Advanced Programming Reflective Report

Introduction

Building the bus stop web application was a real test of my understanding and skills. It was still just as interesting as the more complex projects I've worked on before. At first, I was just trying to get it working, but as the project progressed and the architecture continued to stack up, I began to understand how the pieces fit together.

I learned from this experience that programming is more than just writing code—it requires thinking, experimenting, and figuring out how to fix things when things go wrong.

Web App Development

I started with a large dataset of bus stops, and I wasn't entirely sure if my overall structure was right. So, I decided to build it piece by piece based on what each page needed. The assignment required us to use Flask, but I had never completed a full project with it before. I slowly figured out how to use it to create pages and connect them to a database. I didn't follow a strict plan, but I had one habit: fix one issue at a time and test before moving on, adjusting based on the results. That habit saved me a lot of stress.

Some parts of the project went smoothly. Seeing the data show up correctly on the page felt great. When I added pagination and a working search bar, I was really proud. But there were also frustrating moments—like when database queries failed or the app crashed for no clear reason. I even had trouble with the layout when all the page numbers bunched up at the bottom during pagination. I had to keep searching online, carefully reading error messages, and trying different fixes. Thankfully, even though there were many issues, most could be resolved with enough effort.

Thinking ahead to how someone else—or even future me—might use or edit the app pushed me to refine my structure and keep things simple. Looking back, I can see that developing an app isn't just about functionality, but also about the experience of using and maintaining it. This also reminded me of the importance of balancing the aesthetics of the front end and the reliability of the back end, especially in user-oriented applications, where the first impression is crucial. The aesthetics of the front end can allow users to understand the function and role of the web page at first glance, and increase the amount of information on the web page as much as possible without triggering aesthetic fatigue in users.

Your perception of advanced programming is changing

Before taking this course, I thought advanced programming meant writing complex code. But through this project, I realized that this is not the case. It. But through this project, I realized that's not the case. It's more about writing clear, understandable code that won't break with small changes. Things like regular use of Git and writing tests started to make more sense to me.

I also learned that being a good developer isn't about knowing everything—it's about knowing how to figure things out. Now, I see programming as a creative process where you build something step by step and learn along the way.
This experience has taught me to value good practices over quick fixes. I used to think that as long as something worked, that was enough. Now I see the value in taking time to organize code properly, handle exceptions thoughtfully, and leave useful comments for clarity. These habits don't just make development easier—they also reduce stress when something goes wrong later. I've also come to appreciate the role of planning and consistency across a team—even if this was a solo project, imagining how another developer would understand my code helped improve its structure.

If I Did This Again

If I could redo this project, I would start with more detailed planning. I kind of jumped in without thinking about the big picture. Next time, I'd sketch out what I want the app to look like and how the data should flow through it. I'd also start writing tests earlier instead of adding them at the end.
Another thing I'd change is how I organized the code. At first, everything was packed into one file, which became hard to manage. Once I split it into smaller parts, it became much easier to handle.
I would also spend more time exploring Flask's ecosystem—blueprints, WTForms, and Jinja macros, for instance. I realized too late how helpful these could be. And if I had more time, I would experiment with adding user login functionality or filtering results by categories. These additions would make the app more interactive and closer to a real-world product.
Looking forward, I think incorporating automated testing and continuous integration would be a useful goal. These tools can save time in larger projects, especially when deploying updates. I also want to spend time learning about deployment best practices, since deploying to Render was smooth, but I know larger-scale deployments have more moving parts.

Technical Challenges and Solutions

One of the biggest challenges I faced was dealing with unexpected data formatting issues. Some records in the dataset had missing or inconsistent values, which caused certain queries to fail. To resolve this, I added validation logic during data import and used default values for missing fields. I also used try-except blocks to catch runtime errors more gracefully.
Another challenge was pagination. It seemed simple in theory, but coordinating page numbers, offsets, and links in both the back-end and front-end took multiple attempts. Eventually, I settled on a clean solution that allowed users to navigate easily without overloading the page.
By documenting each issue and my attempts to solve it, I built a mini knowledge base that helped me debug faster later. It also gave me a better understanding of patterns - what to look out for when things go wrong, and how to more effectively identify the cause.

My thoughts and conclusions

This project, while not particularly difficult, was a great experience in my programming career. Each test and code fix felt like replacing a support beam in the structure - well thought out and ultimately courageous in execution.And when everything finally worked, the satisfaction was incredible. I've gained more confidence in building something from scratch, and I now enjoy programming more. To me, advanced programming means building things thoughtfully and with care. It's not about showing off fancy code—it's about solving problems in clever ways and learning through doing. I now better understand the value of planning ahead, testing often, and designing with the user in mind. These are lessons I'll carry with me into future work. Most importantly, this course has helped shift my mindset from "make it work" to "make it right."

Reference:

Beck, K. (2003) *Test-Driven Development: By Example*. Boston: Addison-Wesley.

Flask Documentation (no date) *Flask: Web development, one drop at a time*. Available at: https://flask.palletsprojects.com/ (Accessed: 16 May 2025).

SQLite Documentation (no date) *SQLite Documentation*. Available at: https://www.sqlite.org/docs.html (Accessed: 16 May 2025).

Git Documentation (no date) *Git - Documentation*. Available at: https://git-scm.com/doc (Accessed: 16 May 2025).