



Πανεπιστήμιο Κρήτης
Τμήμα Επιστήμης Υπολογιστών

SnipShare: Μια εφαρμογή διαμοιρασμού, σχολιασμού και αναζήτησης αποσπασμάτων κώδικα λογισμικού

SnipShare: A Web Application for Sharing,
Commenting and Searching Code Snippets

Γιώργος Ψηλός
csd4316@csd.uoc.gr

Επόπτης: Γιάννης Τζιτζίκας
Επιβλέπων: Γιάννης Μαркеτάκης

Φεβρουάριος 2025

Περιγραφή εργασίας

Η ανάπτυξη λογισμικού είναι μια συνεργατική διαδικασία, στην οποία οι προγραμματιστές συχνά εργάζονται μαζί για την δημιουργία νέων λειτουργιών και την επίλυση προβλημάτων. Είναι πολύ συνηθισμένο οι προγραμματιστές να μοιράζονται αποσπάσματα κώδικα για αναθεώρηση και αποσφαλμάτωση, καθώς αυτό βοηθά στη βελτίωση της ποιότητας του κώδικα και στην ανεύρεση σφαλμάτων πιο γρήγορα και αποτελεσματικά. Επιπλέον ο κώδικας λογισμικού σε αντίθεση με άλλα είδη κειμένου, έχει συνήθως συγκεκριμένη μορφοποίηση (π.χ. κενά, σύμβολα, ειδικούς χαρακτήρες) η οποία έχει σημασία. Είναι λοιπόν σημαντικό η ανταλλαγή τέτοιων πληροφοριών να γίνεται με τρόπο που δεν τροποποιεί την σημασιολογία τους. Η ανταλλαγή για παράδειγμα ενός αποσπάσματος κώδικα μέσω μια εφαρμογής άμεσων συνομιλιών (π.χ. skype) ενδέχεται να οδηγήσει σε ένα εντελώς παραποιημένο απόσπασμα λόγω των συντομεύσεων που χρησιμοποιούν οι εφαρμογές συνομιλιών (π.χ. emoticons).

Ο σκοπός της εργασίας είναι ο σχεδιασμός και η υλοποίηση του SnipShare, μιας διαδικτυακής εφαρμογής διαμοιρασμού αποσπασμάτων κώδικα λογισμικού με ένα εύκολο, γρήγορο και ασφαλές τρόπο. Συγκεκριμένα δίνει την δυνατότητα στους προγραμματιστές να στείλουν ένα απόσπασμα κώδικα σε συναδέλφους τους, απλά αποθηκεύοντας το στην εφαρμογή η οποία τους επιστρέφει ένα μοναδικό αναγνωριστικό, με την μορφή ενός URL. Έτσι πλέον οι προγραμματιστές μπορούν να διαμοιράσουν το URL αντί για το κομμάτι του κώδικα, δίνοντας την δυνατότητα στους παραλήπτες να το δουν μέσω της εφαρμογής. Επιπλέον το SnipShare τους παρέχει τη δυνατότητα να σχολιάσουν το απόσπασμα κώδικα απευθείας μέσω της εφαρμογής. Εκτός από τον σχολιασμό των αποσπασμάτων, η εφαρμογή παρέχει και την δυνατότητα επισήμανσης με την χρήση ετικετών καθώς και αναζήτησης είτε μέσω κειμένου είτε μέσω των ετικετών. Τα αποσπάσματα κώδικα είναι δημόσια προσβάσιμα ώστε να μπορούν να αναζητηθούν. Επιπλέον, αξίζει να αναφερθεί ότι το SnipShare, μπορεί να χρησιμοποιηθεί για τον διαμοιρασμό οποιουδήποτε κειμένου είτε δομημένου είτε αδόμητου, και όχι μόνο για αποσπάσματα κώδικα. Τέλος οι χρήστες έχουν την δυνατότητα να ανεβάσουν ψηφιακά αντικείμενα (π.χ. εικόνες) και να τα διαμοιραστούν με άλλους χρήστες.

Ακολουθεί μια εκτενής περιγραφή των εργασιών που έγιναν για τον σχεδιασμό και την υλοποίηση του εργαλείου SnipShare.

***SnipShare* : A Web Application for Sharing, Commenting and Searching Code Snippets**

Giorgos Psilos

Computer Science Department, University of Crete, Heraklion, Greece
csd4316@csd.uoc.gr

Abstract. Code sharing is one of the most common activities between developers, as it facilitates the collaborative debugging, documentation and improvement of the quality of software code. Unlike other types of text, software code usually has specific formatting that is important. Therefore, it is essential that the exchange of such information occurs in a way that does not alter its semantics. In this work, we describe the design and implementation of *SnipShare*, an online application for sharing code snippets in an easy, fast and secure way. It supports storing code snippets, providing developers with a short and concise URL that can be used for sharing it. In addition, it provides the facilities for commenting code snippets directly through the online application, as well as searching and providing analytics of code snippets.

Keywords: code · collaborative software development · code sharing · code snippet · screenshot sharing · code tagging

1 Introduction

Software development is a collaborative process in which developers often work together to create new features and solve problems. It is very common for developers to share code snippets for review and debugging, as this helps improve code quality and identify errors more quickly and effectively. Additionally, unlike other types of text, software code usually has specific formatting (e.g., indentation, symbols, special characters) that is crucial. Therefore, sharing such information in a way that preserves its semantics is important. For instance, sharing a code snippet via an instant messaging application (e.g., Skype) might result in a completely distorted snippet due to shortcuts used by such applications (e.g., emoticons).

SnipShare is an online application for sharing software code snippets in an easy, fast, and secure manner. Specifically, it allows developers to send a code snippet to their colleagues simply by saving it in the application, which returns a unique identifier in the form of a URL. Developers can share this URL instead of the actual code snippet, enabling recipients to view it via the

application. Moreover, *SnipShare* provides the ability to comment on the code snippet directly through the application.

In addition to commenting on snippets, the application offers tagging features and a search functionality that works through either text or tags. Moreover, it is worth noting that *SnipShare* can be used to share any text, whether structured or unstructured, and not just code snippets. Finally, users can also upload any digital resource (e.g. a screenshot in JPEG format) instead of a code snippet and share it with other users.

The rest of this paper is organized as follows: Section 2, describes the main requirements, Section 3, provides details about the design and implementation of *SnipShare*, Section 4, evaluates our approach with respect to the main requirements, Section 5, describes application usage scenarios and shows detailed screenshots to facilitate understanding, Section 6, elaborates on related works and applications, Section 7, concludes the paper and elaborates issues for further research. Finally, Appendix A provides the documentation of the implemented RESTfull services of *SnipShare*.

2 Requirements

In this section, we describe the main requirements of *SnipShare*.

- **R1: Store and share code snippets**
Allow users to store their code snippets and share them with others easily.
- **R2: Attach metadata to code snippets**
Each code snippet should be associated with a short title and tags.
- **R3: Allow commenting on code snippets**
Users should be able to add comments to shared code snippets, enabling discussions or feedback.
- **R4: Show code formatting to facilitate developers**
Proper syntax highlighting and code formatting should be supported to enhance readability and understanding.
- **R5: Allow browsing code snippets**
Users must be able to browse through all the publicly available code snippets.
- **R6: Allow searching code snippets**
A search feature is required, enabling users to find specific snippets by title, tags, or content.
- **R7: Allow storing and sharing images**
Users should have the option to attach and share images along with code snippets.
- **R8: Allow storing and sharing any digital resource**
SnipShare should support sharing files such as documents, PDFs, or other digital resources.

- **R9: Ability to work using a single browser (no need for specific application)**

SnipShare must be implemented as a web application, allowing access from any browser without the need for additional software installation.

3 The Application

In this section, we provide the details about the implementation of *SnipShare*. First, in Section 3.1 we introduce information about the design of the system, based on the key requirements. Section 3.2 we provide the high-level architecture of the system. Section 3.3 describes the schema of the permanent storage of the application. Finally, Section 3.4 elaborates on the implementation details of the application.

3.1 Design

The design of *SnipShare* is driven by the following key requirements:

- **User-Friendly Interface:** The application provides an intuitive interface for submitting, managing, and discovering code snippets (snips).
- **Modularity and Scalability:** The backend and frontend are designed to be modular, making it easier to maintain and extend the application.
- **Search and Tagging:** Users can attach tags to their code snippets, making it easier to search and filter through the available snips.
- **Media Support:** Support for image uploads to associate screenshots or diagrams with code snippets.
- **Dark Mode:** Provides a dark mode feature for a better user experience during extended use.
- **Database Management:** Relational database (MySQL) for storing snips, comments, tags, and associated digital resources.

The frontend is developed with HTML, CSS, and JavaScript, while the backend uses Node.js with the Express framework. The system is designed to ensure a seamless user experience and efficient data handling.

3.2 High-Level Architecture

The high-level architecture of *SnipShare* consists of six primary components that are depicted in Fig. 1. More details about these components are given below.

Frontend (Client-Side): The frontend is built using HTML, CSS, and JavaScript. It enables users to submit, view, and search for snips. Additionally, it offers features such as dark mode, tag management, and image uploads to enhance user experience.

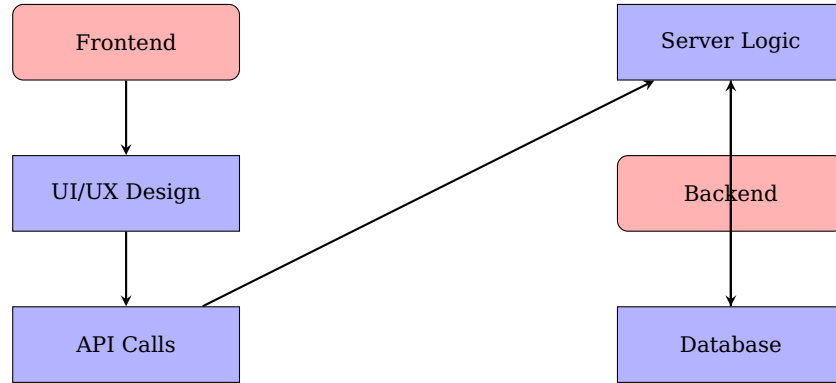


Fig. 1. High-level architecture of *SnipShare*

Backend (Server-Side): The backend is developed using Node.js and the Express framework. It manages API requests for submitting snips, fetching snips, managing comments, and uploading images. The backend provides RESTful endpoints to facilitate interaction with the database.

Database (MySQL): The database stores snips, comments, tags, and images. It includes tables such as `snip`, `comments`, `tags`, `images`, and `snip_tag`. Relational constraints are implemented to ensure data consistency and integrity.

Api Calls: This component handles the communication between the frontend and the backend. It is responsible for making HTTP requests to the backend's RESTful endpoints and processing the responses to update the frontend accordingly.

Server Logic: The Server Logic component contains the core business logic of the application. It processes incoming requests from the Api Calls, performs necessary operations such as data validation and transformation, and interacts with the database to retrieve or store data.

The communication flow is as follows: the frontend sends requests to the backend server, which processes the requests and interacts with the MySQL database. The backend then returns the appropriate response to the frontend.

3.3 Database Schema Description

The database schema is designed to efficiently manage and organize information related to code snippets, supporting functionalities such as tagging, commenting, and associating with external resources. The central entity in the schema is the **snip** table, which stores the primary data for each code snippet. It includes attributes such as `snip_id` (serving as the primary key), `title`, `snip` (containing the code content), `language` (indicating the

programming language), and `created_at` (recording the timestamp of creation).

The schema incorporates several related entities to enhance the data associated with each snippet. The **comments** table enables the storage of user-generated comments linked to specific snippets through the `snip_id` foreign key. This table includes attributes such as `comment_id` (primary key), `comment` (the text of the comment), and `created_at` (timestamp of the comment).

The **images** table is designed to associate visual content with code snippets. It contains `image_id` (primary key), `snip_id` (foreign key referencing the `snip` table), and `image_data`, which stores the image in a `LONGBLOB` format to accommodate large binary files.

To support external references, the **url** table stores URLs related to code snippets. This table includes `url_id` (primary key), `snip_id` (foreign key), and `url_link` (the external URL), allowing for the integration of additional resources.

Tagging functionality is implemented through the **tags** and **sniptag** tables. The `tags` table stores unique tags with attributes such as `tag_id` (primary key) and `tag_name`. Since a many-to-many relationship exists between code snippets and tags, the **sniptag** junction table is employed to manage this relationship, consisting of composite keys `snip_id` and `tag_id` that link snippets to their respective tags.

Overall, the schema is normalized to promote data integrity and reduce redundancy, while its relational structure facilitates efficient querying and management of code snippets and their associated metadata. The Entity Relationship diagram of the database is shown in Fig. 2.

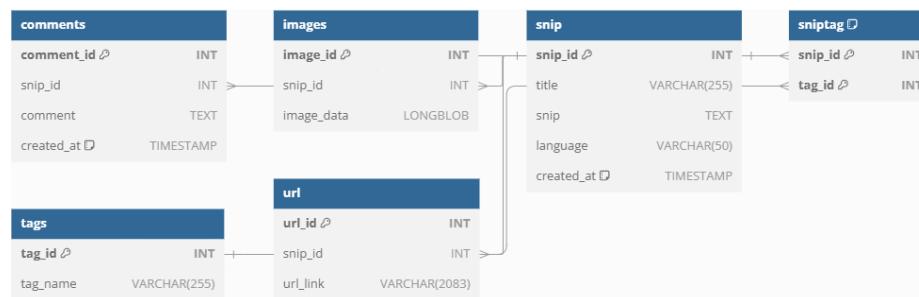


Fig. 2. The E-R diagram of the database

3.4 Implementation Details

SnipShare has been implemented as a web-based application using Node.JS¹ and Express² frameworks. The source code of the application has been deposited in a GitHub repository³. More details about the implementation, configuration and maintenance of the application are given in the sequel.

Updating the Frontend Configuration The `config.json` file in the project directory must be updated to reflect the desired URL. This file contains the base URL used by client-side scripts. Modify the `baseUrl` value accordingly.

For example, if deploying on a local server:

```
1 {  
2   "baseUrl": "http://localhost:3000"  
3 }
```

For deploying it on a remote server, users must replace the `baseUrl` with the URL of their server (e.g. IP address or server domain name).

Updating the Backend Configuration Similarly, the backend configuration must be adjusted in the `config.js` file. This file ensures that server-side scripts reference the correct API endpoint. The `baseUrl` should be modified in a similar manner as above.

```
1 module.exports = {  
2   baseUrl: "http://your-desired-url.com"  
3 };
```

Applying Changes and Restarting the Server To apply these changes, restart the server. If the application is running using `Node.js`, execute:

```
node server.js
```

For development environments using `nodemon`, use:

```
nodemon server.js
```

Backend Server Configuration The backend server is configured using `Node.js` and `Express`. Below is the key configuration for connecting to the MySQL database:

¹ <https://nodejs.org/en>

² <https://expressjs.com/>

³ <https://github.com/isl/SnipShare>


```

1 // server.js
2 const mysql = require('mysql2');
3
4 const db = mysql.createConnection({
5   host: 'localhost',
6   user: 'user',
7   password: 'pass!',
8   database: 'snip',
9   port: 3306
10 });
11
12 db.connect(err => {
13   if (err) throw err;
14   console.log('Connected to MySQL database');
15 });

```

To use a remote MySQL database, update the host, user, password, and port fields accordingly.

API Endpoints The following API endpoints are implemented to handle core functionality (detailed examples and JSON responses are provided in Appendix A). Table 1 lists all the available methods that can be used.

API method	Type	Description
/submit-snip-with-tags	POST	Handles submission of a new snip along with tags and an optional image
/api/snip/:id	GET	Fetches a snip by its unique ID, including associated tags and images
/api/discover-snips	GET	Retrieves all snips with previews and associated tags
/api/comments	POST	Allows users to add comments to a specific snip
/api/tags/search	GET	Provides tag suggestions based on user input

Table 1. The API methods of *SnipShare*

4 Evaluation

In this section, we evaluate the implementation of *SnipShare* with respect to the defined requirements (from Section 2). Table 2 outlines each requirement and how it is addressed in the implementation.

Req.	How It Is Covered
R1	<i>SnipShare</i> allows users to store and share code snippets ensuring that their work is safely preserved. Moreover collaboration is streamlined as users can share their code snippets with others by simply exchanging a short and unique URL, eliminating the need to send entire code blocks.
R2	<i>SnipShare</i> requires any code snippet to have a short title (of a maximum of 255 characters), to allow users quickly understand the context of a code snippet. Moreover, users can attach specific tags to a code snippet, which among others facilitate their retrieval.
R3	<i>SnipShare</i> includes a commenting feature integrated into each code snippet, enabling users to provide targeted feedback, and collaborate directly on the shared code. It is an interactive environment where developers can exchange ideas, ask questions and refine their work seamlessly within the context of code snippets. Moreover, comments are timestamped and presented in chronological order within the particular code snippet.
R4	Syntax highlighting is provided, in order to enhance the readability and comprehension of code snippets. This feature provides clear, font and color-coded differentiation of code elements for the most widely-used programming languages, including Java, Javascript, Python, C and many others.
R5	Users can explore over the saved and shared code snippets through the <i>Discover</i> page. This feature provides access to a dense version of code snippets, showing their title, tags, and a small part of the code snippet.
R6	<i>SnipShare</i> provides a search functionality that allows users to find specific code snippets by entering keywords related to the titles or tags. This mechanism facilitates users find their code snippets, even if they have lost their unique URL, enabling them to share it again or comment it.
R7-R8	Digital resources, such as images, documents and relevant files, can be attached to code snippets. These resources are securely preserved within the application's database, stored as binary long objects (BLOBs), ensuring that they remain safely archived, easily accessible, and protected from corruption or loss.
R9	<i>SnipShare</i> has been implemented as a web application and is accessible through any web browser, without the need for users to install additional software or plugins.

Table 2. Evaluation of *SnipShare* against defined requirements.

5 Application usage and Screenshots

This section illustrates two scenarios demonstrating the application's core functionalities through code submissions and error reporting.

5.1 Scenario 1: Posting a Code Implementation

In this scenario, the user submits a code snippet to showcase their work. The user provides a title, the source code, and relevant tags. This example focuses on the submission of a Bubble Sort Algorithm implemented in Python. The exact steps are shown below.

1. Step 1: The user navigates to the code submission page.

The user accesses the page where they can submit their code snippets. This page provides fields for entering a title, a text area for the code, and options to select relevant tags.

2. Step 2: The user fills in the title, code snippet, selects tags, and submits the post.

The user provides the following details:

- **Title:** "Bubble Sort Algorithm in Python"

- **Code Snippet:**

```
def bubble_sort(arr):
    n = len(arr)
    for i in range(n):
        for j in range(0, n-i-1):
            if arr[j] > arr[j+1]:
                arr[j], arr[j+1] = arr[j+1], arr[j]
    return arr
```

Example usage:

```
array = [64, 34, 25, 12, 22, 11, 90]
sorted_array = bubble_sort(array)
print("Sorted array:", sorted_array)
```

- **Tags:** The user selects "Python" and "Sorting" as the relevant tags.

- **Formatting:** The user selects "Python" as the Formatting.

- **Images:** The user provides no image for this snip.

After filling in the details, the user submits the post. The first two steps are visualized in Figure 3.

3. Step 3: The submission is displayed, allowing other users to view, comment, and interact.

Once submitted, the "Bubble Sort Algorithm in Python" code snippet is displayed on the platform with the correct syntax highlighting. Other users can now view the code, leave comments, and interact with the code snippet, as shown in Figure 4. The tags help in categorizing and searching for the code snippet within the platform as shown in Figure 5.

Title

Bubble Sort Algorithm in Python

```
def bubble_sort(arr):  
    n = len(arr)  
    for i in range(n):  
        for j in range(0, n-i-1):  
            if arr[j] > arr[j+1]:  
                arr[j], arr[j+1] = arr[j+1], arr[j] # Swap if the element is greater  
    return arr  
  
# Example usage  
numbers = [64, 34, 25, 12, 22, 11, 90]  
sorted_numbers = bubble_sort(numbers)  
print("Sorted array is: ", sorted_numbers)
```

Choose Formatting:
Python

Tags
Search for a tag...

Upload Image
Choose File No file chosen

Add Tag

Python X Sorting X

Submit

Fig. 3. Creating a new code snippet and adding details (e.g. title, code, tags)

URL

http://192.168.1.101:3000/after_submit.html?snip_id=78

COPY

Tags: Python Sorting

Bubble Sort Algorithm in Python

Export Code Choose Formatting: Python

```
def bubble_sort(arr):
    n = len(arr)
    for i in range(n):
        for j in range(0, n-i-1):
            if arr[j] > arr[j+1]:
                arr[j], arr[j+1] = arr[j+1], arr[j] # Swap if the element is greater
        return arr

# Example usage
numbers = [64, 34, 25, 12, 22, 11, 90]
sorted_numbers = bubble_sort(numbers)
print("Sorted array is:", sorted_numbers)
```

Attached Files

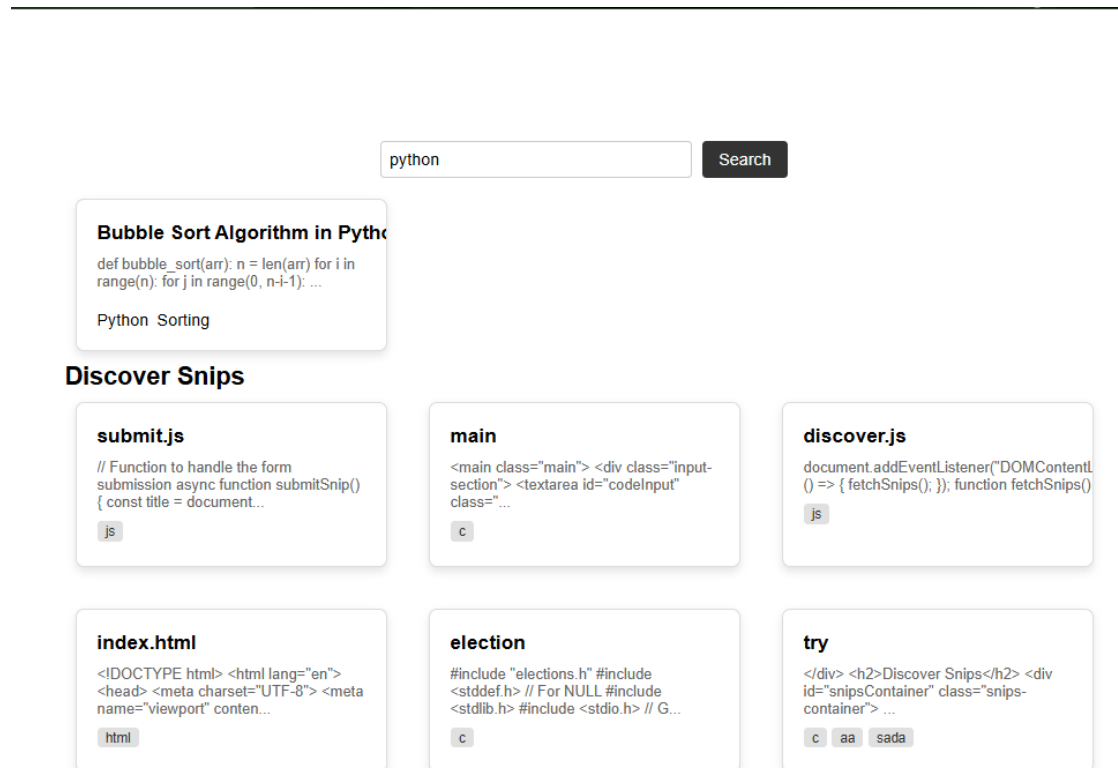
No files attached.

Comments

Add a comment...

Submit

Fig. 4. Visualizing and sharing a code snippet

**Fig. 5.** Searching for code snippets

5.2 Scenario 2: Reporting an Error with Code and Screenshot

In this scenario, the user posts a code snippet that results in an error during execution. Alongside the code, the user uploads a screenshot of the error message to seek assistance from the community. The exact steps are shown below.

1. Step 1: The user accesses the submission page and enters the code that causes the error.

The user provides the following details:

- **Title:** "Java NullPointerException Error"
- **Code Snippet:**

```
public class Main {
    public static void main(String[] args) {
        String str = null;
        System.out.println(str.length());
    }
}
```

- **Tags:** The user selects "Java" and "Error Handling" as the relevant tags.
- **Formatting:** The user selects "Java" as the Formatting.
- **Images:** The user provides an image "Scenario2error.png" highlighting the error output.

After filling in the details, the user submits the post. The first two steps are visualized in Figure 6.

2. Step 2: The submission allows others to review the code and provide suggestions or solutions in the comments section.

Once submitted, the code snippet is displayed on the platform. Other users can review the code and the error screenshot, and provide suggestions or solutions in the comments section. For example, a user might comment:

To fix the NullPointerException, ensure that the 'str' variable is initialized properly before calling the 'length()' method. For example:

```
String str = "Hello, World!";
System.out.println(str.length());
```

All of the above are shown in Figure 7

Title

Java NullPointerException Error

```
public class Main {  
    public static void main(String[] args) {  
        String str = null;  
        System.out.println(str.length());  
    }  
}
```

Choose Formatting:
Java

Tags
Search for a tag...

Upload Image
Choose File Scenario2error.png

Add Tag

Java X Error Handling X

Submit

Fig. 6. Create a snippet with a code snippet in Java

URL

http://192.168.1.101:3000/after_submit.html?snip_id=79

COPY

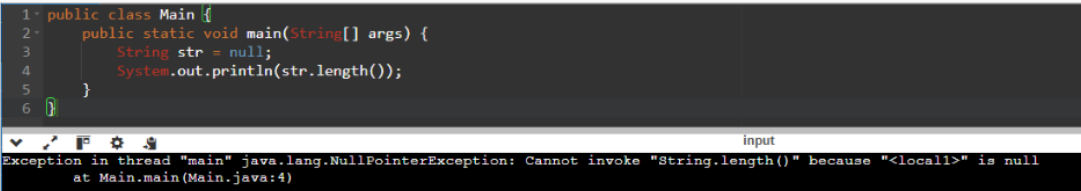
Tags: java Error Handling

Java NullPointerException Error

Export Code Choose Formatting: Java

```
public class Main {  
    public static void main(String[] args) {  
        String str = null;  
        System.out.println(str.length());  
    }  
}
```

Attached Files



Comments

user: To fix the NullPointerException, ensure that the 'str' variable is initialized properly before calling the 'length()' method. For example: String str = "Hello, World!"; System.out.println(str.length()); (2/11/2025, 12:02:20 PM)

Add a comment...

Submit

Fig. 7. Visualizing a code snippet with image, and comments

6 Related Work

In this section, we discuss existing code and text-sharing platforms that are similar to *SnipShare*. These platforms have been widely used for sharing and storing snippets of code, text, and other information. We also highlight the key differences between our application and these existing tools.

6.1 PasteBin

PasteBin [1], is one of the oldest and most widely used online tools for storing and sharing plain text and code snippets. Users can paste their text or code, select a syntax highlighting option, and share a generated URL. PasteBin supports public, private, and unlisted pastes.

Key Features:

- Syntax highlighting for various programming languages.
- Option to make pastes public, private, or unlisted.
- Ability to set expiration times for pastes.

Limitations:

- Limited tagging and search functionality.
- No support for image uploads or rich media.
- User interface is not customizable.

6.2 Hastebin

Hastebin [4] is a lightweight and minimalist paste tool designed for quick and easy sharing of text and code snippets. It offers a simple user interface and supports syntax highlighting.

Key Features:

- Minimalist interface with fast paste creation.
- Supports syntax highlighting for code.
- URL generation for easy sharing.

Limitations:

- No user accounts or authentication.
- Pasted content has a limited retention period.
- No support for comments, tagging, or image uploads.

6.3 Rentry.co

Rentry.co [3] is a markdown-based paste tool that allows users to create and share formatted text and code snippets. It supports markdown for formatting, making it ideal for documentation and rich text.

Key Features:

- Markdown support for rich text formatting.
- Custom URLs for pastes.
- Option to edit pastes with a secret key.

Limitations:

- Limited support for code-specific features.
- No syntax highlighting for advanced languages.
- No tagging or search functionality.

6.4 Paste.ee

Paste.ee [2] is a paste tool that supports rich text, images, and various other media types. It offers user accounts and privacy controls for managing pastes.

Key Features:

- Supports rich text and image uploads.
- Privacy settings for controlling access to pastes.
- User accounts for managing pastes.

Limitations:

- Limited search and tagging functionality.
- Interface can be overwhelming for new users.
- No collaborative features.

6.5 How *SnipShare* Differentiates

SnipShare offers a unique blend of features that address the limitations of existing platforms. The key differentiating factors are:

- **Advanced Tagging System:** Allows users to tag their snips for better organization and searchability.
- **Image Support:** Users can upload images to accompany their code snippets, useful for including diagrams or screenshots.
- **User-Friendly Interface:** A clean and intuitive interface with support for both light and dark modes.
- **Comments and Collaboration:** Enables users to comment on snips, fostering collaboration and discussion.

- **Customizable Backend:** The backend is built with Node.js and MySQL, making it easy to deploy and customize for different environments.
- **Discoverability:** Provides a “Discover” feature that allows users to explore popular and recent snips.
- **Configuration Flexibility:** Uses a configuration file to easily switch between local and remote databases or servers.

These features make *SnipShare* a versatile tool for developers, students, and anyone who needs a comprehensive platform for sharing and managing code snippets.

7 Conclusion

In this paper, we presented *SnipShare*, a web-based platform designed for developers to share, organize, and manage code snippets. The platform integrates essential features such as tagging, image uploads, and collaboration through comments, which set it apart from existing solutions like PasteBin, Hastebin, Rentry.co, and Paste.ee.

Our implementation leverages modern technologies including Node.js for the backend, MySQL for the database, and Express for handling API endpoints. The user interface is intuitive and supports both light and dark modes to enhance user experience. Additionally, it offers configuration flexibility, making it adaptable to both local and remote deployment scenarios.

7.1 Future Work

While **SnipShare** meets the current requirements for code snippet sharing and management, there are several areas for future improvements:

- **User Authentication:** Implementing user accounts and authentication to provide personalized experiences and manage user contributions.
- **Syntax Highlighting for More Languages:** Expanding support for additional programming languages and integrating advanced syntax highlighting libraries.
- **Real-Time Collaboration:** Enabling real-time editing and collaboration for multiple users on the same snippet.
- **Enhanced Search Functionality:** Adding advanced search features, including filtering by tags, languages, and user contributions.
- **Mobile Optimization:** Improving the platform’s responsiveness and usability on mobile devices.
- **API Documentation:** Providing comprehensive API documentation for developers who wish to integrate it into their workflows.

We believe these enhancements will further solidify *SnipShare* as a robust and versatile tool for the developer community.

References

1. Pastebin: Pastebin - 1 paste tool since 2002! (2025), <https://pastebin.com/>, accessed: 2025-02-07
2. Paste.ee: Paste.ee - Advanced Pastebin (2025), <https://paste.ee/>, accessed: 2025-02-07
3. Rentry: Rentry - Markdown Pastebin (2025), <https://rentry.co/>, accessed: 2025-02-07
4. Toptal: Hastebin - Simple Text Sharing (2025), <https://www.toptal.com/developers/hastebin>, accessed: 2025-02-07

A API Documentation

This section provides details about the API calls and their example JSON responses used in *SnipShare* .

A.1 Submit Snip with Tags

POST /submit-snip-with-tags

Request Body:

```
{
  "title": "Sample Snip",
  "snip": "console.log('Hello, World!');",
  "language": "javascript",
  "tags": ["js", "example"]
}
```

Response:

```
{
  "snip_id": 123,
  "message": "Snip created successfully"
}
```

A.2 Get Snip by ID

GET /api/snip/:id

Response:

```
{
  "snip_id": 123,
  "title": "Sample Snip",
  "snip": "console.log('Hello, World!');",
  "language": "javascript",
  "tags": ["js", "example"],
  "created_at": "2025-01-01T12:34:56Z"
}
```

A.3 Get All Snips for Discovery

GET /api/discover-snips

Response:

```
[
  {
    "snip_id": 123,
    "title": "Sample Snip",
    "preview": "console.log('Hello, World!');",
    "tags": ["js", "example"]
  },
  {
    "snip_id": 124,
    "title": "Another Snip",
    "preview": "print('Hello, Python!')",
    "tags": ["python", "example"]
  }
]
```

A.4 Add Comments

POST /api/comments

Request Body:

```
{
  "snip_id": 123,
  "comment": "This is a great example!"
}
```

Response:

```
{
  "message": "Comment added successfully"
}
```

A.5 Search Tags

GET /api/tags/search

Response:

```
[
  { "tag_name": "javascript" },
  { "tag_name": "java" },
  { "tag_name": "json" }
]
```