FlixBook: A Movie and Tv Tracker from the Future

High-Level Design

<COP 4331C, Fall, 2014>

Version	Date	Who	Comment
v0.0	**10/22/14**	**Ramses M.**	**Template, added High-Level Architecture**
v0.1	**10/22/14**	**Lakshmidhar C.**	**Added Design Issueus**

Team Name:

Group 15

Team Members(click on name for website):

Roman Larinov - rlarionov@knights.ucf.edu

Ramses Mederos - ramsesmederos@knights.ucf.edu

Lakshmidhar Chigurupati - lakshmidharc@knights.ucf.edu

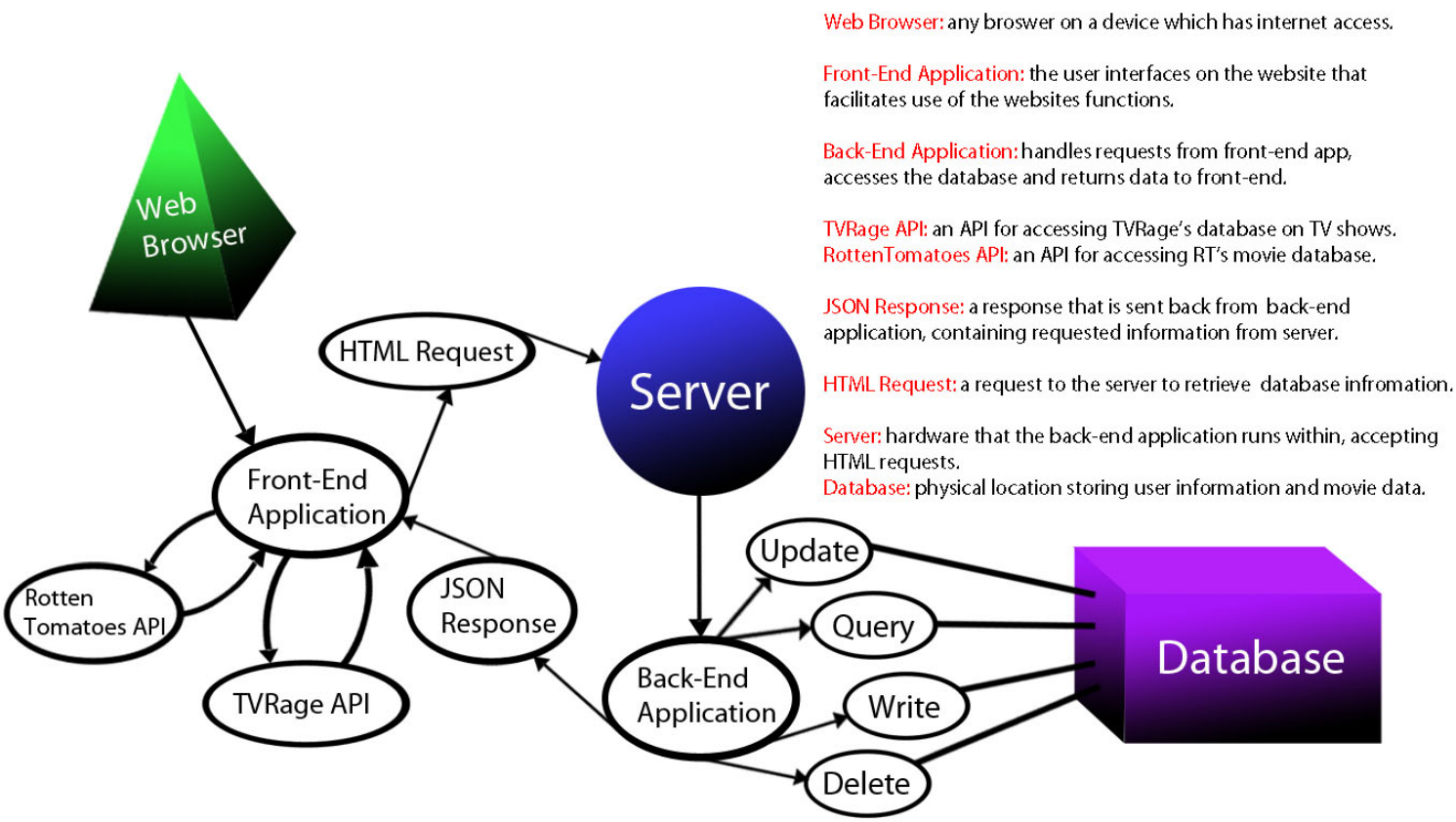
Michael Wahlberg - wahlberg2012@knights.ucf.edu

Michael Pittmman - m.pittman@knights.ucf.edu

Benjamin Kirksey -

Table of Contents

High-Level Architecture:



Design Issues:

During the planning process, we decided that the above architecture would best suit our needs for our product. We arrived at this conclusion after considering such things as maintainability, performance, scalability across different devices (i.e. different browsers/versions of browsers), and safety.

Since we are following the Agile method of software development, our maintainability will be rapid. In other words, the ability of our development team to fix issues will be quick and responsive. This ensures that the current product is as free of issues as possible. The product, FlixBook, will be tested after every sprint to ensure that the product functions as it should and that no previous features have been broken. This will ensure that at the end of each sprint, the release candidate of the product will always be a more functional edition of the product in production.

A key component that we must consider in the development of the product is the performance of the product. This means that throughout the whole development process, we must keep in mind the speed of the response to requests made by the user.This ensures that the user will have a pleasurable experience while using the site and will also ensure that the user is more likely to return as the site is quick to respond. To ensure that we, as a company, have a secure product, we will attempt to prevent the user from having access to the backend as much as possible. This can be done by putting most, if not all, the business logic

in server side files. This ensures that the only source files that the user has access to are the files that are associated with the frontend of the website (i.e. the html/css/javascript).

In order to ensure the website is scalable among different devices as well as different browsers, we will be using a CSS framework that automatically scales the website based on the screen resolution. This ensures that the user, regardless of which device they're on, be it mobile, tablet, phablet, laptop, or desktop, will always be shown a site that has an excellent user experience.

We decided to go with the above architecture because it allows us to modularize as much of the code as possible. By modularizing, we are able to group together parts of code that function in the same way. For example, we are able to keep all user interface related code in the frontend of the application and keep all the server and database code in the backend of the application. Not only does this allows for a faster isolation of the problem, which allows for quicker responses to bugs and issues in production, but it also adds a layer of security by ensuring that the user does not have access to the business logic or database of the product. The architecture was based on our design, so no valuable design tradeoffs were made.

Some of the technical difficulties that we expect to encounter during the development of the product include, but are not limited to, improper user authentication, failure to save user data, incorrectly saving user data, and slow response times from the server. Improper user authentication is a major technical issue because it means that the user cannot login even when they are providing the correct login credentials. This would mean there is a problem with the way the password checking or username checking. To fix this problem, we would have to reexamine our hashing function as well as the way in which the data is being stored and retrieved from the database.

Technical difficulties can also occur when attempting to save user data. This could mean anything from the user data being lost while being stored in the database to the user data not saving at all in the database. This is a major problem because it would mean that users will never be able to use a significant portion of the website as it requires a user account. To fix this, we would need to ensure that all the data is being passed correctly to the database and that the database handles the storing correctly. This means that we have to check and make sure that the proper commands are being called to link user interaction with the server and database.

Finally, a minor technical difficulty that we can encounter is slow response times from the server. This is an issue because if there is a large amount of user requests, the server could suffer from a DDOS. This would mean the server either crashes or has large amounts of latency due the large number of requests. The way to fix this would be to ensure that the server uses some sort of queue to ensure that all requests are server in a first in, first out fashion. This will ensure that all users have similar response times as well as short response times. Another solution is to present the user with a loading screen so that they are not inclined to refresh the page or repeat their actions, which if they did, would further exacerbate the issue.