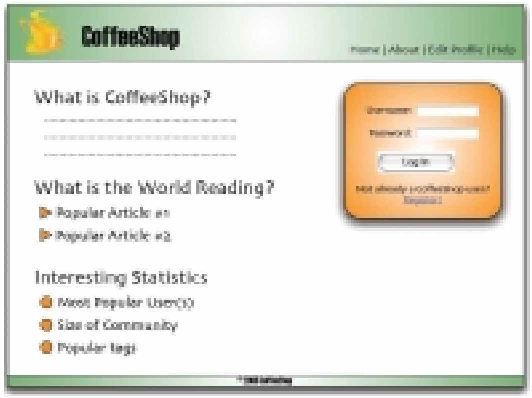
CoffeeShop Overview

Lifecycle Architecture

Features: What can you do with it?

- Read RSS feeds via our web portal
- Search and subscribe to feeds, then use tags to organize them.
- Express your opinions by commenting on articles for the world to see.
- Become somebody's fan – subscribe to the articles they find interesting.
- Find and view users with similar interests
- Rate the articles you've read
- Track article/feed popularity

Design: What will it look like?

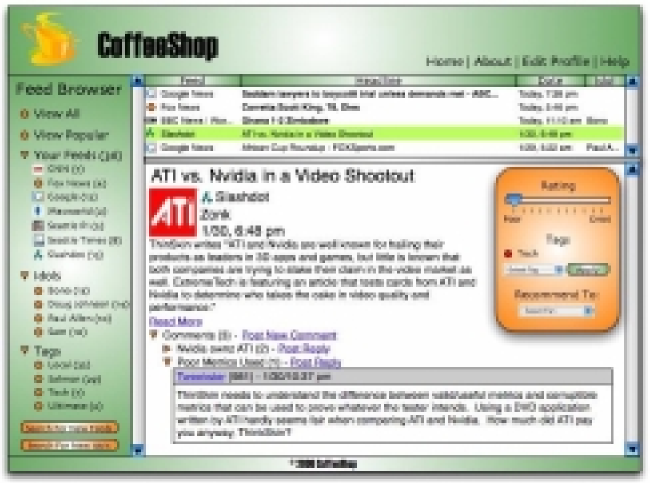


CoffeeShop
Home | About | Edit Profile | Help
Username:
Password:
Log in
Not already a coffeeshop user?
Register!

What is CoffeeShop?

What is the World Reading?
- Popular Article #1
- Popular Article #2

Interesting Statistics
- Most Popular User(s)
- Size of Community
- Popular tags



ATI vs. Nvidia in a Video Shootout

Slashdot

Zonk

1/30, 6:48 pm

[illegible]

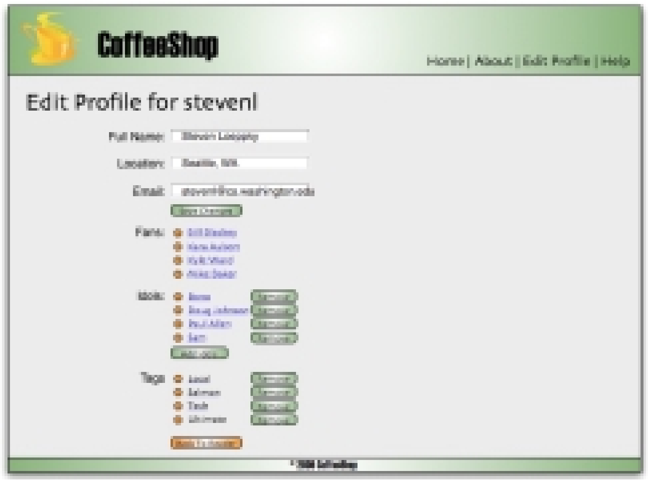
Read More

Comments (3) - Post New Comment

Nvidia owns ATI (2) - Post Reply

Poor Metrics Used (1) - Post Reply

[illegible]



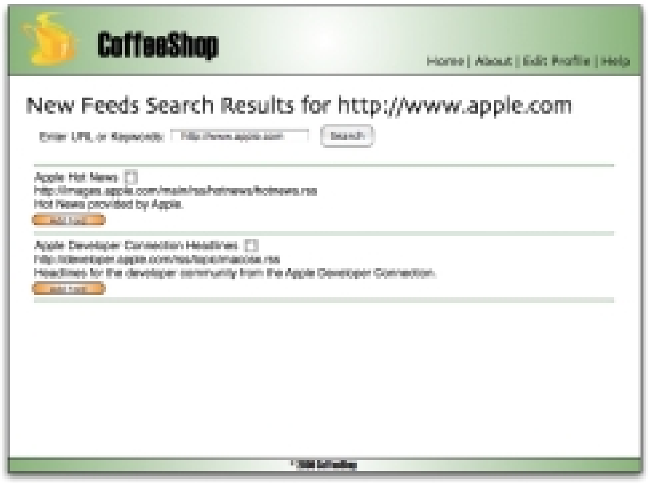
Edit Profile for stevenl

Full Name: Steven Loeppky
Location: Seattle, WA
Email: stevenl@cs.washington.edu

Fans:
- Bill Bradley
- Kara Aulsen
- Kyle Ward
- Riley Baker

Idols:
- Bono
- Doug Johnson
- Paul Allen
- Sam

Tags:
- Local
- Salmon
- Tech
- Ultimate

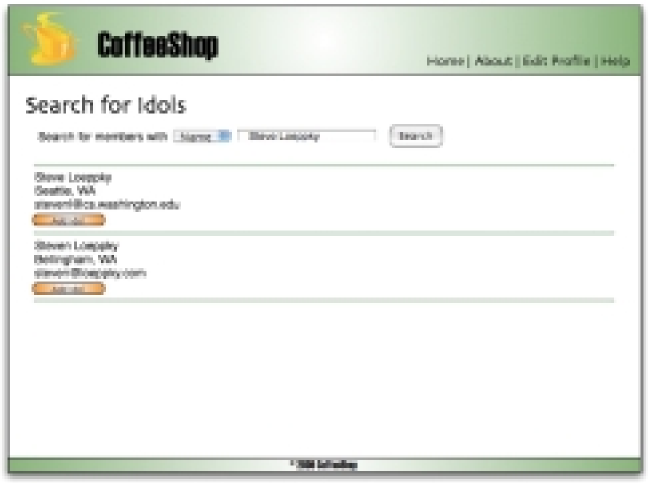


New Feeds Search Results for http://www.apple.com

Enter URL or Keywords: http://www.apple.com

Apple Hot News
http://images.apple.com/main/rss/hotnews/hotnews.rss
Hot News provided by Apple.

Apple Developer Connection Headlines
http://developer.apple.com/rss/topic/macosx.rss
Headlines for the developer community from the Apple Developer Connection.



Search for Idols

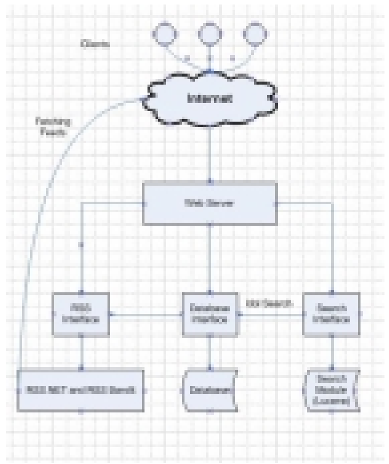
Search for members with: Name | Steve Loeppky

Steve Loeppky
Seattle, WA
stevenl@cs.washington.edu

Steven Loeppky
Bellingham, WA
steven@loeppky.com

Architecture: How?

- 3 Main Interfaces
  - Database
  - RSS
  - Search (DB)
- Classes/Objects
  - User
  - Feed
  - Article
  - Comment



Architecture

- Users
  - Objects containing pertinent user info, such as login, email, idols, subscriptions
- Feeds
  - Objects containing feed data, such as articles, URL, ratings, tags
- Articles
  - Similar to feed, contains source URL, associated feed, comments
- Comments
  - Object containing comments about articles. Contain an associated username, article, and feed.

Architecture

- Database Interface
  - Main interface that communicates with the webserver
  - Contains methods for adding/updating object info (login info, feeds, idols, etc.)
  - Communicates with RSS Bandit/Lucene for fetching/parsing feeds and searching for items in the database
  - Indexed DB for quick results

Architecture

- RSS Interface
  - Interface which facilitates fetching/parsing feeds
  - Web Server will query the DB for a feed
    - If present, DB will return it
    - If not, will use RSS Bandit to fetch the feed from the provided URL
  - DB will return results to the webserver

Architecture

- Search Interface
  - Uses the “Lucene” tool, which will allow for an indexed search of the database.
  - Will take a request from the webserver and use Lucene to fetch the data, returning it to the webserver
    - If a feed is not present, this is where the DB will make use of RSS bandit to get the feed, then return the search results

Architecture

- Class Properties
  - User
    - Subscriptions
    - Name
    - Email
    - Username
    - Idols
    - Fans
  - Feed
    - Tags
    - GlobalTags
    - Articles
    - Name
    - Description
    - URL

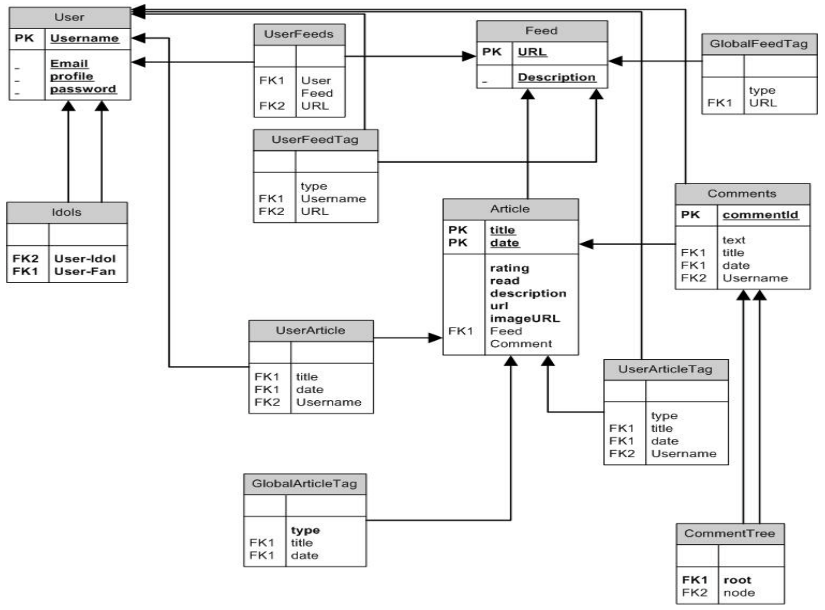
Architecture

- Class Properties (cont.)
  - Article

Read	Read by user
Rating	Rating given by user
Feed	Associated Feed
Tags	Unique user-given tags
Comments	Comments about article
Description	Desription of article
Title	Title of article
Date	Date written (filled in by RSS channel)
Copyright	Copyright information for use by the channel
PubDate	Date published

  - Comments

Article	Article belonging to
Comments	The comments in response to comment
Text	The text of the comments
User	User associated with the comments



Architecture

- High Risk/Problem Areas
  - Lucene/RSS Bandit Interfaces
    - Not defined by us, prewritten code
    - Design depends on functionality of modules and ability to adapt
  - Database Design
    - Too complex of design can result in too much wasted time trying to integrate all the tables and data
    - Simplistic design which will not encompass all related data will help save time while still providing basic functionality

Schedule and Task Assignments: Who and when?

- Milestones
- Tasks
- Schedule

Milestones

- **February 16** – Beta 1 Release
  - Create a working RSS reader. Should be able to:
    - Create a new account
    - Log in
    - Edit their user profile
    - Add a feed
    - Remove a feed
    - Read unread articles
    - Search for feeds
    - Tag feeds
    - Tag articles

Milestones

- **March 1** – Beta 2 Release
  - Finish community features. Users should be able to:
    - Comment on articles
    - Rate articles
    - Search for users
    - Add idols
    - View fans
    - View idols' interesting articles

Milestones

- **March 7** – Final Release
  - Bug fixes. Users should be able to:
    - Click on anything without breaking it
    - No unexpected behavior
    - See friendly error messages
  - Maybe Version 2 features. Users might be able to:
    - View suggested articles
    - View an article's popularity
    - View a feed's popularity
    - Be introduced to users with similar interests
    - Be notified of new articles containing a particular keyword
    - View a graph of a particular keyword's frequency in articles over time.

Schedule

- **Wednesday, February 1**
  - Administrative tasks completed.
    - Garrett will finish these administrative tasks.
  - LCA completed.
    - The team will work together on this.

Schedule

- **Monday, February 6**
  - Training completed.
    - Every member of the team should go through the training necessary to be fluent in ASP.NET and our development environment.

Schedule

- **Friday, February 10**
  - Skeleton code and unit tests for Beta 1 features completed.
    - The classes listed in the CoffeeShop Architecture document should be created and added to source control with method and property stubs, comment, and unit tests.

Schedule

- **Thursday, February 16**
  - Beta 1 Release completed.
    - Each member of the team will be assigned a component/module of the architecture and will be responsible for implementing its features.

Schedule

- **Monday, February 20**
  - <u>Skeleton code and unit tests for Beta 2 features completed.</u>
  - <u>Testing and usability studies.</u>
    - Identify the bugs and usability flaws found in the first round of testing on the Beta 1 build.

Schedule

- **Thursday, March 2**
  - <u>Beta 2 Release completed.</u>
    - Bugs from the Beta 1 testing cycle should all be fixed. All features from the Beta 2 specification should be completed.

Schedule

- **Friday, March 3**
  - <u>Testing and usability studies.</u>
    - Identify bugs and usability flaws found in the second round of testing right away so that we can begin bug fixing over the weekend.

Schedule

- **Tuesday, March 7**
  - <u>Final Release completed.</u>
    - All bugs fixed (yeah right!).

Feasibility Rationale: why it will work

- RSS readers are a proven entity: the fundamentals of our project exist in one form or another, however not in the unified manner that we are proposing.
- The components of this system are designed for high modularity making for easy replacement of parts and future add-ons.
- Version 1.0 components have been narrowed down to include only those that will result in a highly robust system.

Feasibility Rationale: why people want it

- Our system will provide a highly intuitive user interface. Included in this is a highly polished feature set that will allow the user to maximize the information gained.
- RSS is already highly popular and its user base continues to expand.
- Benefits over other readers:
  - Easier to find interesting and relevant articles:
    - Article/feed suggestions
    - Tagging
    - “Interesting” articles distributed to fans
  - Leverages community participation
    - See what others have to say (comments)
    - Organize articles by popularity (community buzz).