SAVANA

An Interactive Learning Platform

Bsc. Software Development

Kamau Samuel Gachunga

19/02761

Supervisor: Prof. Ezekiel Kuria

Table of Contents

Abstract

online learning, also known as e-learning has emerged as a transformative approach to education in the digital age. It leverages the power of technology and the internet to deliver educational content and facilitate learning experiences beyond the traditional classroom setting.

Online learning offers numerous advantages, including flexibility in scheduling, geographical independence, and personalized learning experiences. Learners can access educational resources and participate in courses from anywhere, at any time, using various devices.

However, online learning also poses challenges. It requires learners to be self-motivated, disciplined, and proactive in managing their own learning journey. Technical issues, such as unreliable internet connectivity or inadequate access to technology, can create barrier to participation. Maintaining learner engagement and interaction can also be challenging without face-to-face interactions and the immediate support of instructors.

Introduction

The aim if this project is to leverage the power of interactive screencasts, to enhance the learning experience for students and learners. Interactive screencasts combine the benefits of video tutorials with hands-in coding practice, allowing users to actively engage with the content in real-time. By intergrating interactive elements directly into the screencasts, this projects seeks to create an immersive and effective learning environment.

The utilization of interactive screencasts offers several advantages over traditional video tutorials. Learners can actively participate in the learning process, reinforcing their understanding by experimenting with the concepts being taught. This hands-on approach fosters greater comprehension and retention of knowledge. Additionally, interactive screencasts allow for personalized learning experiences, as learners can progress at their own pace, revisit specific sections, and tailor their learning journey to their individual needs.

Through this project, I aim to create a user-friendly platform that seamlessly integrates interactive screencasts into a diverse range of educational content. By combining the benefits of engaging video tutorials, hands-on coding practice, and interactive elements, I aim to revolutionize online learning and provide learners with an immersive and effective way to acquire new skills and knowledge across various disciplines.

Literature Review

In 2018, Scrimba.com, an interactive screencast learning platform known for its engaging web development tutorials started providing exceptional web development resources. It is essential to note that its content primarily centers around web development techniques and may not cover a comprehensive range of programming languages or other technical domains. As such, for learners seeking in-depth knowledge beyond web development, exploring additional educational platforms with broader content may be beneficial.

In 2020, Jungkook Park, Yeong Hoon Park, and Alice H. Oh present an innovative editing tool equipped with a non-linear editing algorithm for text-based screencasts. Their tool empowers users to modify any part of a text-based screencast while preserving overall consistency, addressing the challenge of ambiguity that arises when text editing operations are interconnected in sequence. This feature allows users to make edits with precision, ensuring that changes do not disrupt the coherence of the screencast.

In their research, Shinpei Hayashi, Takayuki Omori, Teruyoshi Zenmyo, Katsuhisa Maruyama, and M. Saeki propose a concept and technique for refactoring the edit history of source code, along with an automated tool to support the process. The primary goal of their history refactoring approach is to enhance the clarity and usefulness of the code history while preserving its overall effect. They define primitive history refactorings with preconditions and procedures, as well as larger refactorings composed of these primitives.

Yi Li, Chenguang Zhu, Julia Rubin, and Marsha Chechik propose a divide-and-conquer-style partitioning approach enhanced by dynamic delta refinement to achieve minimal semantic history slices. By utilizing dynamic invariants generated from successive test executions, their technique accurately ranks changes based on relevance to the target functionality, resulting in efficient and effective partitioning of history slices. This research contributes valuable insights into semantic history slicing, which could have practical implications in interactive learning platforms for software development by enabling learners to focus on relevant changes related to specific functionalities.

SOFTWARE DESIGN SPECIFICATION (SDS)

The purpose of this Software Design Specification (SDS) is to outline the design and architecture of the Interactive Screencast Learning Platform. It provides a comprehensive overview of the system's functionality, components, and interactions.

System overview

The Interactive Screencast Learning Platform will consist of several main components, including:

a. User Interface: The user interface will provide an intuitive and user-friendly interface for learners to access and interact with the interactive screencasts. It will include features such as video playback, code editor, interactive elements, and progress tracking.

b. Interactive Screencast Player: The interactive screencast player will enable learners to watch the screencasts while actively participating in the learning process. It will support real-time code editing, allowing learners to modify code examples and instantly see the results.

c. Content Management System (CMS): The CMS will facilitate the creation, organization, and management of interactive screencasts. It will provide an interface for content creators to upload videos, add interactive elements, and structure the learning modules.

d. User Management: The user management component will handle user authentication, registration, and profile management. It will allow learners to track their progress, save their work, and access personalized recommendations based on their learning history.

Functional Design

3.1 User Authentication and Profiles

Users will be able to create accounts, log in, and manage their profiles. User authentication will be implemented using secure protocols (e.g., OAuth, JWT) to ensure data confidentiality and integrity.

3.2 Screencast Management

Admin users will have the ability to upload and manage screencasts. Each screencast will consist of video content, synchronized code snippets, and interactive elements. The platform will support various multimedia formats (e.g., videos, audio, images) to accommodate diverse learning materials.

3.3 Interactive Elements

The platform will allow content creators to embed interactive elements directly into the screencasts. Examples of interactive elements include code editors, quiz questions, drag-and-drop exercises, and real-time preview panes. Users will be able to interact with these elements, modify code, and observe immediate results.

3.4 User Progress Tracking

The system will track user progress, recording completed screencasts, exercises, and achievements. Learners will have access to their personalized learning history, enabling them to resume where they left off and track their learning journey.

Non-functional Design

4.1 Performance

The system should handle a large number of concurrent users and deliver screencasts with minimal latency. Caching mechanisms and content delivery networks (CDNs) may be employed to optimize performance.

4.2 Security

The platform should implement robust security measures to protect user data and prevent unauthorized access. This includes secure user authentication, encryption of sensitive information, and adherence to security best practices.

4.3 Scalability

The architecture should be designed to accommodate future growth and increased user demand. The system should be scalable, allowing for the addition of more servers or cloud resources as needed.

SOFTWARE REQUIREMENTS SPECIFICATION (SRS)

2.1 Functional Requirements

User Management:

User Registration: Users should be able to create accounts by providing necessary information.

User Login: Users should be able to log in using their credentials.

User Profiles: Users should have personalized profiles to manage their preferences and track their progress.

Screencast Management:

Screencast Upload: Admin users should be able to upload screencasts with video content, code snippets, and interactive elements.

Screencast Categorization: Screencasts should be categorized based on topics or domains.

Search and Filtering: Users should be able to search and filter screencasts based on keywords, categories, and other relevant criteria.

Screencast Playback: Users should be able to view screencasts with synchronized code snippets and interactive elements.

Interactive Elements:

Code Editor: Users should be able to interact with embedded code snippets, make modifications, and see real-time results.

Quizzes and Exercises: Users should be able to complete quizzes and exercises embedded within the screencasts.

Drag-and-Drop Interactions: Users should be able to perform drag-and-drop exercises as part of the learning process.

Progress Tracking:

User Progress Monitoring: The system should track user progress, including completed screencasts, exercises, and achievements.

Resume Capability: Users should be able to resume from where they left off in their learning journey.

2.2 Non-functional Requirements

Performance: The system should provide responsive and fast loading times to ensure a smooth learning experience.

Security: User data should be securely stored and transmitted, using encryption and following best security practices.

Scalability: The system should be able to handle increasing user load and accommodate future growth.

User Interfaces

The user interface should be intuitive, visually appealing, and accessible across different devices (e.g., desktops, tablets, mobile devices). It should provide easy navigation, clear instructions, and a seamless experience for users to interact with screencasts and complete exercises.

System Constraints

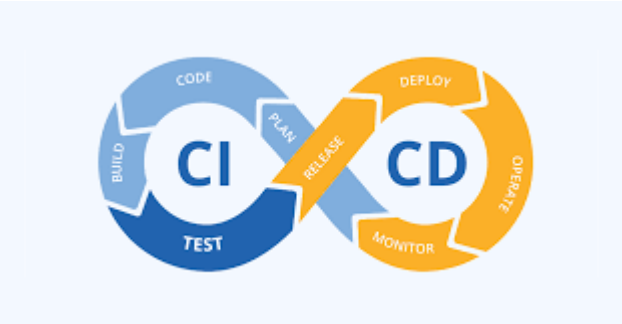
Technology Stack: The system should be developed using specific technologies, frameworks, and programming languages (e.g., HTML5, CSS, JavaScript, Python, Django).

Compatibility: The system should be compatible with major web browsers (e.g., Chrome, Firefox, Safari) and accessible on different operating systems.

Architecture and Design

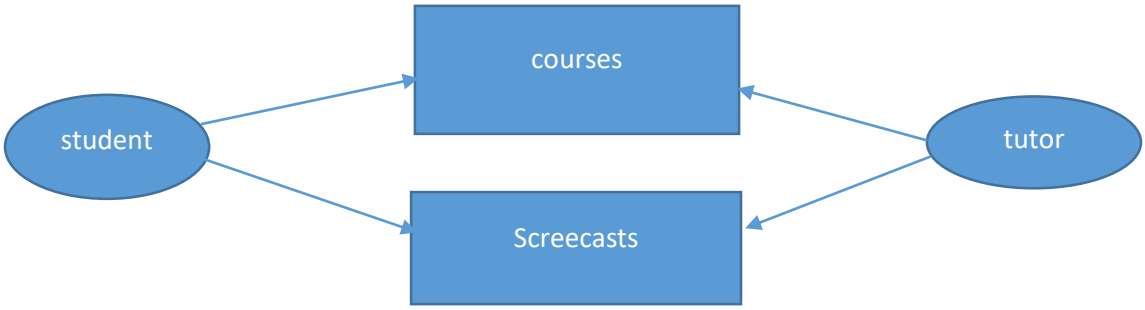
Workflow

In the development process, Continuous Integration and Continuous Deployment (CI/CD) were employed to streamline and automate the software development lifecycle. CI/CD pipelines were set up to automatically build, test, and deploy code changes, ensuring that new features and bug fixes were thoroughly tested and promptly delivered to the production environment, minimizing manual intervention and accelerating the delivery of updates to the interactive learning platform.

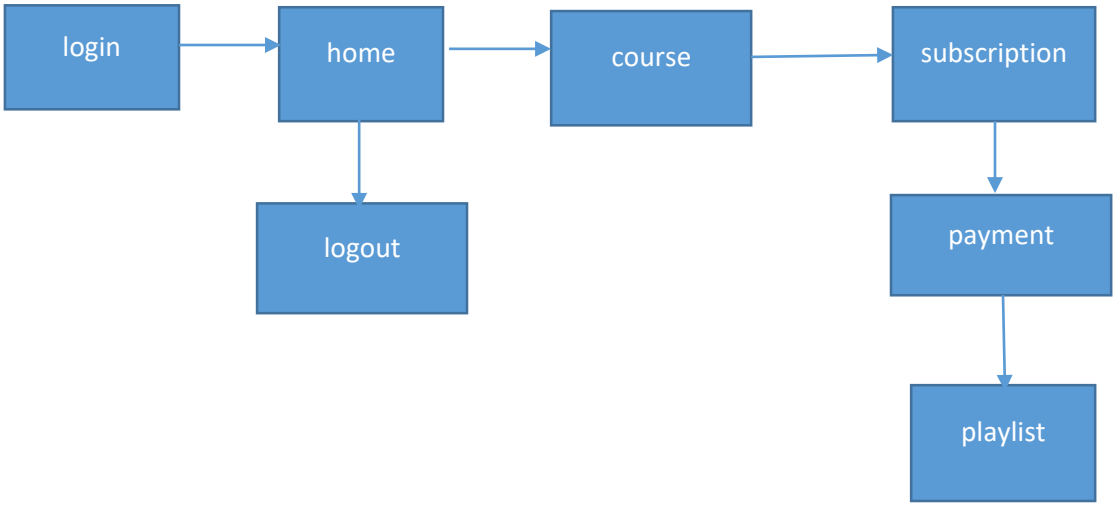


Class Diagram

The class diagrams showcase the relationships and associations between the "Student" and "Tutor" classes, indicating that both user types can access "Screencasts" and enroll in "Courses." This design ensures that both students and tutors have equal access to the educational content available on the platform, fostering a collaborative learning environment.


courses
student
tutor
Screecasts

Wireframe


login
home
course
subscription
logout
payment
playlist

Implementation

For the implementation of my interactive learning platform, I utilized a variety of technologies and libraries, including jQuery, npm, Prism, Skulpt, JavaScript, HTML, and CSS. One of the key pieces of code we implemented is a custom extension for jQuery, which adds a method to retrieve the unique path of a given DOM element.

```
// import $ from "jquery";

window.$ = window.jQuery;

var $ = window.$;

console.log("imported", $);

$.fn.getPath = function() {
 // Get path of element
 // borrowed from http://stackoverflow.com/a/2068381/1376627
 if (this.length !== 1) throw new Error("Requires one element.");
 var path,
  node = this;
 while (node.length) {
  var realNode = node[0],
   name = realNode.localName;
  if (!name) break;
  name = name.toLowerCase();
  var parent = node.parent();
  var siblings = parent.children(name);
  if (siblings.length > 1) {
   name += ":eq(" + siblings.index(realNode) + ")";
  }
  path = name + (path ? ">" + path : "");
  node = parent;
 }
 return path.split("html>")[1];
}export default $;
```

Testing

In the implementation of our interactive learning platform, we employed unit testing as an integral part of the development process to ensure the reliability and correctness of our codebase. Unit testing involves testing individual components or units of code in isolation, verifying that each unit functions as intended and meets its specified requirements.

For this project, we utilized unit testing frameworks, such as Mocha and Chai, to write and execute tests for our JavaScript code. Here's how we utilized unit testing throughout the development:

1. **Test Coverage**: We aimed to achieve high test coverage by writing test cases to cover various scenarios and edge cases for each function and module in our codebase. This helped us identify and address potential bugs or errors early in the development process.

2. **Testing jQuery Extensions**: Since we extended the jQuery library with a custom `getPath` method, we created unit tests specifically to verify the accuracy of this extension. These tests evaluated whether the method correctly returned the unique path of a given DOM element for different cases, such as elements with different nesting levels and those with sibling elements.

3. **Integration Testing**: In addition to unit tests, we performed integration testing to ensure that different components of the platform work seamlessly together. We tested interactions between modules and components to verify that they correctly communicate and exchange data as intended.

4. **Automated Testing**: We automated the execution of unit tests, enabling us to run them automatically whenever code changes were made. This automation reduced the manual testing effort and provided immediate feedback on code changes, ensuring that any regressions were promptly identified and addressed.

5. **Continuous Integration (CI):** To streamline the testing process, we integrated unit tests into our CI/CD pipeline. This way, every code commit triggered a series of automated tests, ensuring that the main branch always remained stable and functional.

6**. Test-Driven Development (TDD):** In some cases, we adopted a test-driven development approach, where we first wrote unit tests for new features or bug fixes before implementing the actual code. This approach guided our development process, ensuring that the code met the specified requirements outlined in the tests.

By utilizing unit testing throughout the development of our interactive learning platform, we were able to improve code quality, identify and fix issues early, and maintain a reliable and robust codebase. The comprehensive test suite provided us with confidence in the correctness of our implementation and contributed to a more stable and user-friendly learning platform.

Deployment and Installation

If you want to run the platform locally, you can first clone the repository from git@github.com:treezy254/S-editor.git

Then install the necessary dependencies with

```
npm i
```

```
npm start
```

To build and run for productions, you can use

```
npm run build
```

Maintenance and Support

1. Software Updates and Bug Fixes:

As the interactive learning platform evolves and adapts to meet the needs of learners and educators, regular software updates and bug fixes will be essential. The development team commits to maintaining an agile development process to address any reported issues promptly and provide regular updates to enhance the platform's functionality, performance, and security.

2. Version Control and Change Management:

To ensure transparency and accountability in the maintenance process, the project will adopt version control systems, such as Git, to track code changes and manage software updates effectively. Any modifications, improvements, or bug fixes will be documented in detail, enabling stakeholders to understand the changes made and facilitating seamless collaboration among developers.

3. User Support and Feedback:

Providing exceptional user support is a cornerstone of our commitment to the interactive learning community. We will establish user support channels, including email support and community forums, where learners and educators can seek assistance, report issues, and offer feedback. Timely responses and proactive engagement with user feedback will enable us to continuously improve the platform and address user needs effectively.

4. Security and Data Privacy:

As the custodians of user data, we prioritize the security and privacy of our users. We will employ robust security measures to safeguard user information and ensure compliance with relevant data

protection regulations. Regular security audits and vulnerability assessments will be conducted to maintain a secure environment for our users.

5. Compatibility and Accessibility:

The interactive learning platform will be designed to be compatible with a wide range of devices and web browsers to cater to diverse user preferences. Additionally, we will ensure that the platform complies with accessibility standards to make it inclusive and accessible to users with disabilities.

6. Scalability and Performance:

As the user base grows, we are committed to ensuring the platform's scalability and performance. Regular performance monitoring and optimization efforts will be carried out to maintain a smooth and responsive learning experience, even during peak usage.

7. Documentation and Knowledge Base:

A comprehensive and up-to-date documentation and knowledge base will be maintained to assist users and developers with platform functionalities, troubleshooting, and best practices. This documentation will serve as a valuable resource to help users navigate the platform effectively.

8. Continuous Improvement:

The interactive learning platform will be a living project that continuously evolves based on user feedback and emerging technologies. We will actively engage with our community of learners and educators to gather insights, identify areas for improvement, and implement new features that align with the evolving needs of our users.

Conclusion

In conclusion, our commitment to maintenance and support ensures that the interactive learning platform remains a reliable, secure, and user-friendly environment for learners and educators alike. By staying proactive in addressing issues, offering robust user support, and embracing continuous improvement, we aim to provide an exceptional and enriching learning experience for all users of the platform.

References

Tufano, 2018 – 2018 IEEE/ACM 15th International Conference on Mining Software Repositories (MSR)

Park, 2018 – ACM Symposium on User Interface Software and Technology

Li, 2016 – automated software engineering

Hayashi, 2012 – 2012 28th IEEE International Conference on Software Maintenance (ICSM)

Saca, 2017 – 2017 IEEE 37th Central America and Panama Convention (CONCAPAN XXXVII)

https://codesandbox.io/s/jquery-playground-forked-suqgzz?file=/src/jquery.js

scrimba.com