

Learning Log

Submitted in partial fulfilment of the
requirements for the award of
Bachelor of Science degree in Computer Science

by

Rishikanth K C (39290086)
Varun Prasath K (39290121)



DEPARTMENT OF COMPUTER SCIENCE
SCHOOL OF COMPUTING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A" by NAAC

JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI - 600119

MARCH - 2022



SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE

BONAFIDE CERTIFICATE

This is to verify that this project report is the bonafide work of **Rishikanth.K.C (39390086)** and **Varun Prasath.K (39290121)** who carried out the project entitled as "Learning Log" under our supervision from Dec 2021 to March 2022.

Internal Guide

Dr. Mrs. Ulagamuthalvi

Head of the Department

Dr. L. Lakshmanan M.E, Ph.D

Submitted for Viva voce Examination held on_____

Internal Examiner

External Examiner

DECLARATION

I, Rishikanth K C (39290086) and Varun Prasath K (39290121) hereby declare that the Project Report entitled “**Learning Log**” done by me under the guidance of Dr. Mrs. Ulagamuthalvi is submitted in partial fulfilment of the requirements for the award of Bachelor of Science degree in Computer Science.

DATE: 01.03.2022

PLACE: Chennai

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to the Board of Management of SATHYABAMA for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T.Sasikala M.E., Ph.D, Dean**, School of Computing **Dr. L. Lakshmanan M.E., Ph.D. ,** and **Dr.S.Vigneshwari M.E., Ph.D.** Heads of the Department of Computer Science and Engineering for providing necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide Dr./Mr./Ms for his valuable guidance, suggestions and constant encouragement paved the way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department Of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

How about we take a look at today's social networks? One that is filled with advertising, notorious trackers, third-party content trackers or unencrypted sites that could leak data to intermediary eavesdroppers like ISP. That's the total sum of flaws that a major percentage of sites have in particular except for the major companies like Google, Apple, Facebook etc. Early web developers don't seem to grasp the concept of security and privacy well enough, they don't really know if someone likes to upload their contents publicly or privately with their true consent, Developers take everything, never pays attention to privacy of user as long the site brings user traffic into the server which means real profit for them, Unlike any of those things, A site should be built solely for one purpose, where users can just blog about anything or create logs as they come across their own learning sessions with privacy enabled by default, A place without trackers, Advertisers, insecure sites that users cannot rely on. Of course there are millions of sites that are online with the features I mentioned but none of them were built with privacy right from the moment it started. That's where the real problem is. Each beginner for web developing should be aware of what it means to build such features from the beginning of the project and not in the middle of gradual updates over the period of time. We hope to change this trend of idiocy by building a blogging site with the privacy features enabled right from the word 'Go!'. Expecting in goodwill that developers embrace the idea of building a site with all the essential features from beta tests. These kinds of blogging sites must have a reliable server in order to run perfectly. These servers provide the features of database management and application's live interaction with the user, even the site's connection gets secured when it's been offered a certificate from the web hosting company.

These criterias are not just to sites that've been managed by developers with some reputation, Even the new developers should not miss providing the features that pass the criteria. That being said, Learning Log is built to provide a secure way for the users to blog.

TABLE OF CONTENTS

Chapter No	TITLE	Page no
	ABSTRACT	v
	LIST OF FIGURES	viii
	LIST OF ABBREVIATIONS	viii
1	INTRODUCTION	
	1.1 Overview of the project	1
2	LITERATURE REVIEW	3
3	AIM AND SCOPE OF PRESENT INVESTIGATION	6
	3.1 Aim of the Project	6
	3.2 Scope and Objective	6
	3.3 System Requirements	
	3.3.1 Hardware Requirements	
	3.3.2 Software Requirements	
	3.4 Overview of the Project	7
	3.4.1 Software Used	7
	3.4.1.1 Django Framework	7
	3.4.1.2 Visual Studio Code	10
	3.4.1.3 Bootstrap	11
	3.4.1.4 Heroku	12
4	EXPERIMENTAL OR MATERIAL METHODS	14
	4.1 Design Methodology	
	4.1.1 Existing System	
	4.1.2 Proposed System	
	4.2 Module Description	15
	4.3 System Implementation	17
	4.4 Architecture Explanation	17
	4.4.1 User Registration	19

	4.4.2 Login	19
	4.5 Working Procedure	19
5	RESULTS AND PERFORMANCE ANALYSIS	21
6	CONCLUSION AND FUTURE ENHANCEMENTS	25
	6.1 Conclusion	
	6.2 Future Enhancements	
	REFERENCES	27
	APPENDIX	28
	A. Source Code	
	B. Publication	

LIST OF FIGURES

Figure no	Figure name	Page no
3.4.1.1	Django Framework	7
4.4	Django Architecture	18
5.1	Web app Homepage	21
5.2	Login Page	22
5.3	Entries Page	23
5.4	Certification Page	24

LIST OF ABBREVIATIONS

PIP - Preferred Installer Package
PEP - Python Enhancement Proposal
IDE - Integrated Development Environment
HTML - Hypertext Markup Language
RAM - Random Access Memory
GUI - Graphical User Interface
CMD - Command Line
OOP - Object Oriented Programming
VENV - Virtual Environment
IDE - Integrated Web Development

Chapter 1

Introduction

1.1 Overview of the Project

The web is an incredible platform. Its mix of ubiquity across devices and operating systems, its user-centred security model, and the fact that neither its specification nor its implementation is controlled by a single company makes the web a unique platform to develop software on. Combined with its inherent linkability, it's possible to search it and share what you've found with anyone, anywhere. Whenever you go to a website, it's up-to-date, and your experience with that site can be as ephemeral or as permanent as you'd like. Web applications can reach anyone, anywhere, on any device with a single codebase. Platform-specific applications are known for being incredibly rich and reliable. They're ever-present, on home screens, docks, and taskbars. They work regardless of network connection. They launch in their own standalone experience. They can read and write files from the local file system, access hardware connected via USB, serial or bluetooth, and even interact with data stored on your device, like contacts and calendar events. In these applications, you can do things like take pictures, see playing songs listed on the home screen, or control song playback while in another app. Platform-specific applications feel like part of the device they run on. Progressive Web Apps (PWA) are built and enhanced with modern APIs to deliver enhanced capabilities, reliability, and installability while reaching anyone, anywhere, on any device with a single codebase.

At their heart, Progressive Web Apps are just web applications. Using progressive enhancement, new capabilities are enabled in modern browsers. Using service workers and a web app manifest, your web application becomes reliable and installable. If the new capabilities aren't available, users still get the core experience.

The numbers don't lie! Companies that have launched Progressive Web Apps have seen impressive results. For example, Twitter saw a 65% increase in pages per session, 75% more Tweets, and a 20% decrease in bounce rate, all while reducing the size of their app by over 97%. After switching to a PWA, Nikkei saw 2.3 times more organic traffic, 58% more subscriptions, and 49% more daily active users. Hulu replaced their platform-specific desktop experience with a Progressive Web App and saw a 27% increase in return visits.

Progressive Web Apps provide you with a unique opportunity to deliver a web experience your users will love. Using the latest web features to bring enhanced capabilities and reliability, Progressive Web Apps allow what you build to be installed by anyone, anywhere, on any device with a single codebase. Therefore we have decided to build our web application 'Learning Log' through Django as PWA for users to interact and record entries in a reliable and secure way.

CHAPTER 2

LITERATURE REVIEW

H.W.Gellersen, M.Gaedke Proposed the theory of high level abstractions for design level modelling in a markup language.

Many of the studies in the area of web application development have mainly focused on the evolution of web application and comparison of web application development languages. Gellersen and Gaedke in their article ,overviewed object oriented web applications and identified object-oriented models for web applications,they found that XML technology contributes in enabling high level abstractions for design level modelling in a markup language. Mattsson identified the strengths and weaknesses of object oriented frameworks. Finifter and Wagner explored the relation between web application development tools and security. Chatzigeorgiou et al, evaluated object oriented design with link analysis. Paikens and Arnicans explored the use of design patterns in PHP- based web application frameworks. French presents a new methodology for developing web applications and web development life cycle. Copeland et al, in their article titled “Which web development tool is right for you” discussed and compared various tools for web application development.

Mehdi Jazayeri ,IEEE - Some trends in web application development

A similar perspective has been provided by another researcher about the growing trends of web application. It was cited by Mehdi Jazayeri in the paper called ‘Some trends in web application’.

He dealt with topic of how web application has transformed the internet into a much bigger virtual world,

Ever since the web came into existence it has evolved into something more in terms of scale, reliability, design, and in trends. It deals with a constant entity called 'change'. The paper presented by mehdi brilliantly captures the essence of what did really change therefore making us realise that the current commercial market relies heavily on PWA (Progressive web apps) along with the changing trends.

Ming-te Lu, Wing-lok Yeung - A framework for effective commercial web application development

The World Wide Web (WWW) or the Web has been recognized as a powerful new information exchange channel in recent years. Today, an ever-increasing number of businesses have set up Web sites to publicise their products and services. However, careful planning and preparation is needed to achieve the intended purpose of this new information exchange channel. This paper proposes a comprehensive framework for effective commercial Web application development based on prior research in hypermedia and human-computer interfaces. The framework regards Web application development as a special type of software development project. At the onset of the project, its social acceptability is investigated. Next, economic, technical, operational, and organisational viability are examined. For Web page design, both the functionality and usability of Web pages are thoroughly considered. The use of the framework should result in more effective commercial Web application development.

A.Jeff, J.T.Rayfield - Web application development using model/view/control design pattern.

The Model/View/Controller design pattern is very useful for architecting interactive software systems. This design pattern is partition-independent, because it is expressed in terms of an interactive application running in a single address space. Applying the Model/View/Controller design pattern to web-applications is therefore complicated by the fact that current technologies encourage developers to partition the application as early as in the design phase. Subsequent changes to that partitioning require considerable changes to the application's implementation-despite the fact that the application logic has not changed. This paper introduces the concept of Flexible Web-Application Partitioning, a programming model and implementation infrastructure, that allows developers to apply the Model/View/Controller design pattern in a partition-independent manner. Applications are developed and tested in a single address-space; they can then be deployed to various client/server architectures without changing the application's source code. In addition, partitioning decisions can be changed without modifying the application.

CHAPTER 3

AIM AND SCOPE OF PRESENT INVESTIGATION

3.1 Aim of the project

The aim is to provide users a blogging web application with the features that are solely reasoned on issues of privacy, better reliability, much better interaction between the client and the server with a neater GUI. The application can be used at any time or anywhere you want irrespective of the platform you are on, all it takes is a browser and internet.

3.2 Scope and Objective

The scope of web application is very huge as it paves the way for commercial markets and enterprise applications. They have managed to grow more and more with advancements in its development procedures like better application testing, flexibility in terms of system compatibility and much easier to program and deploy. It has grown significantly well in the aspects of application security compared to what it was back then.

The objectives are stated below:

- To identify the scope of progressive web application and the important role it plays in web development.
- To identify the security improvements in web applications and how it has evolved more secure towards exploitations over the course of time.
- To interpret the architecture and reliability of a web application for a better understanding of how the application works.

3.3 System Requirements

3.3.1 Hardware Requirements

- Hard Disk - 40GB and above
- RAM - 1 GB and above
- Processor - Intel I3 and above
- Mobile, PC , Laptop

3.3.2 Software Requirements

- Django
- Python
- Visual Studio Code
- Git
- Bootstrap
- Browser - Chrome/Firefox/Edge or of any kind
- Cmd - Command Line Terminal

3.4 Overview of the Project

3.4.1 Software Used

3.4.1.1 Django

Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source.

Library: Django includes dozens of extras you can use to handle common web development tasks. Django takes care of user authentication, content administration, site maps, RSS feeds, and many more tasks — right out of the box.

Security:

Django takes security seriously and helps developers avoid many common security mistakes, such as SQL injection, cross-site scripting, cross-site request forgery and clickjacking. Its user authentication system provides a secure way to manage user accounts and passwords.

Scale:

Django uses a component-based "shared-nothing" architecture (each part of the architecture is independent of the others, and can hence be replaced or changed if needed). Having a clear separation between the different parts means that it can scale for increased traffic by adding hardware at any level: caching servers, database servers, or application servers. Some of the busiest sites have successfully scaled Django to meet their demands (e.g. Instagram and Disqus, to name just two).

Django helps you write software that is:

Complete

Django follows the "Batteries included" philosophy and provides almost everything developers might want to do "out of the box". Because everything you need is part of the one "product", it all works seamlessly together, follows consistent design principles.

Versatile

Django can be (and has been) used to build almost any type of website from content management systems and wikis, through to social networks and news sites. It can work with any client-side framework, and can deliver content in almost any format (including HTML, RSS feeds, JSON, XML, etc). The site you are currently reading is built with Django!

Internally, while it provides choices for almost any functionality you might want (e.g. several popular databases, templating engines, etc.), it can also be extended to use other components if needed.

Maintainable

Django code is written using design principles and patterns that encourage the creation of maintainable and reusable code. In particular, it makes use of the Don't Repeat Yourself (DRY) principle so there is no unnecessary duplication, reducing the amount of code. Django also promotes the grouping of related functionality into reusable "applications" and, at a lower level, groups related code into modules (along the lines of the Model View Controller (MVC) pattern).

Portable

Django is written in Python, which runs on many platforms. That means that you are not tied to any particular server platform, and can run your applications on many flavours of Linux, Windows, and Mac OS. Furthermore, Django is well-supported by many web hosting providers, who often provide specific infrastructure and documentation for hosting Django sites.

3.4.1.2 Visual Studio Code

Visual Studio Code is a code editor in layman's terms. Visual Studio Code is an IDE (Integrated Development Environment) that helps the programmer write code, helps in debugging and corrects the code using the intelli-sense method. In normal terms, it facilitates users to write the code in an easy manner. Many people say that it is half of an IDE and an editor, but the decision is up to the coders. Any program/software that we see or use works on the code that runs in the background. Traditionally coding was used to do in the traditional editors or even in the basic editors like notepad! These editors used to provide basic support to the coders.

Some of them were so basic that it was very difficult to write basic English level programs in them. As time went by, some programming languages needed a specific framework and support for further coding and development, which was not possible using these editors. VI Editor, Sublime Text Editor, is one of the many kinds of editors that came into existence. The most prominent and which supports almost every coding language is VISUAL STUDIO CODE. Its features let the user modify the editor as per the usage, which means the user is able to download the libraries from the internet and integrate it with the code as per his requirements.

Features of VSC:

- Support for multiple programming languages
- Intellisense
- Easy to navigate between code and folders
- Built in git support

3.4.1.3 Bootstrap

Bootstrap is a free and open-source CSS framework directed at responsive, mobile-first front-end web development. It contains HTML, CSS and (optionally) JavaScript-based design templates for typography, forms, buttons, navigation, and other interface components.

Bootstrap provides basic style definitions for all HTML elements. The result is a uniform appearance for prose, tables and form elements across web browsers. In addition, developers can take advantage of CSS classes defined in Bootstrap to further customise the appearance of their contents. For example, Bootstrap has provisioned for light- and dark-colored tables, page headings, more prominent pull quotes, and text with a highlight. Bootstrap also comes with several JavaScript components in the form of jQuery plugins. They provide additional user interface elements such as dialog boxes, tooltips, and carousels. Each Bootstrap component consists of an HTML structure, CSS declarations, and in some cases accompanying JavaScript code. They also extend the functionality of some existing interface elements, including for example an auto-complete function for input fields.

Features:

Bootstrap comes with a whole barrellful of components you can easily tack onto your web page, including:

- Navigation bars
- Dropdowns
- Progress bars
- Thumbnails

Bootstrap comes with its own code for automatically resizing images based on the current screen size. Just add the `.img-responsive` class to your images, and the predefined CSS rules take care of the rest. No more spending hours coding your own grid—Bootstrap comes with its own grid system predefined.

3.4.1.4 Heroku Cloud

Heroku is a platform as a service based on a managed container system, with integrated data services and a powerful ecosystem, for deploying and running modern apps. The Heroku developer experience is an app-centric approach for software delivery, integrated with today's most popular developer tools and workflows.

Runtime: Heroku runs your apps inside dynos — smart containers on a reliable, fully managed runtime environment. Developers deploy their code written in Node, Ruby, Java, PHP, Python, Go, Scala, or Clojure to a build system which produces an app that's ready for execution. The system and language stacks are monitored, patched, and upgraded, so it's always ready and up-to-date. The runtime keeps apps running without any manual intervention.

Developer Experience: The Heroku Developer Experience is an app-centric approach to software delivery so developers can focus on creating and continuously delivering applications, without being distracted by servers or infrastructure. Developers deploy directly from popular tools like Git, GitHub or Continuous Integration (CI) systems. The intuitive web-based Heroku Dashboard makes it easy to manage your app and gain greater visibility into performance.

Heroku makes it easy to deploy and scale Python apps. Whether you prefer frameworks like Django or Flask, or getting your hands dirty with Twisted or raw sockets, Heroku helps you build things your way with the tools you love.

Security: Heroku offers developers a secure way of developing applications due to its set of security features. This PaaS platform reduces developers' requirement to push constant security patches, which can be a hassle, especially in more sophisticated web applications. Heroku ensures an optimal level of security for servers, application code and prevents any possible issues.

Powerful CLI: Heroku Command Line Interface or CLI is a feature that helps to develop and manage Heroku applications conveniently. The CLI is, in fact, one of the most reliable components of the Heroku PaaS.

Easy To Scale: Heroku PaaS relies on a Dyno for running the code written by developers. Each dyno is fundamentally a container encapsulating resources such as memory and CPU and application code and related dependencies.

Heroku users can then choose to create any number of independent application copies once it has been prepared, a process known as horizontal scaling. Applications that receive an unprecedented influx of traffic can be handled through the one-click creation of hundreds of app instances.

Heroku's auto-scaling feature helps it easily detect traffic spikes and create more Dynos accordingly.

CHAPTER 4

EXPERIMENTAL OR MATERIAL METHODS

4.1 DESIGN METHODOLOGY

4.1.1 Existing System

Today's Blog site provides users an opportunity to keep track of what the users share over a period of time on the website. It can either be private or public but these kinds of sites are extremely vulnerable in terms of reliability of servers, and data exploitation which often happens through scripting attacks. The data has to be managed in real time with robust connection to the internet. These servers can get overloaded or shut down through malfunction. The User's private entry can easily be exploited by degrading the site to HTTP protocols for insecure security restrictions which will make you visible to your ISP. They can not only record your browsing history but also monitor what kind of credentials you are using to login and the entries that you have uploaded in your blog. Very few blogging sites have HTTPS protocols available. The primary concern of today's blogging sites is that companies like Google and Facebook are starting to sell user data in the name of personalised suggestions and analytics purposes. These kinds of things make today's internet and the websites totally insecure.

4.1.2 Proposed System

In the Proposed system, The Web application can handle tasks like processing user requests such as creating entries and topics for their personal journal entries and collective recording of user data over a secure

connection through the network provided by HTTPS protocol.

Each user will be able to create their own account on the registration page, so that they can access their session through encryption provided by their username and password. The User can write and read their requested entries by requesting the URL for each site , The journal entries will be automatically sorted through data added for personalised journal experience. Users can respond to the site in a better way through interactive design of the website (GUI) that is provided by bootstrap preset themes. The site gives better search results and faster response for each user request to provide the best web application experience. Each user's data gets stored in database SQL and the website will be running on a live server from heroku.

4.2 MODULE DESCRIPTION

1. Interactive Website

The django framework and bootstrap 4 preset themes gives the website a pleasant look to interact with, It provides a simple and yet sophisticated look to the users every time they log into the website, GUI is constructively designed to avoid giving users a hard time navigation when navigating through different pages.

2. Database Management

The database is managed locally through SQLite as it provides a clear view over the architecture of how the data are constructed. The data is stored in SQL in the structure of the query set.

3. URL Refinement

Each link that opens a particular website is built with URLs defined for each site so that users can navigate easily with a clear understanding as to which site they are on. Each site is built and arranged in a form of tree structure.

4. User Login Management

The Site has a secure way for users to login privately with the help of username and password for each of the individual sessions. The authorization management is provided by the django framework and is built with care as to not leak the data of other people.

5. Graphical User Interface

Bootstrap 4 polishes the site with a better user interface which is elegant and simple for better user experience. The site features white theme with margins allocated for the title ,topics and user greeting on the home page.

6. Query Search

The data are arranged in a query set to supply data more efficiently at the request from users every time making it more reliable and optimised.

4.3 System Implementation

A user has to head down to the URL <https://learning-log-22.herokuapp.com/> , The homepage with a banner explains the app's monologue with a huge title front. The website displays buttons to register a new account or to login for an existing ID. users can either login or register a new account. Once the user login into the site they will be allowed into the topics section where your personal entries will be displayed and sorted in chronological order with the date. It lets you create new topics for entries and new entries related to a particular topic. Users can log out anytime they want and their activity inside the application will be completely secure.

1. The application has a neat user interface for better interaction
2. The site is reliable and works smoothly
3. The site responds to users over a secure connection provided by HTTPS Protocol

4.4 Architecture Explanation

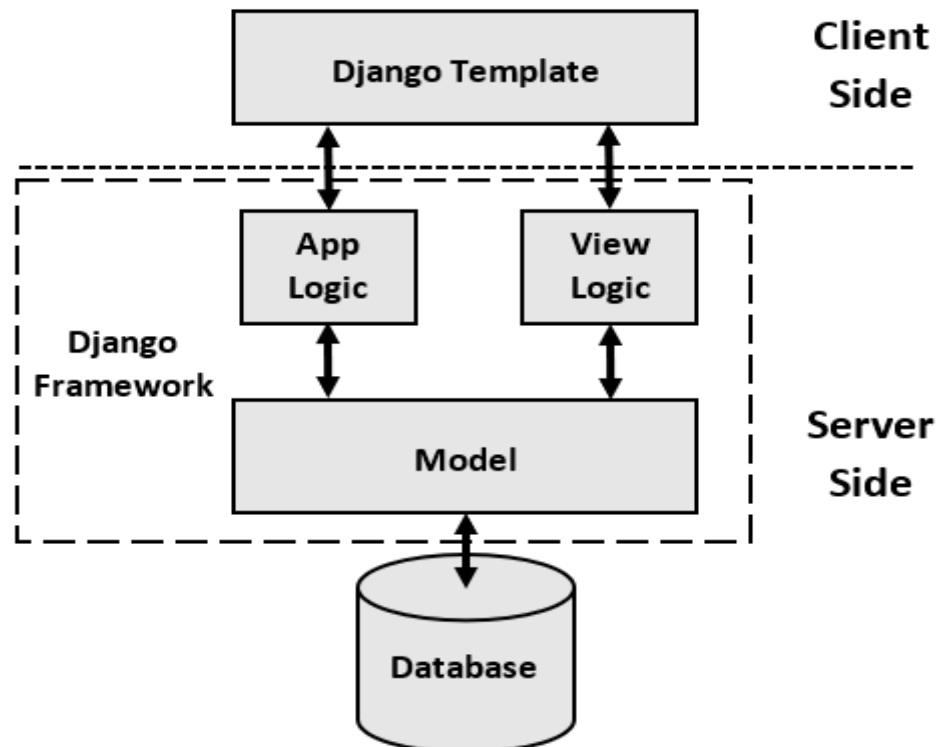
Django follows a Model-View-Controller(MVC) architecture, which is split up into three different parts:

- The Model is the logical data structure behind the entire application and is represented by a database(generally relational databases such as MySQL, Postgres).
- The View is the user interface, what you see in your browser when you visit a website. These are represented by HTML/CSS/Javascript files.

- The Controller is the middleman that connects the view and model together, meaning that it is the one passing data from the model to the view.

With MVC, your application will revolve around the model—either displaying it or manipulating it.

So say a user will enter a URL in their browser, that request will go through the internet protocols, to your server, which will call Django. Django will then process the given URL path, and if it matches an URL path you have explicitly stated, it will call the Controller, which will then perform a certain action, such as get an entry from your Model(database) and then render a View(ie: JSON text, HTML/CSS/JavaScript Web page).



4.4.1 User Registration

Django provides an authentication and authorization ("permission") system, built on top of the session framework discussed in the previous tutorial, that allows you to verify user credentials and define what actions each user is allowed to perform. The framework includes built-in models for Users and Groups (a generic way of applying permissions to more than one user at a time), permissions/flags that designate whether a user may perform a task, forms and views for logging in users, and view tools for restricting content

4.4.2 Login

Users can login into the site to access their individual account. They can login, logout, register accounts provided by django local library sets.

4.5 Working Procedure

Model - A model is a class that represents a table or collection in our Database, and where every attribute of the class is a field of the table or collection. Models are defined in the `app/models.py` (in our example: `myapp/models.py`).

Views - A view function, or "view" for short, is simply a Python function that takes a web request and returns a web response. This response can be the HTML contents of a Web page, or a redirect, or a 404 error, or an XML document, or an image, etc. Example: You use view to create web pages, note that you need to associate a view to a URL to see it as a web page.

URL Mapping - When a user makes a request for a page on your web app, Django controller takes over to look for the corresponding view via the url.py file, and then returns the HTML response or a 404 not found error, if not found. In url.py, the most important thing is the "urlpatterns" tuple. It's where you define the mapping between URLs and views.

RESULTS AND PERFORMANCE ANALYSIS

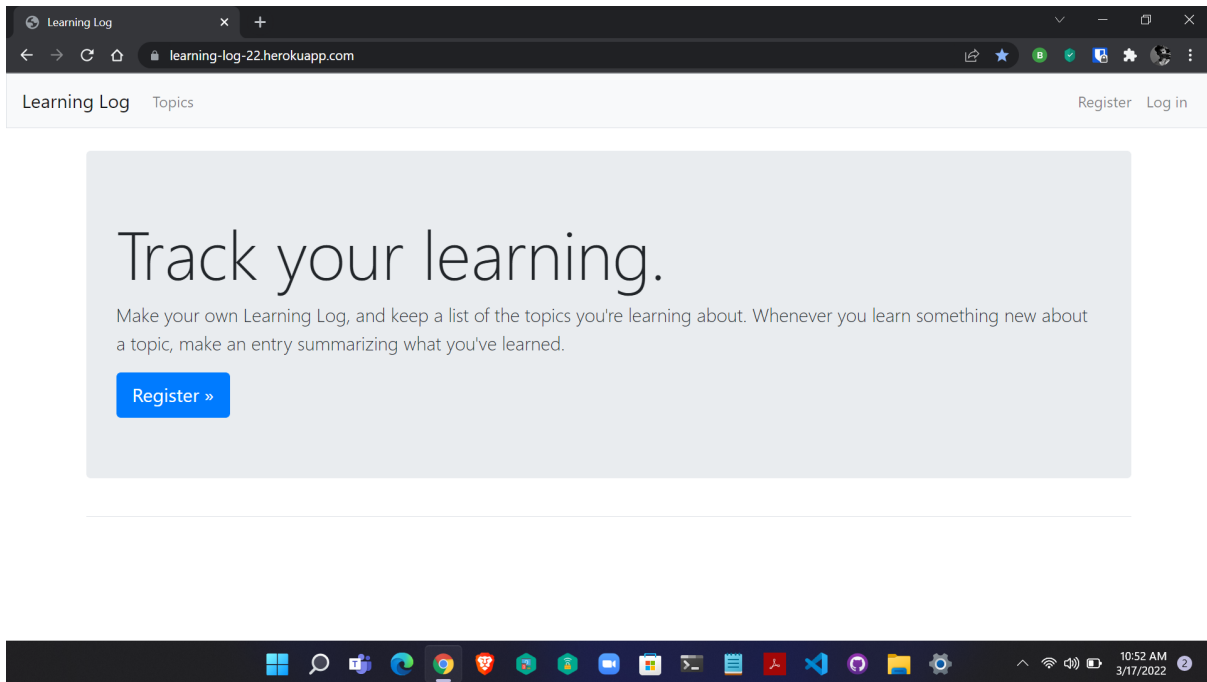


Fig: 5.1 Homepage

This is the homepage of the web application 'Learning Log'. The homepage covers a navigation bar at the top with the title, topics, register and login buttons embedded on it. A monologue about the page appears up front with a neat UI that reads "Track your learning", You can manually click the register button to create a new account or click the login to get into your existing account. The theme of the web application is provided by bootstrap 4 presets.

The UI buttons respond smoothly making it much more interactive, Django provides excellent performance and reliability to the site's functionality. It has zero clutter and cliché.

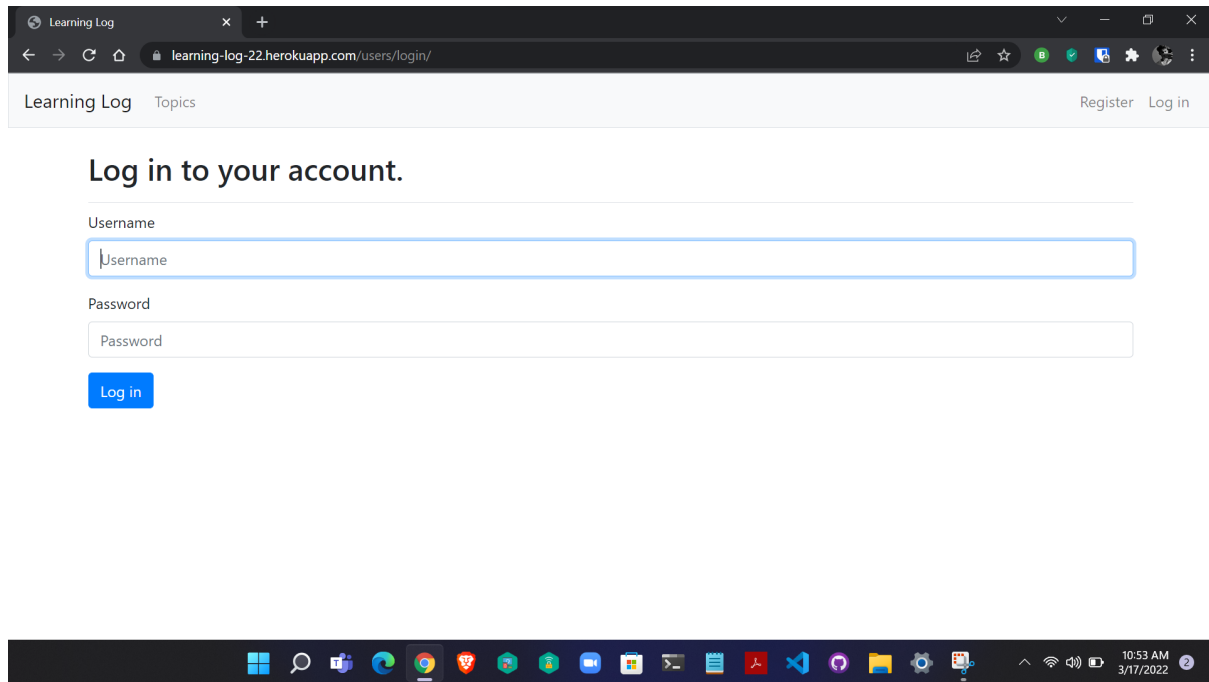


Fig: 5.2 Login Page

The login page displays the text fields for both username and password. The user authentication system is provided by the django local library. If a new user tries to login it will automatically prompt the user to create a new account. If django senses the old user it will open the account for them with ease.

Topics Page

Topics page will appear when you click the topics button on top, This is the place where your entries and blogs will be categorised into a particular topic that you set it as, for instance: Chess contains all the entries that you wrote about it. You can also create a new topic by clicking 'Add new topic'.

New topics page is embedded as an HTML Link so that it can open a new URL structured as https://learning-log-22.herokuapp.com/new_topic/. The page can only be accessed when you are signed in as to prevent the integrity of the data.

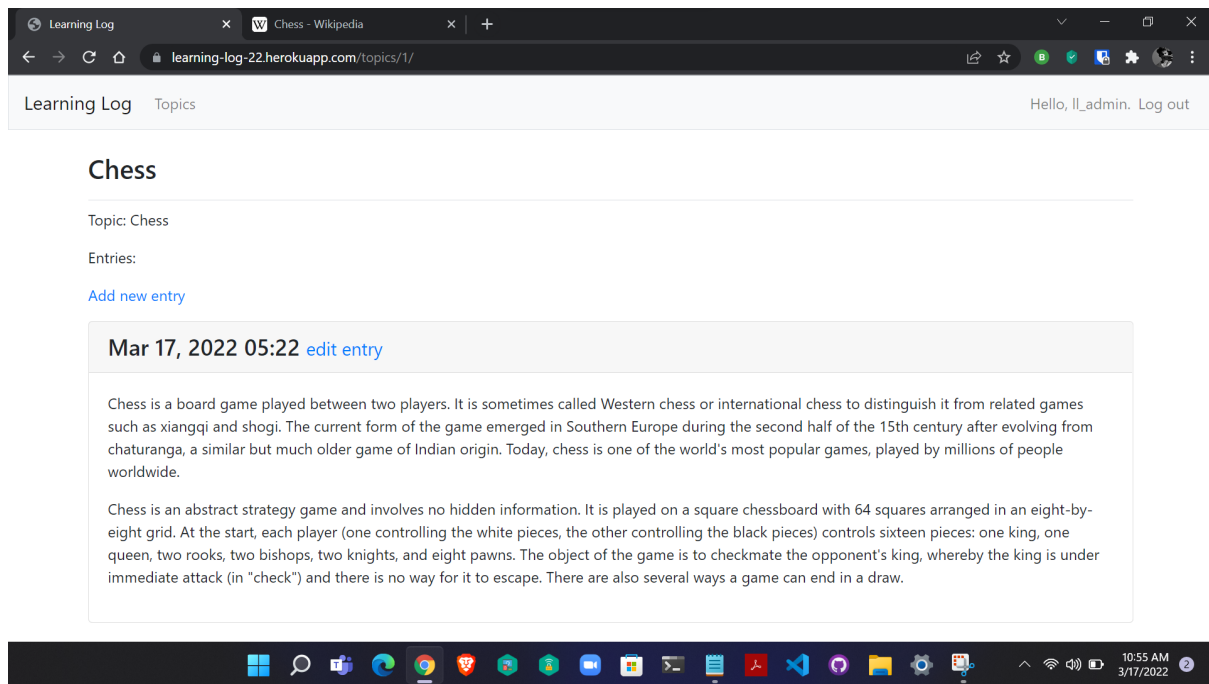


Fig:5.3 Entries Page

By clicking chess topic, You can open the entries that you wrote about that particular topic, as shown above you can see an entry written at that M/D/Y. The entry covers 'what is chess?'. You can edit your existing entry for modifications, you can delete any entries you want, You can create new entries and assign it to a particular topic that you create.

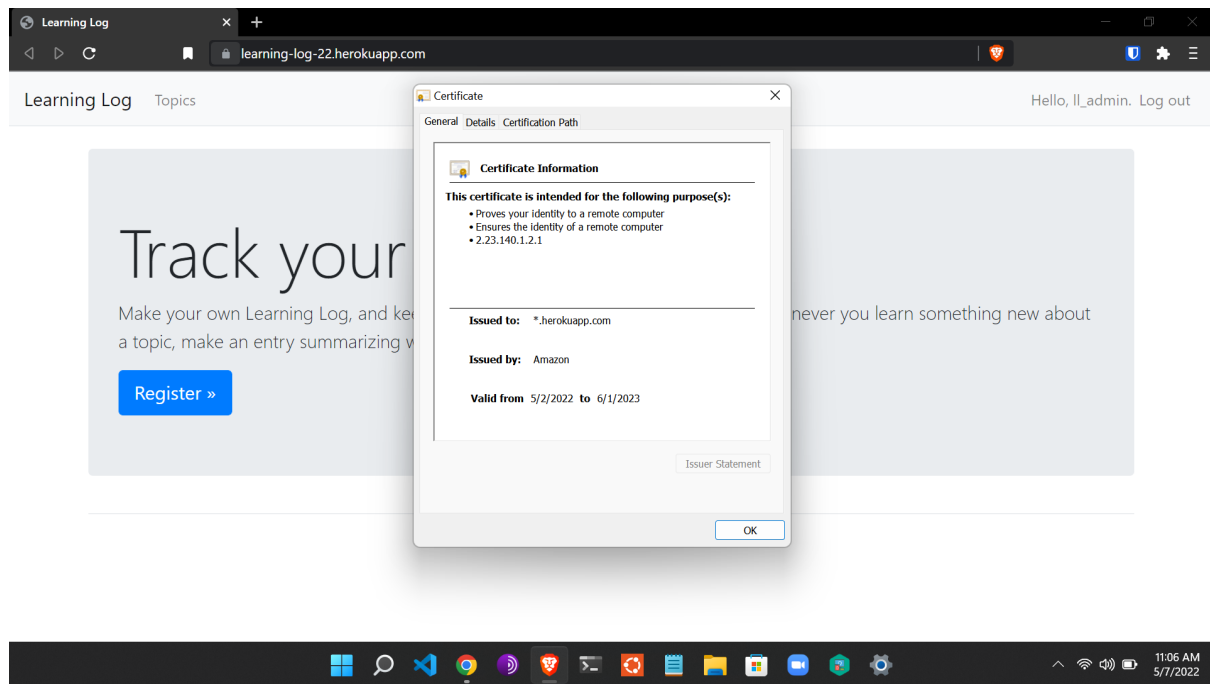


Fig:5.4 SSL Certification

The learning log web application is certified by web hosting service 'Heroku' given the fact that the main objective of the project is to provide users a secure site to prevent their privacy getting exploited in the first place. Whenever one tries to reach the site through a browser they can surf safely knowing the site is protected by HTTPS protocol with no trackers lurking behind.

CHAPTER 5

CONCLUSION AND FUTURE ENHANCEMENTS

6.1 Conclusion

The application has been tested with various browsers such as chrome, firefox, brave and edge to check the performance and runtime of the application on a live server. It has proven successful with all the essential features making it a secure blogging site like it was intended to perform. The site is secure with certification and has no trackers, verified by the results duckduckgo tracker radar set has given. Anyone can access the site using the URL <https://learning-log-22.herokuapp.com/> . Learning log acts as a blogging tool whenever it is needed for recording entries, journals, keeping notes etc. It can be accessed anytime and at any platform with a browser. We hoped to be a part of something that concerns the integrity of privacy and we have achieved a small fraction of it.

6.2 Future Enhancements

We would love to add features that give privacy dominance over the application by giving options such as public or private mode that can enable the exposure of accounts with the user's consent. Adding UI that is on par with major social media web apps like IG, FB etc. It gives us the advantage of handling major data and attraction across the country. Up until now we have managed to provide necessary features for a fully functional web application. New features can be added in the near future.

REFERENCES

- H.W. Gellersen, R. Wicke and M. Gaedke, "WebComposition: An Object-Oriented Support System for the Web Engineering Lifecycle", *Computer Networks and ISDN Systems*, vol. 29, pp. 1,429-1,437, 1997.
- T. Berners-Lee, "The World-Wide Web", *Comm. ACM*, vol. 37, no. 8, pp. 76-82, Aug. 1994.
- F. Coda, "Towards a Software Engineering Approach to Web Site Development", *Proc. Ninth Int'l Workshop on Software Specification and Design (IWSSD-9)*, 1998.
- Lu, M. and Yeung, W. (1998), "A framework for effective commercial Web application development", *Internet Research*, Vol. 8 No. 2.
- J. Coutaz, "PAC An Object-Oriented Model for Dialog Design", *Proceedings of Human-Computer Interaction INTERACT*, pp. 431-436, 1987.
- G.E. Krasner and S.T. Pope, "A Cookbook for Using the Model-View-Controller User-Interface Paradigm in Smalltalk-80" in *Journal of Object-Oriented Programming*, SrGS Publication, pp. 26-49, August/September 1988.
- K. Betz, A. Leff and J. Rayfield, "Developing Highly-Responsive User Interfaces with DHTML and Servlets", *Proceedings of the 19th IEEE International Performance Computing and Communications Conference — IPCCC-2000*, 2000.
- M. Abrams, C. Fanouriou, A. Batongbacal, S. Williams and J. Shuster, "UIML: AnAppliance-Independent XML User Interface Language", *Proceedings of the Eight International World Wide Web Conference*, pp. 617-630, May, 1999.
- A. Sadeghian, M. Zamani and S. Ibrahim, "SQL Injection Is Still Alive: A Study on SQL Injection Signature Evasion Techniques", *Proc. Int'l Conf Informatics and Creative Multimedia (ICICM 13)*, pp. 265-268, 2013.
- Glenn E. Krasner, Stephen T. Pope, "A cookbook for using the model-view controller user interface paradigm in Smalltalk-80", *Journal of Object-Oriented Programming*, vol. 1, no. 3, 1988, pp. 26-49.

- Berners-Lee, T. et al., 2001, The semantic Web. *Scientific American*, May, pp. 34–43.
- Holovaty, A., Kaplan-Moss, J., The Definitive Guide to Django: Web Development Done Right, Berkeley, CA: Apress, 2009.
- Bai X., et al. The Web Warrior Guide to Web Programming. Thomson Learning Course Technology, 2003
- Y.-W. Huang, S.-K. Huang, T.-P. Lin, and C.-H. Tsai. Web application security assessment by fault injection and behaviour monitoring. In WWW '03: Proceedings of the 12th International Conference on World Wide Web, 2003.

APPENDIX

A.SOURCE CODE

Model:

```
from django.db import models
from django.contrib.auth.models import User
# Create your models here.
class Topic(models.Model):
    """A topic the user is learning about."""
    text = models.CharField(max_length=200)
    date_added = models.DateTimeField(auto_now_add=True)
    owner = models.ForeignKey(User, on_delete=models.CASCADE)
    def __str__(self):
        """Return a string representation of the model."""
        return self.text
class Entry(models.Model):
    """Something specific learned about a topic."""
    topic = models.ForeignKey(Topic, on_delete=models.CASCADE)
    text = models.TextField()
    date_added = models.DateTimeField(auto_now_add=True)
    class Meta:
        verbose_name_plural = 'entries'
    def __str__(self):
        """Return a string representation of the model."""
        return f"{self.text[:50]}..."
```

Views:

```
from django.shortcuts import render, redirect, get_object_or_404
from django.contrib.auth.decorators import login_required
from django.http import Http404
from .models import Topic, Entry
from .forms import TopicForm, EntryForm
```

```

# Create your views here.
def index(request):
    """The home page for Learning Log."""
    return render(request, 'learning_logs/index.html')
@login_required
def topics(request):
    """Show all topics."""
    Topic.objects.filter(owner=request.user).order_by('date_added')
    context = {'topics': topics}
    return render(request, 'learning_logs/topics.html', context)
@login_required
def topic(request, topic_id):
    """Show a single topic and all its entries."""
    topic = get_object_or_404(Topic, id=topic_id)
    # Make sure the topic belongs to the current user.
    if topic.owner != request.user:
        raise Http404
    entries = topic.entry_set.order_by('-date_added')
    context = {'topic': topic, 'entries': entries}
    return render(request, 'learning_logs/topic.html', context)
@login_required
def new_topic(request):
    """Add a new topic."""
    if request.method != 'POST':
        # No data submitted; create a blank form.
        form = TopicForm()
    else:
        # POST data submitted; process data.
        form = TopicForm(data=request.POST)
        if form.is_valid():
            new_topic = form.save(commit=False)
            new_topic.owner = request.user
            new_topic.save()
            return redirect('learning_logs:topics')

```

```

# Display a blank or invalid form.
context = {'form': form}
return render(request, 'learning_logs/new_topic.html', context)

@login_required
def new_entry(request, topic_id):
    """Add a new entry for a particular topic."""
    topic = Topic.objects.get(id=topic_id)
    if request.method != 'POST':
        # No data submitted; create a blank form.
        form = EntryForm()
    else:
        # POST data submitted; process data.
        form = EntryForm(data=request.POST)
        if form.is_valid():
            new_entry = form.save(commit=False)
            new_entry.topic = topic
            new_entry.save()
            return redirect('learning_logs:topic', topic_id=topic_id)
        # Display a blank or invalid form.
        context = {'topic': topic, 'form': form}
        return render(request, 'learning_logs/new_entry.html', context)

@login_required
def edit_entry(request, entry_id):
    """Edit an existing entry."""
    entry = Entry.objects.get(id=entry_id)
    topic = entry.topic
    if topic.owner != request.user:
        raise Http404
    if request.method != 'POST':
        # Initial request; pre-fill form with the current entry.
        form = EntryForm(instance=entry)
    else:
        # POST data submitted; process data.
        form = EntryForm(instance=entry, data=request.POST)

```

```

if form.is_valid():
    form.save()
    return redirect('learning_logs:topic', topic_id=topic.id)
context = {'entry': entry, 'topic': topic, 'form': form}
return render(request, 'learning_logs/edit_entry.html', context)

```

URL Refinement:

```

"""Defines URL patterns for learning_logs."""
from django.urls import path
from . import views
app_name = 'learning_logs'
urlpatterns = [
    # Home page
    path('', views.index, name='index'),
    # Page that shows all topics.
    path('topics/', views.topics, name='topics'),
    # Detail page for a single topic.
    path('topics/<int:topic_id>', views.topic, name='topic'),
    # Page for adding a new topic
    path('new_topic/', views.new_topic, name='new_topic'),
    # Page for adding a new entry
    path('new_entry/<int:topic_id>', views.new_entry, name='new_entry'),
    # Page for editing an entry.
    path('edit_entry/<int:entry_id>', views.edit_entry, name='edit_entry'),
]

```

Registration Form

```

from django.shortcuts import render, redirect
from django.contrib.auth import login
from django.contrib.auth.forms import UserCreationForm

```

```

# Create your views here.
def register(request):
    """Register a new user."""

```

```

if request.method != 'POST':
    # Display blank registration form.
    form = UserCreationForm()
else:
    # Process completed form.
    form = UserCreationForm(data=request.POST)
    if form.is_valid():
        new_user = form.save()
        # Log the user in and then redirect to the homepage.
        login(request, new_user)
        return redirect('learning_logs:index')
    # Display a blank or invalid form.
    context = {'form': form}
    return render(request, 'registration/register.html', context)

```

Form (Visual Representation)

```

from django import forms
from .models import Topic, Entry
class TopicForm(forms.ModelForm):
    class Meta:
        model = Topic
        fields = ['text']
        labels = {'text': ''}
class EntryForm(forms.ModelForm):
    class Meta:
        model = Entry
        fields = ['text']
        labels = {'text': 'Entry:'}
        widgets = {'text': forms.Textarea(attrs={'cols': 80})}

```