

Code Docs Automation Technical Report 2020

National College of Ireland, BSc in Computing 2020/2021
Lucian Nechita, x16149505@student.ncirl.ie
Student ID: x16149505

Disclaimer

December 2020/May 2021

This technical report was developed by Lucian Nechita as part of the final project assessment for BSc (Hons) in Computing (Software Development Stream) at National College of Ireland. For further information contact National College of Ireland <https://www.ncirl.ie/>

The information and views set out in this report are those of the author(s) and do not necessarily reflect the official opinion of the National College of Ireland. Neither the National College of Ireland institution and bodies nor any person acting on their behalf may be held responsible for the use which may be made of the information contained therein.

Contents

1. Objective	4
2. Opportunity	4
3. Solution	4
Introduction	5
4. Project Background	5
5. Project Aims	5
6. Project technologies	5
System	7
1. Requirements	7
a. Functional Requirements	7
b. Use Case Modelling	8
c. Data Requirements	19
d. Data Conversion	20
e. Environmental Requirements	21
f. Non-functional Requirements	21
g. Usability Requirements	21
2. Design & Architecture	22
a. System Architecture Back-End	22
b. System Implementation	23
c. User Interface	23
3. Testing	28
4. Evaluation	30
5. Conclusion	30
6. Further Development and Research	31
References	32
Appendices	34
1. Project proposal	34
2. Reflective Journal October	37
3. Reflective Journal November	38
4. Reflective Journal December	39
5. Reflective Journal January	40
6. Reflective Journal February	41
7. Reflective Journal March	42
8. Reflective Journal April	43
9. Showcase Poster	44
10. Attachments	45

Glossary

CI/CD	In software development CI stands for continuous integration and CD for continuous development. It is a software development practice where team members create and integrate their code frequently verified by an automation build including tests to detect errors and updates production environment code if no errors were detected. (Stolberg, 2009)
Production Environment	Production environment is a way for software developers to describe the setting where computer code is executed and put into operation with the goal of being used by end users. (Technopedia, n.d.)
End User/s	An end user is a person for which a device, software or system was designed to interact with. In simple terms it is the person who uses the software or hardware. (TechTerms, n.d.)
Software development pipeline	A software development pipeline is a set of automated tasks from which computer code is passed through from task to task following a set plan to achieve different goals at each step. (Azure, n.d.)
Code documentation	Code documentation is the process in which a programmer describes the code actions or how the system works. (Goldis, 2018)
API Documentation	API documentation is the process of explaining how to use a certain API endpoint, with details about arguments, types, returns etc. (Vasudevan, n.d.)
Cloud computing	Cloud computing is getting access to different computing resources as a service including databases, virtual machines, networking resources, software etc. (Microsoft, 2018)
Use case modelling	A use case model is a representation of how a user or different users interact with a system to access a service, solve a problem, or achieve a certain goal. (UTM, n.d.)
Software Development Automation	Software development automation is the process of transforming manual labour into automated systems which frees up developer time for more important tasks. (StackExchange, 2017)
Software Repository	Software repository is a change management tracker for project code building up a history with all changes made to the code over time and promotes easy collaboration within developer circles. (technopedia, 2016)
Web Application	A web application is a software program which functions on a web server and it is designed to be accessed by a browser with access to internet and installation free applications. (PCMAG, 2016)
Responsive Web Design	Responsive design is an approach to design application that should respond to user behaviour and environment. Content of the application should dynamically adjust and fit on the screen size on different devices with different screen sizes. (Smashing, 2016)

Executive Summary

This technical report provides the analysis and discussion of scope for the development of an automated code documentation system. The strategy deals directly with the code and other tools in gathering the information across platforms and building a production grade document.

1. Objective

The aim of this project is the creation of a system for automating code documentation process which will gather source documents and assemble them into a ready to deploy web application. The system will work within an automated development process, recognize repository changes, and automatically start a task within the software pipeline publishing the web application to the cloud and making it accessible to end users within minutes.

2. Opportunity

Programmers live in a world of plain text, either code syntax or writing documentation its all plain text which computer understands and process it into a different output. Most of the times we want to turn this output into an easy to understand and pretty format which humans can digest with ease. Also, as programmers we have the best tools to track changes to the code and we keep a close eye on the history of our changes but not much monitoring on the documentation side of the application causing outdated descriptions of the application features or code implementations.

The workflow for tracking code changes is very powerful and familiar to any developer so why would we forget about it when it comes to documenting our application?

3. Solution

The solution is to use that powerful workflow discussed above for the documentation side of the application. At its core, utodocs is a dynamic document bundler for modern applications. When utodocs processes your application documentation, it internally builds a table of contents graph which maps every markdown file in your project and generates http link's which point into your application GitHub repository.

Utodocs is an automated process for code documentation which is built at the same time with your application source-code. This enables developers to keep a close eye on the changes of the code and documentation, and the version of the documents will relate to the version of the source-code creating a one-to-one mapping. So, when the version of the source-code is changed, being for a new feature or downgrading the application to an older version; the documents will reflect the respective version of the software.

Introduction

4. Project Background

This technical report provides the analysis and discussion of scope for the development of an automated code documentation system strategy. The strategy deals directly with the management of code documentation at all stages of software development life cycle.

The purpose of this document is to provide the reader with in-depth detail of the project. The proposed project will give developers the ability to replace the manual task of gathering and assembling code documentation with an automated system to build and deploy production grade documents. Writing code functionality documentation takes as much time as writing the code if not longer. This takes precious software development time from the developer and having an automated system which takes care of code documentation will increase productivity and quality of the code.

5. Project Aims

The aim of this project is the creation of a system for automating code documentation process which will gather source documents from different stages of software development life cycle and assemble them into a ready to deploy web application. The system will work within an automated development process, recognise repository changes, and automatically start a task within the software pipeline publishing the web application to the cloud and making it accessible to end uses within minutes.

The automation tool should be coding language agnostic so any developer from any background can benefit.

The automation tool should be “learn free” so any developer can focus on what their best at instead of using company time to learn how to use our tool.

The automation tool should include all stages of the software development life cycle. Building software functionality usually include documents, scattered across different apps or places. For example, a UML use case modelling documentation from requirements elicitation, might be small but powerful into helping us building the documentation for a unit of functionality, like who can use it (actors), what they need(prerequisites), and describe the flow of events.

6. Project technologies

- *NodeJS* – is an open-source JavaScript runtime environment which allows programs written in JavaScript to run outside a browser, for example on a server or on a local machine environment. The project will make use of the NodeJS technology to access operating system resources and run the JavaScript files to parse and build the documentation and web application. The project will also use NodeJS to set up a back-end environment within the pipeline where most of the code documentation automation system will reside. (NodeJS, n.d.)
- *JavaScript* – is a popular programming language also known as the scripting language of the web. The project will make use of the JavaScript language as it is light-weight, high level language and makes it very fast to build software projects. The project will have a web-based component to display the web

application and as the web browsers mostly supports JavaScript language will make it more appropriate to use. (developer.mozilla, n.d.)

- *HTML and CSS Grid* – is the foundation of web applications, HTML stands for Hypertext-Markup-Language and it is designed to work in a browser and display web pages. Usually, it is assisted by CSS technology to improve the design of the webpages. This project will make use of CSS Grid which is a newer technology and can be used to create complex responsive layouts with more consistency between the browsers. (Andrew, 2018)
- *.MD File Types* – the project will make use of GitHub’s standard text files which is .MD or markdown files. It is plain text which include symbol to indicate special text like title or code text and it is used widely across repositories. The system will make use of this filed to parse its content and transform it into a HTML page with the special text indicated by symbol parsed accordingly to display it as a title or code block within our HTML document. (Markdown, 2017)
- *Concourse CI/CD* – it is an open-source software which runs within a Docker container to build an automated system and manipulate code files, test, or deploy applications to the web. The project will make use of the Concourse technology to set up a pipeline between local environment and remote repository, pipeline in which the system will gathers source documents and code, read, parse and test and finally assembly all the pieces of the application and deploy it to the cloud. (Concourse, 2017)
- *Mocha testing framework* – Mocha is a test framework for JavaScript code used for unit and integration testing of applications build in JavaScript. The project will make use of Mocha testing framework to test the code and assure that the previously build features of the application are still functioning as intended. This will increase development speed and system reliability. (Mocha, n.d.)
- *Test Driven Development* – Test driven development is a technique of developing software by building the tests before the functionality is build. Developing the failure tests before the functional code will increase the robustness of the code as functional code has to pass the tests before it even works first time. (Agile, n.d.)
- *Agile methodology* – Agile methodology is an incremental software development technique which encourages iterating quickly through the software development lifecycle, managing the project by breaking up the system in small pieces and incrementally work on each piece from design to deployment. (Alliance, 2018)
- *Gantt Chart* – A Gantt chart is a project management plan which helps software developers track progress and manage tasks to a successful project delivery. It is a visual view of work to be done and its dependencies showing a schedule across time with start and end dates for each task assisting the development of the system and keeping an eye on project deliveries. (Manager, 2020)
- *Heroku Cloud* – Heroku Cloud is a platform from Sales Force that offers computing as a service for building, testing, and managing applications. The project will make use of cloud computing technologies such as database, virtual machine, and network interface to deploy the web application and make it available to potential users without the need of installing the system. (Heroku, 2020)

System

1. Requirements

The following section of this paper will discuss the required specifications of the code documentation automation system. We will describe the functions which the system should fulfil to satisfy stakeholder needs expressed in a textual statement.

a. Functional Requirements

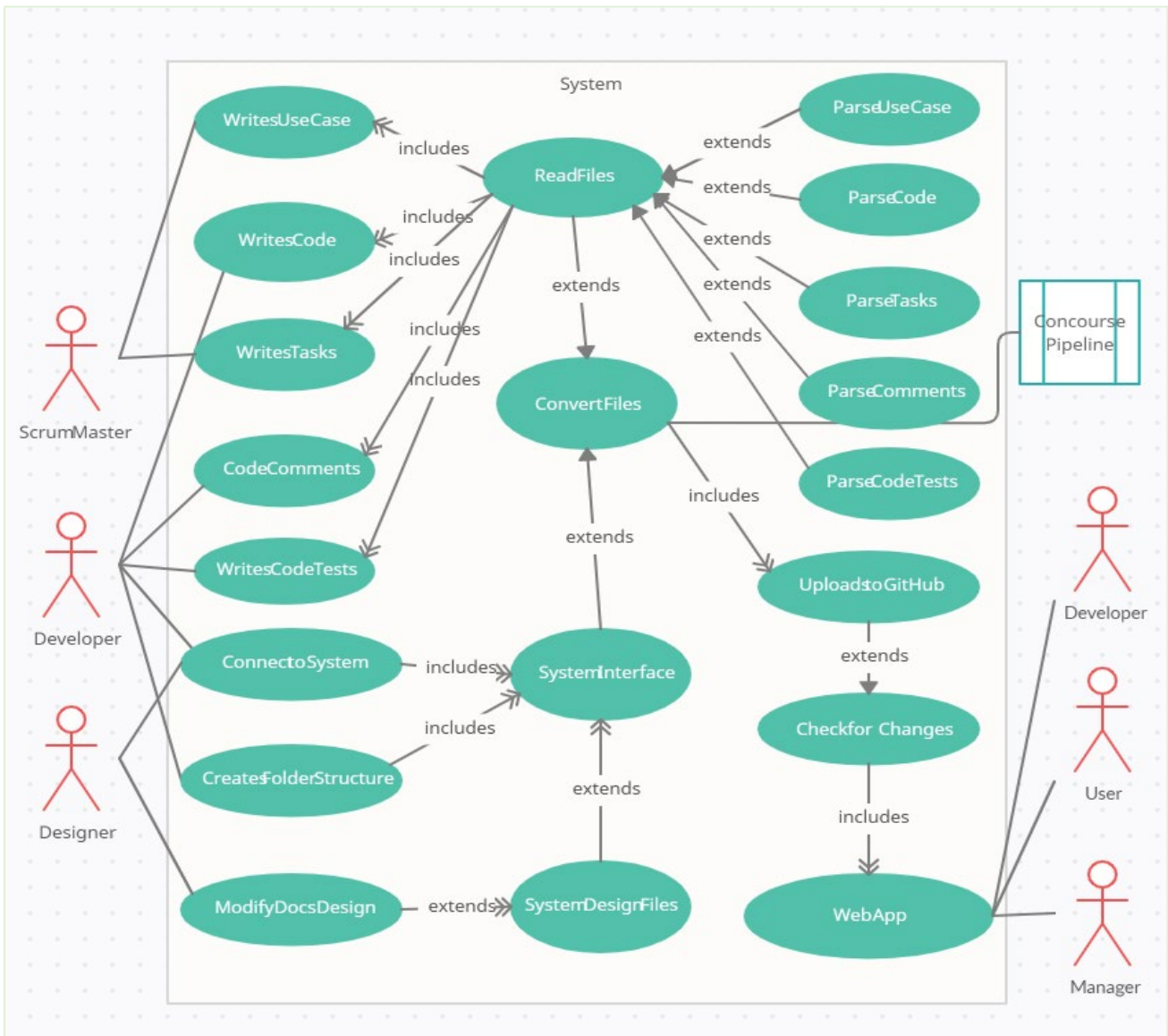
Functional requirements are a list of services a software must have on a component level of functional level. It defines a set of specific functionalities which a system is likely to perform with a given input and output expected. (GeeksForGeeks, 2017)

- User should be able to create a vertical navigation to place at the side of a page
The system will allow a user to create a vertical navigation bar for which the system will read the folder structure from the repository and convert the names of the folders into links within a side navigation bar of the web app.
- User should be able to create a top navigation bar
The system will allow users to customize the header of the web application by reading a predefined file within repository.
- User should be able to customize the design of the application
The system will offer full access to the styling files for the web application where a user can delete, add, or modify the looks, layout, or colours of the web app.
- User should be able to create .MD files
A user should be able to create markdown files within his own repository for which the system will offer connection through web Tokens and APIs.
- Convert markdown files to html
The system will read and convert the markdown text files into appropriate HTML, parsing markdown symbols and transform normal text into titles of different sizes, images, code blocks, external links etc.
- Ignore comments within the markdown files
The system will recognize comments within the markdown files and ignore them when parsing the text and outputting it to HTML syntax.
- Continuous integration development
The system will not reside with developer's code and will allow the user to operate the system through a continuous integration pipeline where it will perform most of the tasks.

- Continuous deployment
A user should be able to deploy to the cloud the web application responsible for code documentation and the system will have the necessary pipeline tasks in place for the deployment process to happen is chosen.
- System Independence
The system will be independent of operating system or platform or developer code base, it will set up its on server within a Docker image.
- Recognize code changes
The system will monitor any code changes happening in the repository and it will automatically build a new version of the documentation if any changes happened.

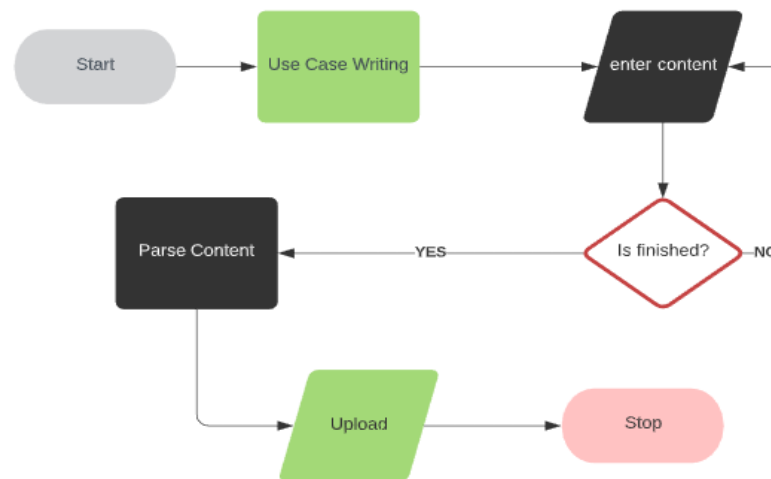
b. Use Case Modelling

The following use case diagram provides a simple representation of a user's interaction with the system and user's involvement with the system describing units of useful functionality performed by the code documentation automation system in collaboration with external actors.



Use Case	Write Use Cases
Description	The user will have the ability to create .MD files with Use Cases, very similar to the use cases on this paper
ID	CDAS001
Scope	The scope of this use case it to read and convert data from use cases docs to HTML
Actors	Scrum Master, GitHub

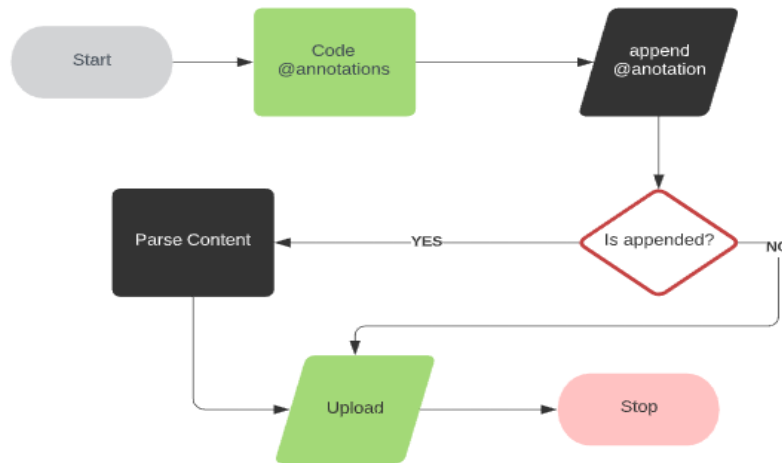
Algorithm
Diagram



- Flow Description
- Precondition
 - The user must use the predefined markdown file
 - Activation
 - Once it detects a non-empty predefined markdown file being uploaded to GitHub
 - Main flow
 - ✓ User writes use cases in the predefined markdown file (See A1)
 - ✓ User pushes the file to the GitHub repository
 - ✓ The system will intercept the file (See A2)
 - ✓ The system will parse the content to HTML describing system functionality
 - ✓ The system will commit the changes to GitHub (See A3)
 - Alternate Flow
 - ✓ A1 – The system will ignore the file and prompt the user for changes
 - ✓ A2 – If is no modification done to the file or simply not present the system will skip the parsing functionality
 - ✓ A3 – If user wants to skip this step and chose to deploy instead the system will deploy/update the web application on the cloud
 - Exceptional Flow
 - ✓ User manually stops the system, will display manually stopped message
 - ✓ The system encounters an unexpected error, it will restart and try to resume the job
 - Termination
 - System is stopped by a user
 - Post Condition
 - The system will sleep until new file is detected

Use Case	Write Code
Description	The user will have the ability to add a custom @anotation in addition to user's code
ID	CDAS002
Scope	The scope of this use case it to read and convert data from source-code files to HTML
Actors	Developer, GitHub

Algorithm Diagram

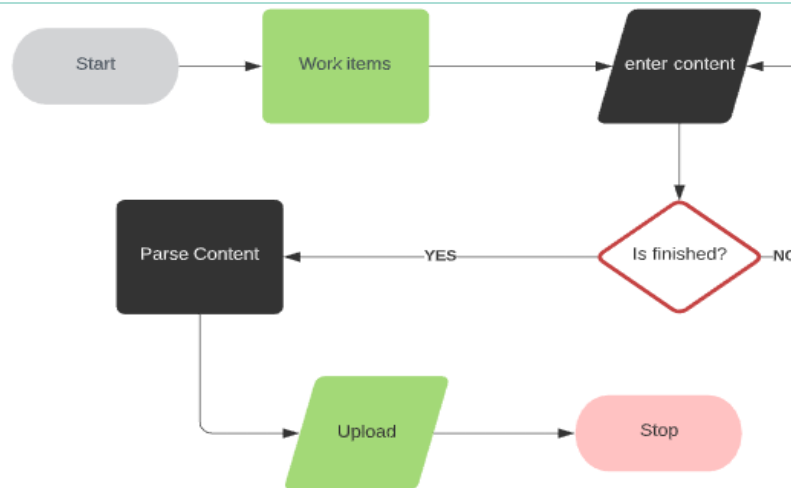


Flow Description

- Precondition
 - The user must use the @annotations within his own source code
- Activation
 - Once it detects @annotation in the code base file being uploaded to GitHub
- Main flow
 - ✓ User appends @annotation to the source code
 - ✓ User pushes the file to the GitHub repository
 - ✓ The system will intercept the source code file (See A1)
 - ✓ The system will parse the function name and parameters and transform them to HTML syntax describing functions functionality
 - ✓ The system will commit the changes to GitHub (See A2)
- Alternate Flow
 - ✓ A1 – If is no @annotation detected system will skip annotation parsing functionality
 - ✓ A2 – If user wants to skip this step and chose to deploy instead the system will deploy/update the web application on the cloud
- Exceptional Flow
 - ✓ User manually stops the system, will display manually stopped message
 - ✓ The system detects improper use of the @annotation and will notify the user of the modifications needed and correct the mistake
 - ✓ The system encounters an unexpected error, restart and resume the job
- Termination
 - System is stopped by a user
- Post Condition
 - The system will sleep until new @annotation within the file is detected

Use Case	Write Tasks
Description	The user will have the ability to write tasks or cards for developers with current and future work items
ID	CDAS003
Scope	The scope of this use case it to read and convert data from tasks/cards to HTML
Actors	Scrum Master, GitHub

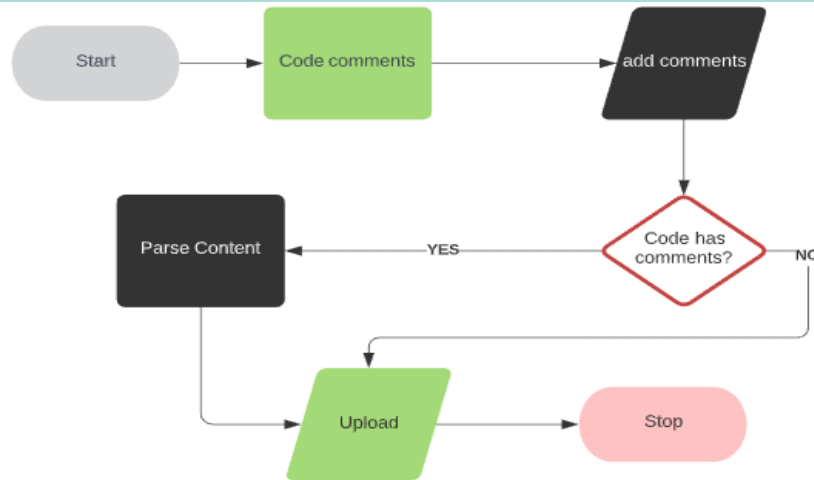
Algorithm
Diagram



- Flow Description
- **Precondition**
The user must use predefined markdown files specially created to serve this purpose
 - **Activation**
Once it detects the predefined markdown files being uploaded to GitHub
 - **Main flow**
 - ✓ User writes use work items in the predefined markdown file (See A1)
 - ✓ User pushes the file to the GitHub repository
 - ✓ The system will intercept the file (See A2)
 - ✓ The system will parse the content to HTML describing the functionality
 - ✓ The system will commit the changes to GitHub (See A3)
 - **Alternate Flow**
 - ✓ A1 – The system will ignore the file and prompt the user for changes
 - ✓ A2 – If is no modification done to the file or simply not present the system will skip the parsing functionality
 - ✓ A3 – If user wants to skip this step and chose to deploy instead the system will deploy/update the web application on the cloud
 - **Exceptional Flow**
 - ✓ User manually stops the system, will display manually stopped message
 - ✓ The system encounters an unexpected error, restart and resume the job
 - **Termination**
System is stopped by a user
 - **Post Condition**
The system will sleep until a new work task file is detected

Use Case	Code Comments
Description	The user will have the ability to write code comments within his source code and use them to build documentations content
ID	CDAS004
Scope	The scope of this use case it to read and convert data from code comments to HTML
Actors	Developer, GitHub

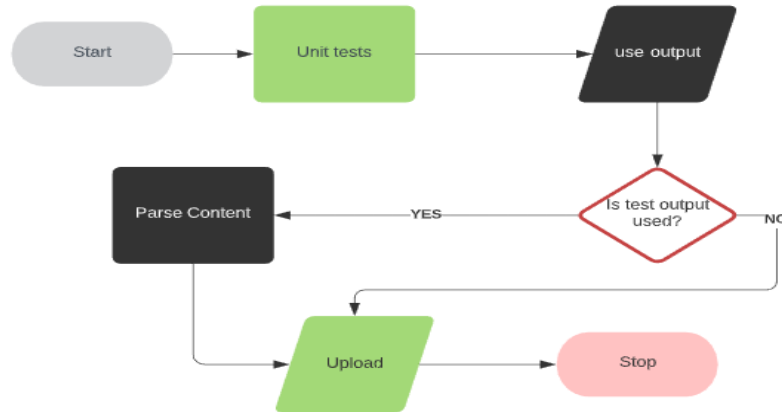
Algorithm
Diagram



- Flow Description
- Precondition
 - The user must use comments within his source code files
 - Activation
 - Once it detects any code comments in the source files
 - Main flow
 - ✓ User writes comments within his code base files
 - ✓ User pushes the file to the GitHub repository
 - ✓ The system will intercept the file (See A1, See A2)
 - ✓ The system will parse the content to HTML describing the functionality
 - ✓ The system will commit the changes to GitHub (See A3)
 - Alternate Flow
 - ✓ A1 – The system will allow to skip any comments within the source files
 - ✓ A2 – If is no modification done to the file or simply not present the system will skip the parsing functionality
 - ✓ A3 – If user wants to skip this step and chose to deploy instead the system will deploy/update the web application on the cloud
 - Exceptional Flow
 - ✓ User manually stops the system, will display manually stopped message
 - ✓ The system encounters an unexpected error, restart and resume the job
 - Termination
 - System is stopped by a user
 - Post Condition
 - The system will sleep until new comments are added to the file

Use Case	Code Tests
Description	The user will have the ability to write tests for his source files and use test results output as part of the code documentation
ID	CDAS005
Scope	The scope of this use case it to read and convert data from unit tests to HTML
Actors	Developer, Pipeline, GitHub

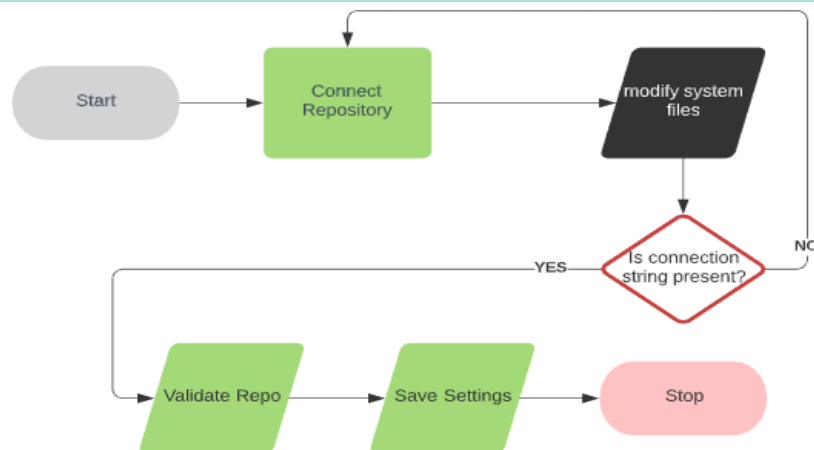
Algorithm Diagram



- Flow Description
- Precondition
 - The user must use unit tests to test his source code files
 - Activation
 - Once unit tests are run within the pipeline system functionality
 - Main flow
 - ✓ User writes unit tests for his source code files
 - ✓ User activates the use of unit tests output towards building the documentation
 - ✓ User pushes the file to the GitHub repository
 - ✓ The system will activate the pipeline and run the unit tests
 - ✓ The system will listen for unit tests output using operating system's STDIN/STDOUT data stream (See A1)
 - ✓ The system will save the output to OS memory
 - ✓ The system will parse the output to HTML describing the test results
 - ✓ The system will commit the changes to GitHub (See A3)
 - Alternate Flow
 - ✓ A1 – The system will allow to skip listening for test inputs/outputs and skip the parsing functionality
 - ✓ A3 – If user wants to skip this step and chose to deploy instead the system will deploy/update the web application on the cloud
 - Exceptional Flow
 - ✓ User manually stops the system, will display manually stopped message
 - Termination
 - System is stopped by a user
 - Post Condition
 - The system will sleep until new tests are created or modified

Use Case	Connect to the System
Description	The user will have the ability to connect his source code repository to the system through systems settings files
ID	CDAS006
Scope	The scope of this use case it to give the user the ability to connect with the system
Actors	Developer, Designer, GitHub

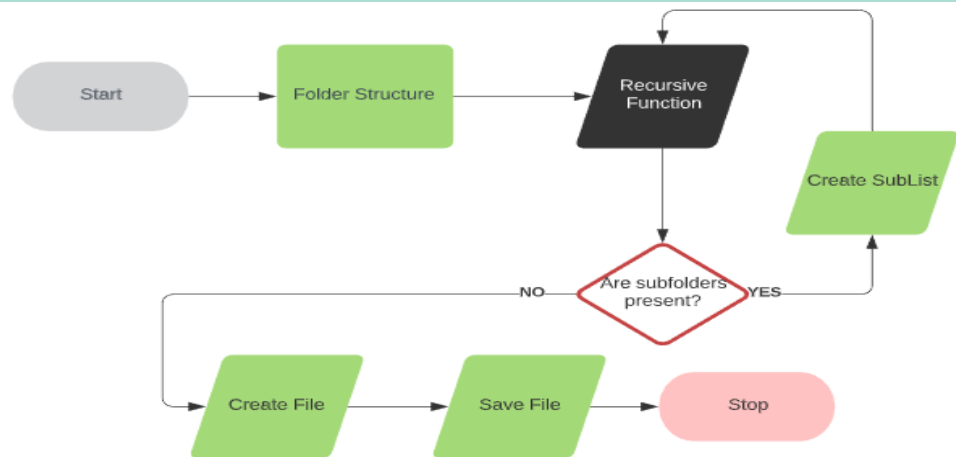
Algorithm Diagram



- Flow Description**
- **Precondition**
The user must download and install code docs automation system files
 - **Activation**
Once user starts the application by entering to any CLI the run command
 - **Main flow**
 - ✓ User opens GitHub repository and retrieves connection string
 - ✓ User opens the settings file provided by the system
 - ✓ User adds connection string to the settings file
 - ✓ User pushes the changes back to development repository
 - ✓ The system will activate the pipeline
 - ✓ The system will validate the connection with the repository (See A1)
 - ✓ The system will save the settings to the repository
 - **Alternate Flow**
 - ✓ A1 – If connection string is not valid the system will notify the user
 - The system will display login GUI to manually login to GitHub
 - The user can enter account name and password
 - The system activates the pipeline
 - **Exceptional Flow**
 - ✓ User manually stops the system, will display manually stopped message
 - ✓ GitHub refuses the connection, the system will notify the user and sleep
 - **Termination**
System is stopped by a user
 - **Post Condition**
The system will sleep until new push to GitHub branch is detected

Use Case	Create Side Navigation
Description	The user will have the ability to create a folder structure from which the system will read and translate the structure into navigation for the web app documentation
ID	CDAS007
Scope	The scope of this use case it to give the user the ability to create a custom navigation
Actors	Developer, GitHub

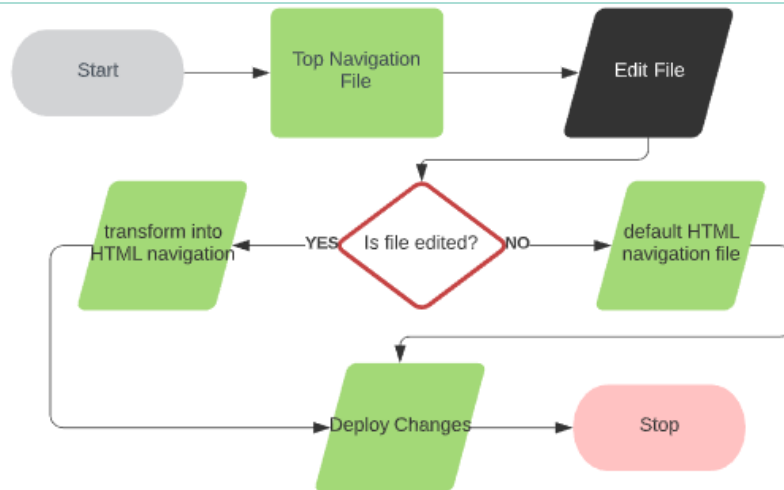
Algorithm Diagram



- Flow Description
- **Precondition**
The user must be connected to the system
 - **Activation**
Once user push his source code to the repo
 - **Main flow**
 - ✓ User creates a desired folder structure which will hold the documentation
 - ✓ User pushes new changes to his source code
 - ✓ The system will activate the pipeline
 - ✓ The system reads the folder structure (See A1)
 - ✓ The system creates a file holding the navigation structure
 - ✓ The system will parse the file and outputs navigation to HTML
 - ✓ The system will save the settings to the repository
 - **Alternate Flow**
 - ✓ A1 – If there are any subfolders the system will create a sub-list
 - The system will append indentation to each name of the folder
 - The system will write the result to a file
 - The system will continue with next folder reading the structure
 - The system will save the settings to the repository
 - **Exceptional Flow**
 - ✓ User manually stops the system, will display manually stopped message
 - **Termination**
System is stopped by a user
 - **Post Condition**
The system will sleep until new push to GitHub branch is detected

Use Case	Create Top Navigation
Description	The user will have the ability to create a file holding the structure of the top navigation bar from which the system will read and translate the structure into top navigation
ID	CDAS008
Scope	The scope of this use case it to give the user the ability to create a custom navigation
Actors	Developer, GitHub

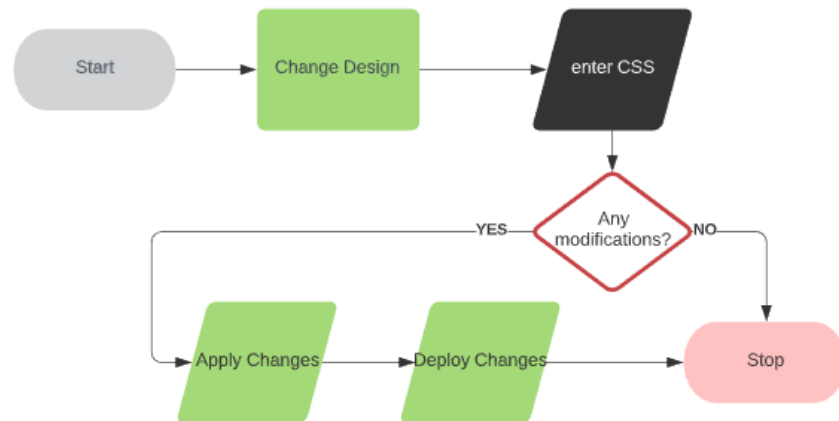
Algorithm Diagram



- Flow Description
- Precondition
 - The user must be connected to the system
 - Activation
 - Once user push his source code to the repo
 - Main flow
 - ✓ User creates a desired structure for the top navigation bar (See A1)
 - ✓ User pushes new changes to his source code
 - ✓ The system will activate the pipeline
 - ✓ The system reads the folder structure
 - ✓ The system will parse the file and outputs navigation to HTML
 - ✓ The system will save the settings to the repository
 - Alternate Flow
 - ✓ A1 – If user did not create the file with top navigation the system will provide a default navigation settings file
 - The system will parse the file and outputs navigation to HTML
 - The system will save the settings to the repository
 - Exceptional Flow
 - ✓ User manually stops the system, will display manually stopped message
 - Termination
 - System is stopped by a user
 - Post Condition
 - The system will sleep until new push to GitHub branch is detected

Use Case	Customise the Design of Documentation
Description	The user will have the ability to change the design of the documentation created by the system
ID	CDAS009
Scope	The scope of this use case it to give the user the ability to customize the design of the documentation
Actors	Designer

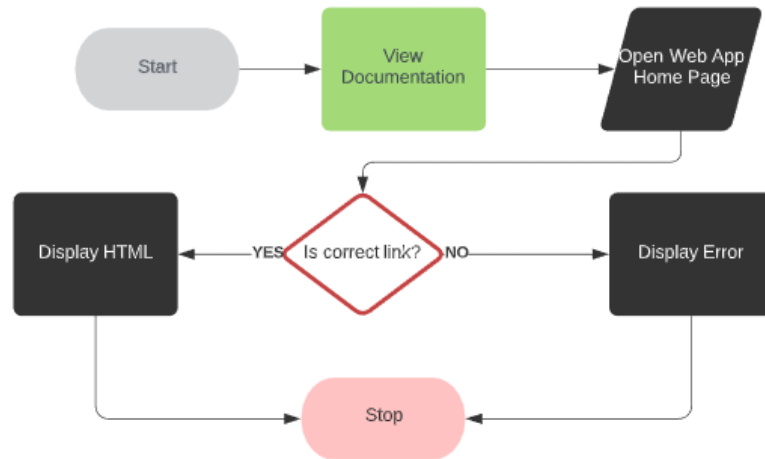
Algorithm Diagram



- Flow Description
- **Precondition**
The user must be connected to the system and a HTML file must be generated at least once
 - **Activation**
Once user push his source code changes to the repo
 - **Main flow**
 - ✓ User opens the CSS settings file on the local repository
 - ✓ User opens the HTML file
 - ✓ User makes the CSS changes needed
 - ✓ User pushes the new code changes to the repository
 - ✓ The system will activate the pipeline
 - ✓ The system reads the CSS modifications
 - ✓ The system applies the changes to the web app
 - ✓ The system deploys the changes
 - **Alternate Flow**
 - ✓ N/A
 - **Exceptional Flow**
 - ✓ User manually stops the system, will display manually stopped message
 - **Termination**
System is stopped by a user
 - **Post Condition**
The system will sleep until new push to GitHub branch is detected

Use Case	View Documentation
Description	The user will have the ability to view the deployed documentation available on the cloud
ID	CDAS010
Scope	The scope of this use case it to give the user the ability to view the documentation produced
Actors	Scrum Master, Designer, Developer, Manager, User

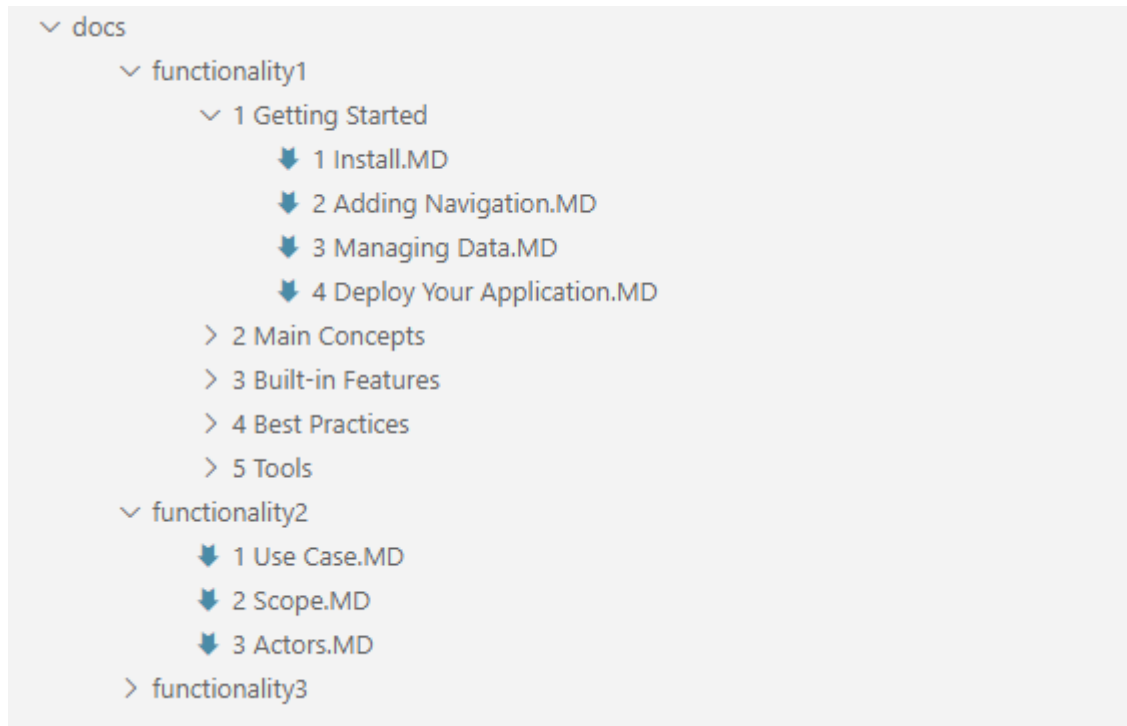
Algorithm Diagram



- Flow Description
- **Precondition**
The user must be having internet connection and system must be run at least one time to generate the files and deploy the web app
 - **Activation**
Once user visit the appropriate link to open web app
 - **Main flow**
 - ✓ User opens an internet browser
 - ✓ User enters the link in a browser (See A1)
 - ✓ The system will present the user with the documentation
 - **Alternate Flow**
 - ✓ A1 – If wrong link entered the system will present user with an error page
 - The system will give instructions of how to access documentation
 - The user clicks on one of the links available on the error page
 - The system will display the appropriate documentation page
 - **Exceptional Flow**
 - ✓ User manually stops the system, will display manually stopped message
 - ✓ The cloud instance is down, the system will display 404 error page
 - **Termination**
System is stopped by a user
 - **Post Condition**
The system will enter a wait condition until an API call is made

c. Data Requirements

This section of the paper will describe the data requirements of the system which is an essential requirement for the system functionality. The system will use GitHub repository as the backend solution to store data generated by the system and by the user. All the documentation details will be stored on GitHub which will consists of markdown files for documents text and a folder structure for the navigation links of the web application. Take for example the sample dataset bellow:



The system will follow the pattern above, recursively reading each file and folder plus the text available in each markdown file after which will create a copy of the above structure with the files converted to HTML. The folders will become part of the navigation system of the web app and the markdown files will be translated to sub-links for each link and the text within each markdown file will be parsed as the content of the document.

This can be compared to a database system for an entity; once any CRUD operation is performed on any of this files or folders the content of the web application will be updated accordingly. The project is designed to use technologies which are closest to the developer and follow GitOps pattern. “GitOps focuses on a developer-centric experience, by using tools developers are already familiar with, including Git and Continuous Deployment tools. The core idea of GitOps is having a Git repository that always contains declarative descriptions of the infrastructure currently desired in the production environment and an automated process to make the production environment match the described state in the repository.” (GitOps, 2017)

The most interesting thing is that we have access to each commit within the GitHub repository which gives us a unique ID and a complete history of files modification. So, let us say you create and deploy an application version 1.0.0, you will have documentation version 1.0.0 available for that version of the application. If any customer has access to an older version of the application, it will have access to an older version of the documentation as well as each commit will match with that version of the application.

d. Data Conversion

The following section of this document will describe the steps taken for the conversion of markdown files to the HTML syntax. The system will have a custom algorithm which will be used to transform markdown symbols into HTML syntax. The project will be following the most common and widely used markdown symbols for markdown syntax provided by GitHub (Guides, n.d.). Let us take for example the table below:

Markdown	HTML	Display Output
# Heading level 1	<h1>Heading level 1</h1>	Heading level 1
## Heading level 2	<h2>Heading level 2</h2>	Heading level 2
### Heading level 3	<h3>Heading level 3</h3>	Heading level 3
#### Heading level 4	<h4>Heading level 4</h4>	Heading level 4
##### Heading level 5	<h5>Heading level 5</h5>	Heading level 5

The system will parse the files and check for any heading symbols like # for heading level one and if found it will delete the symbol and append the <h1> tag element surrounding the text. This will be interpreted by any browser as a heading of size 32 pixels which will make it look larger on the page and on a new line. As documents have headings to differentiate between different chapters and sub-chapters/sections this feature is of high importance for the system. If now heading symbols are found the system will need to convert the text into paragraphs, for example:

Markdown	HTML	Display Output
I really like using Markdown.	<p>I really like using Markdown. </p>	I really like using Markdown.
I think I will use it to format all my documents from now on.	<p>I think I will use it to format all my documents from now on. </p>	I think I will use it to format all my documents from now on.

As you notice we will surround any text without markdown symbols with a HTML <p> tag which will render as a paragraph of size 16 pixels in any internet browser. The same process will happen for each markdown symbol replacing the symbols and appending the appropriate HTML tags depending on the type. The system will support heading, unorganised lists, organised lists, links, code blocks and images.

Using markdown files (also referenced as .MD files in this paper) has been chosen for this project to make it easier for developers to read or edit the documentation files without the need to worry about HTML syntax. Having the HTML files will clutter the documentation and make it harder for developers to read and edit because of the syntax overhead.

e. Environmental Requirements

- The system should have access to internet
As the system works with remote repository, cloud technologies and it is designed such that the user can view the documentation files over the internet it is required that the system should have access to an active internet connection.
- The system should have access to a repository
As the system uses the repository as its primary back-end storage reading and writing to it, the system requires that the user should have access to a repository technology.

f. Non-functional Requirements

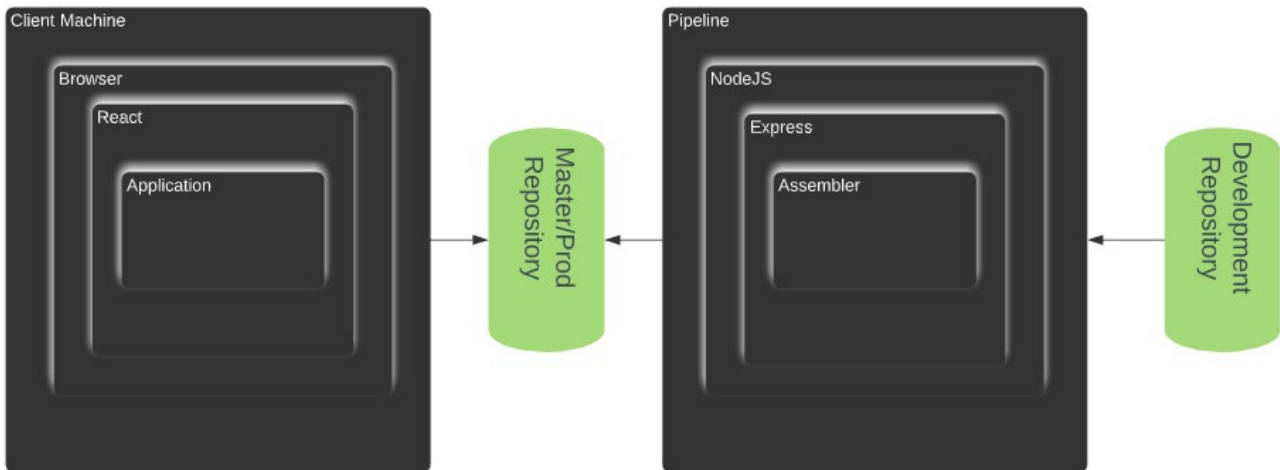
- The system should be available 24/7
As the developers can make code changes at any time of the day the system should be always available, otherwise source code might change, and the documentation might not be created leaving gaps between versions of the software and the documents.
- The system should incur minimal learning
The system should be designed to be “plug and play” as it tries to make it easier for developers to do their jobs and spend their time on more important tasks.
- System performance
The system should complete the jobs required to parse and deploy the new version of the file in a reasonable amount of time, so it will not impact developers speed on updating their source code.
- Docker Orchestration System
As the system will work within a pipeline within a docker container system the user should have a container orchestration system like docker desktop. It will be possible for the user to use a different approach as the system is portable in other pipeline, but it is up to the developer to tweak any settings required.
- Access to cloud compute services
The user should have access to cloud computing services as the documentation will be published on his own domain. If the user does not have access to cloud infrastructure the system will still provide the raw generated files and it will be left to the user to decide the deployment platform.

g. Usability Requirements

- The system should have access to Docker technology
The system is built around Docker technology the user should be able to run docker images on local computer or cloud environment with docker orchestrator technology, otherwise he will not be able to run the system.
- The user should be familiar with technology
The system is designed for users with knowledge of programming languages and IT technology in general and although the system will not have a big learning curve for the user, some general knowledge is required.
- Independent of source code programming language
The system should be independent of developer’s source code programming language as it builds only the documentation for the code. As the documentation is a stand-alone part of the software it should not impact the source-code and just read through the files provided not run them.

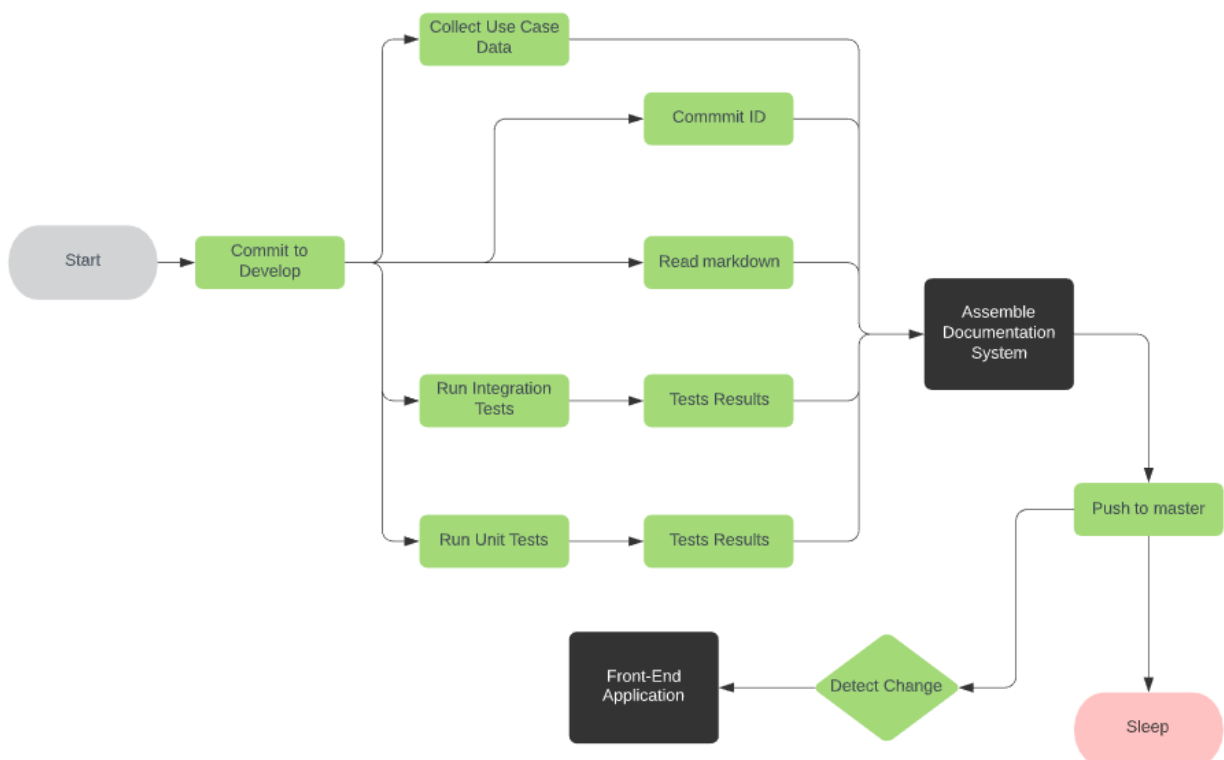
2. Design & Architecture

The system will function on a NodeJS server technology serving front-end documents through HTTP request/response methods. The front-end technology will be built with Angular framework which will display a simplistic design to make the documentation pleasant to read. The web-application will be deployed on Heroku cloud which will communicate with the master repository and update documents published by the back-end system.



a. System Architecture Back-End

The back-end system will also be built on NodeJS technology and will live within the Concourse pipeline activated by any commit to the development branch, building all the documentation and push it to the master repository from where the front-end application will access the latest version of the documentation. The following diagram should offer a better visual description of the system



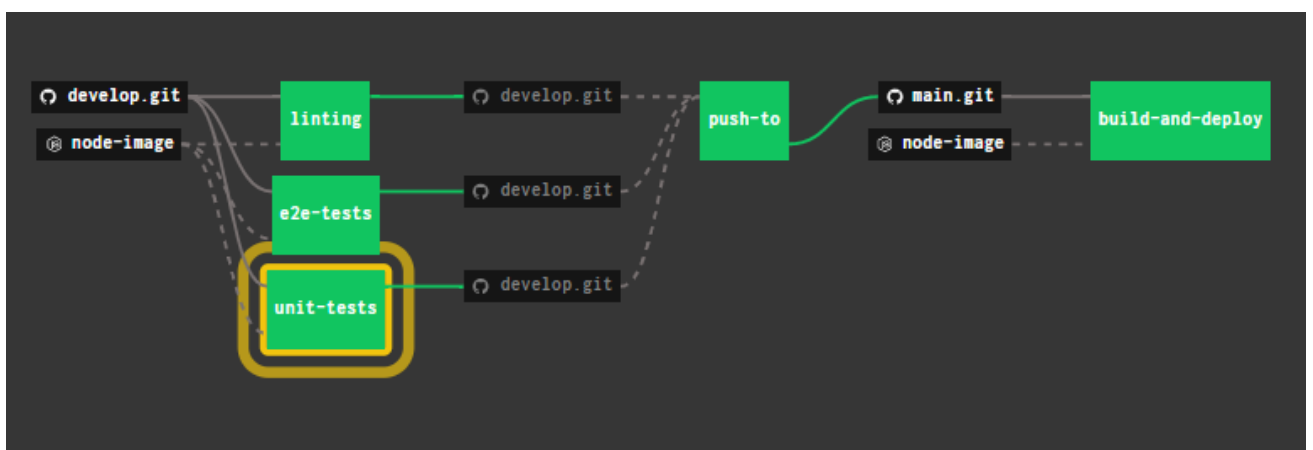
b. System Implementation

- The system will have a continuous integration/continuous deployment pipeline where it will detect any commits to the development branch. The system has access to any changes made to the development branch and all git commands run against the branch from where we can access commit ID, committer name. The system will use this information to version the documentation keeping the same identifier as the software deployed.
- Upon detecting any changes, the pipeline will run all prerequisite jobs like unit tests, integration tests, download repository etc. setting up the automation flow from end to end and validate the code and the documentation created. The system will stop and notify of any errors encountered.
- The system is designed to be dynamic and for each file or folder created the system must create the necessary endpoints where documentation can be accessed.

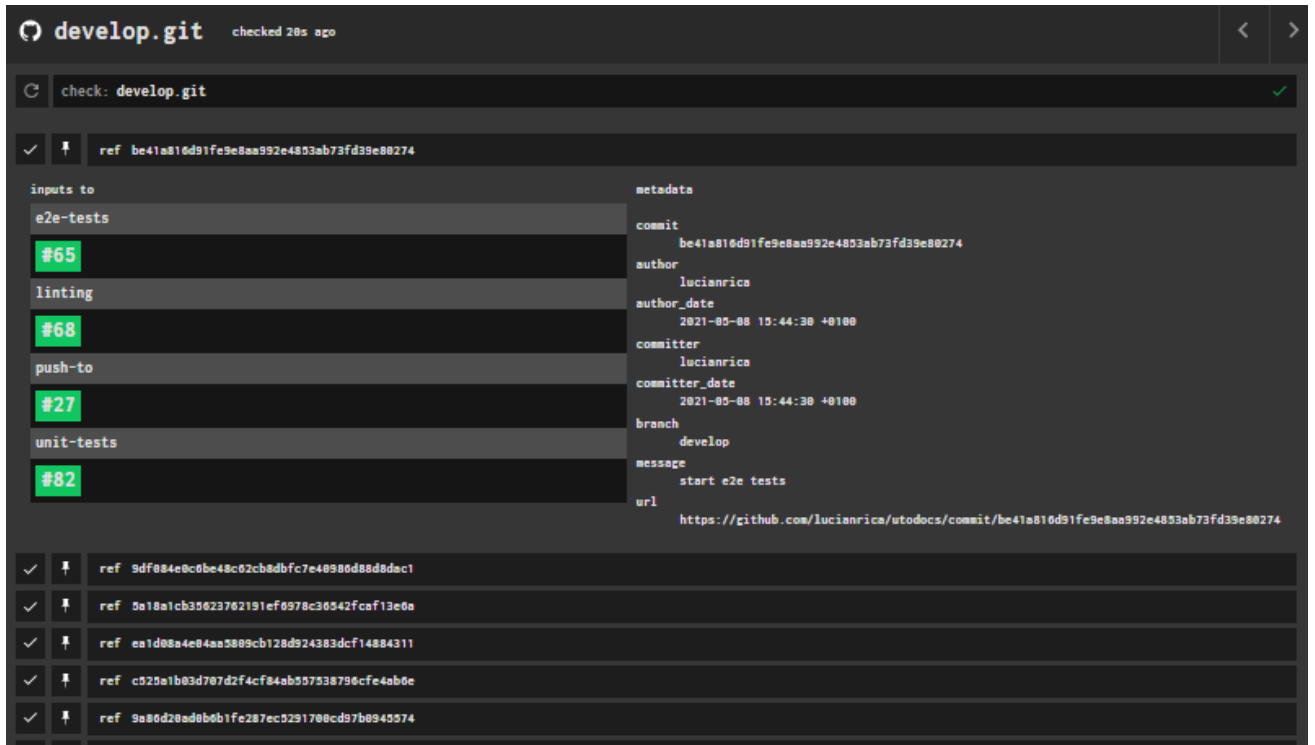
The HTTP GET method which is responsible for any endpoint request incoming from the client for which we have created a custom method “readAndConvertFiles”. The method is taking any request in the form of URL where we split each part and remove any invalid characters such as % and append an empty space to match the folder name by traversing each directory recursively and comparing to the string in the array of strings resulted.

c. User Interface

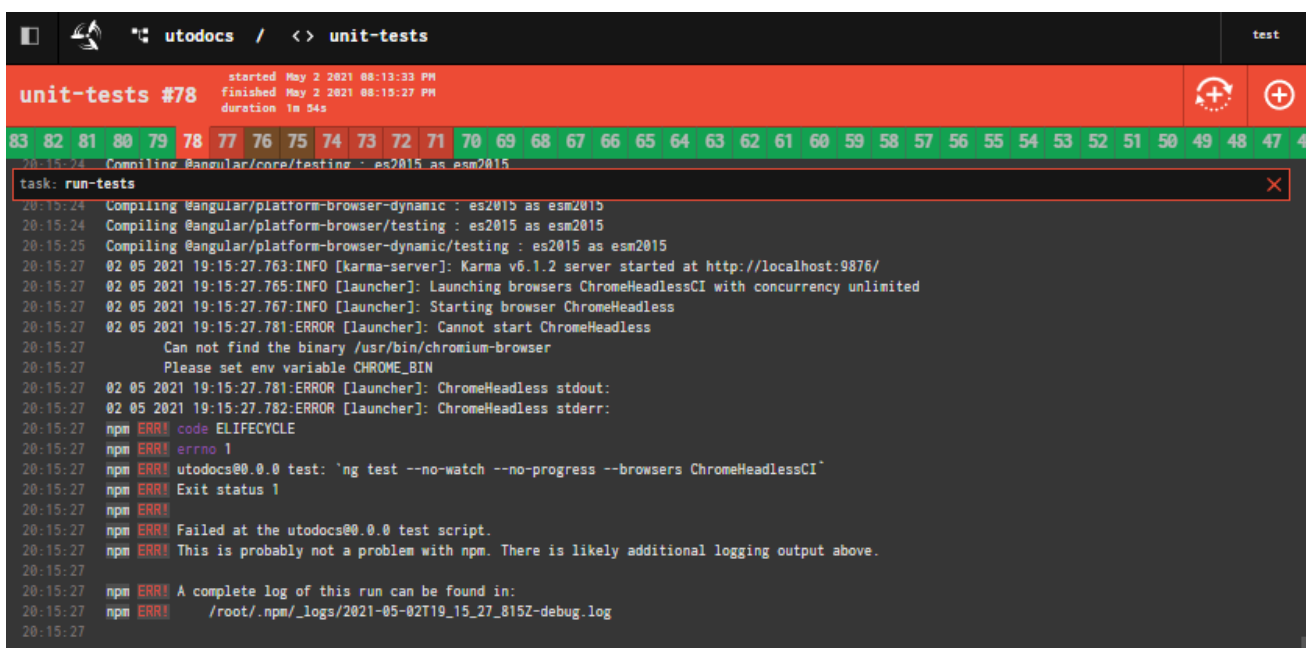
For the back-end service, we do not have user interface as the work done by developers is done mostly in Command Line Interface. Once the back-end service is installed and configured developers pretty much will need to forget about it, it is designed to work in the background but available to the developer needs to delve in and inspect the current status of the process.



The developer has access to all the commits which will reference the specific version of the source-code and documentation which is quite handy as we can inspect, debug, and revert any changes committed in case a bug was discovered in production environment of our application. The documentation will be updated and reflect the “old” version of the software documentation accessible to users within minutes of reverting the source-code.



At the same time, the developer can see real time progress of their application status of their unit test or integration test and any other required tasks just as a normal flow of a software deployment life cycle and can track any version of a deployment with its associated output, e.g. a failed unit test job.



For the front-end application, I used Angular framework to create a single page application and as this is a proof-of-concept application, I added few additional pages to spark interest and attract people in supporting the system. The application is live and available at: <https://utodocs.herokuapp.com>

- Landing page is simplistic with few calls to action messages as we don't want people to get lost of what we are trying to achieve. Snippet from landing page:

Document Your Code

```
## You write markdown

This is where things get interesting, you
provide me with the description of your application
functionality, can even provide me with the code...

```javascript
 console.log("my log")
```

same as writing a markdown file and utodocs will
transform, build and publish a production
gradle web application containing
your updated version of the documentation
```

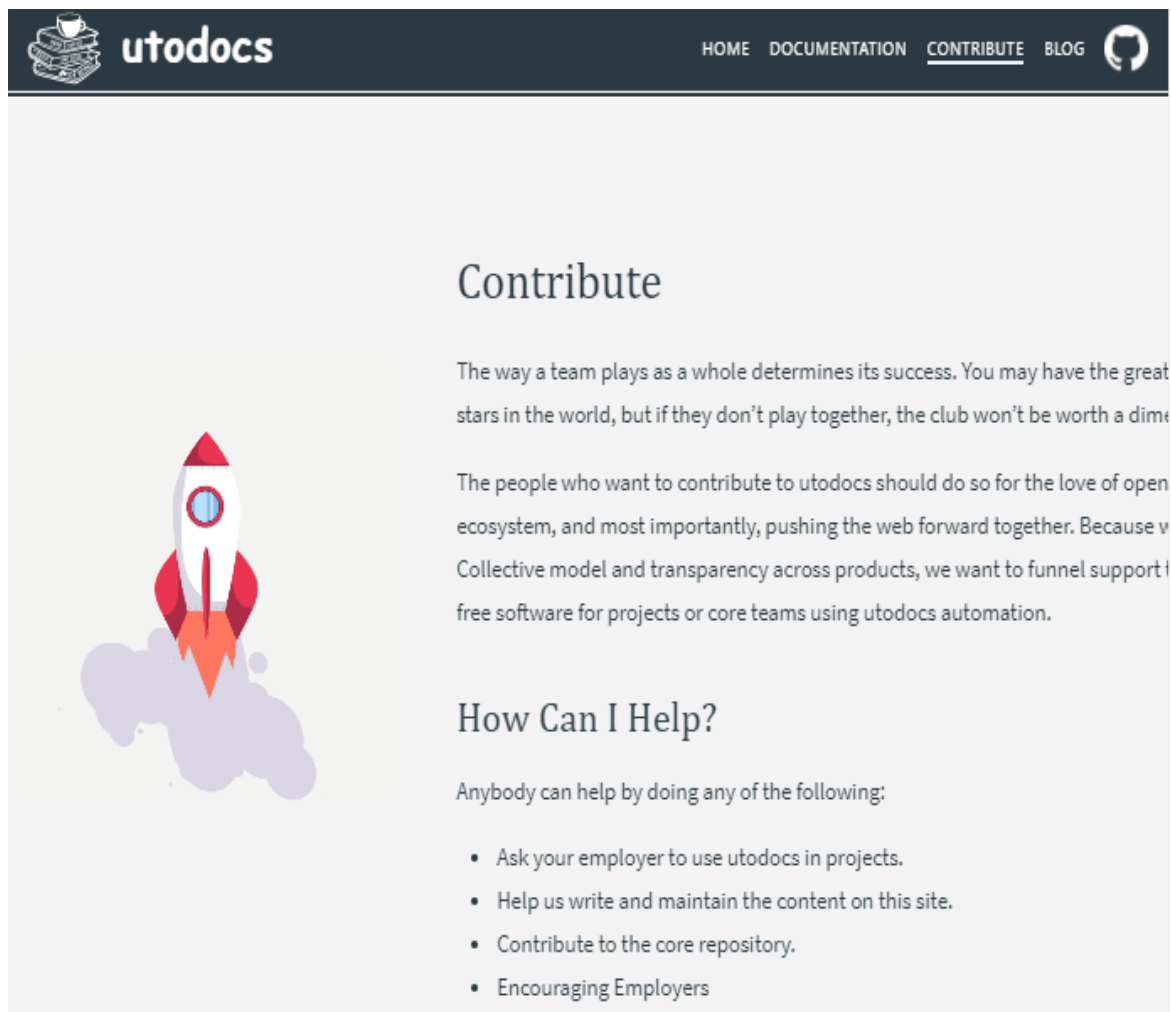
```
<h2>Utodocs makes HTML</h2>
<p>
  This is where things get interesting, you
  provide me with the description of your application
  functionality, can even provide me with the code...
</p>
console.log("my log")
<p>
  same as writing a markdown file and utodocs will
  transform, build and publish a production
  gradle web application containing
  your updated version of the <b>documentation</b>
</p>
```


the automation

Utodocs is provided and executed as a step within the development pipeline which automatically builds and publishes the latest version of your documentation along side your code. The documentation itself will be transformed into an exact replica of this angular web-app and published to a cloud provider of choice. Learn more about utodocs [here](#).

```
graph LR
    subgraph Dev
        D[develop.git]
        NI1[@ node-image]
    end
    subgraph Pipeline
        L[linting]
        E[e2e-tests]
        U[unit-tests]
    end
    subgraph Prod
        M[main.git]
        NI2[@ node-image]
    end
    subgraph Actions
        P[push-to]
        B[build-and-deploy]
    end
    D -.-> L
    D -.-> E
    D -.-> U
    L -.-> P
    E -.-> P
    U -.-> P
    P -.-> M
    NI1 -.-> P
    P -.-> B
    NI2 -.-> B
```

- I have also added a blog and contribute section for any developer who enjoys the idea and wants to contribute into creating a better experience for the developers using utodocs or simply needs more insights into why which I reflected into my blog.



utodocs HOME DOCUMENTATION CONTRIBUTE BLOG 

Contribute

The way a team plays as a whole determines its success. You may have the great stars in the world, but if they don't play together, the club won't be worth a dime.

The people who want to contribute to utodocs should do so for the love of open ecosystem, and most importantly, pushing the web forward together. Because of our Collective model and transparency across products, we want to funnel support to free software for projects or core teams using utodocs automation.

How Can I Help?

Anybody can help by doing any of the following:

- Ask your employer to use utodocs in projects.
- Help us write and maintain the content on this site.
- Contribute to the core repository.
- Encouraging Employers

- The documentation is by far the most interesting feature of the application. Simply put I used utodocs/the system to document itself so that is a live sample of the documentation created by utodocs on how to use and understand utodocs.

The screenshot shows the utodocs website with a dark header containing the logo and navigation links: HOME, DOCUMENTATION, CONTRIBUTE, BLOG, and a GitHub icon. Below the header, there are links for "Writing Documentation" and "Markdown Cheatsheet".

Navigation Menu:

- 1 Welcome
 - 1.1 Introduction
 - 1.2 Glossary
 - 1.3 Technologies
 - 1.4 Licence
- 2 Understand Utodocs
- 3 Getting Started
 - 3.1 Prerequisites
 - 3.2 Setting Up Concourse
 - 3.3 Install
- 4 Setup

Welcome to utodocs

First of all I would like to say thank you for taking an interest into utodocs automation. We are thrilled to see you here and hope you are enjoying our product so far, and just to keep you posted, we are always working to make utodocs exactly what you need for your live application and your feedback helps us decide which features to build, and what improvements should be made to our product.

Project Background

This technical report provides the analysis and discussion of scope for the development of an automated code documentation system strategy. The strategy deals directly with the management of code documentation. The proposed project will give developers the ability to replace the manual task of gathering and assembling code documentation with an automated system to build and deploy production grade documents.

Project Aims

The aim of this project is the creation of a system for automating code documentation process which will gather source documents and assemble them into a ready to deploy web application. The system will work within an automated development process, recognise repository changes, and automatically start a task within the software pipeline publishing the web application to the cloud and making it accessible to end users within minutes.

Version controlled plain text

As programmers we live in a world of plain text. Our documentation tooling should be no exception. We want tools that turn plain text into pretty HTML. We also have some of the best tooling available for tracking changes to files. Why would we forgo using those tools when writing documentation? This workflow is powerful, and familiar to developers.

Markdown

Your first steps in documentation should go into your Markdown. Code hosting services will render your

- Heroku Cloud – Initially I had the application deployed to Microsoft’s Azure cloud by my account credit recently expired so I had to redeploy my application to Heroku cloud. The application is deployed automatically on every GitHub push so as a developer I do not have to take any steps into creating a new version of the app.

I simply work on a new feature on my local environment and when it is ready, I simply push it to the develop branch and everything else is taken care of, unit tests, deployments, conversions etc. P.S. In a real-life scenario this should be done by a pull request where developers will

inspect the code at hand and approve/reject the pull request which will then be merged into develop if approved and continue with testing etc.

3. Testing

For the testing part of the system, I used Karma test runner for JavaScript code that runs on NodeJs and have them executed in the browser. The test runner is running the tests and checks the within the browser to see if the desired output matches what the browser displays.

Karma v 6.1.2 - connected; test: complete; DEBUG

Chrome 90.0.4430.212 (Windows 10) is idle

Jasmine 3.6.0 Options

24 specs, 0 failures, randomized with seed 47369 finished in 0.319s

- MainSidenavComponent
 - should create
- MarkdownCheatsheetComponent
 - should create
- DocsInteractionService
 - should be created
- BlogComponent
 - should render The Quantum Leap title
 - should create
- AppComponent
 - should create the app
- ContributeComponent
 - should create
- MainDocsComponent
 - should create
- SubTopnavComponent
 - should render Writing Documentation link title
 - should render separator
 - should render Writing Documentation link title
 - should create sub-navbar
- HomeComponent
 - should create
- TopnavComponent
 - should render BLOG link title
 - should render DOCUMENTATION link title
 - should render HOME link title
 - should have as logo 'utodocs'
 - should render CONTRIBUTE link title
 - should render logo title
 - should create navbar
- WritingDocumentationComponent
 - should create
- NotFoundComponent
 - should create
- FooterComponent
 - should create
- SafePipePipe
 - create an instance

DISCLAIMER, SEPTEMBER/MAY 2021

THIS PROJECT WAS DEVELOPED BY LUCIAN NECHITA AS PART OF THE FINAL PROJECT ASSESSMENT FOR BSC (HONS) IN COMPUTING (SOFTWARE DEVELOPMENT STREAM) AT NATIONAL COLLEGE OF IRELAND. FOR FURTHER INFORMATION CONTACT NATIONAL COLLEGE OF IRELAND AT <https://www.ncirl.ie>

THE INFORMATION AND VIEWS SET OUT IN THIS REPORT ARE THOSE OF THE AUTHOR(S) AND DO NOT NECESSARILY REFLECT THE OFFICIAL OPINION OF THE NATIONAL COLLEGE OF IRELAND.











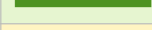
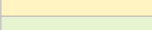


This works well on the local environment as the developer can inspect the tests and see any failures with the reason why it fails. When we run this test within the pipeline it is a different story as there is no developer to watch this test and having chrome opening within the pipeline will result in an error. To make it work in the pipeline I had to use a HeadlessChrome which is a version of Chrome browser working in a sandbox environment without being the full UI, essentially running Chrome without Chrome, is just the underling code and I have to say it was quite challenging to set it up.

Jasmine is the framework used for creating the unit tests for the application. Jasmine is a behaviour driven testing framework for testing JavaScript code. Initially I planned to use Mocha testing framework, but I experienced many issues when running the tests within the pipeline again as they require a Browser DOM. So, I had to change to Jasmine testing framework as it does not require a DOM and it is the most popular within Angular framework. Jasmine also comes with a very hand test reporter which builds a status page of our overall test coverage.

All files

91.83% Statements 65/71 0% Branches 0/6 81.67% Functions 19/24 84.91% Lines 51/57

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

File ▲		Statements	Branches	Functions	Lines	
app/main		100%	2/2	100%	0/0	100%
app/navigation/sidenav/main-sidenav		100%	3/3	100%	0/0	100%
app/navigation/sub-topnav		100%	3/3	100%	0/0	100%
app/navigation/topnav		100%	4/4	100%	0/0	100%
app/pages/blog		76.67%	5/6	100%	0/0	83.33%
app/pages/contribute		76.67%	5/6	100%	0/0	83.33%
app/pages/docs/main-docs		66.67%	4/6	100%	0/0	63.33%
app/pages/home		71.43%	5/7	100%	0/0	66.33%
app/services		96.67%	14/15	0%	0/6	96%
app/shared/footer		100%	2/2	100%	0/0	100%
app/shared/markdown-cheatsheet		66.67%	4/6	100%	0/0	63.33%
app/shared/not-found		100%	2/2	100%	0/0	100%
app/shared/writing-documentation		66.67%	4/6	100%	0/0	33.33%
main		100%	3/3	100%	0/0	100%

At this stage, the pipeline does not run tests for the “feature” git branch and in a real-life scenario it should run the unit-tests on any branch that requested a “pull request”, as I am a one-man team it didn’t make sense to enable such functionality. You can also see above that the unit tests are not 100% code coverage but again as this is more a proof-of-concept/start-up type of application it is recommended not to create any unit/integration tests etc. as it is a waste of time and money. Usually in a start-up/proof-of-concept requirements will change very often making any tests created redundant and the market still needs to accept,

application needs to prove itself worthy and potential to generate income before we can spend money on testing. As such I have created end-to-end/unit-tests to cover the National College of Ireland continuous assessment requirements but in a real-life scenario I would not.

I have also added a linting task within the pipeline which checks the code for programmatic and stylistic errors. It is a static code analyser which will format the code before it's deployed to the version control provider. The process should be done before each pull request but as I am a one-man team this step is done between develop and main branch. If code is not properly formatted the pipeline will output an error and refuse the deployment.

4. Evaluation

The evaluation of the system was done with creating the documentation for itself. The system is highly scalable as it makes use of the latest industry tools as Docker images for pipelines where we can instantiate multiple pipelines to work depending on the lead and requirements and as the system makes use of cloud platform the front-end application can scale horizontally or vertically depending on the cloud provider but with no impediments from the system itself.

The version of the documentation will reflect the latest version at the version control provider, such as GitHub, making it quite robust in terms of correctness of the documentation. So as long the developer has the desired version of the source-code and documentation on GitHub the system is guaranteed to display the correct version.

The documentation is deployed within 3-4 minutes of creating a new version of the document and as it is an Angular front-end application it only has few kilobytes in size making it very fast into accessing it from a browser, around two hundred milliseconds until the application is fully loaded into the browser.

5. Conclusion

Considering that I am part-time student with full time job and more modules to work on, building the application took a very long time but even to my surprise using the application into creating the documentation was done very quick. We can fully document an application within one day as long as we know the content of the documentation which is quite awesome.

The application strength is that developers can just create documentation without learning new tools or this tool. As long as they have some knowledge of Angular and CSS, they can easily manipulate it to create a completely different User Interface. Also, the documentation will reflect the latest source-code change as it is deployed or reverted at the same time with the source-code which is a very robust way in making sure the documents reflect the actual source-code software functionality.

One of the disadvantages of the application is that it was only tested with GitHub version control platform, it might work with other platforms but as I did not test it, I'll put it as a disadvantage. Another disadvantage is that the angular application needs a server to run on the cloud which feels a bit like a waste of money to have a server running on the cloud just to serve documentation, better approach will be to have a fully static website which will be cheaper.

6. Further Development and Research

I really like the end result of my application and as a future research I will get feedback from developers and try to understand what other people need from a document automation system, where I can improve. Also, in real world scenario businesses have multiple version control repositories and even in different providers. Having the system act as a platform for the organisation code documentation and serve documentation from multiple repositories from different providers will make it more robust and even more attractive to developers.

The system will keep the same direction in terms of “learn free” feature and try to avoid having “settings” within the system as the more “settings” we have the more time the developer needs to spend in learning the application making his life harder instead of easier.

References

Agile, n.d. *Introduction to Test Driven Development (TDD)*. [Online]

Available at: <http://agiledata.org/essays/tdd.html>

[Accessed 17 12 2020].

Alliance, A., 2018. *What is Agile?*. [Online]

Available at: <https://www.agilealliance.org/agile101/>

[Accessed 17 12 2020].

Andrew, R., 2018. *Grid by Example*. [Online]

Available at: <https://gridbyexample.com/examples/>

[Accessed 17 12 2020].

Azure, M., 2020. *Get to know Azure*. [Online]

Available at: <https://azure.microsoft.com/en-us/overview/>

[Accessed 17 12 2020].

Azure, n.d. *Azure Pipelines*. [Online]

Available at: <https://azure.microsoft.com/en-us/services/devops/pipelines/>

[Accessed 15 12 2020].

Concourse, 2017. *Concourse*. [Online]

Available at: <https://concourse-ci.org/pipelines.html>

[Accessed 17 12 2020].

developer.mozilla, n.d. *JavaScript*. [Online]

Available at: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>

[Accessed 15 12 2020].

GeeksForGeeks, 2017. *Functional vs Non Functional Requirements*. [Online]

Available at: <https://www.geeksforgeeks.org/functional-vs-non-functional-requirements/>

[Accessed 17 12 2020].

GitOps, 2017. *GitOps*. [Online]

Available at:

<https://www.gitops.tech/#:~:text=GitOps%20is%20a%20way%20of,Git%20and%20Continuous%20Deployment%20tools>

[S](#).

[Accessed 18 12 2020].

Goldis, A., 2018. *How to document source code responsibly*. [Online]

Available at: [Code documentation](#)

[Accessed 15 12 2020].

Guides, G., n.d. *Mastering Markdown*. [Online]

Available at: <https://guides.github.com/features/mastering-markdown/>

[Accessed 18 12 2020].

Manager, p., 2020. *The Ultimate Guide to Gantt chart*. [Online]

Available at: <https://www.projectmanager.com/gantt-chart>

[Accessed 17 12 2020].

Markdown, 2017. *What is Markdown?*. [Online]

Available at: <https://www.markdownguide.org/getting-started/>

[Accessed 17 12 2020].

Microsoft, 2018. *What is cloud computing?*. [Online]
Available at: <https://azure.microsoft.com/en-us/overview/what-is-cloud-computing/>
[Accessed 15 12 2020].

Mocha, n.d. *Mocha*. [Online]
Available at: <https://mochajs.org/>
[Accessed 17 12 2020].

NodeJS, n.d. *About Node.js*. [Online]
Available at: <https://nodejs.org/en/about/>
[Accessed 15 12 2020].

PCMAG, 2016. *Definition of Web application*. [Online]
Available at: <https://www.pcmag.com/encyclopedia/term/web-application>
[Accessed 16 12 2020].

Smashing, 2016. *What Is Responsive Web Design?*. [Online]
Available at: <https://www.smashingmagazine.com/2011/01/guidelines-for-responsive-web-design/#:~:text=Responsive%20Web%20design%20is%20the,use%20of%20CSS%20media%20queries.>
[Accessed 15 12 2020].

StackExchange, 2017. *What is development automation?*. [Online]
Available at: <https://softwareengineering.stackexchange.com/questions/178311/what-is-development-automation>
[Accessed 15 12 2020].

Stolberg, S., 2009. *Enabling Agile Testing through Continuous Integration*, Chicago, IL, USA: IEEE.

technopedia, 2016. *Software Repository*. [Online]
Available at: <https://www.techopedia.com/definition/32890/software-repository>
[Accessed 15 12 2020].

Technopedia, n.d. *Production Environment*. [Online]
Available at: <https://www.techopedia.com/definition/8989/production-environment#:~:text=Production%20environment%20is%20a%20term,intended%20uses%20by%20end%20users.>
[Accessed 06 12 2020].

TechTerms, n.d. *End User*. [Online]
Available at:
<https://techterms.com/definition/enduser#:~:text=An%20end%20user%20is%20the,or%20programmers%20of%20the%20product.>
[Accessed 06 12 2020].

UTM, n.d. *Concept: Use-Case Model*. [Online]
Available at:
http://www.utm.mx/~caff/doc/OpenUPWeb/openup/guidances/concepts/use_case_model_CD178AF9.html#:~:text=A%20use%20case%20model%20is,system%20to%20solve%20a%20problem.&text=The%20most%20important%20mode!%20elements,the%20model%20to%20simplify%20communications
[Accessed 15 12 2020].

Vasudevan, K., n.d. *What is API Documentation, and Why It Matters?*. [Online]
Available at: <https://swagger.io/blog/api-documentation/what-is-api-documentation-and-why-it-matters/#:~:text=API%20documentation%20is%20a%20technical,and%20integrate%20with%20an%20API.&text=API%20Description%20formats%20like%20the,to%20generate%20and%20maintain%20them>
[Accessed 15 12 2020].

Appendices

1. Project proposal

Code Docs Automation

Overview

The purpose of this document is to provide the reader with more in-depth detail of my final year project for National College of Ireland BSc (Hons) in Computing programme. The proposed project will give developers the ability to document their code. Documenting the code, the developer usually takes twice the time to write the code (personal experience). Then the developer spends at least one time to create and publish the documentation.

Goal

Free the developer time by reducing the amount of time needed for writing documentation to at least half the time necessary to write the actual code.

The automation tool should be “LEARN FREE” meaning that the developer should not spend time learning how to use the automation functionality instead of writing code.

Objectives

Create a tool for automating code documentation process which will read the text files from the code repository and transform the text into HTML. Create the automation process within a Continuous Integration/ Continuous Delivery pipeline and publish the HTML package online so other developers can access. Automation process should read the folders and paths of the documents repository and automatically create the navigation of the website.

Benefits

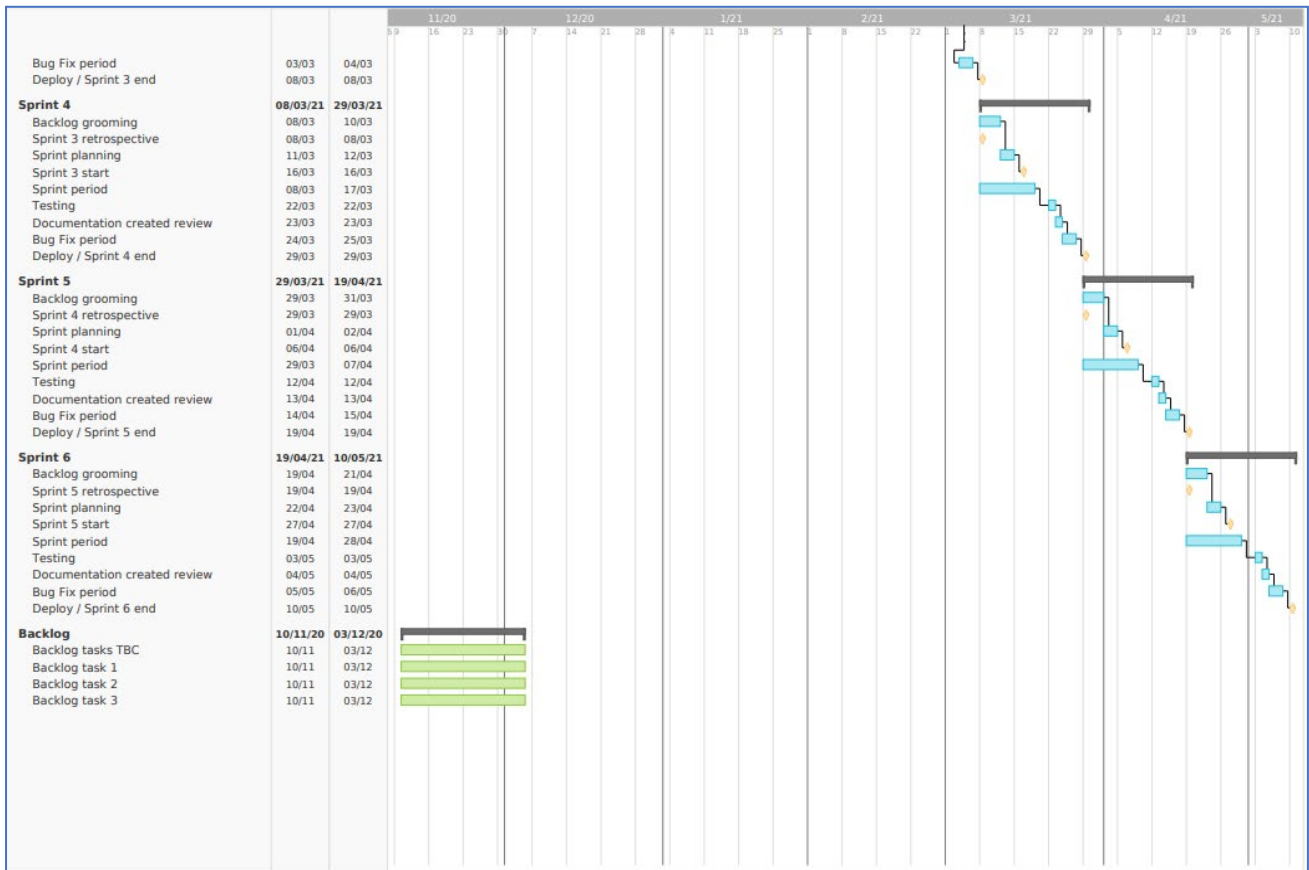
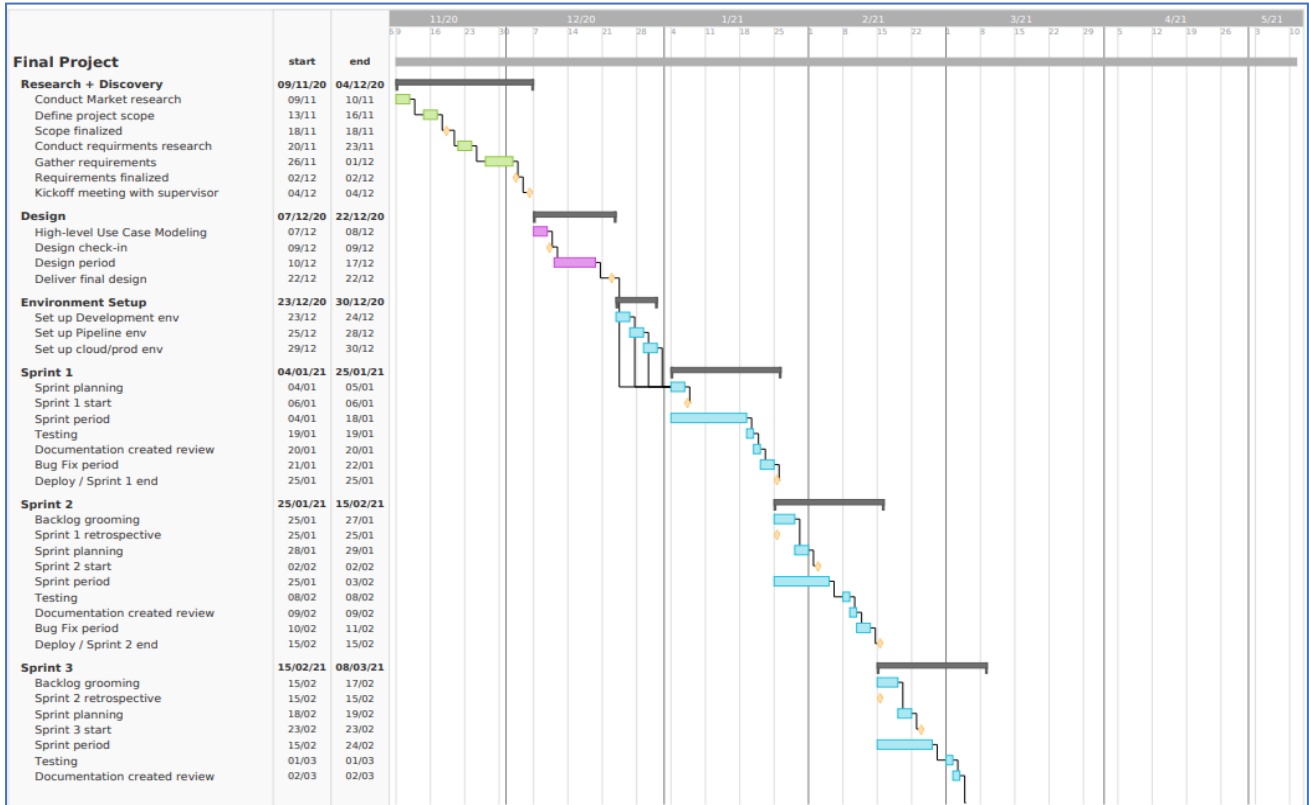
Freeing up developer time will give him time to do more important jobs, improve code quality and fix more bugs and overall improving the business logic and deliver more value to the customer and business itself.

Key Success Factors

The automation tool should be coding language agnostic so any developer from any background can benefit. The automation tool should be “learn free” so any developer can focus on what their best at instead of using company time to learn how to use our tool.

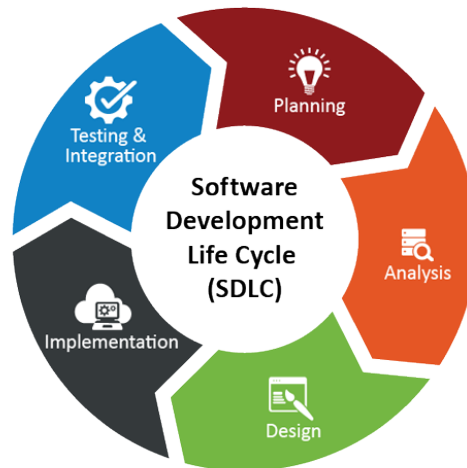
The automation tool should include all stages of the software development life cycle. Building software functionality usually include documents, scattered across different apps or places. For example, a UML use case modelling documentation from requirements elicitation, might be small but powerful into helping us building the documentation for a unit of functionality, like who can use it (actors), what they need (prerequisites), and how to use it (flow of events).

Project Plan



Technical Details

- The project will make the use of NodeJS engine which will give us the ability to run the application within a pipeline.
- The project will be built mainly with JavaScript language, most likely can be built with any language but my skills with JavaScript are greater than other languages.
- The project will make use of the use case modelling technique in building the actual end user published documentation.
- The project will follow the software documentation life cycle and gather as much information as possible from each step for building the documentation.



- The project will be used to document itself during the process of building the automation
- The project will make use of HTML and CSS Grid technologies for the front-end resulting package.
- The project will use .md type (markdown extension) files where all the data will be gathered by the developers.
- The project will use Concourse pipelines technologies to implement continuous integration/continuous deploy and docker images for the underling infrastructure
- The project will deploy the documentation for itself created by itself on a cloud platform like Heroku or Azure.
The project will use Mocha framework for unit and integration tests and to convey information to the documentation automation system for building the documentation.

Evaluation

The system will be tested by creating the documentation for itself while building it and I should be able to read the documentation online.

2. Reflective Journal October

National College of Ireland
Course: BSc (Hons) in Computing –Software Development
Month: October 2020
Lucian Nechita, x16149505

Code Docs Automation

Background

My final project is an automated application to help developers document their software. The application will be designed to live within a Continuous Integration / Continuous Development pipeline, it will read hand-written documents and convert it to a production ready website.

Planning

As any good project implementation starts with few sessions of planning. First week of October is all about planning how I can, and how I want to build my application. Discovering the required technologies involved and assessing critical skills necessary in building the application, is at highest priority.

Research

In the second week of October, I put all my efforts into research on competitive applications, their market share and highlight their achievements, where exactly people connected with their own technology and what they loved about it. I believe it is important to see what other people will love to see achieved from such technologies

Brainstorming

Third and fourth week of October was all about brainstorming features on my application. I allowed two weeks for this task as I want to make sure I will build exactly what I need and diminish the chances to miss on opportunities, missing features.

Achievements

The month of October helped me better understand the problem I face. Although I will continue over the month of October to iterate over and over the planning part of the project as a better knowledge of project requirements will speed up the development progress later down the road

Student Signature: Lucian Nechita
Supervisor's Signature:

Date: 27 /09 /2020
Date: / /2020

3. Reflective Journal November

National College of Ireland
Course: BSc (Hons) in Computing –Software Development
Month: November 2020
Lucian Nechita, x16149505

Code Docs Automation

Background

For the month of November, I spend time on planning and research for my final project which is the automated application to help developers document their software features. As it is quite a challenging project, I need to make sure I know what I want to build before I start building it, otherwise I will spend more time redoing the functionalities which I will build.

Project Submissions

For the first week of November, I completed the ethics form and assembled the project proposal report required by National College of Ireland BSc in computing programme. This is very important as communicating your project and ideas to others will help you better promote your product.

Gantt Chart

Project management took my second week, and it is another important step in building my automation software and Gantt chart is one of the most popular way of showing, planning, and illustrate project's schedule. Doing this I can better plan my activities and keep an eye on the deadlines which makes me more suitable to deliver my project successfully.

Following the Plan

Following my Gantt chart, I spent third week of November researching the market in dept and I found some similar applications, but they are limited to only one programming language and it is mostly just for APIs. While researching I start to get more and more idea's, so I defined the scope of the project relative to the time I have. I will try to stick to the scope as I do not want to get overwhelmed by the amount of work.

Gathering Requirements

My final week in the month of November was mostly about making more research, on requirements gathering best practices and the actual requirements of my automation framework/application.

Achievements

Having the requirements finalised is a major milestone and I managed to keep progress in sync with my project management plan. I have a better idea of what my software should do and what it should not do.

Student Signature: Lucian Nechita
Supervisor's Signature:

Date: 27 / November /2020
Date: / November /2020

4. Reflective Journal December

National College of Ireland
Course: BSc (Hons) in Computing – Software Development
Month: December 2020
Lucian Nechita, x16149505

Code Docs Automation

Background

For the month of December, I spend my time on design and design related research for my final project, the automated application to help developers document their software features.

Design

My first two weeks in the month of December was mostly about the design of the features for my application and create the use case modelling of my application which I will need to use to build my software documentation. So, I had to be careful about the quality of my use case modelling as it will impact the end-result of the automated documentation.

Requirements Elicitation

As use case modelling is a really good tool for requirements elicitation and represent system's requirements, I spend time researching best practices and iterating over key elements in my use case's several time to make sure I really understand what they supposed to deliver. It was good as my knowledge grow in requirements elicitation and describe how actors use a system to achieve a particular goal.

Project Deliverables

Final Project midterm was in the month of December and I had to put some time aside to complete project's deliverables required by National College of Ireland BSc in computing programme. Really happy that I managed to plan for these steps in my project's management Gantt chart and it was easier to manage.

Environment Setup

For the last week of December, I set up the development environment which involved setting up GitHub and my computer's environment variables to development, created the pipeline and run a hello world NodeJS application within the pipeline, and connected the pipeline with the cloud environment for CI/CD.

Achievements

I have gained valuable knowledge for requirements elicitation and set-up the environments as planned. I feel ready to start building my application.

Student Signature: Lucian Nechita
Supervisor's Signature:

Date: 27 / December / 2020
Date: / December / 2020

5. Reflective Journal January

National College of Ireland
Course: BSc (Hons) in Computing – Software Development
Month: January 2020
Lucian Nechita, x16149505

Code Docs Automation

Summary

In the month of January, I spent most of my time with exams and building/playing around my December build of the software prototype.

Exams

My first week in the month of January was mostly about the upcoming exams and I spent all my time studying and getting the exam papers ready for the deadline. As we had six modules last semester and only four this semester, I decided to delay project work and focus on exams only considering that my project plan included six modules for the second semester as well. Having just four will free up much time for project work enabling me to have this delay.

Environment Setup

As I finalised use case modelling documentation during December month and I have built a software prototype I was able to start connecting the application with all my environments. I have created a GitHub account to hold my software source code and created a pipeline in Concourse with Docker orchestrator which will allow me to push the code through the pipeline, have all the tests executed and the documentation created and save it to the GitHub repository.

Project Availability

As the documentation needs to be published to a cloud vendor, I have connected GitHub repository with Heroku cloud provider which will publish my code online once a new version of the source-code reaches the master branch. This is relatively fast making the documentation available on the internet within minutes which is great as the latest version of the software will have the latest version of the documentation

Documentation Setup

The docs automation project makes use of files and folders to generate web app documentation links and content, and although the content generated by the software is working as intended there are some challenges generating the links (or table of contents) as it has to be dynamic, and users should never manually create the links. The issue is within the file-paths as they differ from OS to OS (Linux, Windows etc.) and which link goes where, the OS is sorting the folder structure and we do not need it to be sorted.

Student Signature: Lucian Nechita
Supervisor's Signature:

Date: 27 / January / 2020
Date: / January / 2020

6. Reflective Journal February

National College of Ireland
Course: BSc (Hons) in Computing –Software Development
Month: February 2020
Lucian Nechita, x16149505

Code Docs Automation

National College of Ireland
Course: BSc (Hons) in Computing –Software Development
Month: February 2020
Lucian Nechita, x16149505

Code Docs Automation

Overview

For the month of February, I had a very big unforeseen issue. My personal computer crashed beyond normal recovery options. Not really sure why but the motherboard stopped showing any signs of life and I performed all possible troubleshooting methods with no success. My only option was to buy a new motherboard which took a lot of time as I had to make sure it's compatible with other parts. Of course, is not that straightforward as newer motherboards doesn't like older CPU's, so I had to order a new CPU and Random-Access-Memory as well.

I would've taken it to a shop to be quickly fixed but due to Corona virus all shops were closed, and my only option was to fix it myself. So, I ordered the necessary parts but again due to COVID and Brexit combined the parts were very slow to arrive as the delivery was stuck on United Kingdom customs.

This problem really hurt my progress on the final project so far as I was not able to do anything. Only progress I done was to attend the semester two modules on my mobile phone which wasn't a great experience but hey, better than nothing.

Recover lost time

I don't have a really good method of recovery apart of sacrificing few hours of sleep and work on my project. We have 730 hours in a month from which I spend 160 at work and 240 sleeping (8 hours) giving me 330 hours left for college. Dividing that by the number of modules this semester I'm left with 66 hours for each module in a month. So, I will have to recover 66 hours, considering that I can cut two hours from my sleeping that will give me roughly 33 hours or one month of sleeping two hours less and get back on track with my project.

Conclusion

I believe I still have a chance to finalize my final year project and deliver most of the project requirements. My other option was to apply for a deferral, but I prefer to go forward and try my best, just a little tougher.

Student Signature: Lucian Nechita
Supervisor's Signature:

Date: 27 / February /2020
Date: / February /2020

7. Reflective Journal March

National College of Ireland

Course: BSc (Hons) in Computing – Software Development

Month: March 2020

Lucian Nechita, x16149505

Code Docs Automation

Overview

For the month of March, I followed my previous plan of sleeping two hours less which was quite challenging and tiring but it did pay off and I managed to bring the project up to date. Also, I was very busy in pushing out assignments for other modules in semester two way before their deadlines so I can get more time to work on my final project.

Project profile

Project's profile was created and submitted on teams to be reviewed by the teachers and supervisors, it was quite small task but quite important into advertising my project, so special care was given on the selected profile photos and target audience messages.

Project Documentation

For the project documentation I took the feedback received from the mid-point submission and I added a whole system overview as it was quite difficult for the audience what I was trying to achieve and how the system will look like on my old documentation. Also, another feedback was to include more details on the technology used within my project which I successfully added. Only thing left is the testing and evaluation which will have to wait until I implement the tests and evaluate my project

The system

The system is my only concern at this stage as I do not have any other assignments due. Currently is back up and working as intended on the developing/local environment managing to translate any markdown files into browser viewable pages. I start working on the user interface and hopefully deploy it live on the cloud within a week.

Conclusion

Keeping a cool head and being able to adapt even in the most dire and unforeseeable of circumstances helped me recover and stay on the track with my project.

Student Signature: Lucian Nechita

Supervisor's Signature:

Date: 27 / March /2020

Date: / March /2020

8. Reflective Journal April

National College of Ireland

Course: BSc (Hons) in Computing – Software Development

Month: April 2020

Lucian Nechita, x16149505

Code Docs Automation

Overview

For the month of April, I followed the requirement on the documentation in building the automation system and created the unit test plus the documentation regarding the testing part of the system.

Unit Testing

Project's unit tests were performed with Mocha framework for NodeJS and JavaScript language and is the most popular framework out here at the moment. The testing was quite difficult to perform as the system involves a lot of terminal commands to talk to the operating system and transform the individual parts into beautiful documentation.

User Interface

For the user interface I used Angular framework in the end as I was a bit undecided between front-end frameworks. I choose Angular for the simple reason that is the most used framework within large organisations and this is a great skill and know-how to have under my development "belt" which will boost my professional life.

The system

The system is capable of taking markdown document and transform them into beautiful code documentation which is deployed alongside the source-code itself making the documentation more relevant to the deployed source-code and give developer less chances to have outdated documentation. Developers are also able to tweak the document looks and feel by adding custom settings into the system giving them the ability to keep the uniqueness of their application look and feel.

Conclusion

It was a tough problem try and solve and yet the biggest challenge is to figure out the human inputs towards the documentation and discover a way to automate that, so in the near future I plan to play with speech code documentation automation with the hope of capturing the developer thoughts while he codes and develop his solution.

Student Signature: Lucian Nechita

Supervisor's Signature:

Date: 27 / April / 2020

Date: / April / 2020

9. Showcase Poster



utodocs

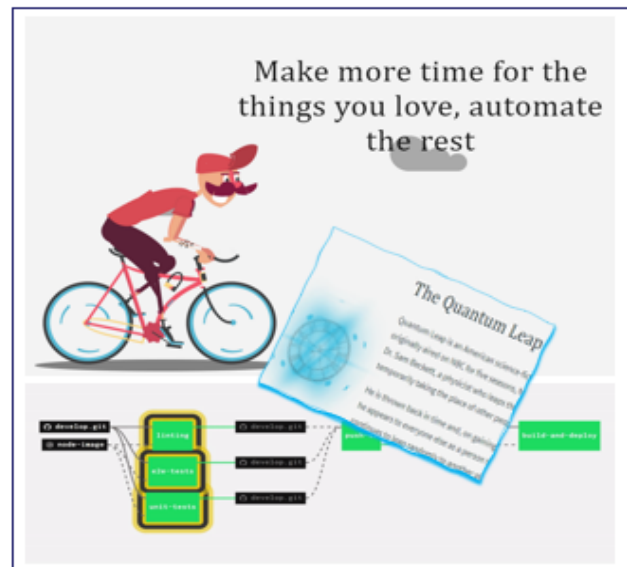
National College of Ireland, BSc (Hons) in Computing 2020/2021
Lucian Nechita, Student ID x16149505

The aim of this project is the creation of a system for automating code documentation process which will gather source documents and assemble them into a ready to deploy web application. The system will work within an automated development process, recognize repository changes, and automatically start a task within the software pipeline publishing the web application to the cloud and making it accessible to end users within minutes.

"As programmers we live in a world of plain text. Our documentation tooling should be no exception. We want tools that turn plain text into pretty HTML." (write the docs)











"We also have some of the best tooling available for tracking changes to files. Why would we forgo using those tools when writing documentation? This workflow is powerful, and familiar to developers." (write the docs)

At its core, utodocs is a dynamic document bundler for modern applications. When utodocs processes your application documentation, it internally builds a table of contents graph which maps every markdown file in your project and generates http link's which point into your application GitHub repository.



10. Attachments

Original files for above appendices.

 x16149505 Reflective Jurnal Par	 x16149505 Reflective Jurnal Par	 Lucian Nechita Project Proposal .do	 x16149505 Reflective Jurnal Par	
 Project Proposal Lucian Nechita.pdf				
 x16149505 Reflective Jurnal Par	 x16149505 Reflective Jurnal Par	 Lucian_Nechita_Mar ch_Reflective_Journ	 Lucian_Nechita_Apr il_Reflective_Journal	 Showcase Poster.pptx