Web application frameworks have been a staple in the web development realm for many years, and many developers have a heavy reliance on such frameworks for their ability to mitigate a large amount of overhead when implementing common activities such as templating, session management, database access, and more.

However, development with a reliance on frameworks is not always easy. There are a number of difficulties and disadvantages that exist with the utilization of web application frameworks. First, version control makes the continuing development with a specific framework difficult in the sense that frameworks typically evolve rapidly. When a new version of a framework is released, often times old library versions no longer are supported by the latest framework. One instance of this being the case is with the Paperclip gem for Ruby on Rails. Paperclip has been a popular library for developers required a file attachment library for Active Record. When Rails 3 was released, it was quickly discovered that the generator architectures for the framework had been changed so drastically that no implementation using the Rails 2.x syntax would work any longer. This led to a number of quick-fix and patching solutions that various developers implemented, which broke away from the standard for implementing generators in the past.

Secondly, when comparing between separate frameworks, it becomes apparent that many frameworks handle production deployments completely differently. For this example, let’s imagine the difference between Django and Rails for pushing a project to Heroku. NEED EXAMPLE

Also, the interaction between the framework and databases is highly different between the various frameworks. This again makes a consistent approach for developers between frameworks almost impossible, as each time the developer works with a different framework, their thought process and approach methodology must also adjust. As an example, let us imagine the differences between Django and Rails for migrating a database. NEED EXAMPLE

One of the largest difficulties is simply in the opposing design choices used between various frameworks. While most web application frameworks take advantage of an MVC architecture, their own implementations vary widely between one another. This exists across the entire implementation of the frameworks, and can easily be seen in the underlying file structure of popular frameworks such as Django and Rails. NEEDS CLARIFICATION This ultimately makes the process of learning new frameworks as, or even more, difficult than learning an entirely new language for developers.

In an effort to reduce the difficulties that exist for developers that work between multiple frameworks, we propose a universal web application framework. This proposed framework would retain a similar design pattern and syntactical approach across the most popular languages being utilized today, with an emphasis on Ruby, Python, and PHP for the sake of the proposal. The following sections outline the clear motivation and past research involved with attempts at a universal web framework, and are concluded with our proposal for the design of a modern and effective universal framework.