Coursework 2

Ovidiu-Andrei Radulescu
40283288@live.napier.ac.uk
Edinburgh Napier University - Advanced Web Technologies (SET09103)

1 Introduction

Communication is an important element in our day to day digital lives. Since the inception of the internet, people have found new and various ways to communicate between each other faster and over longer distances with the help of technology. But that technology isn't limited to one service or one application, therefore different technologies need to communicate between each other as well. This is where APIs and webhooks come in, both of these services are used to send and sync data between different services.

The website I have built for this coursework uses the Discord [1] and Slack [2] webhook integration in order to create a system that can help the user into sending messages. The most common use of this would be similar to announcements on a message board, which is why the website will be called a WebHook Board. The way the Slack and Discord webhooks work is by creating a link(token) to one of the channels on a server, that an outside service can POST a message in a JSON format.

Because of the privacy nature of a webhook(anyone with a token can use it) the website offers an authentication system, and no feature is accessible to an unregistered user [Figure 1].The website allows for users to sign up for an account [Figure 2], where they can store and access their submitted webhooks [Figure 3].When submitting(or editing) a webhook [Figure 4], an user must enter the webhook's URL, a name and an avatar. Also of note, the user must choose the correct service option(Discord or Slack) as the services use different JSON notations for processing webhooks. After a webhook is set up, the user can then send messages to Discord or Slack, and the message will be displayed on the selected server and channel [Figure 5].

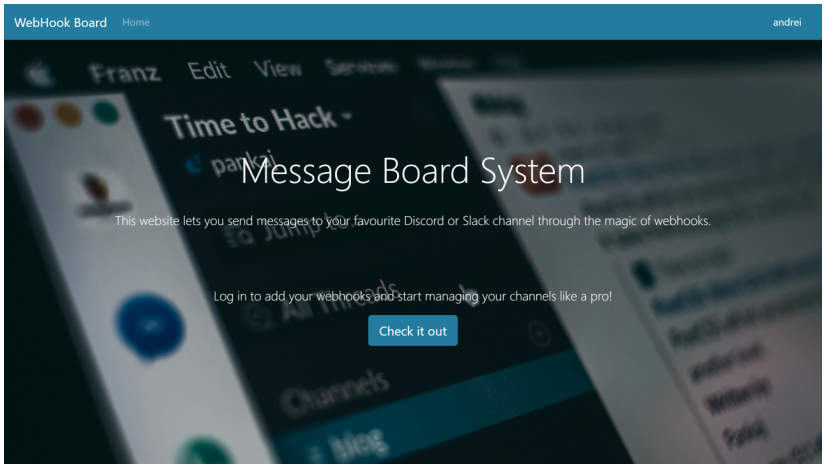

Figure 1: **Main Page** - Where there is a description and the user is asked to log in

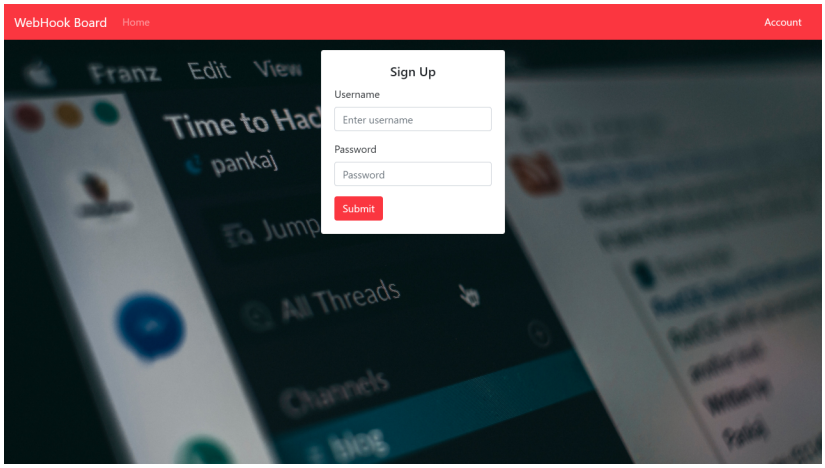

Figure 2: **Sign Up Page** - Where the user can register for an account

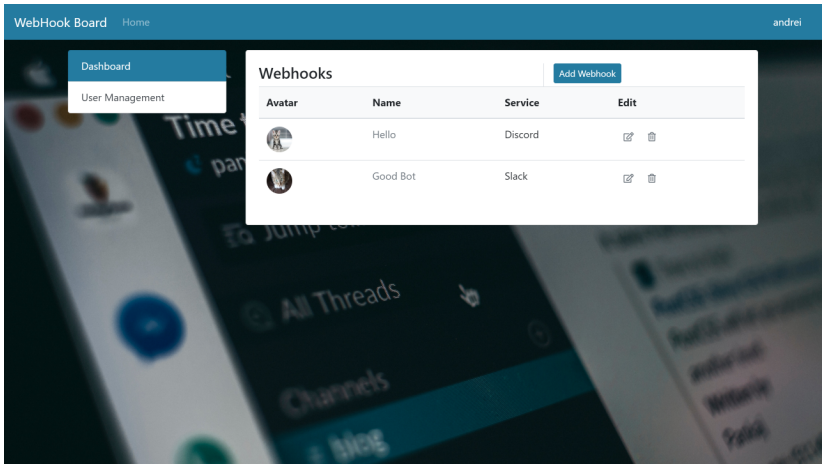

Figure 3: **User Dashboard** - Where the user can add, edit and remove webhooks

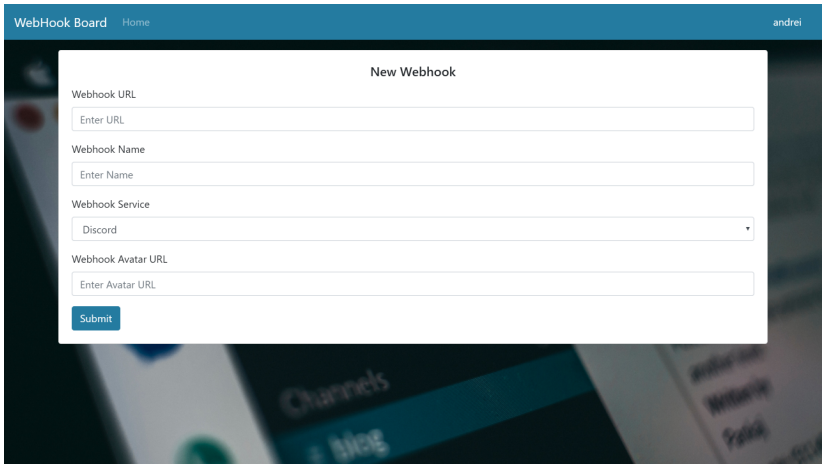

Figure 4: **Adding a new webhook** - The form used for adding a webhook

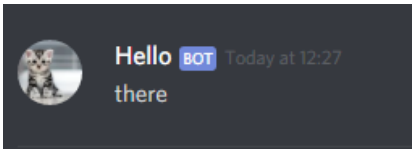

Figure 5: **A webhook message** - as shown on Discord

2 Design

After completing the previous coursework and getting a good grasp on Python and Flask, I wanted to improve on the skills obtained from working the workbook practicals and the coursework. I decided to do an user system authentication as the main focus of the development, making it as robust as possible. I have also used SQLite to store the user accounts and their webhooks, and I have used Bcrypt to secure their passwords. The passwords are stored hashed and salted, and any tampering with the database can only be done by a logged user and only to their own account and their webhooks. I then decided a good use for the system would be webhooks, as I have read about them in the past and they seemed like an interesting project idea that perhaps not many other students have done and they also required the privacy aspect of the website, in order to keep the wrong people away from an user's personal data.

With a project in mind , I went to Discord and Slack's documentation to learn about how to implement webhooks. Adding a webhook to Discord was very easy, as the feature is built into any channel and easy to access, but the Slack integration proved to be a bit harder, as it required creating a bot project that requested webhook access to a server, which proved to be more limiting than the Discord implementation, but at the same time more secure. At this stage, I made a diagram for the website structure and how the user would navigate it [Figure 6].

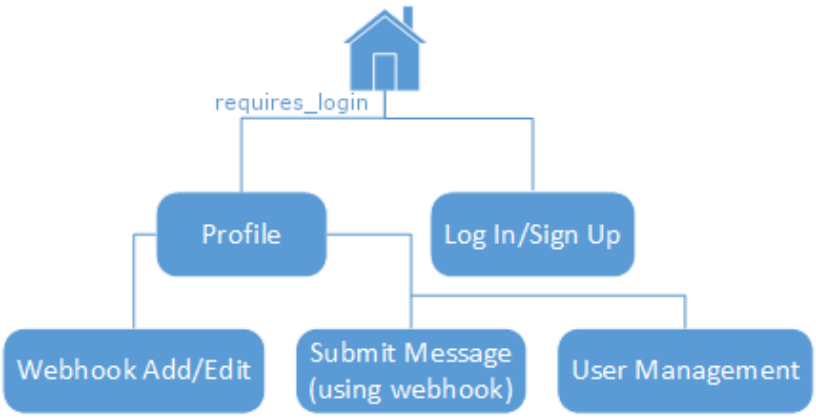

Figure 6: **Adding a new webhook** - The form used for adding a webhook

I have used Bootstrap[3] for the CSS and JavaScript of the website.The various sign in forms use Bootstrap's JavaScript for checking if the fields are empty, with password checking being done on the server side. Apart from these, I have used the Flask template method that uses Jinja to reuse my templates for multiple pages and to keep duplicated code to a minimum.The website also offers the user the option of changing the colour scheme, the colour scheme changes the colour of the navbar and the various buttons and hover effects around the website.

3 Enhancements

The website has, in my opinion, a good core functionality that can help users have quick access to sending messages, notes, announcements to their favourite servers. This is where a big improvement could be made, the message content. A big feature that could be added is automated messages, a user could set up specific messages to be sent when events trigger, such as a countdown to exams, or new events happening in the city. This could also be set up in an API form, where you could connect an external service, such as your e-mail, to your webhook, and it will send the reminders through a webhook.

Another feature that could be added could be password recovery system. Because of a busy semester, I did not have time to implement this before the deadline, as I considered it an optional feature that did not affect the core usability of the website, and I also did not want to collect people's emails unless needed. A password recovery system would work by sending a one-time use link to the user's selected email, where they can change their password. An additional security measure would be the addition of two-factor authentication.

4 Critical Evaluation

I started this coursework with the goal of making a robust and reliable website, that can be used to send messages using webhooks, its use being varied from simple messages to reminders, announcements, whatever a user might find it useful for. I am pleased with the final result, although I would have preferred to introduce the password recovery system as well as a 3rd party API support, even if limited.

5 Personal Evaluation

This project has been a great opportunity to expand and improve my knowledge of Python and Flask, especially by building a user system, as I have never done so for the past courseworks, this being a very important feature that is universal across the internet. My conclusions from the past coursework hasn't changed much, Python and Flask being very intuitive to use and making it very easy to manage the server-side development.

References

[1] Discord, "Discord Webhooks,"

[2] Slack, "Slack Webhooks,"

[3] Bootstrap, "Bootstrap Website,"