Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

Advanced Related-Post Recommendations (ARP Rec) WordPress Add-On

Group 8 - Requirements Document

Brandon Dooley
Matthew Flynn
Eoin Dowling
Kevin Fallon
Joey Curran
Ruairi Gielty

Introduction

I. Overview – Purpose of System

The purpose of our system is to provide an add-on for WordPress blogs that give “related blog-post” recommendations. The add-on will use an innovative Recommendations-as-a-Service API system which adapts its recommendation approach on each request to provide the most relevant recommendations. This is done using machine learning to select the best recommendation approach for the given request and applying that approach to generate and return high quality recommendations.

Our task is to develop the WordPress add-on which makes use of this existing API functionality. This add-on will display the recommendations received from our client’s API in a graphical and easy-to-use manner to users of a given WordPress site.

II. Scope

The scope of this project is to encapsulate the current existing Darwin & Goliath API into a WordPress add-on which graphically renders recommended blog posts from the API’s response. It should be possible for the add-on to be seamlessly integrated into all existing WordPress blogs. The add-on should be ported with a proxy for every blog to protect users’ information with regards to requests.

III. Objective and Success Criteria

Phase 1 – Market Research

- Investigate current competitors and record their strengths and weaknesses.
- Decide upon any additional features we intend to add to give an advantage.
- Update client’s requirement specifications with any necessary changes.

Phase 2 – Proof of Concept

- Set up a test WordPress blog on a dummy server with fake posts for testing.
- Implement a basic add-on which makes requests to client’s API using a simple JavaScript client.
- Request recommendations from client’s API, parse result JSON and graphically render the results in an appropriate manner.
- Confirm recommendations displayed accordingly to API.

Phase 3 – Simple Prototype

- Expand upon the POC add-on developed during phase 2 and extend this to a fully working add-on that can work and be integrated seamlessly with any WordPress site.
- Test this prototype on client’s sites and ensure full working functionality.

Phase 4 – Proxy Server

- Develop a proxy server for all requests to be relayed to. This way all requests hitting the API will come from the proxy and not the user themselves.
- This proxy should be on a per-blog basis.
- This should ensure that all requests to the API are coming from the WordPress installation site’s IP and not the user’s IP for data protection purposes.

IV. Definitions/Abbreviations

- **API:** Application Programming Interface.
- **Proxy Server:** A server that acts as an intermediary for requests from clients seeking resources from other servers.
- **AI:** Artificial Intelligence.
- **RESTful API:** An API that uses HTTP requests to GET, PUT, POST and DELETE data.
- **CMS:** Content Management System.
- **JSON:** JavaScript Object Notation (JSON) is an open-standard file format that uses human-readable text to transmit data objects consisting of attribute–value pairs and array data types (or any other serializable value). It is a very common data format used for asynchronous browser–server communication.
- **Recommendations-as-a-Service:** This is a subclass of information filtering system that seeks to predict the "rating" or "preference" a user would give to an item.

V. References

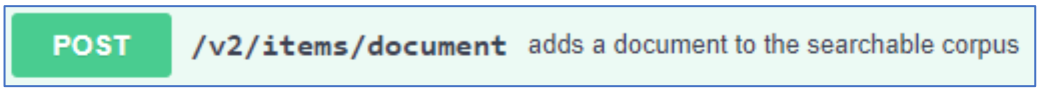
- https://darwingoliath.com/
- https://wordpress.com/
- https://codex.wordpress.org/Plugin_API
- https://en.wikipedia.org/wiki/Recommender_system

Current System

Darwin & Goliath exists as a Recommendation Framework API incorporating the “Recommendations as a Service” paradigm. It consists of three services; Features DarwinAI Meta-Recommendation Framework, a Micro-optimization Algorithm and Embedded Deep Learning Technology.

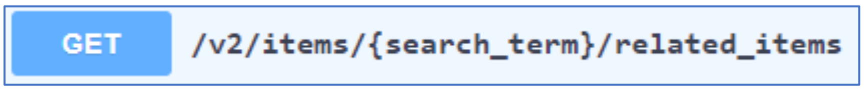
The Darwin&Goliath API is accessible through the RESTful API. The system relies on a basic set of POST and GET methods to upload content and return recommendations.

The current system allows client content to be uploaded through the following POST request:



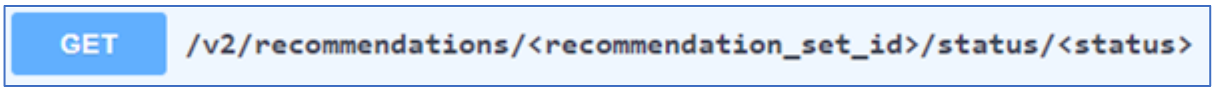
POST **/v2/items/document** adds a document to the searchable corpus

Two GET requests enable clients to search for related content and confirm the relevant recommendations have been displayed:



GET **/v2/items/{search_term}/related_items**

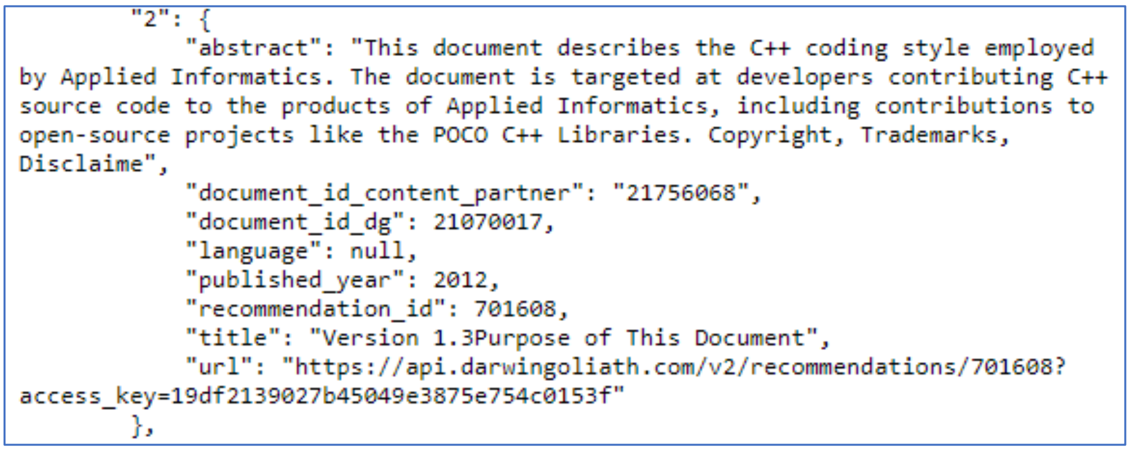
- Searches the corpus for related documents



GET **/v2/recommendations/<recommendation_set_id>/status/<status>**

- Confirms the recommendations have been displayed

The recommendations are currently returned in JSON format ready to be displayed:



```
        "2": {
            "abstract": "This document describes the C++ coding style employed
by Applied Informatics. The document is targeted at developers contributing C++
source code to the products of Applied Informatics, including contributions to
open-source projects like the POCO C++ Libraries. Copyright, Trademarks,
Disclaime",
            "document_id_content_partner": "21756068",
            "document_id_dg": 21070017,
            "language": null,
            "published_year": 2012,
            "recommendation_id": 701608,
            "title": "Version 1.3Purpose of This Document",
            "url": "https://api.darwingoliath.com/v2/recommendations/701608?
access_key=19df2139027b45049e3875e754c0153f"
        },
```

Proposed System

I. Overview

The proposed system is a WordPress add on, which will be able to search webpages/blogs for related/recommended articles, which it will display in a clean and clear way, displaying the article title and any image related to the article using the Darwin&Goliath software.

II. Functional Requirements

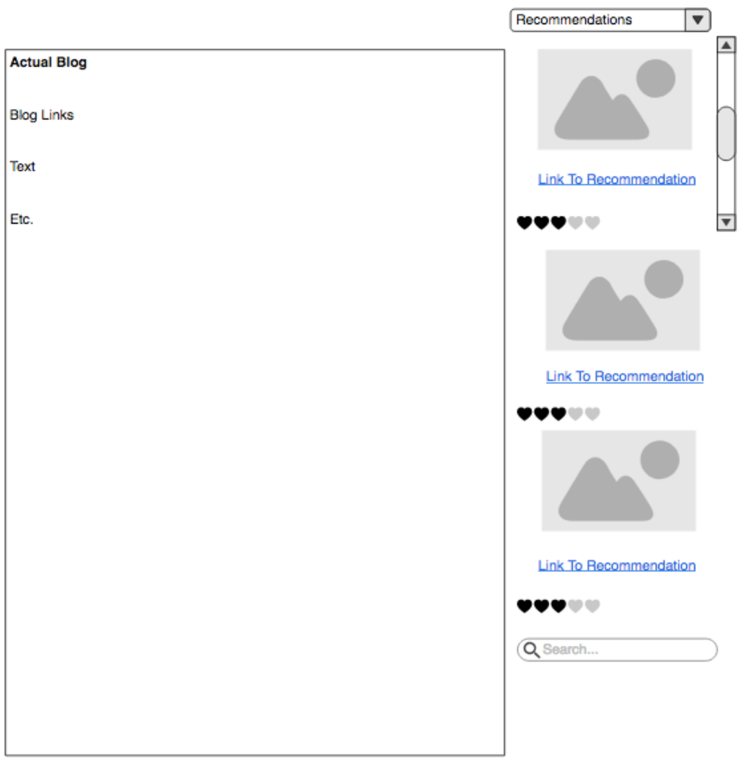
- Interact with the Darwin&Goliath software to request recommendations and receive results
- Display the results in a clean and clear way, using images related to the article when available.
- Creating a proxy server when interacting so as not to receive user IP address without consent.
- Being usable for WordPress software.

III. Non-functional Requirements

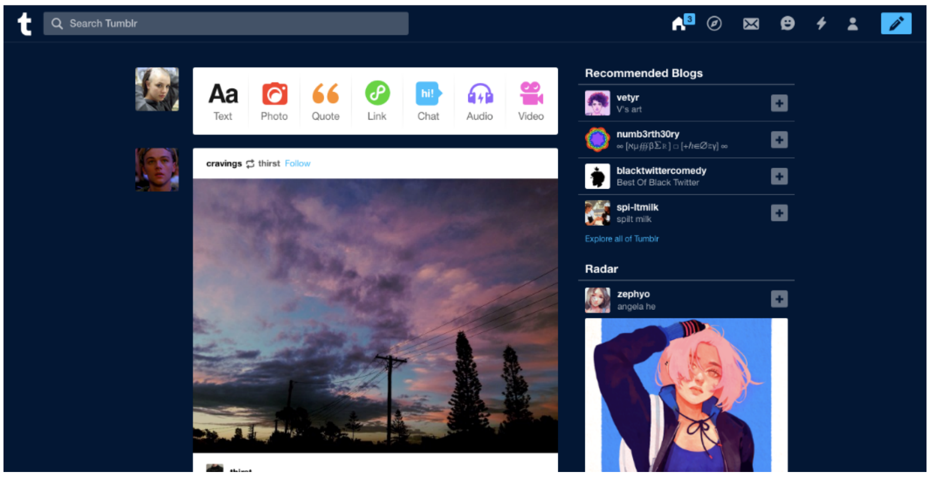
- Display article rating if desired/exists.
- Criteria that caused the article to display.
- Ability to see different recommendations based on related criteria.
- Ability for webmaster to decide position of recommendations.

IV. System Prototype (Models)

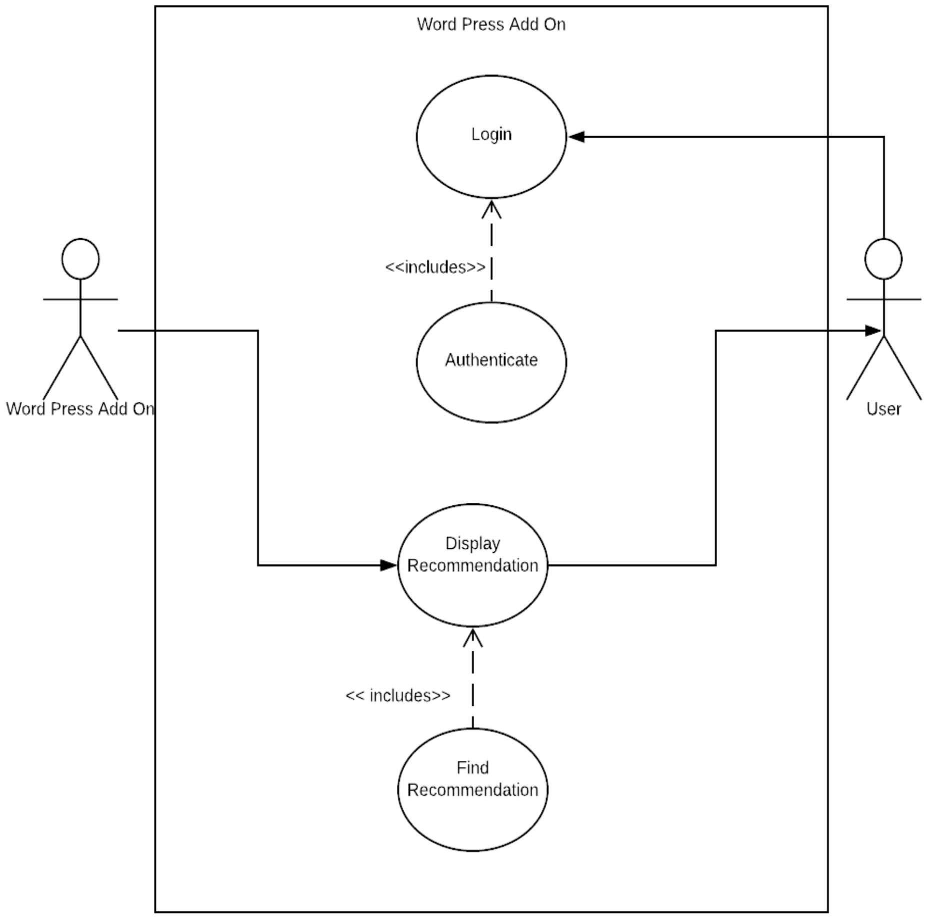
1. User Interface Mock-up:



Based on Tumblr’s Recommendations Application:



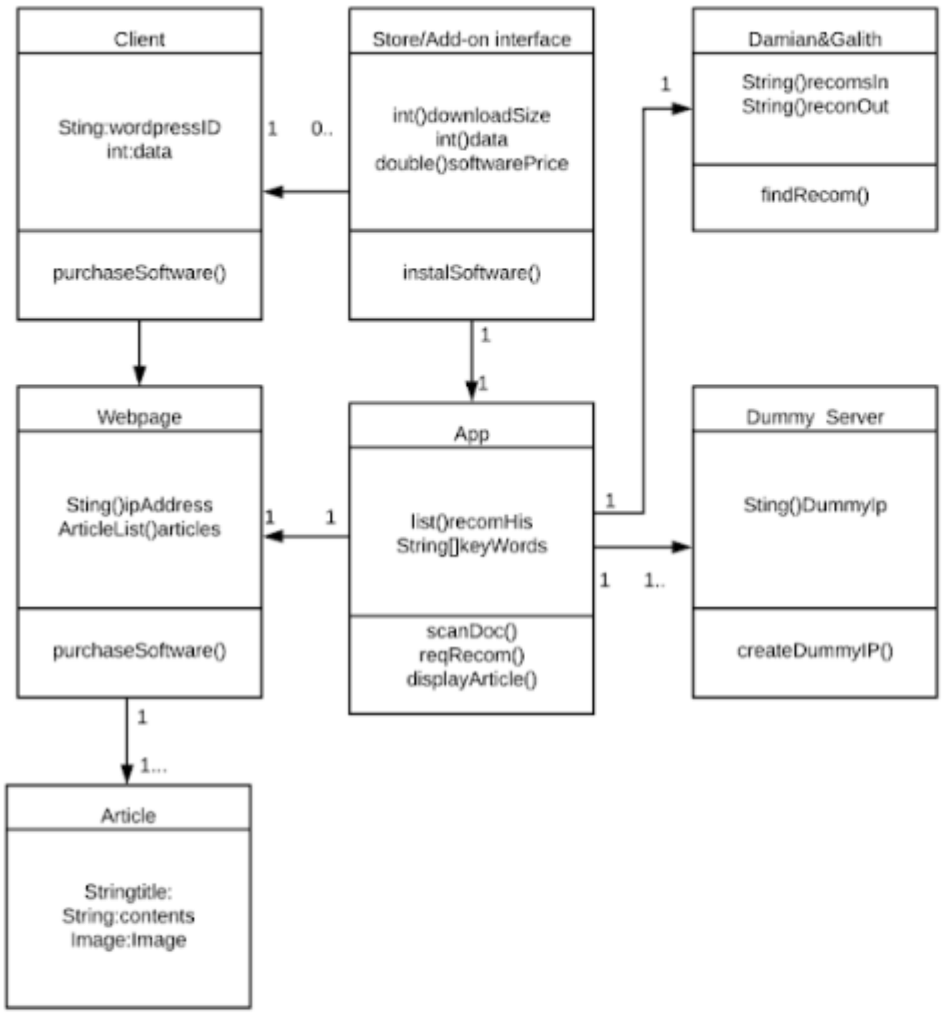
2. Use Cases (Including Text Narratives):



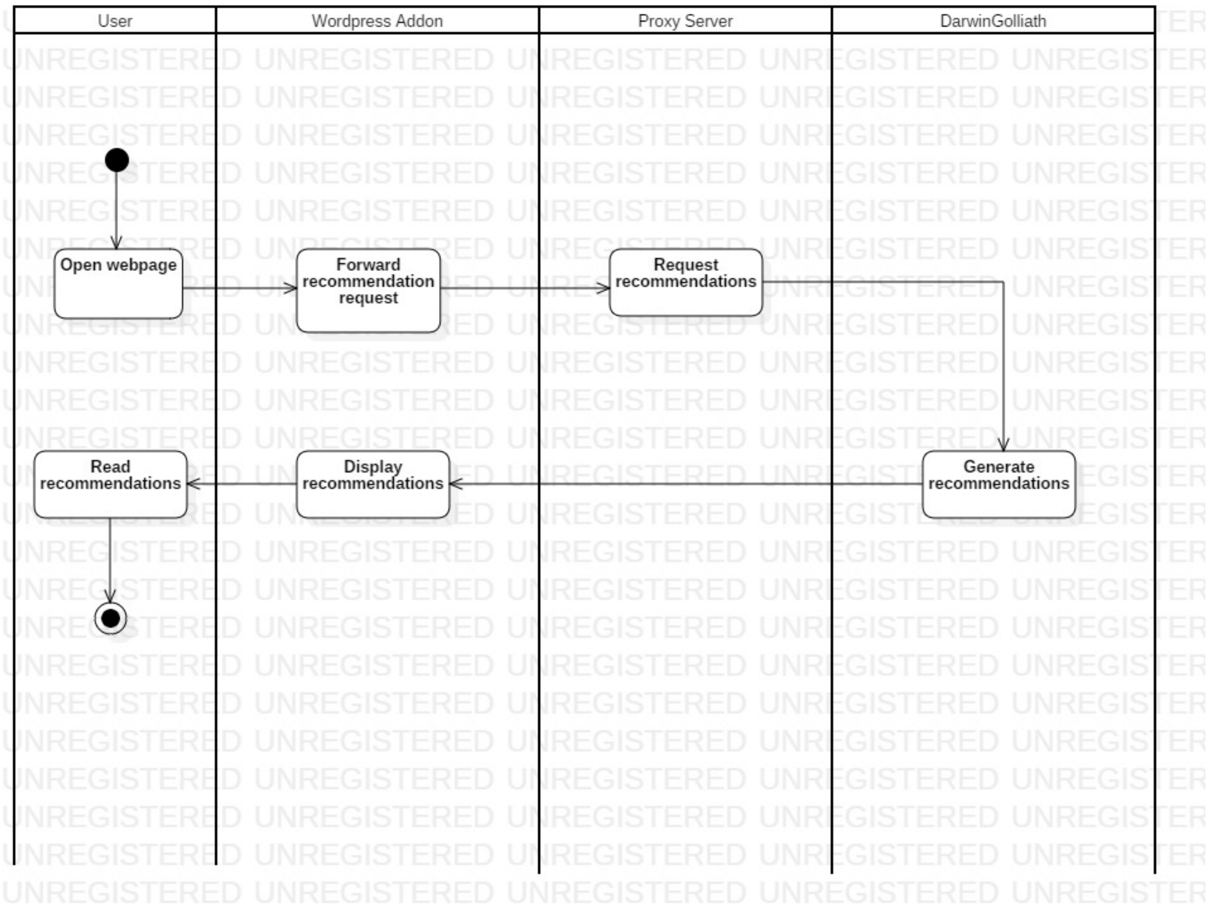
Use Case Name	Login
Primary Actors	WordPress Add On
Secondary Actors	User
Preconditions	User on WordPress login page
Normal Flow	• A user enters their credentials for logging on to their WordPress account • WordPress Authenticates the user’s information.
Post Conditions	User is logged into their WordPress account.
Alternative Flows/ Exceptions	• User enters a wrong password • User exits web page before authentication is complete

Use Case Name	Display Recommendations
Primary Actors	WordPress Add On
Secondary Actors	User
Preconditions	User is logged into WordPress
Normal Flow	• Add-on sends blogpost metadata (title, excerpt, author, category etc.) to D&G API. • Add-on requests recommended blog posts from the D&G API. • Add-on processes API response of recommended posts and graphically renders the recommendations.
Post Conditions	Recommendations are displayed on user’s WordPress
Alternative Flows/ Exceptions	User is yet to set up a WordPress/has no posts.

3. Object Model:



4. Dynamic Model:



User
Wordpress Addon
Proxy Server
DarwinGolliath
Open webpage
Forward recommendation request
Request recommendations
Generate recommendations
Display recommendations
Read recommendations