipe(
            map(
                data => {
                    sessionStorage.setItem(TOKEN_EXPIRY, data.expiry);
                }
            )
        );
    }
}

```

```

        sessionStorage.setItem(USER_ID, "" + data.userId);
        sessionStorage.setItem(AUTHENTICATED_USER, username);
        sessionStorage.setItem(TOKEN, `Token ${data.token}`);
        sessionStorage.setItem('name', data.first_name);
        return data;
    }
    )
    );
}

isUserLoggedIn() {
    let user = sessionStorage.getItem(AUTHENTICATED_USER);
    return !(user == null);
}

logout() {
    return this.http.post(`${APP_URL}/user/logout`, {})
}

getAuthenticatedUserId(): number {
    return parseInt(sessionStorage.getItem(USER_ID));
}

getAuthenticatedUser() {
    return sessionStorage.getItem(AUTHENTICATED_USER);
}

getAuthenticatedToken() {
    if (this.getAuthenticatedUser())
        return sessionStorage.getItem(TOKEN);
}
}

```

- http-interceptor.service.ts

```

import { Injectable } from '@angular/core';
import { HttpInterceptor, HttpRequest, HttpHandler } from '@angular/common/http';
import { AuthService } from '../auth.service';

@Injectable({
    providedIn: 'root'
})
export class HttpInterceptorBasicAuthService implements HttpInterceptor {

```

```

constructor(private us: AuthService) { }

intercept(request: HttpRequest<any>, next: HttpHandler) {

    let basicAuthHeaderString = this.us.getAuthenticatedToken();
    let username = this.us.getAuthenticatedUser();
    if (basicAuthHeaderString && username) {
        request = request.clone({
            setHeaders: {
                Authorization : basicAuthHeaderString
            }
        })
    }
    return next.handle(request);
}
}

```

- route-guard.guard.ts

```

import { Injectable } from '@angular/core';
import { CanActivate, ActivatedRouteSnapshot, RouterStateSnapshot, UrlTree, Router } from '@angular/router';
import { Observable } from 'rxjs';
import { AuthService } from '../auth.service';

@Injectable({
    providedIn: 'root'
})
export class GuardGuard implements CanActivate {

    constructor(private us:AuthService,private router:Router){}

    canActivate(
        route: ActivatedRouteSnapshot,
        state: RouterStateSnapshot): Observable<boolean | UrlTree> | Promise<boolean | UrlTree> | boolean | UrlTree {
        if(this.us.isUserLoggedIn()){
            return true;
        }else{
            this.router.navigate(['forbidden']);
        }
    }
}

```

```
}
```

- app.routing.module.ts

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
import { DashboardComponent } from '../dashboard/dashboard.component';
import { ForbiddenComponent } from '../forbidden/forbidden.component';
import { LoginComponent } from '../login/login.component';
import { PageNotFoundComponent } from '../page-not-found/page-not-found.component';
import { GuardGuard } from '../service/guard.guard';

const routes: Routes = [
  { path: '', redirectTo: '/login', pathMatch: 'full' },
  { path: 'login', component: LoginComponent },
  { path: 'dashboard', component: DashboardComponent, canActivate: [GuardGuard] },
  { path: 'forbidden', component: ForbiddenComponent },
  { path: '**', component: PageNotFoundComponent }
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

- app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { HttpClientModule, HTTP_INTERCEPTORS } from '@angular/common/http';
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { LoginComponent } from '../login/login.component';
import { FormsModule } from '@angular/forms';
import { DashboardComponent } from '../dashboard/dashboard.component';
import { AuthService } from '../service/auth.service';
import { HttpInterceptorBasicAuthService } from '../service/http/http-interceptor-basic-auth.service';
import { ForbiddenComponent } from '../forbidden/forbidden.component';
```

```

import { PageNotFoundComponent } from './page-not-found/page-not-found.component';

@NgModule({
  declarations: [
    AppComponent,
    LoginComponent,
    DashboardComponent,
    ForbiddenComponent,
    PageNotFoundComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    FormsModule,
    HttpClientModule
  ],
  providers: [
    AuthService,
    { provide: HTTP_INTERCEPTORS, useClass: HttpInterceptorBasicAuthService, multi: true }
  ],
  bootstrap: [AppComponent]
})
export class AppModule { }

```

- style.scss

```

$main : #1a1a1a;

html{
  background-color: $main;
}

::-webkit-scrollbar{
  width: 7px;
}

::-webkit-scrollbar-track{
  background: transparent;
  box-shadow: inset 0 0 4px black;
}

::-webkit-scrollbar-thumb{

```

```

background: linear-
gradient(to bottom, rgba(180, 179, 179, 0.637), rgba(128, 128, 128, 0.
712));
border-radius: 100px;
}

```

- accounts/views.py

```

from django.shortcuts import render

# Create your views here.
from rest_framework import generics, permissions, status
from rest_framework.response import Response
from knox.models import AuthToken
from rest_framework.views import APIView
from .serializers import UserSerializer, RegisterSerializer

# Register API
class RegisterAPI(generics.GenericAPIView):
    serializer_class = RegisterSerializer

    def post(self, request, *args, **kwargs):
        serializer = self.get_serializer(data=request.data)
        if serializer.is_valid(raise_exception=True):
            user = serializer.save()
            return Response({
                "success": 1,
                "message": "User Created",
                "user": UserSerializer(user, context=self.get_serializer_co
ntext()).data,
                # "token": AuthToken.objects.create(user)[1],
            }, status.HTTP_201_CREATED)
        return Response({
            "success": 0,
            "message": serializer.errors,
        }, status.HTTP_409_CONFLICT)

from django.contrib.auth import login

from rest_framework import permissions
from rest_framework.authtoken.serializers import AuthTokenSerializer
from knox.views import LoginView as KnoxLoginView

class LoginAPI(KnoxLoginView):

```



```

permission_classes = (permissions.AllowAny,)

def post(self, request, format=None):
    serializer = AuthTokenSerializer(data=request.data)
    if(serializer.is_valid(raise_exception=True)):
        user = serializer.validated_data['user']
        login(request, user)
        print(type(request.user))
        data = super(LoginAPI, self).post(request, format=None).data
        data["success"]=1
        data["userId"]= user.id
        data["name"]=user.first_name
        data["message"]="Login success"
        return Response(data=data,status=status.HTTP_202_ACCEPTED)
    return Response(data={"message":serializer.errors,"success":0},status=status.HTTP_401_UNAUTHORIZED)

from django.contrib.auth.models import User

class getUserByUsername(APIView):
    def get(self, request, username, format=None):
        print(username)
        user = User.objects.get(username = username)
        return Response({
            "user": UserSerializer(user).data,
        })

```

- dashboard/views.py

```

from .models import Note
import arrow
from django.contrib.auth.models import User
from django.shortcuts import render
from rest_framework import status
from rest_framework.response import Response
from rest_framework.views import APIView
from .serializers import NoteSerializer
from django.db.models.query_utils import Q

# Create your views here.
class createNote(APIView):
    def post(self, request, uname, color, format=None):
        obj = request.data
        obj["color"] = color

```

```

    obj["uname"] = User.objects.get(username = uname)
    serializer = NoteSerializer(data=obj)
    if(serializer.is_valid(raise_exception=True)):
        print(obj)
        note = serializer.save()
        return Response({
            "success": 1,
            "message": "Note Created",
        }, status.HTTP_201_CREATED)
    return Response({
        "success": 0,
        "message": serializer.errors,
    }, status.HTTP_409_CONFLICT)

class setPin(APIView):
    def put(self, request, nid, uname, format=None):
        note = Note.objects.get(id = nid)
        note.pinned = not(note.pinned)
        obj = {}
        obj["title"] = note.title
        obj["content"] = note.content
        obj["uname"] = User.objects.get(username = uname)
        serializer = NoteSerializer(note, data=obj)
        if(serializer.is_valid(raise_exception=True)):
            n1 = serializer.save()
            return Response({
                "success": 1,
                "message": "Note Updated",
            }, status.HTTP_201_CREATED)
        return Response({
            "success": 0,
            "message": serializer.errors,
        }, status.HTTP_409_CONFLICT)

class updateNote(APIView):
    def put(self, request, uname, nid, format=None):
        note = Note.objects.get(id = nid)
        obj = request.data
        obj["uname"] = User.objects.get(username = uname)
        serializer = NoteSerializer(note, data=obj)
        if(serializer.is_valid(raise_exception=True)):
            n1 = serializer.save()
            return Response({
                "success": 1,

```

```

        "message": "Note Updated",
    }, status.HTTP_201_CREATED)
    return Response({
        "success": 0,
        "message": serializer.errors,
    }, status.HTTP_409_CONFLICT)

class getNote(APIView):
    def get(self, request, nid, format=None):
        # print(nid)
        note = Note.objects.get(id = nid)
        return Response({
            "user": NoteSerializer(note).data,
        })

class removeNote(APIView):
    def delete(self, request, nid, format=None):
        note = Note.objects.get(id = nid)
        note.delete()
        return Response({
            "msg": "Note id: "+str(nid)+" deleted successfully",
        })

class getNotesBetween(APIView):
    def get(self, request, from_, to_, uname, format=None):
        if(from_ == '0'):
            user = User.objects.get(username = uname)
            from_ = user.date_joined
            note = Note.objects.filter(Q(created_on__gte = from_) | Q(created_on__lte = to_), uname = uname)
            result = {}
            for x in note:
                utc = arrow.get(x.created_on)
                local = utc.to('Asia/Kolkata')
                curr = arrow.utcnow()
                prev = curr.shift(days=-1)
                prevDate = prev.strftime("%B")+" "+prev.strftime("%d")+", "+prev.strftime("%Y")
                currDate = curr.strftime("%B")+" "+curr.strftime("%d")+", "+curr.strftime("%Y")
                obj = local.strftime("%B")+" "+local.strftime("%d")+", "+local.strftime("%Y")
                if(obj == currDate):
                    obj = "TODAY "+currDate
                elif(obj == prevDate):

```

```

        obj = "YESTERDAY "+prevDate
        data = NoteSerializer(x).data
        if obj not in result.keys():
            result[obj] = [data,]
            continue
        result[obj].append(data)
    # data = NoteSerializer(note,many=True).data
    return Response({
        "notes": result,
        "keys": result.keys()
    })

```

- config/settings.py

```

from pathlib import Path

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent

# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/3.2/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = ['127.0.0.1', 'localhost',]

# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'accounts',
    'dashboard',
    'rest_framework',
    'knox',

```

```

        'corsheaders',
        'arrow'
    ]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
    'corsheaders.middleware.CorsMiddleware',
]

ROOT_URLCONF = 'config.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

WSGI_APPLICATION = 'config.wsgi.application'

# Database
# https://docs.djangoproject.com/en/3.2/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}

```

```

# Password validation
# https://docs.djangoproject.com/en/3.2/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]

# Internationalization
# https://docs.djangoproject.com/en/3.2/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'Asia/Kolkata'

USE_I18N = True

USE_L10N = True

USE_TZ = True

# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/3.2/howto/static-files/

STATIC_URL = '/static/'

# Default primary key field type
# https://docs.djangoproject.com/en/3.2/ref/settings/#default-auto-field

```

```
DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'

CORS_ORIGIN_ALLOW_ALL = False

CORS_ORIGIN_WHITELIST = (
    'http://127.0.0.1:4200',
)

CORS_ALLOWED_ORIGINS = ['http://localhost:4200', 'http://127.0.0.1:4200',]

CORS_ALLOW_CREDENTIALS = True

# AUTH_USER_MODEL = 'accounts.myuser'

# 'rest_framework.authentication.BasicAuthentication',
# 'rest_framework.authentication.SessionAuthentication',

REST_FRAMEWORK = {
    'DEFAULT_AUTHENTICATION_CLASSES': [
        'knox.auth.TokenAuthentication',
    ]
}
```

CHAPTER-9

REFERENCES

- [1]. <https://notetakingexpert.com/>
- [2]. <https://en.wikipedia.org/wiki/Note-taking>
- [3]. <https://www.genietutors.co.uk/>
- [4]. <https://www.google.co.in/>
- [5]. <https://www.study.com/>