

LIST OF FIGURES

Fig 4.1 : Home Screen.....	13
Fig 4.2 : Adding item into the List.....	14
Fig 4.3 : Added item into the List.....	14
Fig 4.4 : Checked Checkbox.....	15
Fig 4.5 : Deleted Item.....	15
Fig 4.6 : Creating new module.....	16
Fig 4.7 : Creating new module.....	16
Fig 4.8 : Running Server file.....	17
Fig 4.9 : Starting process of MongoDB 1t.....	18
Fig 4.10 : Starting process of MongoDB 2t.....	18
Fig 4.11 : todolistdb Database.....	19
Fig 4.12 : Datas of todolistdb Database.....	19
Fig 4.13 : Database representation when an item is added.....	20
Fig 4.14 : College List.....	20
Fig 4.15 : Database representation of College List.....	20
Fig 4.16 : App.js Code.....	21
Fig 4.17 : App.js Code.....	21
Fig 4.18 : style.csst.....	22
Fig 4.19 : EJS & HTML files.....	22
Fig 4.20 : GitHub Repository To-Do List Web App.....	23
Fig 4.21 : Language percentages used to build To-Do List Web App.....	23

LIST OF TABLES

Table No. 1 : Web Application vs Website.....	8
---	---

ACKNOWLEDGEMENT

The project work in this report is an outcome of continuous work over a period and drew intellectual support from various sources. I would like to articulate our profound gratitude and indebtedness to those persons who helped me in completion of the project. I take this opportunity to express my sincere thanks and deep gratitude to all those people who extended their wholehearted co-operation and have helped me in completing this project successfully.

I am thankful to Training Associates for teaching and assisting me in making the project successful. Also I am thankful to Mr. Vatsya Tiwari, Mr. Amit Verma for guiding me throughout the completion of the report and Mrs. Durga Puja (Head of the Department, Computer Science) for her support. I would also like to thank my parents & other fellow mates for guiding and encouraging me throughout the duration of the project.

Priyank Saxena

ABOUT UDEMY



Udemy, Inc. is a global destination for teaching and learning online. It was founded in May 2010 by Eren Bali, Gagan Biyani, and Oktay Caglar. As of July 2022, the platform has more than 54 million students, 204,000 courses, and 71,000 instructors teaching courses in over 75 languages. There have been over 741 million course enrollments. Students take courses primarily to improve job-related skills. Some courses generate credit toward technical certification. Udemy has made a special effort to attract corporate trainers seeking to create coursework for employees of their company.

Udemy has not yet generated a profit as is common among high-growth startups who invest heavily in their own growth. Udemy reported net losses of \$69.7 million for 2019 and \$77.6 million in net losses for 2020. By June 30, 2021, Udemy had an accumulated deficit of \$407.9 million. In 2020, Udemy spent \$192.6 million on marketing and advertising.

Udemy is a platform that allows instructors to build online courses on their preferred topics. Using Udemy's course development tools, instructors can upload videos, source code for developers, PowerPoint presentations, PDFs, audio, ZIP files and any other content that learners might find helpful. Instructors can also engage and interact with users via online discussion boards.

The headquarters of Udemy is located in San Francisco, US, with hubs in Denver, US; Dublin, Ireland; Ankara, Turkey; São Paulo, Brazil; and Gurugram, India.

1.INTRODUCTION

1.1. Web Application

A web-application is an application program that is usually stored on a remote server, and users can access it through the use of Software known as web-browser. It is a type of computer program that usually runs with the help of a web browser and also uses many web technologies to perform various tasks on the internet. A web application can be developed for several uses, which can be used by anyone like it can be used as an individual or as a whole organization for several reasons.

1.2. Website

A website is a collection of many web pages, and web pages are digital files that are written using HTML(HyperText Markup Language). To make your website available to every person in the world, it must be stored or hosted on a computer connected to the Internet round the clock. Such computers are known as a Web Server.

All publicly accessible websites collectively constitute the World Wide Web. There are also private websites that can only be accessed on a private network, such as a company's internal website for its employees. Websites are typically dedicated to a particular topic or purpose, such as news, education, commerce, entertainment, or social networking. Hyperlinking between web pages guides the navigation of the site, which often starts with a home page.

When we type a certain URL in a browser search bar, the browser requests the page from the Web server and the Web server returns the required web page and its content to the browser. Now, it differs from how the server returns the information required in the case of static and dynamic websites.

Types of Website:

- Static Website
- Dynamic Website

1.2.1. Static Website

In Static Websites, Web pages are returned by the server which are prebuilt source code files built using simple languages such as HTML, CSS, or JavaScript. There is no processing of content on the server (according to the user) in Static Websites. Web pages are returned by the server with no change therefore, static Websites are fast. There is no interaction with databases. Also, they are less costly as the host does not need to support server-side processing with different languages.

1.2.2. Dynamic Website

In Dynamic Websites, Web pages are returned by the server which is processed during runtime means they are not pre built web pages, but they are built during runtime according to the user's demand with the help of server-side scripting languages such as PHP, Node.js, ASP.NET and many more supported by the server. So, they are slower than static websites but updates and interaction with databases are possible. Dynamic Websites are used over Static Websites as updates can be done very easily as compared to static websites (Where altering in every page is required) but in Dynamic Websites, it is possible to do a common change once, and it will reflect in all the web pages.

1.3. Why do we need a Website?

An effective method to showcase your products and services. Developing a site helps you to create your social proof. Helps you in branding your business. Helps you to achieve your business goals. Allows you to increase your customer support.

1.4. Why do we need a Web Application?

Compared to desktop applications, web applications are easier to maintain as they use the same code in the entire application. There are no compatibility issues. Web applications can be used on any platform: Windows, Linux, Mac... as they all support modern browsers. Mobile App store approval not required in web applications. Released any time and in any form. No need to remind users to update their applications. You can access these web applications 24 hours of the day and 365 days a year from any PC. You can either make use of the computer or your mobile device to access the required data. Web applications are a cost-effective option for any organization. Seat Licenses for Desktop software are expensive where SasS, are generally, pay as you go. Web-Based Apps are Internet-enabled apps that are accessed through the mobile's web browser. Therefore, you don't need to download or install them.

1.5. Web Application vs. Website

Parameter	Web Application	Website
Created for	A web application is designed for interaction with the end user	A website mostly consists of static content. It is publicly accessible to all the visitors.
User interaction	In a web application, the user not only reads the page content but also manipulates the restricted data.	A website provides visual & text content which users can view and read, but not affect its functioning.

Authentication	Web applications need authentication, as they offer a much broader scope of options than websites.	Authentication is not obligatory for informational websites. The user may ask to register to get a regular update or to access additional options. This feature is not available for the unregistered website visitors.
Task and Complexity	Web application functions are quite higher and complex compared to a website.	The website displays the collected data and information on a specific page.
Type of software	The web application development is part of the website. It is itself not a complete website.	The website is a complete product, which you access with the help of your browser.
Compilation	The site must be precompiled before deployment	The site doesn't need to be pre-compiled
Deployment	All changes require the entire project to be re-compiled and deployed.	Small changes never require a full re-compilation and deployment. You just need to update the HTML code.

Table No. 1 : Web Application vs Website

1.6. Summary

A website is a group of globally accessible, interlinked web pages which have a single domain name. App store approval is not required in web applications. Quality and relevant Web Content are the most important characteristics of a good website. Cloud-hosted and highly scalable are the most vital characteristics of a good Web App.

2.PROJECT

2.1. Introduction

To-Do List Web Application is an application specially built to keep track of errands or tasks that need to be done. This application will be like a task keeper where the user would be able to enter the tasks that they need to do. Once they are done with their tasks they can also remove them from the list.

2.2. Functionalities

To-Do List Web Application functionalities can be divided into two parts :

2.2.1. Features functionalities of the application

- We can add the tasks that are to be done in a descriptive way.
- We will be able to add as many tasks as you have.
- Once the task is completed, you will be able to remove it automatically by clicking on the “Task Completed” button.
- We can make a separate module for a particular task to do list.

2.2.2. User interface components that will be shown in our application

- There will be an add task button on the right top of the application
- There will be a check box for the task completion, so it will be able to be deleted automatically.
- The tasks that will be added by the users are stored in the list view.
- There will be a current date label at the top-centered position.
- Copyright and creator name at the bottom.

3. TOOLS & TECHNOLOGY USED

The following tools were used to build the project :

3.1. Integrated Development Environment (IDE) : It is a software application which helps programmers develop software code efficiently. It has increased developer productivity by combining capabilities such as software editing, building, testing, and packaging in an easy-to-use application. We have used *ATOM* as an IDE.

- **ATOM :** Atom is a free and open-source text and source code editor for macOS, Linux, and Microsoft Windows with support for plug-ins written in JavaScript, and embedded Git Control. Developed by GitHub, Atom is a desktop application built using web technologies. Most of the extending packages have free software licenses and are community-built and maintained. It is based on the Electron framework, which was developed for that purpose, and hence was formerly called Atom Shell.

3.2. Command-Line Interface : A command-line interface (CLI) is a text-based user interface (UI) used to run programs, manage computer files and interact with the computer. Command-line interfaces are also called command-line user interfaces, console user interfaces and character user interfaces.

- **Hyper :** Hyper is an Electron-based terminal. Built on HTML/CSS/JS. Fully extensible experience for command-line interface users, built on open web standards. In the beginning, focus will be primarily around speed, stability and the development of the correct API for extension authors.

3.3. Front End : The part of a website that the user interacts with directly is termed the front end. It is also referred to as the ‘client side’ of the application. Responsiveness and performance are two main objectives of the Front End.

- **HTML :** HTML stands for Hypertext Markup Language. It is used to design the front-end portion of web pages using a markup language. HTML is the combination of Hypertext and Markup language. Hypertext defines the link between web pages. The markup language is used to define the text documentation within the tag which defines the structure of web pages.
- **CSS :** Cascading Style Sheets fondly referred to as CSS is a simply designed language intended to simplify the process of making web pages presentable. CSS allows you to apply styles to web pages. More importantly, CSS enables you to do this independent of the HTML that makes up each web page.
- **JavaScript :** JavaScript is a famous scripting language used to create magic on sites to make the site interactive for the user. It is used to enhance the functionality of a website to run cool games and web-based software. Applicable in both front-end and back-end, Javascript is a key to becoming a good developer.
- **Bootstrap :** Bootstrap is a free, open source front-end development framework for the creation of websites and web apps. Designed to enable responsive development of mobile-first websites, Bootstrap provides a collection of syntax for template designs.
- **jQuery :** jQuery is an open-source JavaScript library that simplifies the interactions between an HTML/CSS document, or more precisely the Document Object Model (DOM), and JavaScript. Elaborating the terms, jQuery simplifies HTML document traversing and manipulation, browser event handling, DOM animations, Ajax interactions, and cross-browser JavaScript development.

3.4. Back End : The backend is the server-side of the website. It stores and arranges data, and also makes sure everything on the client-side of the website works fine. It is the part of the website that you cannot see and interact with. It is the portion of software that does not come in direct contact with the users.

- **Node.js :** Node.js is an open-source, cross-platform, back-end JavaScript runtime environment that runs on a JS Engine and executes JS code outside a web browser, which was designed to build scalable network applications. Node.js lets developers use JavaScript to write command line tools and for server-side scripting—running scripts server-side to produce dynamic web page content before the page is sent to the user's web browser.
- **Express.js :** Express is a Nodejs framework used for backend/server-side development. It is used to build single-page, multi-page, and hybrid web applications. With its help, you can handle multiple different HTTP requests.
- **Mongoose :** Mongoose is an Object Data Modeling (ODM) library for MongoDB and Node.js. It is relationships between data, provides schema validation, and is used to translate between objects in code and the representation of objects.

3.5. Database : A database is an organized collection of structured information, or data, typically stored electronically in a computer system. A database is usually controlled by a database management system (DBMS).

- **MongoDB :** MongoDB is a non-relational document database that provides support for JSON-like storage. The MongoDB database has a flexible data model that enables you to store unstructured data, and it provides full indexing support, and replication with rich and intuitive APIs.

4. SNAPSHOTS

4.1. Home Screen

This would be a Home Screen when this web app starts running. The first look of the web app. It would have a background and also highlighted some instructions to use the application.

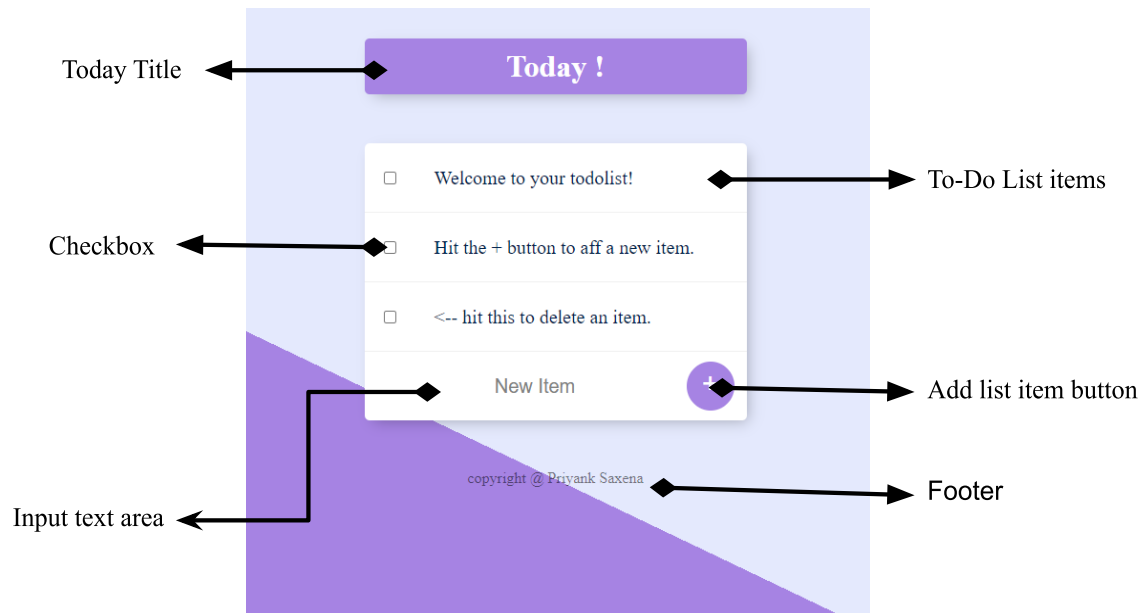


Fig 4.1 : Home Screen

- **Today Title:** It displays today's date title. The day for making the list to be done.
- **Checkbox:** It's for when the work is done, the user can click the checkbox after work completion.
- **Input text area:** Input area to type the title or related word to signify the work as a reminder.
- **To-Do List items:** Displaying a list of items of To-Do List.
- **Add list item button:** A button to add items typed in the input Text area into a To-Do list.
- **Footer:** A small Caption for the Footer field.

4.2. Adding item into the list

We have set up an array where we'll place the to-do list items. Each To-Do item will be an object with three properties: *text*, a string which holds whatever the user types into the text input, *checked*, a boolean which helps us know if a task has been marked completed or not, and *id*, a unique identifier for the item.

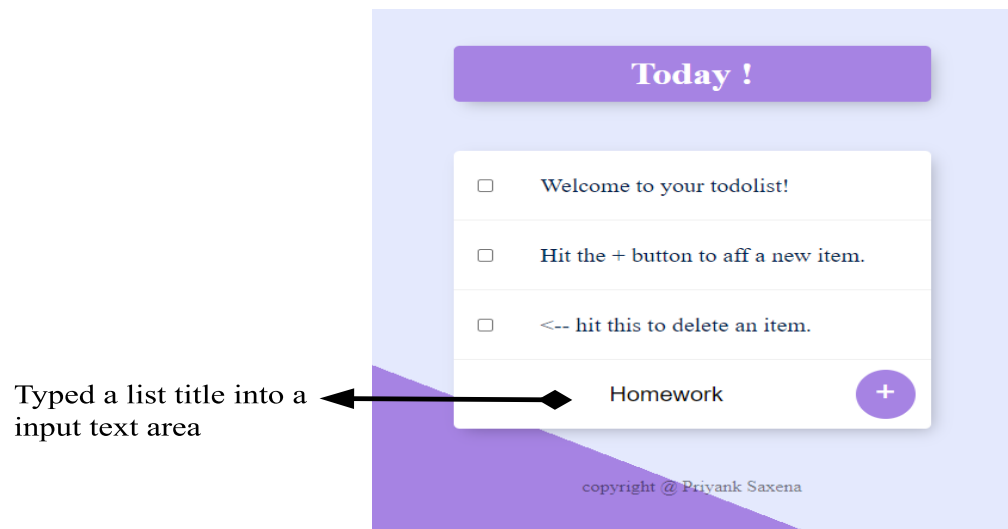


Fig 4.2 : Adding item into the List

Add a few todo items and view the To-Do Items array in your browser console. Each to-do item is represented by an object in the array.

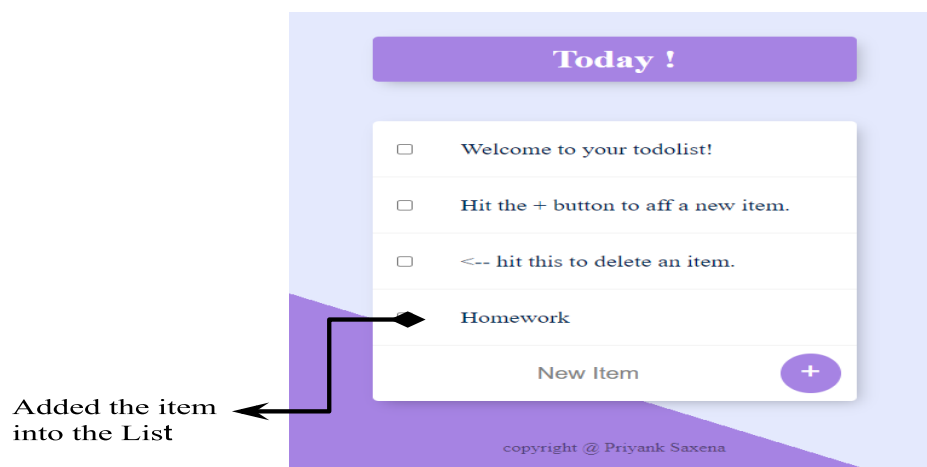


Fig 4.3 : Added item into the List

4.3. Removing/ Deleting item from a list

There is the ability to mark a task as completed by clicking the event on the checkbox, it listens and toggle the checked property on the corresponding To-do List.

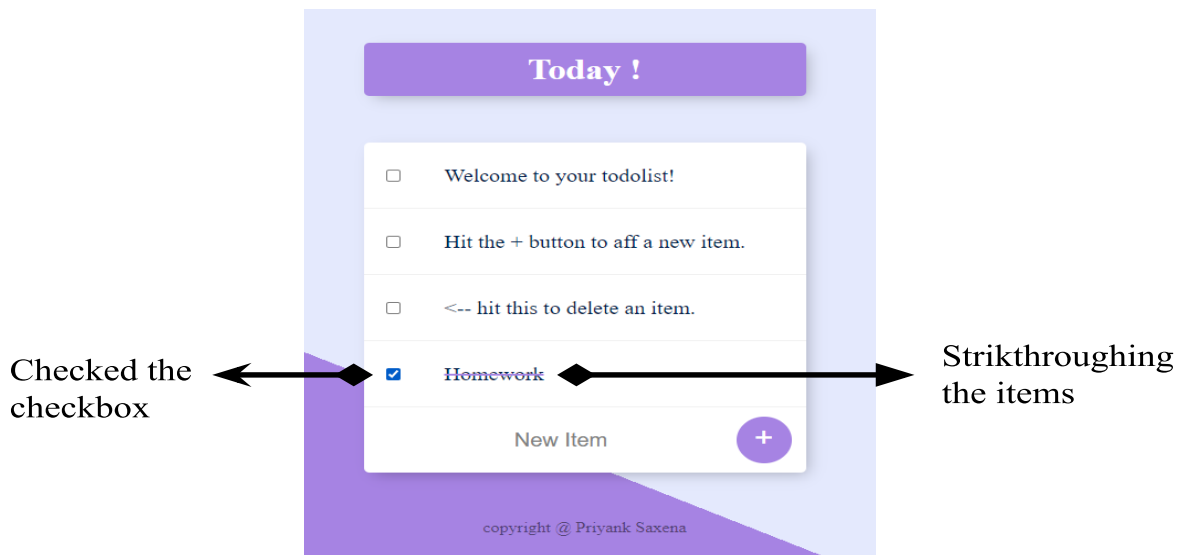


Fig 4.4 : Checked Checkbox

After the checked property is toggled item of the list, the text of the item converts into Strikethrough Text automatically.

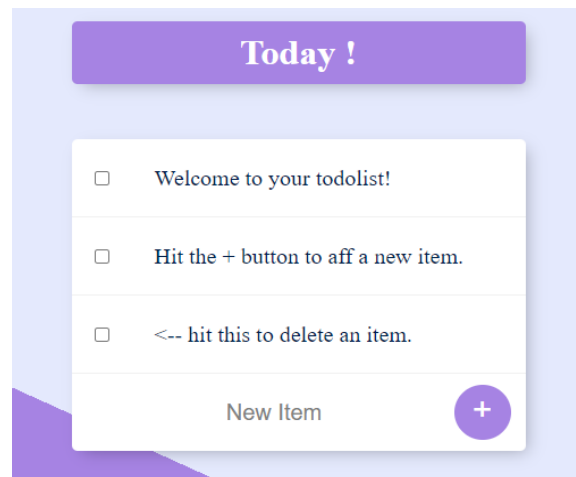


Fig 4.5 : Deleted Item

After Strikethrough of the text(list item) the item of the list automatically deletes the task.

4.4. Creating new module for a list

An additional feature to create a separate list for an individual purpose. Different modules have their own To-Do List and also display in a separate window .

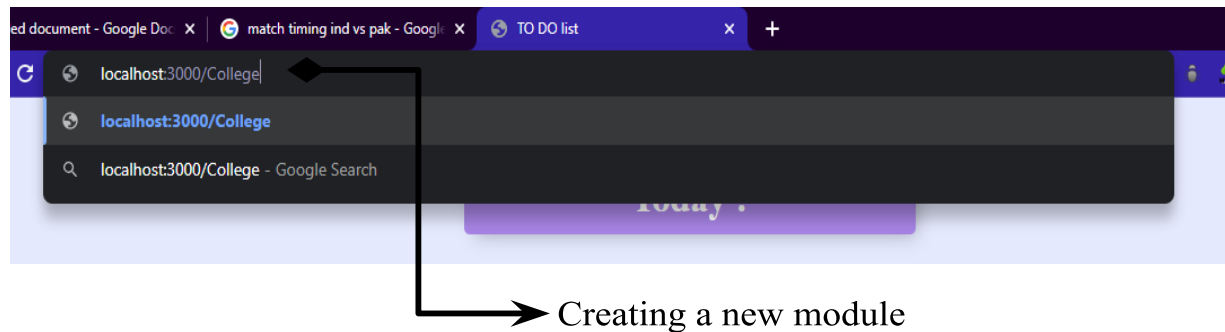


Fig 4.6 : Creating new module

In the url section, to create a new module, we separate by typing “ / ” and can set any name we want to create. For example, we want to create a list for College activities and tasks, so we simply type “ /College ” and press Enter.

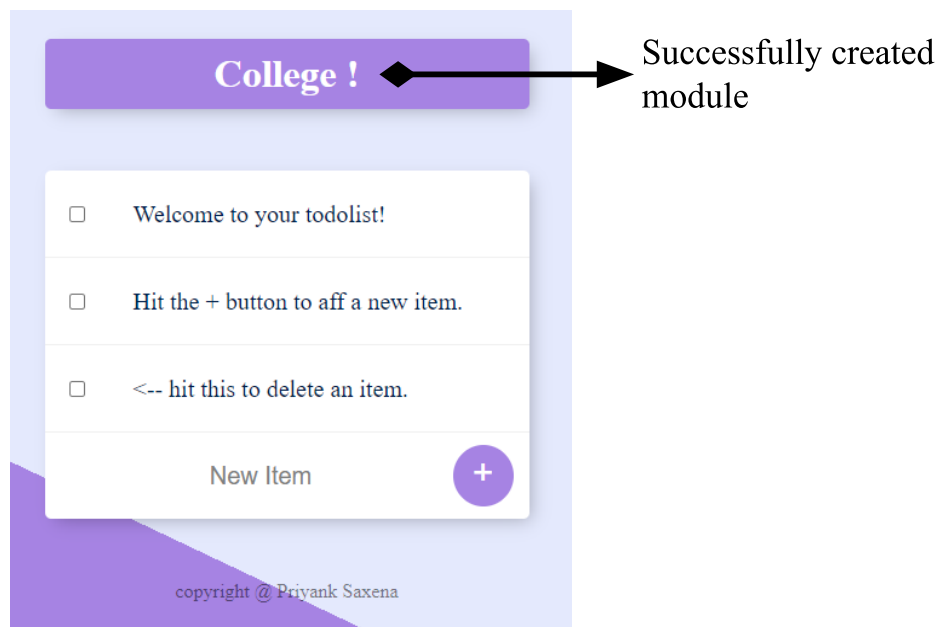
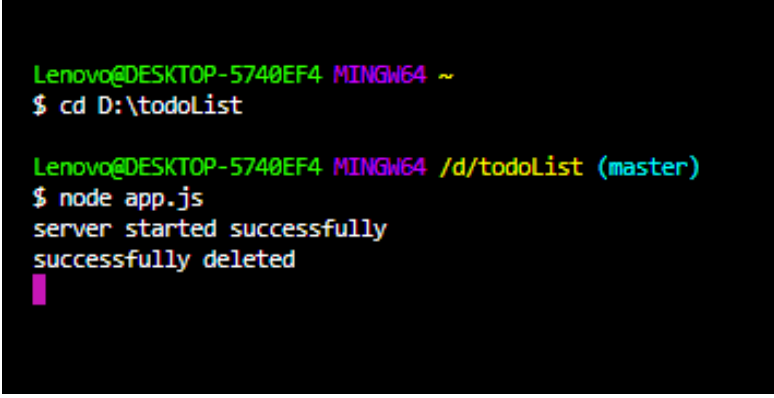


Fig 4.7 : Creating new module

Now, We are ready to make a separate to-do list with the same process mentioned above.

4.5. Server starting process

To run the To-do list web application, we have to start the server part of this application. The app.js is the server file which connects the frontend and backend. This file acts as a communication channel between the frontend and database.

A screenshot of a Windows command prompt window. The title bar reads 'Lenovo@DESKTOP-5740EF4 MINGW64 ~'. The prompt is '\$ cd D:\todoList'. The next line shows the prompt 'Lenovo@DESKTOP-5740EF4 MINGW64 /d/todoList (master)' followed by the command '\$ node app.js'. The output of the command is 'server started successfully' and 'successfully deleted' on two separate lines. A pink cursor is visible at the end of the second output line.

```
Lenovo@DESKTOP-5740EF4 MINGW64 ~  
$ cd D:\todoList  
  
Lenovo@DESKTOP-5740EF4 MINGW64 /d/todoList (master)  
$ node app.js  
server started successfully  
successfully deleted
```

Fig 4.8 : Running Server file

In Starting, we have to type a command in the command line interface “*node app.js*”. The server file name is *app.js* and the technology used to build this file is Node.js. So, to run the node.js file we add a command “*node*”. So, the command node app.js will start running with the specified port number which is **3000**.

An app.js file server is responsible for the storage and management of data files so that other computers on the same network can access the files. It enables users to share information over a network without having to physically transfer files.

4.6. Starting and working process of Database

When we start up a database, we create an instance of that database and you determine the state of the database. Normally, we start up an instance by mounting and opening the database. Doing so makes the database available for any valid user to connect to and perform typical data access operations.

```

Lenovo@DESKTOP-5740EF4 MINGW64 ~
$ mongod
2022-10-22T21:06:41.980+0530 I CONTROL [main] Automatically disabling TLS 1.0, to force-enable TLS 1.0 specify --sslDisabledProtocols 'none'
2022-10-22T21:06:42.000+0530 I CONTROL [initandlisten] MongoDB starting : pid=4464 port=27017 dbpath=C:\data\db\ 64-bit host=DESKTOP-5740EF4
2022-10-22T21:06:42.000+0530 I CONTROL [initandlisten] targetMins: Windows 7/Windows Server 2008 R2
2022-10-22T21:06:42.000+0530 I CONTROL [initandlisten] db version v4.0.28
2022-10-22T21:06:42.000+0530 I CONTROL [initandlisten] git version: af1a9dc12adcfa83cc19571cb3faba26eeddac92
2022-10-22T21:06:42.000+0530 I CONTROL [initandlisten] allocator: tcmalloc
2022-10-22T21:06:42.000+0530 I CONTROL [initandlisten] modules: none
2022-10-22T21:06:42.000+0530 I CONTROL [initandlisten] build environment:
2022-10-22T21:06:42.000+0530 I CONTROL [initandlisten] distro: 2008plus-ssl
2022-10-22T21:06:42.010+0530 I CONTROL [initandlisten] distarch: x86_64
2022-10-22T21:06:42.010+0530 I CONTROL [initandlisten] target_arch: x86_64
2022-10-22T21:06:42.010+0530 I CONTROL [initandlisten] options: {}
2022-10-22T21:06:42.022+0530 I STORAGE [initandlisten] Detected data files in C:\data\db\ created by the 'wiredTiger' storage engine, so setting the active storage engine to 'wiredTiger'.
2022-10-22T21:06:42.022+0530 I STORAGE [initandlisten] wiredtiger_open config: create,cache_size=147M,cache_overflow=(file_max=0),session_max=20000,eviction=(threads_min=4,threads_max=4),config_base=false,statistics=(fast,log(enabled=true,archive=true,path=C:\data\log),statistics_log=(wait=0,verbose=(recovery_progress))),journal,compressor=snappy,file_manager=(close_idle_time=10000),statistics_log=(wait=0,verbose=(recovery_progress)),
2022-10-22T21:06:42.022+0530 I STORAGE [initandlisten] WiredTiger message [1666453003:2811564][4464:140708525921616], txn-recover: Main recovery loop: starting at 12/28000 to 13/256
2022-10-22T21:06:43.623+0530 I STORAGE [initandlisten] WiredTiger message [1666453003:333501][4464:140708525921616], txn-recover: Recovering log 12 through 13
2022-10-22T21:06:43.623+0530 I STORAGE [initandlisten] WiredTiger message [1666453003:623222][4464:140708525921616], txn-recover: Recovering log 13 through 13
2022-10-22T21:06:43.623+0530 I STORAGE [initandlisten] WiredTiger message [1666453003:890198][4464:140708525921616], txn-recover: Set global recovery timestamp: 0
2022-10-22T21:06:43.914+0530 I RECOVERY [initandlisten] WiredTiger recoveryTimestamp. Ts: Timestamp(0, 0)
2022-10-22T21:06:43.927+0530 I STORAGE [initandlisten] Starting to check the table logging settings for existing WiredTiger tables
2022-10-22T21:06:43.943+0530 I CONTROL [initandlisten]
2022-10-22T21:06:43.943+0530 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the database.
2022-10-22T21:06:43.943+0530 I CONTROL [initandlisten] ** Read and write access to data and configuration is unrestricted.
2022-10-22T21:06:43.944+0530 I CONTROL [initandlisten]
2022-10-22T21:06:43.945+0530 I CONTROL [initandlisten] ** WARNING: This server is bound to localhost.
2022-10-22T21:06:43.945+0530 I CONTROL [initandlisten] ** Remote systems will be unable to connect to this server.
2022-10-22T21:06:43.945+0530 I CONTROL [initandlisten] ** Start the server with --bind_ip <address> to specify which IP
2022-10-22T21:06:43.945+0530 I CONTROL [initandlisten] ** addresses it should serve responses from, or with --bind_ip_all to
2022-10-22T21:06:43.945+0530 I CONTROL [initandlisten] ** bind to all interfaces. If this behavior is desired, start the
2022-10-22T21:06:43.945+0530 I CONTROL [initandlisten] ** server with --bind_ip 127.0.0.1 to disable this warning.
2022-10-22T21:06:43.946+0530 I CONTROL [initandlisten]
2022-10-22T21:06:44.025+0530 I STORAGE [initandlisten] Finished adjusting the table logging settings for existing WiredTiger tables
2022-10-22T21:06:45.463+0530 I FTDC [initandlisten] Initializing full-time diagnostic data capture with directory 'C:\data\db\diagnostic.data'
2022-10-22T21:06:45.474+0530 I NETWORK [initandlisten] waiting for connections on port 27017

```

Fig 4.9 : Starting process of MongoDB 1

MongoDB runs as a standard program. You can start MongoDB from a command line by issuing the mongod command and specifying options. By default, MongoDB listens for connections from clients on port 27017, and stores data in the /data/db directory.

```

Lenovo@DESKTOP-5740EF4 MINGW64 ~
$ mongo
MongoDB shell version v4.0.28
connecting to: mongodb://127.0.0.1:27017/?sslapiServiceName=mongodb
Implicit session: session { "id" : UUID("b3fa2eee-558c-4afd-a4e6-96082aebab8a") }
MongoDB server version: 4.0.28
Server has startup warnings:
2022-10-22T21:06:45.823+0530 I CONTROL [initandlisten]
2022-10-22T21:06:45.823+0530 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the database.
2022-10-22T21:06:45.823+0530 I CONTROL [initandlisten] ** Read and write access to data and configuration is unrestricted.
2022-10-22T21:06:45.823+0530 I CONTROL [initandlisten]

Enable MongoDB's free cloud-based monitoring service, which will then receive and display
metrics about your deployment (disk utilization, CPU, operation statistics, etc).

The monitoring data will be available on a MongoDB website with a unique URL accessible to you
and anyone you share the URL with. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---

> dbs
2022-10-22T21:09:56.091+0530 E QUERY [js] ReferenceError: dbs is not defined :
@(<shell>):1:1
> show dbs
admin          0.000GB
config         0.000GB
fruitsDB       0.000GB
local          0.000GB
shopDB         0.000GB
shoppyDB       0.000GB
todolistdb     0.000GB
>

```

Fig 4.10 : Starting process of MongoDB 2

To open up the MongoDB shell, run the mongo command from your server prompt. By default, the mongo command opens a shell connected to a locally-installed MongoDB instance running on port 27017 . Try running the mongo command with no additional parameters: mongo.

```

> show dbs
admin      0.000GB
config     0.000GB
fruitsDB   0.000GB
local      0.000GB
shopDB     0.000GB
shopsyDB   0.000GB
todolistdb 0.000GB
> use todolistdb
switched to db todolistdb
> show collections
items
lists
>

```

Fig 4.11 : todolistdb Database

In MongoDB, you can use the *show dbs* command to list all databases on a MongoDB server. This will show you the database name, as well as the size of the database in gigabytes. We can select any database using the use statement and work on it. So our database name is *todolistdb*. It has two collections *items* and *lists*.

```

> db.items.find().pretty()
{
  "_id" : ObjectId("634441a5a8cb8f5b6bf0aee4"),
  "name" : "Welcome to your todolist!",
  "__v" : 0
}
{
  "_id" : ObjectId("634441a5a8cb8f5b6bf0aee5"),
  "name" : "Hit the + button to aff a new item.",
  "__v" : 0
}
{
  "_id" : ObjectId("634441a5a8cb8f5b6bf0aee6"),
  "name" : "<-- hit this to delete an item.",
  "__v" : 0
}
>

```

Fig 4.12 : Datas of todolistdb Database

All the documents present in the collection *items* of *todolistdb* by using command *db.items.find()*.

```

> db.items.find().pretty()
{
  "_id" : ObjectId("634441a5a8cb8f5b6bf0aee4"),
  "name" : "Welcome to your todoist!",
  "__v" : 0
}
{
  "_id" : ObjectId("634441a5a8cb8f5b6bf0aee5"),
  "name" : "Hit the + button to aff a new item.",
  "__v" : 0
}
{
  "_id" : ObjectId("634441a5a8cb8f5b6bf0aee6"),
  "name" : "<-- hit this to delete an item.",
  "__v" : 0
}
{
  "_id" : ObjectId("635411a8a6e8d8d71c4b0237"),
  "name" : "Homework",
  "__v" : 0
}
>

```

Fig 4.13 : Database representation when a item is added

When a user adds an item into a list typed “Homework”, as we discussed in the section **4.2 Adding item into the list**. In the database of todolistdb this item is stored in the Items collections.

4.7. To-do-list for College

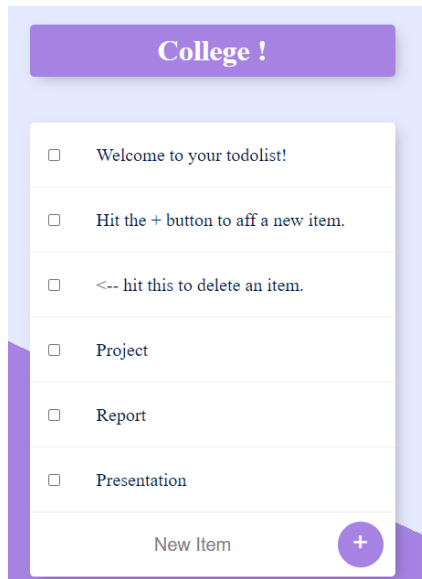


Fig 4.14 : College List

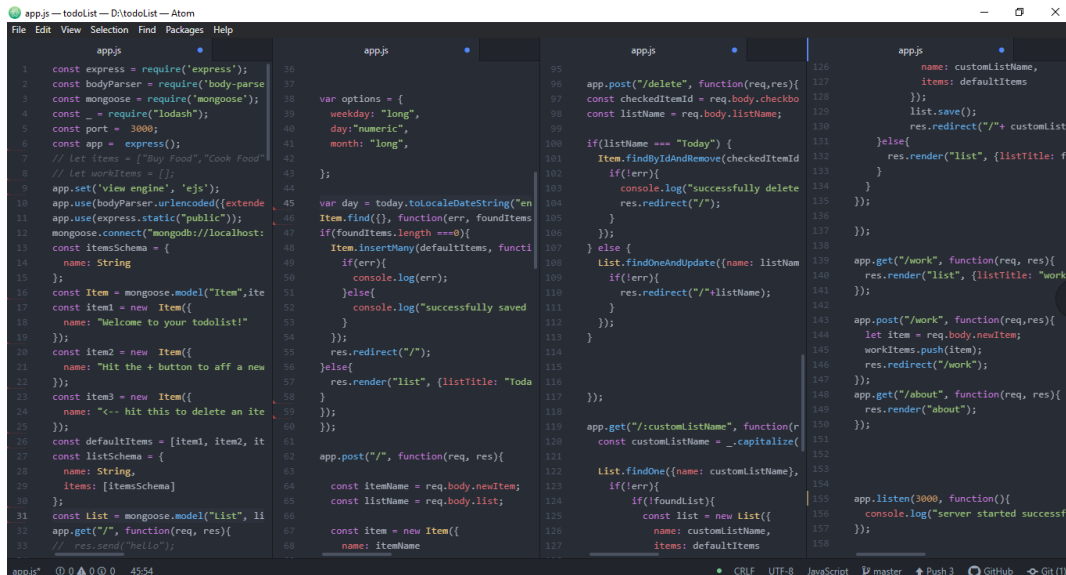
```

{
  "_id" : ObjectId("63540d8fa6e8d8d71c4b022f"),
  "name" : "College",
  "items" : [
    {
      "name" : "Welcome to your todoist!",
      "_id" : ObjectId("63540cc3a6e8d8d71c4b0220")
    },
    {
      "name" : "Hit the + button to aff a new item.",
      "_id" : ObjectId("63540cc3a6e8d8d71c4b0221")
    },
    {
      "name" : "<-- hit this to delete an item.",
      "_id" : ObjectId("63540cc3a6e8d8d71c4b0222")
    },
    {
      "name" : "Project",
      "_id" : ObjectId("63541261a6e8d8d71c4b023d")
    },
    {
      "name" : "Report",
      "_id" : ObjectId("63541270a6e8d8d71c4b0243")
    },
    {
      "name" : "Presentation ",
      "_id" : ObjectId("6354127ba6e8d8d71c4b0249")
    }
  ],
  "__v" : 3
}
>

```

Fig 4.15 : Database representation of College List

4.8. App.js Code (node.js)



```
1 const express = require('express');
2 const bodyParser = require('body-parser');
3 const mongoose = require('mongoose');
4 const _ = require('lodash');
5 const port = 3000;
6 const app = express();
7 // let items = ["Buy Food","Cook Food"];
8 // let workItems = [];
9 app.set('view engine', 'ejs');
10 app.use(bodyParser.urlencoded({extended: true}));
11 app.use(express.static('public'));
12 mongoose.connect("mongodb://localhost:27017/todolist", {
13   name: String
14 });
15 const Item = mongoose.model("Item", itemSchema);
16 const item1 = new Item({
17   name: "Welcome to your todolist!"
18 });
19 const item2 = new Item({
20   name: "Hit the + button to add a new item"
21 });
22 const item3 = new Item({
23   name: "Hit the - button to delete an item"
24 });
25 const defaultItems = [item1, item2, item3];
26 const listSchema = {
27   name: String,
28   items: [itemSchema]
29 };
30 const List = mongoose.model("List", listSchema);
31 app.get("/", function(req, res){
32   // res.send("hello");
33   const options = {
34     weekday: "long",
35     day: "numeric",
36     month: "long"
37   };
38   var day = today.toISOString().replace(/T/, " ").replace(/Z/, " UTC").replace(/:/gm, ":");
39   Item.find({}, function(err, foundItems){
40     if(foundItems.length === 0){
41       Item.insertMany(defaultItems, function(err){
42         if(err){
43           console.log(err);
44         } else {
45           console.log("successfully saved");
46         }
47       });
48       res.redirect("/");
49     } else {
50       res.render("list", {listTitle: "Today", foundItems: foundItems});
51     }
52   });
53   app.post("/", function(req, res){
54     const itemName = req.body.newItem;
55     const listName = req.body.list;
56     const item = new Item({
57       name: itemName
58     });
59     app.post("/delete", function(req, res){
60       const checkedItemId = req.body.checkedItemId;
61       const listName = req.body.listName;
62       if(listName === "Today"){
63         Item.findByIdAndRemove(checkedItemId, function(err){
64           if(!err){
65             console.log("successfully delete");
66             res.redirect("/");
67           }
68         });
69       } else {
70         List.findOneAndUpdate({name: listName}, {$pull: {items: checkedItemId}}, function(err){
71           if(!err){
72             res.redirect("/"+listName);
73           }
74         });
75       }
76     });
77     app.get("/:customListName", function(req, res){
78       const customListName = _.capitalize(req.params.customListName);
79       List.findOne({name: customListName}, function(err, foundList){
80         if(!err){
81           if(!foundList){
82             const list = new List({
83               name: customListName,
84               items: defaultItems
85             });
86             list.save(function(err){
87               if(err){
88                 console.log("Error in saving custom list");
89               } else {
90                 List.findOne({name: customListName}, function(err, foundList){
91                   if(!err){
92                     app.get("/work", function(req, res){
93                       res.render("list", {listTitle: "Work Items", foundItems: foundList});
94                     });
95                     app.post("/work", function(req, res){
96                       let item = req.body.newItem;
97                       workItems.push(item);
98                       res.redirect("/work");
99                     });
100                     app.get("/about", function(req, res){
101                       res.render("about");
102                     });
103                     app.listen(3000, function(){
104                       console.log("server started successfully");
105                     });
106                   }
107                 });
108               }
109             });
110           }
111         }
112       }
113     });
114   });
115 }
```

Fig 4.16 : App.js Code

Node.js is used to build the App.js javascript server file with 157 lines of code.

4.9. package.json



```
1 {
2   "name": "todolist",
3   "version": "1.0.0",
4   "description": "to do list ",
5   "main": "app.js",
6   "scripts": {
7     "test": "echo \"Error: no test specified\" && exit 1",
8     "start": "node app.js"
9   },
10  "author": "Priyank Saxena",
11  "license": "ISC",
12  "engines": {
13    "node": "16.16.0"
14  },
15  "dependencies": {
16    "body-parser": "^1.20.0",
17    "ejs": "^3.1.8",
18    "express": "^4.18.1",
19    "lodash": "^4.17.21",
20    "mongoose": "^6.5.2"
21  }
22 }
```

Fig 4.17 : App.js Code

This file contains all the information about the node.js modules used in this application.

4.10. style.css (CSS)

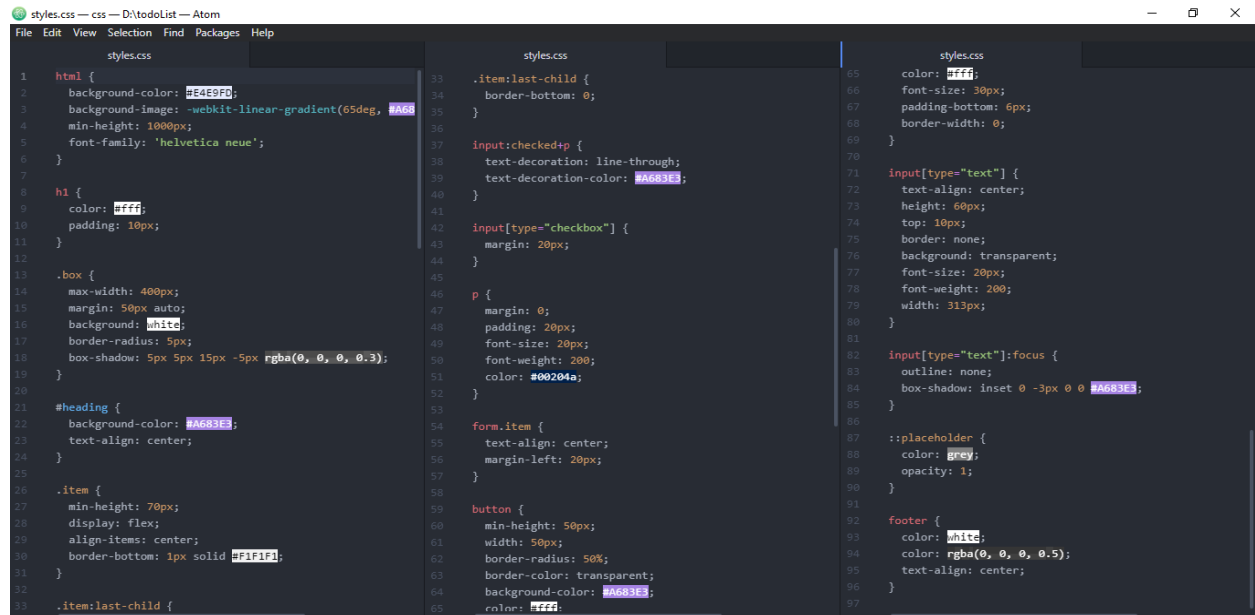


Fig 4.18 : style.css

Cascading Style Sheets (CSS) is used to format the layout of a webpage.

4.11. EJS & Html Files (Express JavaScript and Html)

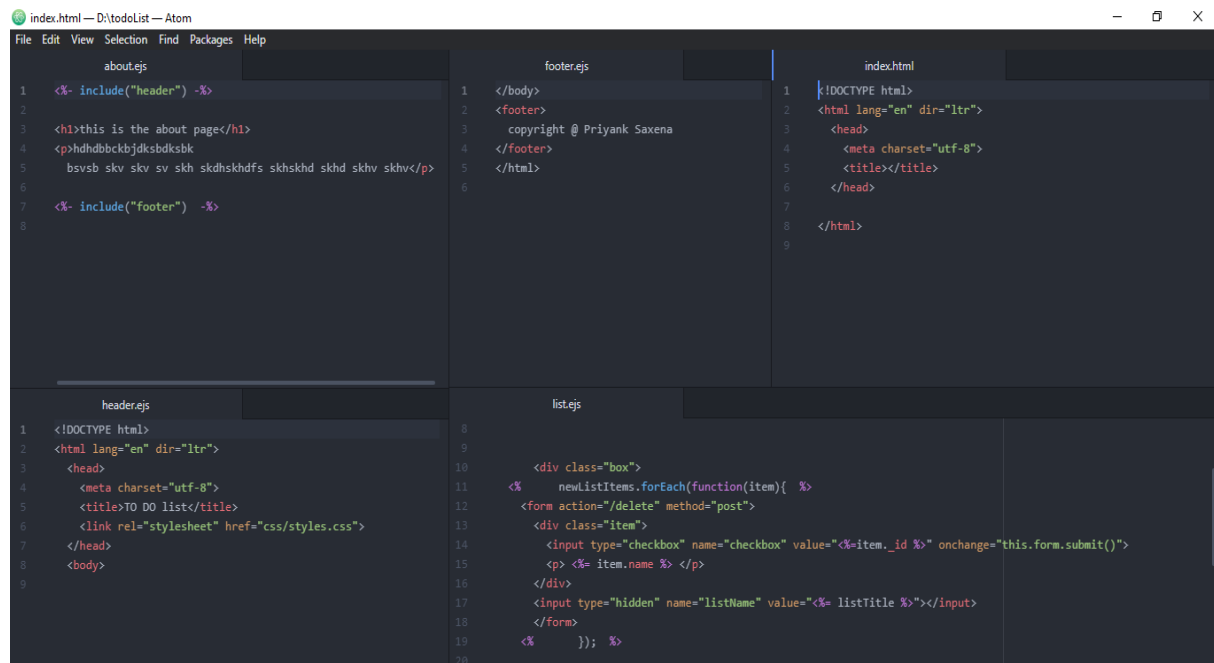


Fig 4.19 : EJS & HTML files

4.12. GitHub Repository

This repository contains all of our project's files and each file's revision history. GitHub allows software developers and engineers to create remote, public-facing repositories on the cloud for free. One of the biggest advantages of Git is its branching capabilities. Unlike centralized version control systems, Git branches are cheap and easy to merge. This facilitates the feature branch workflow popular with many Git users. Feature branches provide an isolated environment for every change to your codebase.

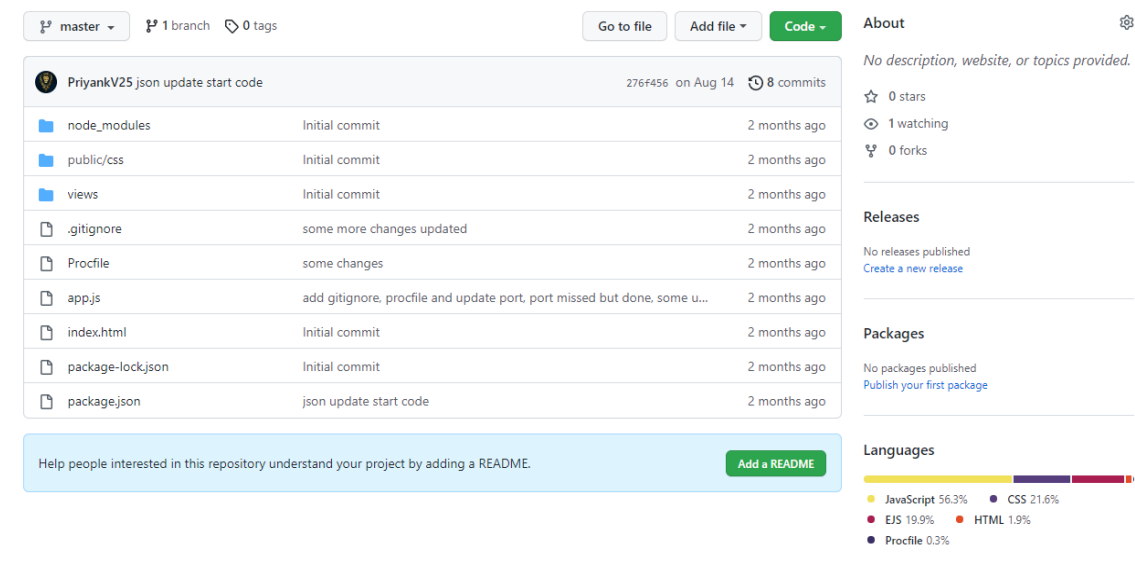


Fig 4.20 : GitHub Repository To-Do List Web App

4.13 Percentages of Languages used in the project

Languages

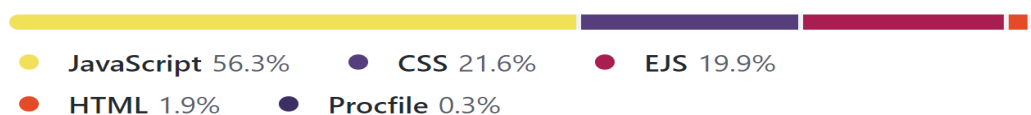


Fig 4.21 : Language percentages used to build To-Do List Web App

5.RESULT & DISCUSSION

TODO List are the lists that we generally use to maintain our day to day tasks or list of everything that we have to do, with the most important tasks at the top of the list, and the least important tasks at the bottom. It is helpful in planning our daily schedules. We can add more tasks any time and delete a task which is completed. The four major tasks that we can perform in a TODO list are:

- Add tasks
- Update tasks
- Read tasks
- Delete tasks

To-do lists can work for you, but if you are not using them effectively, they can actually leave you feeling more disillusioned and stressed than you did before. Think of a filing system: the concept is good, but if you merely file papers away with no structure or system, the filing system will have an adverse effect. The good news is, writing a to-do list doesn't have to be an exercise in futility. It's the same with to-do lists—you can put one together, but if you don't do it right, it is a fruitless exercise.

The first step for creating a to-do list that actually works is to build it in a to-do list app. Written to-do lists provide the wonderfully satisfying feeling of crossing something off, but they're also the worst way to keep track of your to-dos. The truth of the matter is, a written to-do list is disorganized, prone to mistakes, easy to lose, and ineffective.

Instead, get started with a to-do list app. Unlike a written to-do list, the benefits of a to-do list app include:

- **Sorting and prioritizing work.** If you want to change the order of your written to-do list, you have to rewrite the whole thing. But with a to-do list app, you can easily drag and drop items. Not only that—most to-do list apps offer a way to track priority with custom tags.
- **Impossible to lose.** Unlike a handwritten to-do list, you can't "lose" an online to-do list. You'll always have access to the list, so you can jot down to-dos wherever you are.
- **Add additional context to your to-dos.** Most to-do list apps offer a way for you to add additional information in the description. In a written to-do list, all you have are a couple of words to describe what you need to work on. But with a to-do list app each to-do has an expandable description, where you can add any relevant task details, working docs, or important information.
- **Create separate lists in the same place.** Before you choose a to-do list app, make sure you can create more than one "list" in the app. You might want to create a to-do list for personal work, another for your team's work, and a third for your professional development, for example. A to-do list app with multiple list options allows you to store all of these to-dos in one place.
- **Set due dates, reminders, and notifications.** Your to-dos don't mean much if they're not done in time. With a to-do list app, you can track when work is due, and set up reminders or notifications to make sure you get your to-dos done in time.
- **Collaboration.** When your individual to-do list is organized and your priorities are clear, you can better contribute to team projects and initiatives. In other words, the more organized you are, the easier it'll be to collaborate with your team.

6.CONCLUSION & FUTURE SCOPE

App *todo-list-app* performs very efficiently compared to competitors. There is a space for further developments with regards to keeping apps small and quick. It would be optimal to use dedicated CSS and continue using vanilla JavaScript.

To-do-app can be developed as a sole application as well as a very efficient module to be combined in a larger project. One of the key challenges is to choose appropriate storage solution, that will allow to maintain its biggest advantages:

- **Improves your memory:** A to do list acts as an external memory aid. It's only possible to hold a few pieces of information at one time. Keep a to-do list and you'll be able to keep track of everything, rather than just a few of the tasks you need to do. Your to do list will also reinforce the information, which makes it less likely you're going to forget something.
- **Increases productivity:** A to do list allows you to prioritize the tasks that are more important. This means you don't waste time on tasks that don't require your immediate attention. Your list will help you stay focused on the tasks that are the most important.
- **Helps with motivation:** To do lists are a great motivational tool because you can use them to clarify your goals. You can divide your long-term goal into smaller, more achievable short-term goals and as you tick each one off your list, your confidence will increase.

To make this list work effectively for you, it needs to become a daily tool that you use to manage your time and you review it regularly. There is no point in only having the list to record everything that you need to do, but you don't utilize it as part of your bigger time management plan. It seems such a simple solution by putting pen to paper and taking time out of your day to create a to do list, a plan for your day helps define your

challenges and goals. Preventing time from being wasted trying to identify what is the next most important task to tackle next and even more important makes sure you don't forget to do something important.

To-do lists offer a way to increase productivity, stopping you from forgetting things, helps prioritize tasks, manage tasks effectively, use time wisely and improve time management as well as workflow.

Succeeding in today's work environment is tough. There are too many projects and tasks to manage. With new things popping up and your personal commitments, things can get overwhelming. Add the huge amount of distractions and you're in for a difficult climb up the ladder of success.

Having a to-do list can make things much easier. Whether you're looking to achieve more of your goals or controlling your time better, a to-do list will help you. You can get a positive boost to your career by becoming the person who is always on top of things and feel good every day.

It lets you make large and overwhelming projects manageable. Also, you get more done by focusing on high-value activities. Once you have a list of things you need to-do, it's much easier to prioritize the tasks on it. This will ensure you're always working on the right things.

After our research of classical to-do lists, we would like you to know about such online tools that give the possibility to make up digital to-do lists with more functions and features. Such task management software examples are great helpers in more serious business work planning and can be used not only individually but also by all the working staff and teams.

To sum up, a compilation of to-do lists is a peculiar, very creative and at the same time a serious process – it is definitely worth spending time on it and making your personal work more logical and consistent. Also, we have learned that special task management software can help specialists with their to-do listing.

7.REFERENCES

- <https://www.ntaskmanager.com/blog/best-to-do-list-apps/>
- <https://hygger.io/blog/to-do-list-its-benefits-and-assets-for-business>
- <https://www.lifehack.org/articles/productivity/why-lists-dont-work-and-how-change-that.html>
- <https://todo-list-app.readthedocs.io/>
- <https://checkify.com/blog/what-is-a-todo-list/>
- <https://www.mongodb.com/docs/>
- <https://freshman.tech/todo-list/>
- https://www.w3schools.com/howto/howto_js_todolist.asp
- <https://www.geeksforgeeks.org/node-js-date-format-api/>
- https://www.google.com/search?q=udemy&rlz=1C1CHZN_enIN992IN992&oq=udemy&aqs=chrome.O.35i39j46i67i199i433i465j69i59joi67j69i60l4.2542j0j7&sourceid=chrome&ie=UTF-8
- <https://nodejs.org/en/about/>
- <https://ide.atom.io/>