Our approach to building this system is going to be fairly straightforward for the most part. We will combine all of our cumulative knowledge that we’ve gained from our time here at Lane and divvy up responsibilities based on our strengths. Some examples include the use of the MVC Architecture, use of visual studio as our IDE, and Microsoft Azure as a web hosting solution during the development process.

In addition to these, some of us have experience with technologies outside of what we’ve been taught here at lane and we intend to utilize these wherever we decide they are the best fit for the task. Some examples of these technologies include Bootstrap for elegant styling of our views, JQuery for appropriate manipulation of the HTML, and a template structure for our project that mimics the way Content Management Systems like Expression Engine function. This is important because it represents one of our primary design principles: DRY, or Don’t repeat yourself. Rather than utilize unique views for individual web pages in our application, we will try to instead utilize a general structure and layout that most if not all views will share, and represent unique data as partial views instead, reducing the overall amount of HTML necessary to create the application as well as make our project more succinct and easy to read.

In addition to these technologies and strategies for constructing the application itself, we will also use GitHub as a means for tracking various facets of the application as it is being developed. These things include:

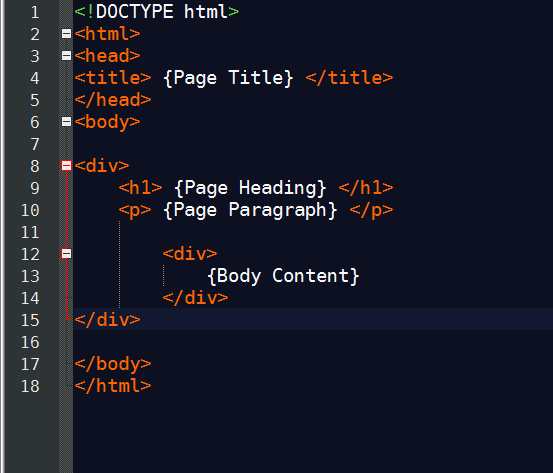
* Each team member’s contributions to the project
* Week-to-week milestones or “sprints”
* Issues and task assignments to individual team members
* Version control
* Minutes from our weekly meetings
* Interactions with the client

As Patrick is currently the strongest member of our team in regards to Git, he has accepted the role of task manager for our team ensuring the efficient tracking and organizing of these things on behalf of the team on GitHub.

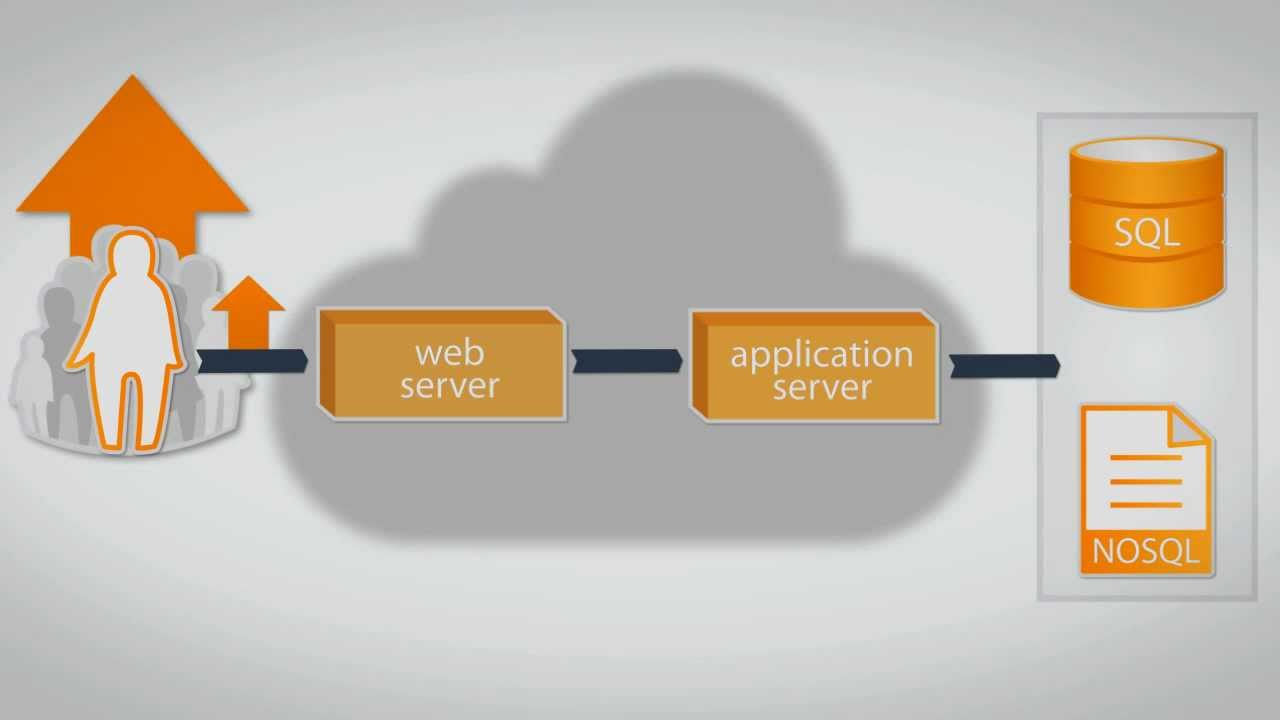
In addition to keeping our source code DRY, our guiding principles for developing the Keller-Williams Family Check-In system are as follows in order of highest priority first:

* **Functionality first** : when it comes down to making decisions, we will always prioritize getting the system functioning before any other concerns are addressed. Our philosophy is that once its working, then we can go back and adjust as needed, whether its refactoring, changing aesthetics, or finding a less resource-intensive method of implementation.
* **Scalable** : Keller-Williams has informed us that they currently have under 100 employees in their current office but that they hope to expand to 300 in the future. Additionally, they have expressed interest in sharing the application we make for this office with other offices within the Keller-Williams family, and not knowing precise numbers means that we will have to design the system to work well with at least 300 people in mind.
* **Easy to use** : The friendly folks at Keller Williams are wonderful at giving us direction and insight about what things are important to them and what functionality they desire, but something they’ve told us is that they aren’t hugely technically savvy. It is therefore important for us to design the KWFCI system with ease of use at the forefront of our minds.

Img for template



Img for scalability



Img for github

