Bilkent University
Department of Computer Engineering

Senior Design Project

T2310

ReLink

Detailed Design Report

Ayberk Yaşa, 21801847, ayberk.yasa@ug.bilkent.edu.tr
Cemhan Kaan Özaltan, 21902695, kaan.ozaltan@ug.bilkent.edu.tr
Çağatay Şafak, 21902730, cagatay.safak@ug.bilkent.edu.tr
Fatih Kaplama, 21802755, fatih.kaplama@ug.bilkent.edu.tr
Görkem Ayten, 21802399, gorkem.ayten@ug.bilkent.edu.tr

Eray Tüzün

Erhan Dolak, Tağmaç Topal

13th of March 2023

Table of Contents

1. Introduction

In our age of technology, bugs, improvements, and features of a software project are stored and tracked in project management tools that provide development teams with an issue tracker. The rise of agile results in large amounts of data about issues in these project management tools which require much more effort to inspect manually [1]. Moreover, pull requests are fundamental artifacts used in the software development lifecycle (SDLC) [1], [2]. Whereas issues are stored in bug tracking and project management tools such as Jira, Bugzilla, and GitHub Issues, pull requests and commits are stored in servers utilizing a version control system (VCS) such as GitHub, GitLab, and Bitbucket. Such contemporary VCS options offer built-in features for manually linking issues with commits and pull requests through a predefined commit message syntax where the issue ID is explicitly given, allowing for a more structured and traceable project. However, despite the importance of traceability in software development, most pull requests are not explicitly linked to issues by developers due to the lack of the fixed rules [3]. The information contained in the missing links is therefore lost, reducing efficiency in processes such as bug localization, bug prediction, and feature improvements [3]. Since it is highly possible for developers to forget establishing such links or fail tagging a commit due to a typo and the manual cost of recovering these links is very high, we propose an automation tool for tackling this issue.

The sections of this report are structured as follows: Section 1 introduces the purpose and the overall structure of the system, along with technical definitions. Section 2 explains the existing competitors and a comparative analysis. Section 3 demonstrates the detailed architecture of the proposed software system. Section 4 explains the subsystems that the proposed software system composes. Section 5 explains how the components of the system are tested for minimizing error. Section 6 considers factors in engineering design such as economic constraints. Finally, Section 7 investigates our teamwork structure.

1.1 Purpose of the System

ReLink is a web application which serves as a tool for automating the issue-pull request linking process during software development. To be specific, ReLink detects missing links retrospectively and creates a link between issue-pull request pairs. This is done through certain machine learning (ML) algorithms. ReLink also provides the visualization of the analysis of missed links among pull requests and issues by certain filters such as by developer, by time, and by reason. Finally, ReLink offers a graphical interface for visualizing data on the historical evolution and progress of a specific project’s issues. To be specific, there is a historical graph visualization for links among commits, pull requests, and issues.

1.2 Design Goals

1.2.1. Usability

- There should exist a navigational sidebar visible from all pages for allowing the user to easily navigate the app.
- The titles on the navigation bar, the titles on each page, and the labels on all buttons should be meaningful and self-explanatory so that users who do not read the user manual are able to understand how the website is used from the titles on the navigation bar, the titles on the screen and the labels on the buttons.
- All screens including pop-ups should be reached by being clicked at most twice.
- In order to be a user-friendly website, it is imperative to be consistent in the website layout and design by following a design pattern in which all screens accessed from the sidebar use the same main template.
- Except for the pop-ups, none of the screens on the website must be connected to each other, so users don’t have to go backward on the website.
- Web Content Accessibility Guidelines [4] must be followed, which makes content in the website more accessible to users with disabilities.

1.2.2. Reliability

- Users must be able to access a historical graph visualization for links among commits, pull requests, and issues of their projects 90% of the time without failure.
- The data fetched from repositories must be used without changing.
- Users who close the browser without properly logging out of their account will be automatically logged out.

1.2.3. Performance

- The load time of each page must be less than 2 seconds.
- The average response time of each button must be about 2 seconds.
- The website must keep the above-mentioned times the same, up to 100 simultaneous users.

1.2.4. Supportability

- User feedback will be evaluated continuously and if there is any bug reported by users, it will be assigned to a developer within 24 hours.

1.2.5. Scalability

- When the daily traffic of this website, which has 1000 visitors per day, exceeds this number, the max bandwidth limit of this website that is allocated to the hosting plan should not be exceeded.

1.3 Definitions, Acronyms, and Abbreviations

Commit: A change on a file, or a set of files, in a version control system repository.
Pull Request (PR): The event of merging new code to the code repository in software development.
Issue: Descriptions of needed changes and tasks on a software project that allow tracking progress.
Issue-PR Link: Semantic associations created between the given software artifacts, either manually or by the ReLink system.
Traceability: The capacity to explicitly relate associated software artifacts throughout development [2].

1.4 Overview

1.4.1. User Types

The system has three kinds of users:

- Admin
- Developer
- Project Manager

Admin should be able to manage every user and project on the application, just in case if there occurs an issue. Moreover, only the admin should be able to assign a project manager for each project.
Developer should be able to only view the pages related to a project and not be able to make any modifications.
Project Manager should be able to do everything that a developer can do. Additionally, Project Manager should be able to manage the projects, analyzes, and historical graphs. They, also, should have extra permissions compared to a developer, like adding developers to a project, removing developers from a project and authorizing developers to manage projects like Project Manager.

1.4.2. Sign Up & Sign In

The project will have sign up and sign in functionalities. The users will be able to do these operations with a unique email and password combination associated with their account. These accounts can have different levels of authorization for different kinds of projects. For example, there are different project managers and developers for different projects even in the same company. Project Managers can sign in to the web application with the email and password assigned by ReLink to them. They have to change their password when they sign in to their account for the first time. Developers need to sign up to create a new account. ReLink doesn’t assign an account for them.

1.4.3. Connect Project to ReLink

Project Manager should provide a link to a project that he/she works on. ReLink needs the GitHub or Azure DevOps repository and Github Issues, Azure DevOps, or JIRA links of the project to read, analyze and change pull requests and issue links. If the project is publicly available, only url information is sufficient. Extra credentials may be requested when the repository is private.

1.4.4. Recovering Missing Links in the Past

The users will be able to access a dedicated page where the previous PRs made for a specific project, which are not explicitly linked to an issue by the developer who created them can be observed and suitable links may be created for them. This core functionality of ReLink will be given to the user as a recommender system, where for each unlinked PR, the user will be given a list of options determined with a ML model and will be allowed to choose among the options. The unlinked PRs will be shown as a list, each with their own category, and the user will be able to view further details and make the said decision by clicking on a listed item.

1.4.5. Historical Graph Visualization

Project Managers update the graph visualization between commits, pull requests, and issues by taking a snapshot of the project at a specific time. Developers will be able to view this graph representation of the connections between commits, pull requests, and issues, where the connections were created by an ML algorithm. In addition, this page will include a data table that shows who has linked which commits, pull requests, or issues. Therefore, Project Managers will be able to track the history of the linkage process.

1.4.6. Visualization of Analysis of Missed Links

This feature of the application is to create plots, graphs, and charts of analysis of missed links. Thanks to this feature, the user can inspect the results of analysis in a more concrete way. While showing results to the user, the user can filter the plots, graphs, and charts by developer, time, and reason. Moreover, the user can follow the progression of the status of the missed links in his/her project since repository mining will be performed regularly and missed links will be detected each time.

2. Current Software Architecture

2.1. Competitive Analysis

There exist many articles inspecting the traceability and recovery of issue links from various perspectives [1], [2], [3]. In these studies, tools aiming to detect and recover concurrent and past issue links are proposed only for research purposes and a commercial tool with such features does not yet exist. As such, we did not include the proposals given in these studies in our competitive analysis. Moreover, certain GitHub Actions pipeline commands exist in GitHub marketplace, which enforces traceability by not allowing a pipeline to exist without all PR-commit pairs being linked [5], [6]. Since these are pre-existing tools only aimed toward Actions pipelines, we do not consider them in our analysis.

	ZenHub	**Boring Cyborg**	**Quantify**	**ReLink**
Manual Issue-PR Linking	✅	✅	✅	✅
Manual Issue-Commit Linking	✅	✅	✅	❌
Suggestion for possible links	❌	❌	❌	✅
Warning if link is missing	✅	✅	❌	❌
Detecting past missing links	❌	❌	❌	✅
Visualization of missing links	❌	❌	❌	✅
Visualization of links among commits, PRs, issues	❌	❌	❌	✅
Supported environments	GitHub	Jira GitHub	Jira GitHub	Jira Azure DevOps

2.2. Academic Analysis

In this section, a summary of the academic research conducted on this topic will be provided. Aung et al. explain that the recovery of lost artifact links is an important practice in change impact analysis (CIA) which is a method of change affect assessment during software evolution, and further explains several methods of link recovery using ML methods such as the usage of the random forest algorithm and recurrent neural networks (RNNs) [7]. Rath et al. further emphasize the importance of links between commits and issues for software system traceability, providing several methods for text similarity detection to be used with naive bayes, decision tree, and random forest classifiers. This paper also explains the importance of the model's recall, which measures the percentage of correctly detected true positives, is an important metric while recommending developers with possible issue IDs [2]. Lüders et al. investigate links between issues in a similar way, and introduce the BERT deep learning model which is trained on titles and descriptions of issues [3]. Even though we are currently not planning to include link recovery between issues in our project, the BERT model may prove useful during our implementation since it is used for a similar purpose. Finally, DeepLink is introduced by Ruan et al., which is an RNN implementation for recovering commit-issue links along with the usage of methods like word embedding, which is the most recognized existing academic implementation of such a tool, therefore being paramount for the development of our project [7].

3. Proposed Software Architecture

3.1 Overview

The architecture of ReLink consists of 3 layers in terms of high level architecture: presentation, business, and persistence. The presentation layer is responsible for displaying the interface content of ReLink to the user through dynamic and interactive components implemented with React.js. Moreover, it is responsible for handling user requests and generating the appropriate responses.

The business layer provides the application's core logic and functionality, including business rules, data validation, and processing of user inputs. Here, the user data is processed and stored using models, using the Django library. The presentation layer depends on the endpoints provided by the REST API module. The link recovery module is responsible for determining possible issue-PR pairs of a given software projects using word embedding, metadata classification, and heuristic rules. This package is implemented with libraries such as Scikit-learn and Gensim. The data retrieval module is responsible for fetching pull request, commits and issues data of user's repositories from Jira REST API and Azure DevOps REST API. These APIs are external systems and therefore do not reside in the ReLink system. This data, after being saved in the persistence layer, is used by the link recovery module, the graph builder module, and also the REST API module to provide it to the client.

The persistence layer is responsible for storing user and application data for persistence through different sessions. Here, the Django models are turned into database records and are stored.

3.2 Subsystem Decomposition

ReLink consists of 5 main subsystems, which are shown with the following diagram and are explained in detail in Section 4.

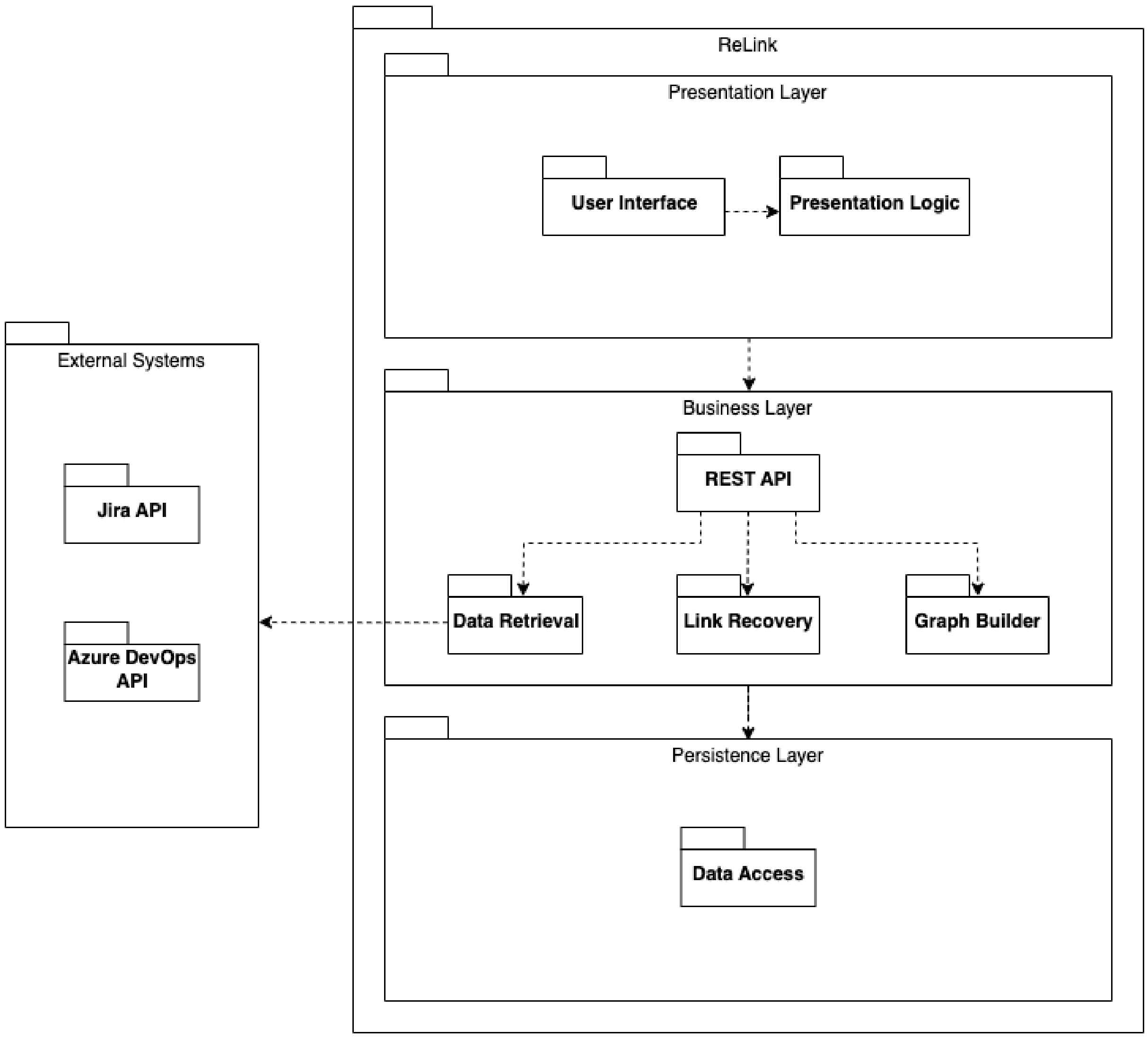


Fig. 1. Package diagram.

3.3 Hardware/Software Mapping

ReLink has two hardware components which are a production server to run successfully on the web browsers and a server to run the ML model. This production server has 2 vCPUs, 2 GB RAM, and 16 GB storage memory and is in Amazon EC2 cloud server. This production server is run on the Linux operating system. Another server for running the ML model has 8 vCPUs, 16 GB RAM, and Amazon EBS storage memory and is in Amazon SageMaker. Moreover, a web browser is required on users' hardware systems such as computers and phones. This web browser can be Google Chrome, Microsoft Edge, Mozilla Firefox, Opera, or Safari. For Google Chrome, the browser version should be 96.0.4664.45 or later. For Microsoft Edge, the browser version should be 96.0.1054.34 or later. For Mozilla Firefox, the browser version should be 94.0.2 or later. For Opera, the browser version should be 81.0.4196.54 or later. For Safari, the browser version should be 15.0 or later. Moreover, the version of Django to be used for the back-end is 4.1.3 and React to be used for the front-end has the version of 18.2.0.

3.4 Persistent Data Management

In this project, a specific database is needed since clients send a request to the server to access the specific information about pull request, issue, repository. These data should be accessed quickly. In this project, PostgreSQL will be used. The reason PostgreSQL is chosen, it is an object relational database that is convenient to apply OOP principles. Moreover, PostgreSQL has various functionalities such as storing array and json as a column. Thanks to PostgreSQL, the project's database will be more functional, speed and safe. ReLink’s main objects such as User, Developer, Project Manager, Project, Repository, Pull Request, Issue, Commit, and Links will be stored in this database that is deployed in our production server by using AWS [9]. Clients can make some operations such as sending GET, PUT, POST and DELETE requests to the server. It is important to provide such operations to client.

3.5 Access Control and Security

Our web application has access control and it provides security. It is provided access control by giving permission to the logged user with the matching used type. In this way, a user cannot use the properties of a user type that they do not belong to. Also, to improve the security of the web application, there will be a good password policy. Users must create a password of at least 8 characters including lowercase, uppercase, special characters and numbers. These improve the security and the access control of the application.

Matrix for Access Control

Actors \ Objects	Project	Repository	PR	Issue	Commit	Link	Analysis
Developer	R	R	R	R	R	R	R
Project Manager	RW	RW	R	R	R	RWX	RX

Note: In the context of our application, X (Execute) means that an actor should be able to start an operation orchestrated by the system on an object. In other words, the actor orders the system to complete the operation instead of directly performing the operation.

3.6. Boundary Conditions

3.6.1 Initialization

The initialization section of the boundary conditions specifies the conditions that must be met for the system to start operating correctly.
Input validation: The system should validate all input data to ensure it is within expected ranges and conforms to the required data types.
Resource allocation: The system should allocate all necessary resources, such as memory or network connections, prior to starting execution.
Configuration loading: The system should load any required configuration files, such as database credentials or API keys.
Data loading: The system should load data from the database, such as project, user, repository mining data.
Data validation: The system should check if access tokens of projects registered by users are expired. The system also should check if URLs of these projects are broken.

3.6.2 Termination

The termination section of the boundary conditions specifies the conditions that must be met for the system to shut down correctly.
Resource deallocation: The system should release all resources, such as memory or network connections, that were allocated during initialization.
Data persistence: The system should save any necessary data to a durable store, such as a database or file system, before shutting down.
Cleanup: The system should perform any necessary cleanup activities, such as cleaning cache, interrupting ongoing processes or terminating network connections.

3.6.1 Failure

The failure section of the boundary conditions specifies the conditions that must be met in the event of a failure or error.
Error handling: The system should gracefully handle any errors or exceptions that occur during execution, such as logging errors and providing appropriate error messages to the user.
Rollback: The system should revert to a known good state in the event of a failure, such as rolling back a transaction or restoring data from a backup.
Notification: The system should alert appropriate parties, such as system administrators or support personnel, in the event of a failure that requires manual intervention.

4. Subsystem services

4.1. Frontend

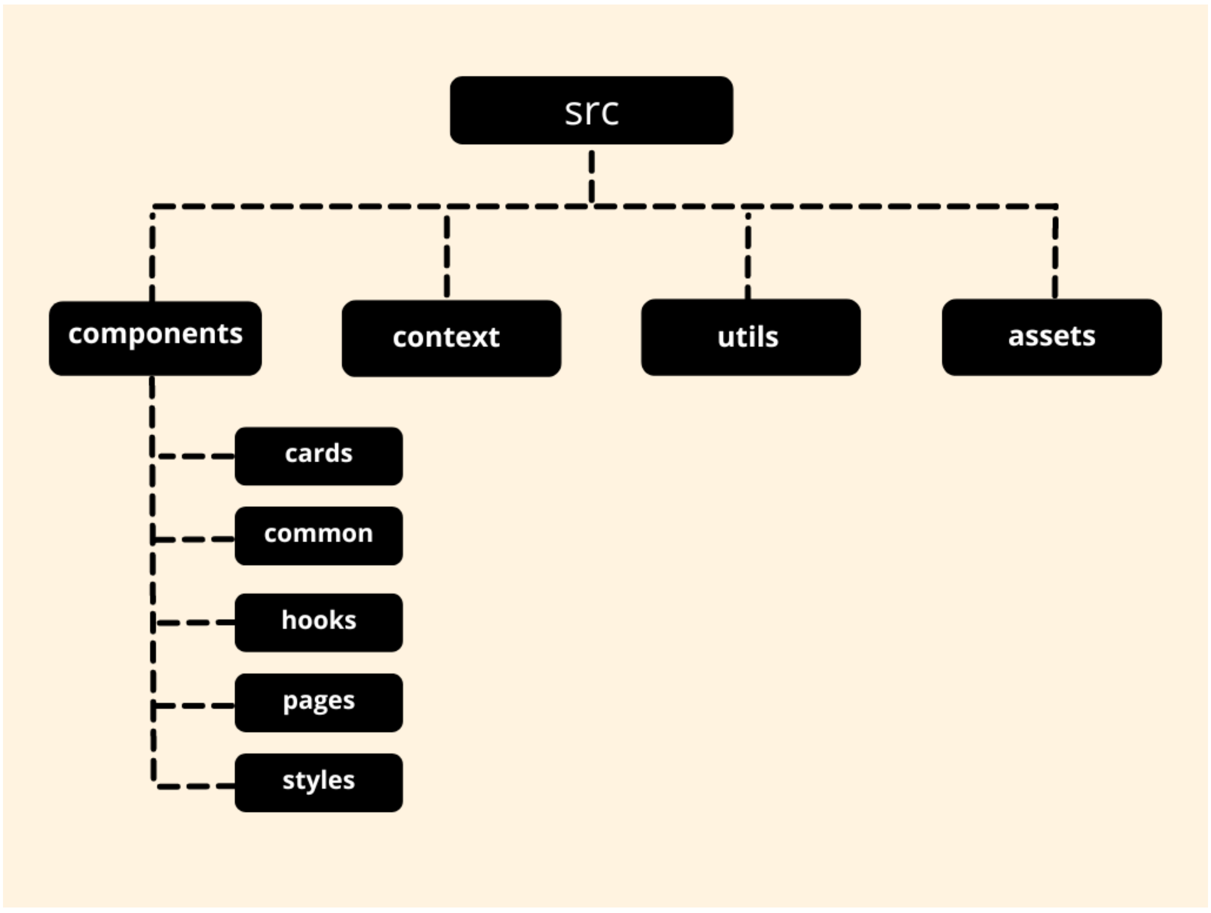


Fig. 2. High-level frontend structure.

In frontend architecture, src folder consists of 4 different folders. In the context folder there is a file called Auth and the task of this file is to store the authentication information in the context API. In the utils folder, there is a file named service. This file is responsible for communication with the backend. All Axios requests are written in this file and the request is sent to the backend thanks to the methods written in this file. In the assets folder there are logo files of the project. The components folder consists of 5 different folders. These are cards, common, hooks, pages, and styles. In the cards folder, there are files that are responsible for the cards in the project such as analysis card, github card, missing links card. In the common folder, there are components used commonly in the project such as navigation bar and side bar. In the hooks folder there are hooks components used for authentication and local storage. In the pages folder. There are page components of the project such as New Project, Historical Graph, Login. In the styles folder there are css and scss files that are responsible for the design of the project.

4.2. REST API

The ReLink API module is a powerful web service that enables communication between the presentation layer and the backend of your application. This advanced module is built using Django and Django REST Framework to provide a scalable, modifiable, and easy-to-use interface.

With ReLink API, resources such as pull requests, issues, users, and commits are exposed through a uniform interface using URIs (Uniform Resource Identifiers) and HTTP methods (GET, POST, PUT, DELETE) to perform actions on them. The API responds with representations of these resources in JSON (JavaScript Object Notation) format.

A key feature of ReLink API is its stateless architecture, which means that each request contains all the information necessary for the server to perform the request. This design makes the API highly scalable and easy to cache, leading to improved performance.

Authentication and authorization are essential aspects of ReLink API. Authentication is used to verify the identity of users or applications, while authorization is employed to determine whether a user or application has permission to perform a specific action on a resource. Okta authentication is used for authentication within the ReLink API, while different types of users, such as developers and project managers, have access to different authorization levels.

4.3. Data Retrieval

This module is responsible for fetching pull request, commit, and issue data from repositories of a project that is created by the user in the application. It interacts with two external systems which are Jira and Azure DevOps REST API to fetch these data. In this module, there are a total of 2 classes, each responsible for retrieving data from a different source. Facade design pattern is used so that the REST API subsystem can easily use the data retrieving action from this subsystem. In this way, the REST API subsystem can initiate the whole process with a single method call without needing to know the details of the data retrieving process. When the data retrieval process is completed, the Data Retrieval module interacts with the persistence layer and saves the data to the database. In this way, REST API, Link Recovery, and Graph Builder subsystems can utilize these data.

4.4. Link Recovery

4.4.1. Word Embeddings

Word embedding is a natural language processing (NLP) technique, where a vector representation is learned for each word. The vector representations of similar words are closer to each other in terms of geometrical distance (either cosine or euclidian distance), making word embeddings a useful technique in many NLP domains such as sentiment analysis, text generation, and text similarity [8], [9]. The most common method for learning word embeddings is the Word2Vec algorithm, which utilizes a shallow neural network for training embedding models and can be accessed through the Python Gensim library. There exist two techniques for implementing a Word2Vec model. The first one is continuous bag of words (CBOW), where a target word is predicted according to the input context words, allowing for missing word prediction. On the other hand, the skip-gram model predicts the context words of the input word, in a similar fashion to an inverted CBOW model [10].

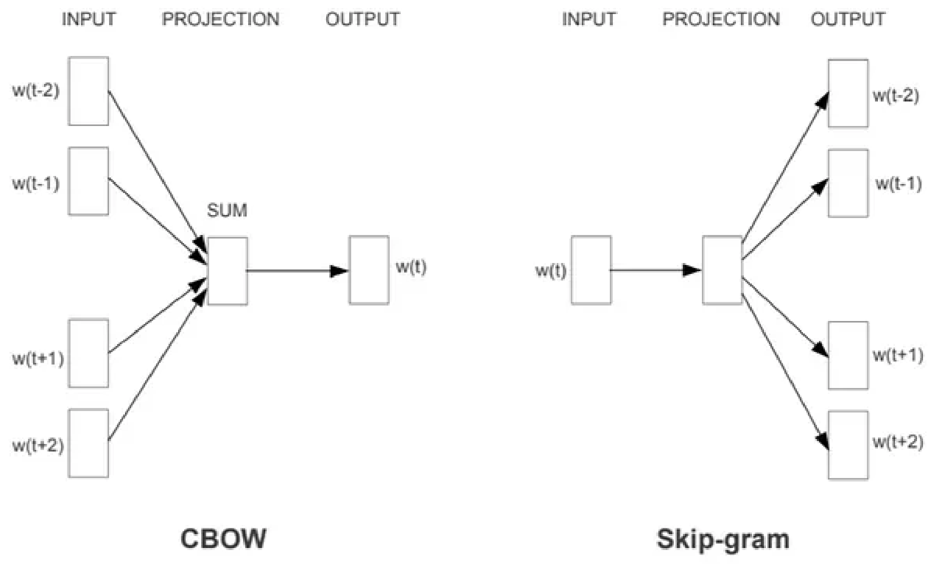


Fig. 3. CBOW vs. skip-gram [9].

Both model types can be trained with the Gensim library, without having to develop a neural network architecture from the ground up. Moreover, existing pre-trained models, such as Google's "GoogleNews-vectors-negative300" model trained on approximately 100 billion words using both the CBOW and skip-gram techniques, can also be used [11]. We used this model since we do not currently have sufficient data for model training and even if we did, it would be difficult to outperform a model trained on such a large dataset. Once the words are embedded into vectors, we must use these vectors for calculating text similarity. For this, we first pre-process our sentence corpus by removing stopwords (common words) and punctuation, changing non-english characters to their english counterparts, turning the entire corpus to lowercase, and lemmatizing the corpus (removing suffixes that add no contextual information). A preprocessing example can be shown as follows:

```
    pre_process([“Missing links page should provide more information”]) ->
                    [“missing link page provide information”]
```

Since we have a model that is already trained for embedding (vectorizing) words, we use this model to find the weighted sum of the word vectors in each sentence. This weighting is done with the term frequency-inverse document frequency (TF-IDF) of each word being multiplied by them while taking the sentence sum, which increases the weights of words with a higher frequency in the corpus for maintaining contextual information. Once all the sentence embeddings are calculated, the cosine similarities of the issue summary and PR title vector sets are calculated by taking the dot products of their embedding vector matrices, as shown below (theta is the angle between vectors):

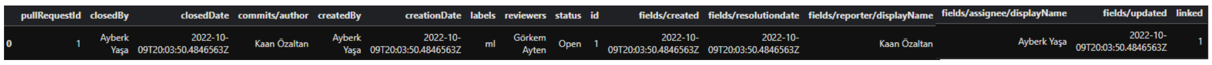
$$\text{cosine similarity} = S_C(A, B) := \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum\limits_{i=1}^{n} A_i B_i}{\sqrt{\sum\limits_{i-1}^{n} A_i^2} \sqrt{\sum\limits_{i-1}^{n} B_i^2}}$$

Fig. 4. Cosine similarity [9].

The cosine similarity is higher for vectors with a smaller angle in-between, meaning that their corresponding sentences are semantically and contextually similar, and the respective issue-PR pair is likely to be linked. This module of the link recovery subsystem is, therefore, responsible for detecting text similarity between issue summaries and PR titles.

4.4.2. Metadata Classification

Further similarity information can be extracted from the metadata of commits and PRs. This information includes opening and closure dates of PRs and issues, their labels, and authors. A binary classifier model is planned to be trained for increasing the accuracy of the text similarity module, where the metadata of an issue and a PR will be fed as input, and the output will predict whether they are linked or not. The data structure we plan to use for this module is as follows:



	pullRequestId	closedBy	closedDate	commits/author	createdBy	creationDate	labels	reviewers	status	id	fields/created	fields/resolutiondate	fields/reporter/displayName	fields/assignee/displayName	fields/updated	linked
0	1	Ayberk Yaşa	2022-10-09T20:03:50.4846563Z	Kaan Özaltan	Ayberk Yaşa	2022-10-09T20:03:50.4846563Z	ml	Görkem Ayten	Open	1	2022-10-09T20:03:50.4846563Z	2022-10-09T20:03:50.4846563Z	Kaan Özaltan	Ayberk Yaşa	2022-10-09T20:03:50.4846563Z	1

Fig. 5. Metadata training sample.

4.4.3. Heuristic Checks

Finally, we will perform heuristic checks on the predicted links to determine whether they can really be linked according to a pre-determined set of rules. These rules will be structured such as, a PR with an opening date 6 months later than a specific issue cannot be linked with the said issue. This rule, along with the others that will be used, enforce the prediction accuracy of our models according to logical rules that must almost certainly hold in all cases. As stated, this is not an ML technique and is purely heuristic.

4.5. Graph Builder

This module utilizes the pull request, issue, and commit data fetched by the Data Retrieval module and saved in the database. The purpose of this module is to create a graph data structure where each pull request is associated with commits and issues. For this, it detects all relationships for these three models in the database and groups them by pull requests. In order to create a visual graph data structure on the frontend, it organizes the data to be sent to the frontend and puts it in the appropriate form.

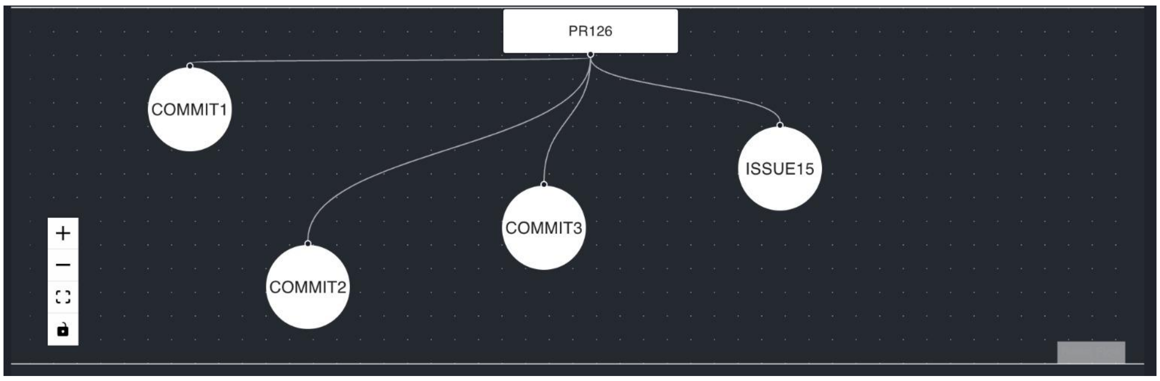


Fig. 6. Example network graph rendered from the data sent by Graph Builder.

5. Test Cases (functional and non-functional test cases, procedures, and expected results)

ReLink’s main priority is to achieve a high accuracy when connecting issues and pull requests. For that reason, our ML model is the most important functionality of our system. In this section, the tests done by humans (developers) will be explained, although the automated script tests will be executed in the following reports. There are 32 test cases in total, including 18 functional test cases, 9 non-functional test cases, and 5 MMR test cases.

5.1. Functional Test Cases

Test Case #F-01
Test Type/Category: Functional

Summary/Title/Objective: Test Case for the Login Feature with Valid Credentials.
Procedure of testing steps:

- Launch the website and navigate to the login page.
- Enter the company email and password.
- Click on the "Login" button.
- Verify that the user is redirected to the overview page.
- Verify that the user's name is displayed in the top right corner of the dashboard.

Expected results/Outcome:

- The user should be able to log in with valid credentials.
- The user should be redirected to the dashboard page after successful login.
- The user's name should be displayed in the top right corner of the dashboard.

Priority/Severity: Critical
Date Tested and Test Result: Pass

Test Case #F-02
Test Type/Category: Functional
Summary/Title/Objective: Test Case for the Login Feature with Invalid Credentials.
Procedure of testing steps:

- Launch the website and navigate to the login page.
- Enter an invalid company email and password.
- Click on the "Login" button.
- Verify that an error message is displayed.

Expected results/Outcome:

- The user should not be able to log in with invalid credentials.
- An error message should be displayed indicating the login was unsuccessful.

Priority/Severity: Major
Date Tested and Test Result: Pass

Test Case #F-03
Test Type/Category: Functional
Summary/Title/Objective: Test Case for the registration feature with valid data.
Procedure of testing steps:

- Launch the website and navigate to the registration page.
- Enter valid data in all required fields.
- Click on the "Register" button.
- Verify that the user is redirected to the login page.
- Verify that a success message is displayed.

Expected results/Outcome:

- The user should be able to log in with valid credentials.
- The user should be redirected to the dashboard page after successful login.
- The user's name should be displayed in the top right corner of the dashboard.

Priority/Severity: Critical
Date Tested and Test Result: To be completed in the final report.

Test Case #F-04
Test Type/Category: Functional
Summary/Title/Objective: Test Case for the Logout Feature.
Procedure of testing steps:
- Launch the website and log in with valid credentials.
- Click on the "Logout" button.
- Verify that the user is redirected to the login page.

Expected results/Outcome:
- The user should be able to log out successfully.
- The user should be redirected to the login page after successful logout.

Priority/Severity: Critical
Date Tested and Test Result: To be completed in the final report.

Test Case #F-05
Test Type/Category: Functional
Summary/Title/Objective: Test Case for the Repository Filtering Feature.
Procedure of testing steps:
- Launch the website and log in with valid credentials.
- Navigate to the Project List screen.
- Navigate to the Project Overview screen.
- Select a repository provider (e.g. GitHub or Azure).

Expected results/Outcome:
- The user should be able to see only the repositories for the selected provider.

Priority/Severity: Minor
Date Tested and Test Result: To be completed in the final report.

Test Case #F-06
Test Type/Category: Functional
Summary/Title/Objective: Test Case for the “Open Project” Button.
Procedure of testing steps:
- Launch the website and log in with valid credentials.
- Navigate to the Project List screen.
- Click on the “Open Project” button of any of the listed projects and verify that the corresponding page is displayed.

Expected results/Outcome:
- The user should be able to navigate to the corresponding page of the project.

Priority/Severity: Major
Date Tested and Test Result: To be completed in the final report.

Test Case #F-07
Test Type/Category: Functional
Summary/Title/Objective: Test Case for the “Manage Project Access” button.
Procedure of testing steps:

- Launch the website and log in with valid credentials.
- Navigate to the Project List screen.
- Click on the “Open Project” button of any of the listed projects and verify that the corresponding pop-up is displayed.

Expected results/Outcome:
- The user should be able to see a corresponding pop-up.

Priority/Severity: Major
Date Tested and Test Result: To be completed in the final report.

Test Case #F-08
Test Type/Category: Functional
Summary/Title/Objective: Test Case for the left frame for general view.
Procedure of testing steps:
- Launch the website and log in with valid credentials.
- Navigate to any possible screen.
- Click on any of the 3 possible buttons at the left frame: “Project List”, “New Project”, or “Logout”.

Expected results/Outcome:
- The user should be able to navigate to the desired screens through the left frame.

Priority/Severity: MInor
Date Tested and Test Result: To be completed in the final report.

Test Case #F-09
Test Type/Category: Functional
Summary/Title/Objective: Test Case for the left frame for project-specified view.
Procedure of testing steps:
- Launch the website and log in with valid credentials.
- Navigate to any possible screen.
- Click on any of the 4 possible buttons at the left frame: “Overview”, “Analysis”, “Missing Links”, or “Historical Graph”.

Expected results/Outcome:
- The user should be able to navigate to the desired screens through the left frame.

Priority/Severity: MInor
Date Tested and Test Result: To be completed in the final report.

Test Case #F-10
Test Type/Category: Functional
Summary/Title/Objective: Test Case for the “Back to Project List” button.
Procedure of testing steps:
- Launch the website and log in with valid credentials.
- Navigate to the Project List screen.

- Click on the “Open Project” button of any of the listed projects and verify that the corresponding pop-up is displayed.

Expected results/Outcome:

- The user should be able to see a corresponding pop-up.

Priority/Severity: Major
Date Tested and Test Result: To be completed in the final report.

Test Case #F-11
Test Type/Category: Functional
Summary/Title/Objective: Test Case for Listing the Repositories of a Project.
Procedure of testing steps:

- Launch the website and log in with valid credentials.
- Navigate to the Project List screen.
- Navigate to the Project Overview screen through any of the possible projects.

Expected results/Outcome:

- The user should be able to view a list of the repositories within the project, along with their authentication codes and the third-party application from which they were obtained.

Priority/Severity: Major
Date Tested and Test Result: To be completed in the final report.

Test Case #F-12
Test Type/Category: Functional
Summary/Title/Objective: Test Case for the Graph Builder.
Procedure of testing steps:

- Launch the website and log in with valid credentials.
- Navigate to the Project List screen.
- Navigate to the Project Overview screen through any of the possible projects.
- Navigate to the Project Analysis screen.

Expected results/Outcome:

- The user should be able to view the graphs for the already executed analysis.

Priority/Severity: Major
Date Tested and Test Result: To be completed in the final report.

Test Case #F-12
Test Type/Category: Functional
Summary/Title/Objective: Test Case for Changing the Chart Type of Graph Builder.
Procedure of testing steps:

- Launch the website and log in with valid credentials.
- Navigate to the Project List screen.
- Navigate to the Project Overview screen through any of the possible projects.
- Navigate to the Project Analysis screen.
- By the selection component at the right, which is labeled as “Chart Type”, try to change the chart type

Expected results/Outcome:
- The user should be able to change the chart type.

Priority/Severity: Minor
Date Tested and Test Result: To be completed in the final report.

Test Case #F-13
Test Type/Category: Functional
Summary/Title/Objective: Test Case for Listing the Missing Links in a Project.
Procedure of testing steps:
- Launch the website and log in with valid credentials.
- Navigate to the Project List screen.
- Navigate to the Project Overview screen through any of the possible projects.
- Navigate to the Project Missing Links screen.

Expected results/Outcome:
- The user should be able to see the missing links in the project, which has at least one proposed solution by the ML model.

Priority/Severity: Major
Date Tested and Test Result: To be completed in the final report.

Test Case #F-14
Test Type/Category: Functional
Summary/Title/Objective: Test Case for Listing the Potential Solutions for Missing Links in a Project.
Procedure of testing steps:
- Launch the website and log in with valid credentials.
- Navigate to the Project List screen.
- Navigate to the Project Overview screen through any of the possible projects.
- Navigate to the Project Missing Links screen.

Expected results/Outcome:
- The user should be able to see the potential relinks provided by the ML model, for each of the missing links.
- The user should be able to see the details of a potential relink, such as the issue ID, confidence (accuracy) rate, name of the related repository, and the issue title.

Priority/Severity: Major
Date Tested and Test Result: To be completed in the final report.

Test Case #F-15
Test Type/Category: Functional
Summary/Title/Objective: Test Case for Checking the Check-box(es) for the Potential Relinks.
Procedure of testing steps:
- Launch the website and log in with valid credentials.
- Navigate to the Project List screen.

- Navigate to the Project Overview screen through any of the possible projects.
- Navigate to the Project Missing Links screen.
- At the listed potential issues for any of the listed pull requests, choose the issue(s) which you think are relevant to the pull request.

Expected results/Outcome:
- The user should be able to choose the issues they like to relink with the pull request of interest.
- The user should be able to see a warning message about selecting more than one issue is allowed, but not recommended.

Priority/Severity: Major
Date Tested and Test Result: To be completed in the final report.

Test Case #F-16
Test Type/Category: Functional
Summary/Title/Objective: Test Case for the Relink Functionality.
Procedure of testing steps:
- Launch the website and log in with valid credentials.
- Navigate to the Project List screen.
- Navigate to the Project Overview screen through any of the possible projects.
- Navigate to the Project Missing Links screen.
- At the listed potential issues for any of the listed pull requests, choose the issue(s) which you think are relevant to the pull request.
- Click on the “RELINK” button at the top right of the component.

Expected results/Outcome:
- The user should be able to choose the issues they like to relink with the pull request of interest.
- The user should be able to receive a status message about whether the relink operation is finished with success.
- If the relinking operation is successful, the pull request should be removed from the screen.

Priority/Severity: Critical
Date Tested and Test Result: To be completed in the final report.

Test Case #F-17
Test Type/Category: Functional
Summary/Title/Objective: Test Case for Ignoring the Relink Functionality.
Procedure of testing steps:
- Launch the website and log in with valid credentials.
- Navigate to the Project List screen.
- Navigate to the Project Overview screen through any of the possible projects.
- Navigate to the Project Missing Links screen.
- At the listed potential issues for any of the listed pull requests, choose the issue(s) which you think are relevant to the pull request.
- Click on the “IGNORE” button at the top right of the component.

Expected results/Outcome:
- The user should be able to choose the issues they like to relink with the pull request of interest.
- The user should be able to receive a status message about their choice.
- The pull request should be removed from the screen.

Priority/Severity: Major
Date Tested and Test Result: To be completed in the final report.

Test Case #F-18
Test Type/Category: Functional
Summary/Title/Objective: Test Case for Ignoral Pop-up of the Relink Functionality.
Procedure of testing steps:
- Launch the website and log in with valid credentials.
- Navigate to the Project List screen.
- Navigate to the Project Overview screen through any of the possible projects.
- Navigate to the Project Missing Links screen.
- At the listed potential issues for any of the listed pull requests, choose the issue(s) which you think are relevant to the pull request.
- Click on the "IGNORE" button at the top right of the component.

Expected results/Outcome:
- When the user clicks the ignore button on the Missing Links page, a pop-up(modal) should appear which says, "This PR will be ignored, so you cannot recover its missing link from now on.".

Priority/Severity: Minor
Date Tested and Test Result: To be completed in the final report.

5.2. Non-Functional Test Cases

Test Case #NF-01
Test Type/Category: Performance
Summary/Title/Objective: Test Case for Load Time of Homepage.
Procedure of testing steps:
- Open the website homepage in a web browser.
- Start the timer as soon as the website starts loading.
- Stop the timer as soon as the website is fully loaded.

Expected results/Outcome:
- The website homepage should load within 2 seconds.

Priority/Severity: Major
Date Tested and Test Result: To be completed in the final report.

Test Case #NF-02
Test Type/Category: Performance
Summary/Title/Objective: Test Case for Load Time of Project Page.
Procedure of testing steps:

- Open the Project Overview screen in a web browser.
- Start the timer as soon as the page starts loading.
- Stop the timer as soon as the page is fully loaded.

Expected results/Outcome:
- The product page should load within 2 seconds.

Priority/Severity: Minor
Date Tested and Test Result: To be completed in the final report.

Test Case #NF-03
Test Type/Category: Security
Summary/Title/Objective: Test Case for Password Strength.
Procedure of testing steps:
- Attempt to create an account using a weak password (e.g. "1234").
- Verify that the system rejects the password.

Expected results/Outcome:
- The system should not allow a weak password to be created and should prompt the user to choose a stronger one.

Priority/Severity: Major
Date Tested and Test Result: Pass

Test Case #NF-04
Test Type/Category: Usability
Summary/Title/Objective: Test Case for Navigation.
Procedure of testing steps:
- Attempt to navigate to a specific page on the website using the navigation menu (left frame).
- Verify that the navigation is intuitive and easy to use.

Expected results/Outcome:
- The navigation should be easy to use and should take no more than 2 clicks to reach any page on the site.

Priority/Severity: Minor
Date Tested and Test Result: Pass.

Test Case #NF-05
Test Type/Category: Compatibility
Summary/Title/Objective: Test Case for Browser Compatibility.
Procedure of testing steps:
- Open the website in different browsers (e.g. Chrome, Firefox, Safari, Edge).
- Verify that the website displays correctly and functions properly in each browser.

Expected results/Outcome:
- The website should be compatible with all major browsers and should display correctly and function properly in each one.

Priority/Severity: Minor

Date Tested and Test Result: To be completed in the final report.

Test Case #NF-06
Test Type/Category: Security
Summary/Title/Objective: Test Case for Login Authentication.
Procedure of testing steps:
- Attempt to log in using incorrect login credentials.
- Verify that the system does not grant access.

Expected results/Outcome:
- The system should not allow access to the account if the login credentials are incorrect.

Priority/Severity: Critical
Date Tested and Test Result: To be completed in the final report.

Test Case #NF-07
Test Type/Category: Usability
Summary/Title/Objective: Test Case for Mobile Responsiveness.
Procedure of testing steps:
- Open the website on a mobile device and navigate to different pages.
- Verify that the website is mobile responsive and easy to use on a smaller screen.

Expected results/Outcome:
- The website should be mobile responsive and easy to use on a smaller screen.

Priority/Severity: MInor.
Date Tested and Test Result: To be completed in the final report.

Test Case #NF-08
Test Type/Category: Performance
Summary/Title/Objective: Test Case for Concurrent User Load.
Procedure of testing steps:
- Simulate multiple concurrent users accessing the website at the same time.
- Verify that the website can handle the load without performance issues or errors.

Expected results/Outcome:
- The website should be able to handle multiple concurrent users without performance issues or errors.

Priority/Severity: Major
Date Tested and Test Result: To be completed in the final report.

Test Case #NF-09
Test Type/Category: Usability
Summary/Title/Objective: Test Case for User Interface Consistency.
Procedure of testing steps:

- Navigate to different pages on the website.
- Verify that the user interface (UI) design elements (colors, fonts, icons, etc.) are consistent across all pages.

Expected results/Outcome:

- The user interface should be consistent across all pages, providing a cohesive and professional look and feel.

Priority/Severity: Minor
Date Tested and Test Result: Pass

5.3. Mean Reciprocal Rank Test Cases

Mean Reciprocal Rank (MRR) is a widely used evaluation metric in the field of information retrieval and machine learning. In this section, we will use MRR to evaluate the performance of our machine learning model and compare it against other models or baseline approaches [12].

$$\text{MRR} = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{\text{rank}_i}.$$

Fig. 7. Mean reciprocal rank formula.

After 5 different test cases for the ML model, by mean reciprocal rank, we obtained 3 out of these cases that resulted with the best success, while one of them gave the correct issue in the 2nd option, and one of them gave the correct issue in the 6th option. Therefore, we obtained the mean reciprocal rank as (⅙ + 1 + 1 + ½ + 1)/5 = 22/30, or about 0.73.

Test Case #MRR-01
Test Type/Category: Mean Reciprocal Rank
Given Data:

- Pull request ID: 07
- Pull request name: Implement retrieval methods.

Expected link:

- Issue ID: 10002
- Issue name: Create Data Retrieval Service for Azure.

Resulting links:

1. (10004, 0.6538699070958309)
2. (10043, 0.5920638481740015)
3. (10006, 0.5867772714700163)
4. (10061, 0.5657587874506174)
5. (10059, 0.5537630073462833)
6. **(10002, 0.5424273213403735)**
7. (10056, 0.5096294871954614)

Rank: 6
Reciprocal rank: 1/6

Priority/Severity: Major
Date Tested and Test Result: Pass.

Test Case #MRR-02
Test Type/Category: Mean Reciprocal Rank
Given Data:
- Pull request ID: 10
- Pull request name: Add branch information to PR model

Expected link:
- Issue ID: 10047
- Issue name: Branch information should exist in PR data

Resulting links:
1. **(10047, 0.800569248490578)**
2. (10033, 0.5269733242444746)
3. (10054, 0.5238443933565953)
4. (10050, 0.4821396587077893)
5. (10019, 0.4798525475325077)
6. (10004, 0.47978947211175793)
7. (10045, 0.4787526031388459)

Rank: 1
Reciprocal rank: 1
Priority/Severity: Major
Date Tested and Test Result: Pass.

Test Case #MRR-03
Test Type/Category: Mean Reciprocal Rank
Given Data:
- Pull request ID: 01
- Pull request name: Initialize Django project on backend

Expected link:
- Issue ID: 10001
- Issue name: Initialize Django project on backend

Resulting links:
1. **(10001, 1.0)**
2. (10023, 0.7692845430370506)
3. (10061, 0.5951890946975835)
4. (10041, 0.5625866478927344)
5. (10059, 0.5554017555065811)
6. (10048, 0.5519805606736923)
7. (10003, 0.5513433182599485)

Rank: 1
Reciprocal rank: 1
Priority/Severity: Major
Date Tested and Test Result: Pass.

Test Case #MRR-04
Test Type/Category: Mean Reciprocal Rank
Given Data:
- Pull request ID: 13
- Pull request name: Create Possible Link model

Expected link:
- Issue ID: 10054
- Issue name: Create PossibleLink database model

Resulting links:
1. (10050, 0.6267598773665553)
2. **(10054, 0.6061555594382552)**
3. (10033, 0.5803833028086245)
4. (10031, 0.5692814990365802)
5. (10032, 0.5505409504680864)
6. (10030, 0.5045170025238684)
7. (10019, 0.5033349571558927)

Rank: 2
Reciprocal rank: 1/2
Priority/Severity: Major
Date Tested and Test Result: Pass.

Test Case #MRR-05
Test Type/Category: Mean Reciprocal Rank
Given Data:
- Pull request ID: 09
- Pull request name: Add code formatter to backend project

Expected link:
- Issue ID: 10023
- Issue name: Add code formatter to backend project

Resulting links:
1. **(10023, 0.9999999999999999)**
2. (10001, 0.7692845430370506)
3. (10059, 0.6877586067003254)
4. (10061, 0.6394603342250512)
5. (10057, 0.6075983829301943)
6. (10020, 0.5734452453783456)
7. (10063, 0.5652294032663305)

Rank: 1
Reciprocal rank: 1
Priority/Severity: Major
Date Tested and Test Result: Pass.

6. Consideration of Various Factors in Engineering Design

There is no external factor except for the economic factor limiting us during the development of ReLink. We need to consider some economic factors during the design phase of the project. These economic factors are database costs and server costs, respectively. Since we will be doing repository mining, we need a database where we can store the data of the project with any scale from GitHub, Jira, or Azure DevOps. This database will require a substantial budget. Moreover, we may need external CPUs to train our models, so we will need more performance than a normal web application. This will increase server costs.

Factor	Effect(s)	Level of Effect (0-10)
Public Health	Has no effect	0
Safety	Has no effect	0
Global	Has no effect	0
Cultural	Has no effect	0
Social	Has no effect	0
Environmental	Has no effect	0
Economic	Database Costs Server Costs	7

7. Teamwork Details

7.1. Contributing and functioning effectively on the team

Ayberk Yaşa: Writing reports, implementation of REST API, implementation of data retrieval, implementation of graph builder modules, regularly attending meetings.
Cemhan Kaan Özaltan: Writing reports, implementation of REST API, implementation of ML modules, regularly attending meetings.
Çağatay Şafak: Writing reports, sketching UI in Figma, implementation of frontend at a low level, attending meetings.
Fatih Kaplama: Writing reports, sketching UI in Figma, implementation of frontend, regularly attending meetings.
Görkem Ayten: Writing reports, implementation of REST API, regularly attending meetings.

7.2. Helping creating a collaborative and inclusive environment

Ayberk Yaşa: Demonstrated an encouraging and critical approach in meetings.
Cemhan Kaan Özaltan: Demonstrated an encouraging and critical approach in meetings.
Çağatay Şafak: Was relatively reluctant in meetings.
Fatih Kaplama: Demonstrated an encouraging and critical approach in meetings.
Görkem Ayten: Demonstrated an encouraging and critical approach in meetings.

7.3. Taking lead role and sharing leadership on the team

Ayberk Yaşa: Took overall leadership in the project management and task distribution along with retrieval modules.
Cemhan Kaan Özaltan: Took leadership in the decision making and implementation of ML related modules.
Çağatay Şafak: Took leadership in the UI/UX design and decision making phases of the project.
Fatih Kaplama: Took leadership in the front-end implementation and decision making stages.
Görkem Ayten: Took leadership in the backend implementation, modeling, and decision making.

8. References

[1] T. W. Aung, H. Huo, and Y. Sui, “A literature review of automatic traceability links recovery for software change impact analysis,” *Proceedings of the 28th International Conference on Program Comprehension*, 2020.

[2] M. Rath, J. Rendall, J. L. Guo, J. Cleland-Huang, and P. Mäder, “Traceability in the wild,” *Proceedings of the 40th International Conference on Software Engineering*, 2018.

[3] C. M. Lüders, T. Pietz, W. Maalej. “Automated Detection of Typed Links in Issue Trackers”. *2022 IEEE 30th International Requirements Engineering Conference (RE)*, 2022.

[4] “Web content accessibility guidelines,” *W3C*. [Online]. Available: https://www.w3.org/TR/WCAG21/. [Accessed: 05-Mar-2023].

[5] “Github Action Check Linked Issues,” *GitHub*. [Online]. Available: https://github.com/nearform/github-action-check-linked-issues. [Accessed: 08-Oct-2022].

[6] H. Shobokshi, “Verify Linked Issue Action,” *GitHub*. [Online]. Available: https://github.com/hattan/verify-linked-issue-action. [Accessed: 08-Oct-2022].

[7] H. Ruan, B. Chen, X. Peng, and W. Zhao, “DeepLink: Recovering issue-commit links based on deep learning,” *Journal of Systems and Software*, vol. 158, no. 110406, 2019.

[8] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” *arXiv preprint arXiv*, vol. 1301, no. 3781, 2013.

[9] “Comparison of different word embeddings on text similarity‐a use case in NLP,” *Medium*, Oct. 2019. [Online]. Available: https://intellica-ai.medium.com/comparison-of-different-word-embeddings-on-text-similarity-a-use-case-in-nlp-e83e08469c1c. [Accessed: 10-Mar-2023].

[10] O. Levy and Y. Goldberg, “Dependency-based word embeddings,” *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, vol. 2, pp. 302–308, 2014.

[11] “Word2vec-google-news-300,” *Hugging Face*. [Online]. Available: https://huggingface.co/fse/word2vec-google-news-300. [Accessed: 05-Mar-2023].

[12] N. Craswell, “Mean reciprocal rank,” *Encyclopedia of Database Systems*, p. 1703, 2009.