FlixBook©

The Movie/TV Tracker from the Future

The Team

Roman Larionov
Back-End Engineer

Lakshmidhar C.
Back-End/Front-End Engineer

Benjamin Kirksey
Quality Assurance Specialist

Ramses Mederos
Front-End Engineer/Designer

Michael Wahlberg
Front-End Engineer

Michael Pittman
Front-End Engineer

What is it?

A State-of-the-Art Movie Tracker

A Cutting Edge Movie Notification System

A New Way to Browse Movies

Concept of Operations

Lakshmidhar Chigurupati

The Current System

NONE!!!!

That remind the user of an upcoming movie release.

Our Solution: FlixBook

Allows users to opt in for email notification for upcoming movies

Allows users to easily access to an extensive movie database

Allows users to rate movies for user reviews

Benefits of Using FlixBook

Never forget a movie release again!

Never rely on just critic or just user reviews again!

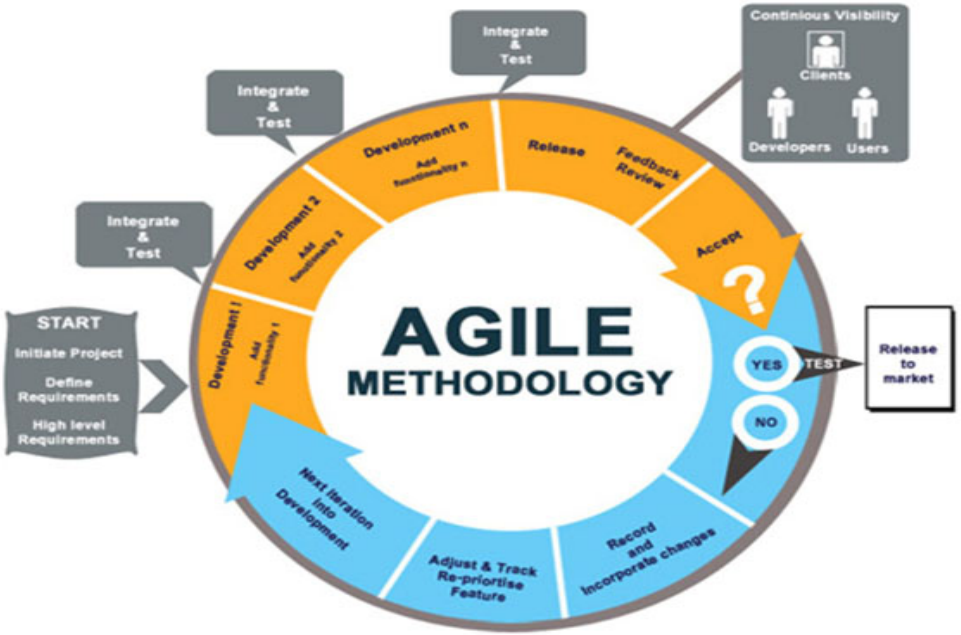
Never run out of movies to watch again!

Software Life Cycle Process and Progress Management

Michael Wahlberg

Agile Method

- Development in sprints
- Testing in intervals
- Final version testing
- Allows repeat of process:
  - Addresses client concerns.
  - Addresses incomplete features.



Tracking, Control, and Reporting of Progress

- Group meets every Tuesday
- The group meets whenever there’s an issue.
- Online communication through Slack
- The group will work around each member’s schedule
- For physical meetings we address absent members through:
    - Skype
    - Calling on speakerphone
    - Filling absent members in later

Progress Metrics

- To determine overall progress we will use:
  - Implemented features.
  - Specific features in both the front end and back end.
  - The rate at which we finish features.

Tools and Configuration Management

Roman Larionov

Development Tools and Environments

Frameworks

- Bootstrap (styling)
- jQuery (IMDb, Rotten Tomatoes)
- Node (modular, efficient server)

Server

- Database established with redis
- Constantly running Node.js server

Environment: Chrome Dev Tools, Raspberry Pi

Version Control

- Git
- Personalized branching
- Segmentation of work

Risk Management and Quality Assurance

- Losing a team member
- Server performance
- Two step testing process

System Requirements

Michael Pittman

Functional Requirements

- Users must have the ability to create accounts and have the ability to login and logout easily.
- Users must have the ability to search movies based on title, actors, year, or genre.
- Users must have the ability to save movies to a library that can only be accessed through the user account on which it was created.
- The web application must have the ability to recommend movies to the users.
- The web application must have the ability to alert/remind users of when specified movies are coming on television or being released into theatres.

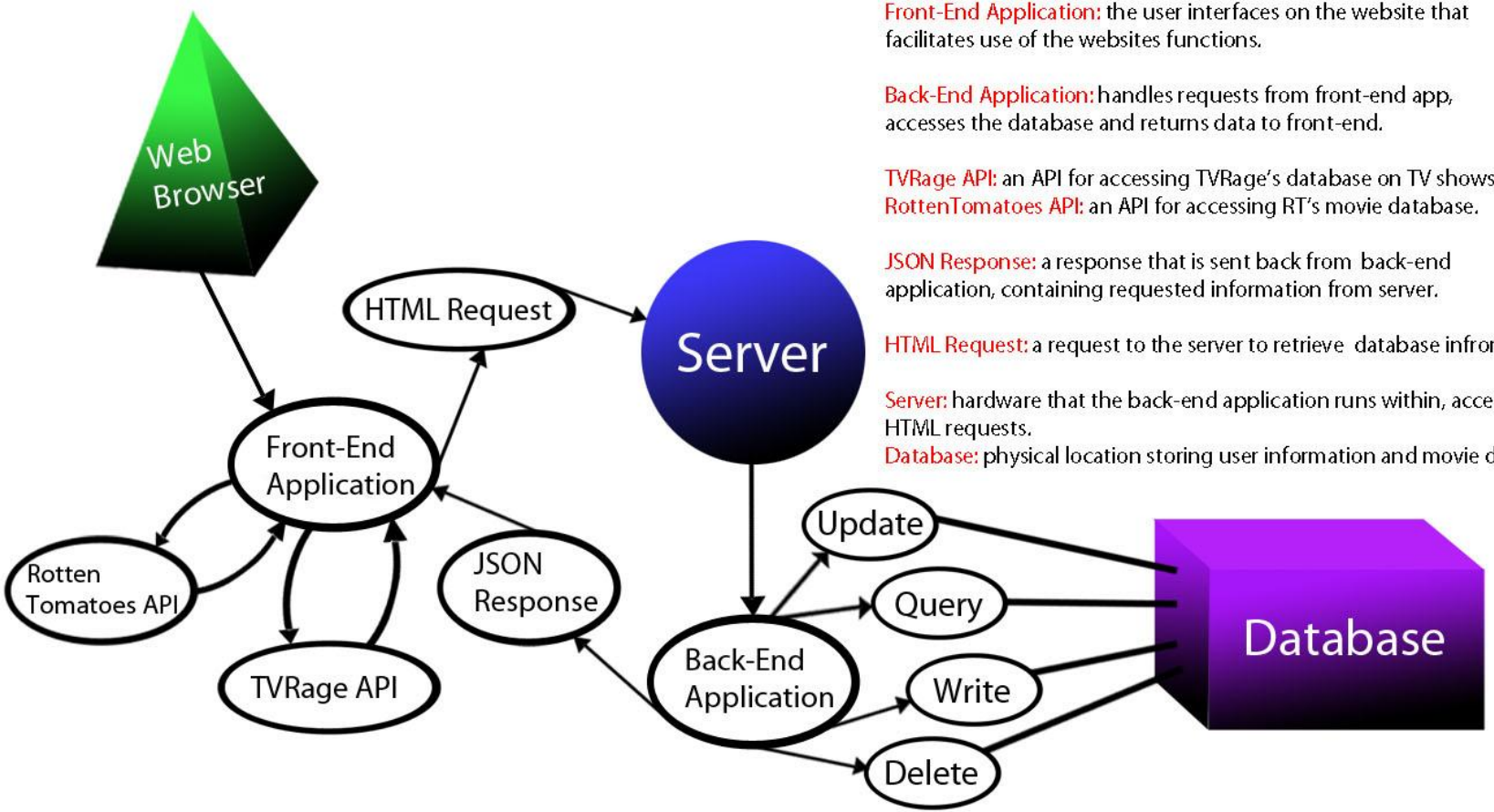
Interface Requirements

- The interface must have a guest user home page which will allow the user to login or create an account.
- Once a user logs in they should be brought to a registered user home page that will show them recommended movies based on the movies the user's library.
- The registered user home page should also have a link to the user's library and account settings.
- A screen for searching for movies must be available to both guest users and to registered users. The input for the search should be a string describing an actor, movie title, or movie genre.
- A button to return the user to the home page should be available at any time when navigating the web application.

System Architecture & Design Issues

Ramses Mederos

High-Level Architecture



Web Browser: any broswer on a device which has internet access.

Front-End Application: the user interfaces on the website that facilitates use of the websites functions.

Back-End Application: handles requests from front-end app, accesses the database and returns data to front-end.

TVRage API: an API for accessing TVRage's database on TV shows.
RottenTomatoes API: an API for accessing RT's movie database.

JSON Response: a response that is sent back from back-end application, containing requested information from server.

HTML Request: a request to the server to retrieve database infromation.

Server: hardware that the back-end application runs within, accepting HTML requests.
Database: physical location storing user information and movie data.

Maintainability

- Agile Method facilitates very rapid maintainability.
  - (Dev. team will be quick and responsive in fixing issues)
- The product will be tested after every sprint of development.
- Constantly checking product functionality ensures that no previous features have broken.
  - This method helps to make each version of the product more functional than the last.

Performance

- Speed and responsiveness of the website have to be carefully considered.

These factors very heavily influence the user experience, and we intend to facilitate a pleasant website interface and speedy loading times.

Users will be able to access all account functionality through a streamlined front-end design in which they never have to worry how back end systems are handling their profile and preferences.

Scalability

- To ensure scalability among different devices and browsers:

  Use a **CSS** framework that automatically scales the website based on screen resolution.

This helps to create a more pleasant overall experience for all users regardless of which computer or mobile device is being used to view the website.

Possible Technical Difficulties

- Improper user authentication (users can’t log in).
  - Could indicate a problem with username/password checking system.
  - Data storing method or password hashing may need to be rechecked.
- Failure to save (or incorrectly saving) user data.
  - Database could not be saving properly, major problem for user accounts
  - All data linking commands would have to be rechecked.
- Slow response times from server.
  - Could occur with overwhelming user requests (high latency or server crash)
  - May require a queue to handle requests (First In First Out)

Testing

Benjamin Kirksey

Testing Structure

- Structure consists of black box testing and white box testing.
  - Black box testing will be performed by several volunteers.
  - White box testing will be performed by the quality assurance specialist.

Volume Testing

- Volume testing will be difficult
  - We have several people to perform testing, but coordinating efforts may be difficult.
- Current server should be able to host 10 simultaneous users.

Human Factors Testing

- None of the black-box testers will initially be given instructions.
  - Each will be asked to give explicit feedback on the UI, as well as information on time to learn the system.
- No feature should require more than one minute to figure out.

Compatibility Testing

- Each tester will be using a unique set of hardware and software, such as OSs and browsers.
  - Each tester will be asked to perform full testing from multiple browsers from a given list.
- Each feature should work properly regardless of OS or browser used.

Documentation Testing

- Each tester will be asked to test each documented feature at least once.
  - This goes hand-in-hand with compatibility testing, as well as human factors testing.
- All documented features should work properly for all testers.

Code Review

- White-box testers will be asked to review the code and propose changes to reduce dependencies and improve performance.
- Code should be highly readable and have minimal dependencies.

Conclusion

Q&A