**Project: FEEDi**

**DATENBANKEN UND WEBTECHNIKEN**

Project Task – Summer Semester 2019

**Author: Shovra Das**

Matriculation # 550659

Program: Master Web Engineering

Date: 23/06/2019

Technische Universität Chemnitz

Abstract

[The abstract should be one paragraph of between 150 and 250 words. It is not indented. Section titles, such as the word Abstract above, are not considered headings so they don’t use bold heading format. Instead, use the Section Title style. This style automatically starts your section on a new page, so you don’t have to add page breaks. Note that all of the styles for this template are available on the Home tab of the ribbon, in the Styles gallery.]

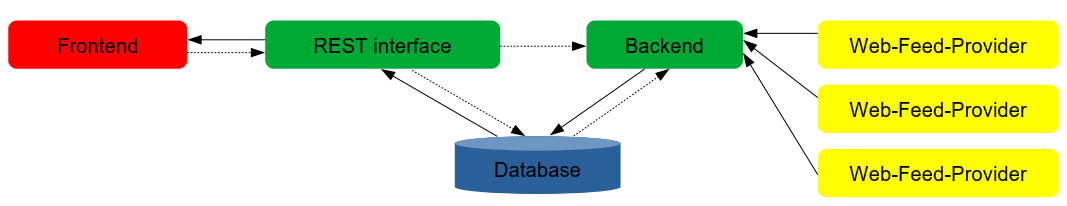
Keywords: rss; atom; web-feed; aggregator

Project: FEEDi

Websites may offer their content in addition to the actual HTML pages as web feeds. Instead of the full content, usually, excerpts are offered and link to the actual HTML pages. Often these web feeds also do not contain the information on all available content, but merely link to a certain number of the latest posts. Users can subscribe to and aggregate these web feeds from multiple providers to access all content through a common interface. The aim of **FEEDi** is to aggregate various web feeds and presents them in a common interface.

# Project Description

This project is best described by the figure below. This is a distributed web application where each tier has their own responsibilities. Each of these layers are listed in below subsections.



**Figure-1**: Application Architecture

## Backend

The backend aggregates and processes the data of the various web feeds. It can handle multiple standards of web feeds. It prevents the updating of web feeds in a too small-time interval, so that it waits at least 10 minutes between two requests for a web feed. Hence it blocks other requests to do so. It ensures a consistent storage of the data in a database. Existing records can also be updated, but no duplicates are created. Records that have reached a certain age are deleted and no longer inserted. By default, this age is 30 days.

## Database

The database stores the aggregated and processed data of the web feeds. It also stores the configuration. The frontend never communicates directly with the database.

## REST based interface

The REST based interface provides communication between the frontend and the database. It delivers the data from the database to the frontend. It also stores configuration settings in the database. It can initiate an update of the web feeds by the backend.

## Frontend

The information of all web feeds can be displayed. The display of the contents of individual providers, i.e. web feed links, can be enabled or disabled.

# Used Technologies

Several technology stacks have been utilized to accomplish the overall project task. They are categorized below.

## Front-end

The front-end layer of this project has utilized several technologies such as jQuery, Bootstrap, Font Awesome, & AngularJS etc. which are briefly described below.

### jQuery

jQuery is the most widely deployed JavaScript library designed to simplify HTML DOM tree traversal and manipulation, as well as event handling, CSS animation, and Ajax [1]. This component is used as the dependency of Bootstrap and Font Awesome module. The version 3.4.1 is used for the application.

### Bootstrap

Bootstrap is a free and open-source CSS framework directed at responsive, mobile-first front-end web development [2]. This component is used to simplify the user interface development and to facilitate an appealing look to the application. The version 4.0 is used in the application

### Font Awesome

Font Awesome is a font and icon toolkit based on CSS and LESS [3]. This tool is used to add some visual smoothness to the application. Font Awesome version 5 is used in this project.

### AngularJS

AngularJS is a JavaScript-based open-source front-end web framework mainly maintained by Google and by a community of individuals and corporations to address many of the challenges encountered in developing single-page applications [4]. This is the heart of this project. All the frontend user interface is managed by this component. To make the routing simpler and more robust a third-party sub-module ui-router has been attached to AngularJS. Here version 1.7.x is used.

#### ui-router

Angular UI-Router is a client-side Single Page Application routing framework for AngularJS. UI-Router applications are modeled as a hierarchical tree of states. UI-Router provides a state machine to manage the transitions between those application states in a transaction-like manner [5]. This module is used to simplify frontend routing of the application.

## Back-end

### NodeJS

Node.js is an open-source, cross-platform JavaScript run-time environment that executes JavaScript code outside of a browser. It is composed of several modules. The listed modules below have been used to implement the backend layer of the application.

#### Express

Express is a web application framework designed for building web applications and APIs. The backend layer of this project is totally dependent of express module. In this project Express is used as the handler of all the request coming from the front end.

#### bcrypt

The bcrypt library on NPM has been used to hash and compare passwords in the application.

#### body-parser

body-parser has been used to extract the entire body portion of an incoming request stream.

#### feedparser-promised

feedparser-promised has been used to retrieve and parse the web feeds from the internet. It can parse almost all the common web feed formats available in the web.

#### jsonwebtoken

jsonwebtoken (JWT) provide a stateless solution for authentication and stateless applications. As this project exposes REST interfaces where traditional session/cookie pair does not make sense, JWT has been used to facilitate some state management features.

#### mongoose

Mongoose is an object data modeling (ODM) library and it has been used in this project to manage the MongoDB database layer.

#### morgan

This module has been used for logging purpose.

#### nodemon

nodemon is a development dependency for the project which provides a virtual development and deployment environment. This has been used as the alternative of the hosting environment while developing the project. Some environment variables are also added to simulate the hosting environment.

### REST Interface

The backend layer of this project exposes its services via a REST (**Re**presentational **S**tate **T**ransfer) based interface.

GET – Used to retrieve a resource

POST – Used to create a resource

PATCH – Used to update an existing resource

DELETE – Used to remove a resource

Further detail can be found at the Appendix section

## Database

For storing and manipulating the data generated by the application MongoDB has been used. Since this application deals with semi structured data therefore MongoDB is thought to be the better choice.

# Project Functionalities

Sdsdds

## Frontend

List of features

|  |  |  |
| --- | --- | --- |
| **Feature** | **Description** | **Corresponding figure** |
| View all entries |  | Figure - 2 |
| View Specific Provider’s entries |  | Figure - 2 |
| View Favorite feeds |  | Figure - 2 |
| Filter Read/Unread feeds |  | Figure - 2 |
| Mark an entry as Favorite |  | Figure - 2 |
| Mark an entry as Read/Unread |  | Figure - 2 |
| Sort feeds by published date |  | Figure - 2 |
| Add new feed |  | Figure - 3 |
| View Providers |  | Figure - 4 |
| Delete Provider with all entries |  | Figure - 4 |
| Update individual Provider |  | Figure - 4 |
| Update all Providers |  | Figure - 4 |
| User Signup |  |  |
| User Login |  |  |
| View Profile |  |  |
| Change Password |  |  |

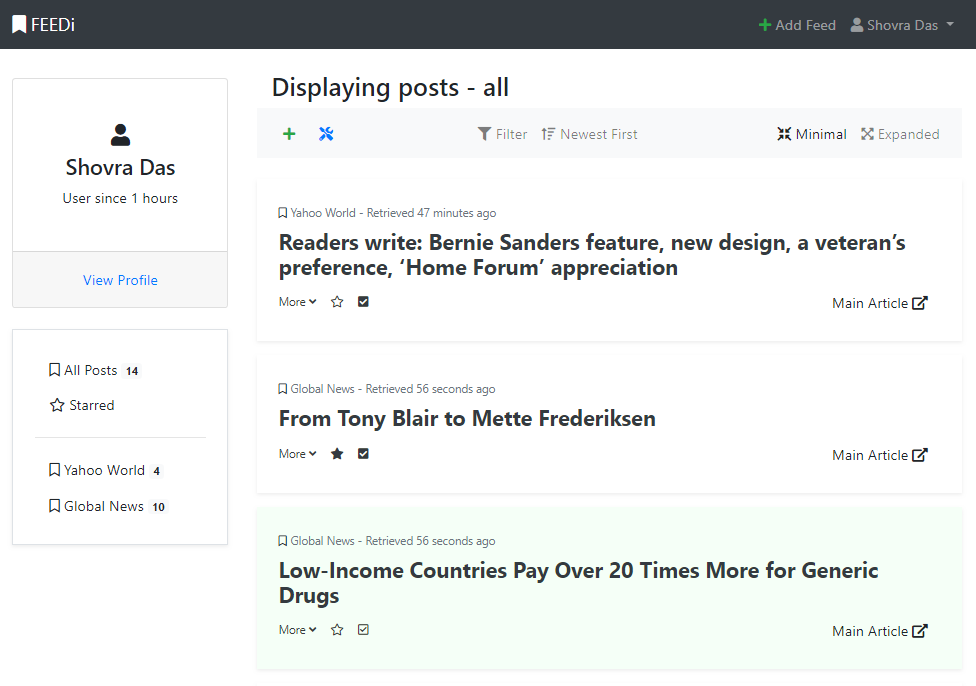


Figure-2: User Home Page

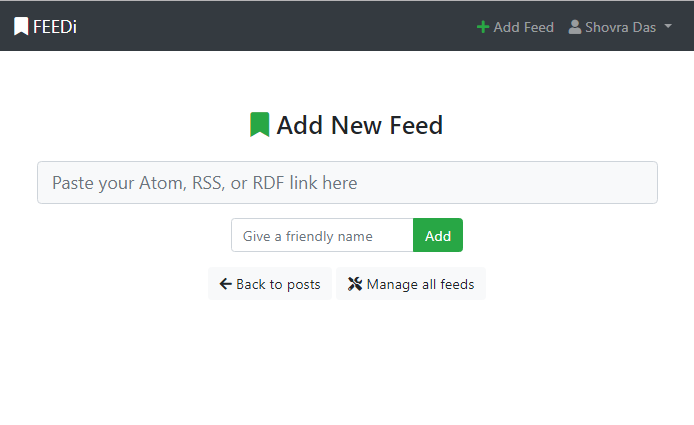


Figure-3: Subscribe for a new feed

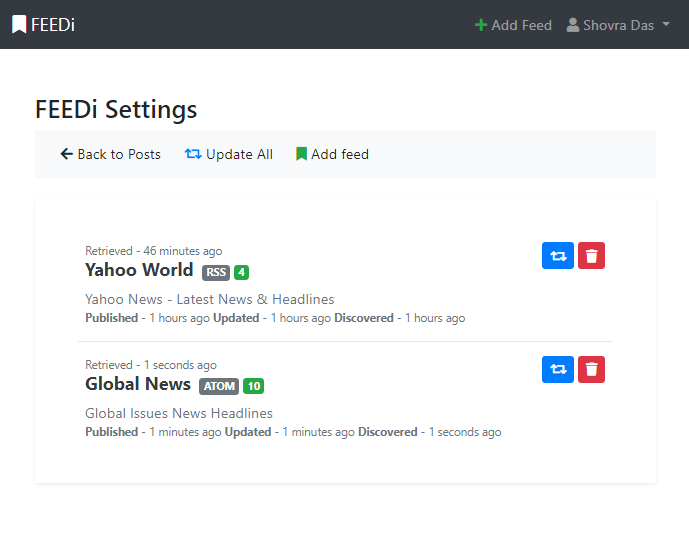


Figure-4: Managing feeds

# Project Challenges

Dfdfdsfdffd

# Conclusion & Future Scope

References

[1] JQuery. (2019, May 22). Retrieved from <https://en.wikipedia.org/wiki/JQuery>

[2] Bootstrap (front-end framework). (2019, May 21). Retrieved from <https://en.wikipedia.org/wiki/Bootstrap_(front-end_framework)>

[3] Font Awesome. (2019, June 21). Retrieved from <https://en.wikipedia.org/wiki/Font_Awesome>

[4] AngularJS. (2019, June 19). Retrieved from <https://en.wikipedia.org/wiki/AngularJS>

[5] Angular-Ui. (2019, June 15). Angular-ui/ui-router. Retrieved from https://github.com/angular-ui/ui-router

Appendix

1[Add footnotes, if any, on their own page following references. For APA formatting requirements, it’s easy to just type your own footnote references and notes. To format a footnote reference, select the number and then, on the Home tab, in the Styles gallery, click Footnote Reference. The body of a footnote, such as this example, uses the Normal text style. (Note: If you delete this sample footnote, don’t forget to delete its in-text reference as well. That’s at the end of the sample Heading 2 paragraph on the first page of body content in this template.)]