Software Developing: Automating with Low-Code Solutions

CS4991 Capstone Report, 2024

Andrew Shin
Computer Science
The University Virginia
School of Engineering and Applied Science
Charlottesville, Virginia USA
ajs5qgm@virginia.com

ABSTRACT

As a computer science intern at Nationwide IT Services, I was given the option of focusing on migrating the company website or creating applications to automate corporate processes. I chose to pursue the automation project, because it would allow me to make a positive change in the company's internal processes. I spent a few weeks learning about Microsoft PowerApps, a tool that lets developers create the base application from scratch without having to code a single line of HTML. Using PowerApps, I implemented a weekly report application that allowed project managers and team members to document what they accomplished that week, and what was to come the following one. Although I was only able to finish the primitive version of the application, one simple yet effective portal did the job and the team's process no longer requires a shared google doc and a string of emails. By designing a low-code course for underclassmen, I hope to draw student insights to the benefits and downsides of a project-based introductory CS course.

1. INTRODUCTION

What happens when professionals with little to no programming knowledge are able to implement full applications from scratch? The learning curve of application development flattens and an extensive understanding of abstract computer science principles is no longer necessary. The user can instead focus on learning and implementing high-level concepts such as database management and app automation. Low code development platforms provide users with an environment to implement applications through a graphical user interface (GUI), allowing them to take advantage of these benefits.

However, simplifying the development process comes with its tradeoffs. Similar to picking a theme for a slide deck, the user can select a color theme for the application before they start their "drag and drop" implementation process. They can then select the size and layouts of the components using the GUI; but that is the extent. The styling of the app will not be better than one created by an expert of web development. This issue, however, can be overcome as the app can be viewed in code form, giving those who wish to make specific changes a direct option.

The University of Virginia does not have a course that teaches students no/low-code development platforms. While some courses like HCI (Human Computer Interaction) encourage students to use no-code software to create mock-ups for their ideas, these applications often have no practical functionality, limiting the application to a graphical prototype. Low-code platforms can be used to introduce CS majors to concepts critical to software development without wasting time learning new programming languages necessary for software development.

2. RELATED WORKS

In a publication about social computing, Przegalinska (2024) considered the methods and impact of teaching low/no-code solutions to non-IT students. She stated after surveying various students and instructors that low-code solutions were easy to utilize even with a group with low technical literacy. My proposal utilized Przegalinska's approach to using low-code solutions to introduce more advanced important methodologies, which mitigates additional difficulties encountered by students learning multiple concepts at the same time.

When surveying fellow developers from my internship about low-code solutions, Ivan (Zheng, personal communications, March 27, 2024) mentioned that low-code solutions were incredibly efficient at automating manual processes, while Sahil (Matre, personal communications, March 27, 2024) stated that a low-code application increased the efficiency of conducting weekly reports by nearly 300%. These comments helped me choose low-code platforms as the center piece of the new course.

3. PROPOSED DESIGN

The purpose of this course will be to provide exposure to first and second-year students to basic software development principles through low-code platforms. As a result, the course will be named Intro to Software Development.

3.1 Course Hierarchy

Intro to Software Development provides a framework for more advanced concepts, making it more beneficial to take early. Thus, Intro to Software Development (CS 2240) would require CS 1110 as a prerequisite and be a prerequisite to CS 3240 (Advanced Software Development). CS 3240 is arguably the most important class for CS majors and any measures taken to prepare students for this class are indispensable.

3.2 Course Material

3.2.1 Software Development Principles

Students enrolled in this course will only have the minimal skills necessary to code a basic application in Python. To introduce the students to the pillars of software development, the first half of the semester's content will be composed of concepts taught in CS 3240. The principles I learned from Advanced Software Development were simple yet valuable. Understanding critical ideas such as the Software Development Lifecycle and User Story management provided me with the basics I needed to start personal projects. Students will also be introduced to Git, a version control system. These are critical skills utilized by all developers. These skills can only be improved through practice, making the earlier the exposure, the better.

3.2.2 Microsoft Power Platform

The second half of the course will teach students how to use Microsoft's Power Platform. The Power Platform is a collection of features that can be used collectively to build awesome applications. Power Apps are used to create applications, Microsoft Data Verse for the database, and Power Automate to automate workflow processes. None of these features require coding knowledge; however, they can reinforce the concepts they've learned that semester without having to create a full Django project.

3.3 Course Assessment

3.3.1 Quizzes

This course isn't suitable for exams. Instead, a mini-quiz administered twice a month that measures concept mastery will sufficiently gauge the class's overall concept mastery.

3.3.2 Homework

Students will work with a partner on homework assignments, getting more practice with Git in the process. Whether the

assignment is to create a requirements document from user stories or to code a simple automated testing program, these relatively small tasks would be designed to provide a manageable method for the students to practice what they've learned in theory.

3.3.3 Project

When the students learn about Microsoft Power Platforms, they will draw natural connections with concepts they've learned and the Power Platform, allowing them to create a "full-stack" project without having to write any code. This project will likely be the first time they create a full application and can be the training grounds they need to learn the software development process.

3.3.4 Final Exam

The final exam of this course will be weighted heavily and assess students' overall understanding of software development principles and proficiency in power platforms.

3.3.4.1 Written Portion

Instead of having a traditional written exam with multiple choice, true and false, etc., students will be given a scenario and be tasked to draw user stories, create requirements, and plan their application. This form of examination better assesses concept mastery and allows students to forego studying every little detail.

3.3.4.2 Programming Portion

This section of the final exam involves implementing their plan using Microsoft's Platform. Given a new set of instructions, students will create an application from the ground up. The student's application would adequately assess their mastery of basic Power Platform features.

3.4 Modifications to CS 3240

While CS 3240 will remain a project-based course that utilizes Django, students will be taught how to use Django over the initial stages of the course. The informative lectures would replace the Django tutorial assignment that requires the student to self-learn their first framework. Space for these new lectures would come from the added mastery of basic software development principles from Intro to Software Development. These lectures will increase the group members' Django mastery, mitigating the chance of dysfunctional project groups.

3.4.1 Modifications to Django Project

The current project for CS 3240 is well-formatted but can be optimized to be more realistic. Since the students have already experienced applying software development basics through their CS 2240 project, CS 3240's project can be reformatted to simulate a real-world software development scenario. This can be done by having TAs and the professor represent relevant roles in the industry, giving teams a specific application to build, and assigning teams sprints with the overall goals. Because of the addition of CS 2240, both the learning curve of Django and the difficulty of applying SDLC concepts will be reduced, setting project teams up for success.

4 ANTICIPATED RESULTS

Results of this project include both potential benefits and potential negatives.

4.1 Benefits

Students who take Intro to Software Development will be stronger candidates when looking for an internship as a first year. Most first years in CS have little to no coursework or projects that are impressive enough to land an internship. However, students who take CS 2240 will not only be able to talk about relevant software

development concepts during their interviews but also have an impressive project on their resume. This advantage cannot be understated because opportunities are given to those who have previous experiences and a first-year summer internship can boost career prospects drastically.

Early exposure to software development environments that utilize frameworks and platforms creates more curious and well-rounded software developers. Students who are motivated, but don't know where to find the resources to get started can obtain inspiration from what they learn in class.

4.2 Negatives

Though software development principles are easy to understand, learning a whole service platform may seem daunting at first. It'll be of utmost importance to prioritize the student's understanding of Microsoft Power Platform. If the roadblock to application development comes from the technology itself, it'll be impossible to apply the software development principles to the project.

5 CONCLUSION

I learned at Nationwide IT Services that different technologies are necessary for different situations, which helped me realize that no/low-code alternatives should be implemented in college CS curriculums. Due to the simplistic nature of low-code solutions, it serves a greater value to introduce them earlier in the student's academic career. Since low-code technologies are avidly used by various companies, it can be easily structured to complement the learning of software development principles.

This class will not only better prepare students for CS 3240, but also equip students with the necessary experience to find their first opportunity. Along with that, students will learn to use unfamiliar technologies, a skill necessary for a field with constantly evolving technologies. Universities design their curriculum to progress and evolve students' understanding of core foundations. From the students' performance and feedback about the class structure, professors can learn about the impacts of introducing difficult yet relevant concepts in an introductory course. Using this information, new emerging technologies that will need to be considered can be crafted into courses more easily.

6 FUTURE WORK

Intro to Software Development will need to be compared to CS 3240 and other similar courses to minimize content overlap. Along with this, professors will need to consider what specific low-code platforms will best complement the concepts within the software development lifecycle. After that, professors will need to hire TAs with specific low-code experience. This is especially important as a strong teaching staff is crucial to the success of a new course. Lastly, it is necessary to acquire feedback from the students who complete this class because of its interesting course structure, especially for a introductory class. After analyzing and incorporating student feedback, Intro to Software Development may become a class that significantly strengthens the CS curriculum of UVA.

REFERENCES

Sońta, M., & Przegalinska, A. (2024). Say 'yes' to 'no-code'solutions: how to teach low-code and no-code competencies to non-IT students. In *Handbook of Social Computing* (pp. 330-342). Edward Elgar Publishing.