REROLL INC

Sprints Overview

Sprint 1 Overview

Sprint 1 - Planning

Project Overall View:

BUILD AN INTERFACE

Interface…

- has a form to be used by anyone to input a text from someone and another input that's potentially from them and compare the similarity between the texts
- potentially can offer similarities/differences details in the documents - highlight the similarities and differences in the document
- potentially add feedback and comments on the analysis
- has the ability for users to create an account and sign in using it
  - accounts will offer the ability to create knowledge banks (i.e.: profiles) for easier comparison
- will be interactive, clean and intuitive

SPRINT GOAL - Low-fidelity and high-fidelity prototypes with fully developed background work

TEAM ROLES

Identified and delegated roles for responsibility: Roles

TASKS:

Motivational Model -

- identity the motivations, needs and goals of the stakeholders of the project
  - will help with creating a final product that aligns with the expected outcome

Select and create **User Stories -**

- goal to create user stories that in summary, form a meaningful aim and direction for this project
- create a breakdown of the current user stories for further identifiable tasks

Identify the **architecture** that will be used

- Front-end, back-end, authorisation (log-in)

Low-fidelity prototype -

- A rough map of what we wantthe web app to look like

High-fidelity prototype -

- Working prototype of what the web app is expected to look like and work

Create a **persona** of a user -

- help better wrap head around users

CURRENT BREAKDOWN OF ROLES/RESPONSIBILITIES:

Motivational Model - Thomas

User Stories - Adrian

API research and decision-making - Hanizam

Prototyping - Lin and Saaiq

- Lin: low fidel

- Saaiq: high fidel

Maintain and update JIRA

TEAM RESPONSIBILITIES:

- ensure everyone is participating, ideas are listened to and respected
  - use when2meet for optimal involvement!
- standups every week for progress checks, designated day: Monday 8:30pm

NOTES:

- Meeting minutes will be recorded
  - will include decisions made

Sprint 1 - Review

August 25 - Friday

Review of the sprint

- Sprint goal: Low-fidelity and high-fidelity prototypes with fully developed background work
  - In accordance with the team and what was presented to the product owner, the sprint goal was met and ample planning was done throughout in expectation and preparation for the next phase of the sprint

Product Preview

- Product website prototypes, both high and low fidelity, were presented to Eduardo. The prototypes were created in accordance of what was given to us and to the user stories we created based off of it
  - Project overview: Project Overview
  - User stories: User Stories - had a follow up in a second meeting (1st Sept)
  - Prototypes
    - Low-Fidelity: Low-Fidelity Prototypes
    - High-Fidelity: High-Fidelity Prototype
- We also presented a rough overview of what was completed for this, summary of this can be found in Sprint 1
- Initial presentation included log in, signups, knowledge banks for profile storage after signup (knowledge banks - storage of multiple texts added for more precise analysis)

Feedback

- Full details can be read about here: Meeting with Eduardo
- Summary:
  - Feedback to focus on basic implementation for the website, moving extension additions to a version after Version 1.0 (text input of known and unknown texts, comparison via model, presentation of results)
    - Given advice to have a nice documentation of current and extensions for the potential of handing the language processing websites to future group to implement
  - Text corrects and additions to sections of the website such as titles and instructions where needed
  - Minor experience notes
  - Eduardo was largely happy with what was presented and plans going forward.
- 1st September Follow up meeting
  - Updated user stories presented and received a thumbs up
  - Eduardo let us know that the website was primarily designed for **Teacher users** and we proposed the alternative of **Author users** and minor addition of **Student users** which he gave the thumbs up to both

Moving ahead:

- Start with the basic stuff for the website - with UI and comparing of texts using model. Treat the profiles as an extension and a version to be added on afterwards.
- Less is more - make something that really works (what file types are accepted?), can extend later
- Work on version 1.0 as a primary basis - Jira board has been updated

Sprint 1 - Retrospective

WHAT WENT WELL

- Prototyping of the website
  - Rough, low and high fidelity prototypes liked by everyone and got everyone's input
- Good feedback
  - everyone gave good feedback to each other when help and opinions was needed
- Good communication
  - kept up to date with all the tasks and got feedback from everyone on what they're doing on a weekly basis, everyone knew what everyone was doing with their tasks largely
- Achieved sprint goals
  - Sprint goals achieved!
- Solid planning from first sprint
  - First sprint planning laid out the tasks that we felt were required and we were able to refer back to planning to ensure we were on track and had all tasks ticked off
- Laid a good groundwork and direction for second sprint
  - including understanding model and technologies (big picture and small picture)
  - sprint 1 kept the second sprint in mind where decisions also held sprint 2 in mind, made sprint 2 beginning a lot easier/smoother
- Everyone did their tasks and did it well
  - very proud of everyone!

WHAT CAN WE DO BETTER

- Manage our Jira board more consistently
  - There were pockets of space and time where the Jira board required a mass update and breakdown of tasks to properly ensure delegation of tasks in the right direction
  - ⭐ plans to update consistently per progress in tasks, team members will keep each other reminded in collaboration
- Document all of our processes in one place
  - maintain consistent documentation, team members documented relevant tasks and progress on different platforms which required collection and merging of them all
  - ⭐ ensure documentation occurred in a consistent space - **Confluence, Shared Notion**
- More frequent standups
  - ⭐ Whilst standups up happened, we aim to have an additional stand up per week (moving it up to two standups a week) to keep track of progress and have a frequent update between members and delegated tasks
- Creating more specific and target driven breakdown of tasks
  - Is more individual based as tasks per individual could be broken down into more specific targets and tasks to help each member themselves and others with progress and recordkeeping
  - ⭐ Breakdown of tasks during backlog creation to help future selves
- Communicate more over slack
  - Whilst main communication was happening over slack, we had more individual communications directly with team members happening on different platforms due to preference.
  - ⭐ We aim to have more communication on slack to help everything be more cohesive especially for decision making

ACTION

- Sprint 2 planning!

    - begin sprint -
        - start development for both back-end and front-end
    - create backlog of tasks on Jira, ensure good breakdown of individual tasks
    - identify goals we aim to hit
- create a recurring standup meeting twice a week to meet and achieve agile process

Sprint 2 Overview

- Sprint 2 - Planning
- Sprint 2 - Review
- Sprint 2 - Retrospective

Sprint 2 - Planning

SPRINT GOAL - Create a functioning website with UI and modelling

TASKS:

- Jira board backlog creation
- Github notes and maintenance

Create the **front end** of the website

- include all the functioning sections that was planned in the prototype
- build in preparation for future inclusion
- ensure all functionality works as intended and *user experience* is optimized

Understand the **backend**

- be able to optimise the backend where options and applications are available
- utilise the backend to help design the frontend
    - extract information from the backend to apply/present

Create text for:

- buttons, instructions, about, welcome screen, learn more
- ensure all instructions and text portions

CURRENT BREAKDOWN OF ROLES/RESPONSIBILITIES:

Thomas and Adrian: Breakdown of front-end tasks

- delegation
- update Jira board
- **Landing page + welcome screen**
    - probably the same page, welcome screen as an overlay
        - decision pending on fade or slide-out
- Navigation

Saaiq and Hanizam: Focus on the model and linkage to frontend

- full understanding of the code
- understanding the front end needs to provide to the backend to run
- linkage focus
- backend optimisation

Lin: Clean up of functioning prototype/design

- semantics
- all text and instructions
- button names

TIMELINE ESTIMATIONS:

Done by week 8-9

TEAM RESPONSIBILITIES:

- ensure everyone is participating, ideas are listened to and respected

- use when2meet for optimal involvement!

NOTES:

Sprint 2 - Review

Review of the sprint

- Sprint goal: Create a functioning website with UI and modelling
  - Through a meeting with Yue Song (tutor) in the absence of our client, and a team meeting, the present sprint goal was met and achieved. Groundwork was laid to move forward with changes as stated below under *Moving Ahead*

Product Preview

- The fully functioning website created by the view of the prototype was presented to Yue. Changes were made since the prototype and have been documented as such.
  - Current website: Current Website
  - Updated and changes to prototype: Prototype Changes
- Presentation included the full functionality of the website from its text input, document upload, comparison and results.
- The user stories, motivational model and personas were updated to fit the current rendition and feedback from previous sprints: Motivational Modelling [UPDATED] User Persona
- We also presented a rough overview of what was completed for this, summary of this can be found in 2 Sprint 2
- Presentation also included the plans we intend to complete moving forward including but not limited to: sign up, knowledge banks and testing/deployment (sprint 3 planning)

Feedback - 22nd September

Details of the meeting

- Feedback we received was largely functionality-based and on user experience:
  - Due to the long length of our textbox, users would have to scroll more than usual to see the documents that they have uploaded (which is present underneath the text input area). A shorter text area would benefit user experience
  - The button to add the current text area's text as a document was unlabeled and thus for a user to understand the buttons use, it would be beneficial to include a description.
  - Upload button's visibility increase to users
- The feedback has been documented and will be worked on immediately after the wrap up of this sprint.

Moving ahead:

- Wrap up and implement the feedback received
- Given enough time after sufficient research into testing and deployment, we aim to complete the implementation of signups, testing, and deployment.
- Visually, our website feels a bit lacking which will be a first priority due to user experience.

Sprint 2 - Retrospective

WHAT WENT WELL

- Completed the website version 1.0
  - Got the link with the back-end and front-end running
  - Sped up algorithm processing time using saved models
  - Relatively in line with initial prototype design but adjusted to match needs
  - (and beyond! version 1.1) Upload files for known and unknown texts with various options and file types
- Good feedback, teamwork and communication
  - good responses and feedback through all the tasks along the way
  - everyone communicated well through allocated channels
- Everyone did their tasks and did it well
  - very proud of everyone again!
- Handled our Jira board a lot more consistently
  - was updated along the way and were checked in on more frequently
- The learning process!
  - Everyone had to learn new things in order to complete this part of the sprint and it was handled really well
- More frequent meetings were held compared to last sprint
  - two to three meetings a week was achieved! We were able to keep up to date with members more frequently

WHAT CAN WE DO BETTER

- Document all of our processes in one place (carried over from last sprint)
  - There were less documentation to keep track of this time but was still a little scattered, updates to specific portions were not always easy to find or were missing
    - maintain consistent documentation, team members documented relevant tasks and progress on different platforms which required collection and merging of them all
  - ⭐ ensure documentation occurred in a consistent space - **Confluence, Shared Notion, Slack**
- Communicate more over slack
  - We had improved communication compared to last sprint, with more sharing of updates and resources but there were still moments where process updates and a few commits could’ve been notified
  - Communication happened largely over slack this time which was an improvement
  - ⭐ We aim to have more communication once again on slack to help everything be more cohesive during the update of progress and decision making
- Better handling of our Git repository
  - Due to ease of merging straight to main since less merge conflicts happened when merges were organized between members, branches and commit guidelines were largely neglected thus we aim to bring it back on track
  - ⭐ Follow the git, coding and merging guidelines that are now currently in place for future progress from henceforth

ACTION

- Sprint 3 planning!
  - begin sprint -
    - start research and planning into testing and deployment for our website
  - create backlog of tasks on Jira, ensure good breakdown of individual tasks
  - identify goals we aim to hit for the final and last sprint
- Ensure everything is coherent and up to date for the last stretch

Sprint 3 Overview

- Sprint 3 - Planning
- Sprint 3 - Review
- Sprint 3 - Retrospective

Sprint 3 - Planning

SPRINT GOAL - Complete fully functional website, alongside successful testing and deployment

TASKS:

Round out the website for **User Experience**

- create instructions
- identifying labels for button
- styling explainations
- updating/recreating home page
- background for website

Testing

- functional, non-functional and structural testing documentation
  - acceptance criteria → acceptance test → test cases
- record/update bugs list

Deployment

- CI/CD
- implement github action for testing
- AWS for deployment (look into S3 bucket and EC2)

CURRENT BREAKDOWN OF ROLES/RESPONSIBILITIES:

- Saaiq:
  - Deployment process documentation
  - implementation of CI/CD
  - deployment
- Thomas:
  - Non-functional testing
- Adrian:
  - Test cases and bug recording
- Hanizam:
  - Structural testing
- Lin:
  - Acceptance criteria and acceptance tests

TEAM RESPONSIBILITIES:

- ensure everyone is participating, ideas are listened to and respected
- standups every week for progress checks, designated day: Monday 8:30pm
- Communicate effectively and frequently to ensure everyone is on the same page for testing and deployment

NOTES:

- Last stretch, lets go guys!

Sprint 3 - Review

Review of the sprint

- Sprint goal: Complete fully functional website, alongside successful testing and deployment
  - In the absence of our client, through an internal review, we determined that the current sprint goal has been met and achieved. All documentation has been recorded for any potential future implementation.

Product Preview

- Only minor changes were made to the website including the updated home page for aesthetics
- The product was showcased today on the 20th October, demonstrating key functionality and features of the website that was developed across all the sprints
- Testing notes include that of functional, non-functional and structural. This can be found here: Testing
- Added tutorial button to show instructions for website use alongside an updated help page with more in-depth assistance. Tutorial Images
- Changes that have been made since beginning (things that have been moved to potential future extensions) and since sprint 2: Changes Made Since
- Deployment was successfully achieved, notes of which can be found here: Deployment Process & CI/CD
- Website update with the new GitHub Repositories and deployed link: Website Updates
- User stories have also been finalised to the single teacher user, and has been polished: Updated User Stories

Feedback - 13th October

Summary of the meeting

- Feedback we received was largely positive, the few things Eduardo had mentioned were related to help functionality such as more clarity in explanations and a filled help page. Details can found
  - Feedback was implemented into the final version and presented in the product showcase

Moving ahead:

- Prepare documentation for handover

Sprint 3 - Retrospective

WHAT WENT WELL

- Successful deployment of the website
  - includes successful implementation of the CI/CD pipeline
  - and GitHub actions added in for fast unit testing
- Overall full tested website
  - was able to cover a large scope of the website under our testing
  - testing was largely all very successful!!
- No cost deployment
  - Definitely a pride point
- Our front page is animated and got complimented on!
- The overall completion of the entire website
  - despite having plans changed, the overall completion of the website was still to everyones satisfaction
- Good feedback, teamwork and communication
  - good responses and feedback through all the tasks along the way
  - everyone communicated well through allocated channels
- Everyone did their tasks and did it well once again
  - very proud of everyone again!

WHAT CAN WE DO BETTER

We have improved a lot on everything that we have stated before but to wrap up this sprint:

- Better time management to better accomplish things that were once established in our very first sprint session.
  - There were a lot more features that we had envisioned but were not able to implement in our current rendition but will be a lesson we can take into the future.
- Communicate more over a single platform
  - We had improved communication compared to last sprint again, with more sharing of updates and resources but we had split into two platforms towards the end due to the creation of a discord group to keep track of more outside standup, updates ended up split between two channels. This did lead to some confusion along the way
  - ⭐ We aim to have more communication on a single platform in the future to help everything be more cohesive during the update of progress and decision making
- Documentation of testing along the way
  - Testing documentation was something that we only looked at towards the end of sprint despite having done testing throughout the entire progress
  - This led to a few confusions and a lot of time sunk into trying to traceback our progress through confluence to identify where testing had occurred
  - ⭐ We aim to keep testing documentation as a forefront though in future projects

Organization

Roles

Product Owner

Eduardo Oliveira - eduardo.oliveira@unimelb.edu.au

Scrum Master

Lin Xin Xie - linxx@student.unimelb.edu.au *1231766*

Development Team

Adrian Podiono - apodiono@student.unimelb.edu.au *1269317*

Hanizam Zaharyudin - hzaharyudin@student.unimelb.edu.au *1230020*

Jia Jie Zheng - jiajizheng@student.unimelb.edu.au *1271813 (Product owner)*

Lin Xin Xie - linxx@student.unimelb.edu.au *1231766*

Saaiq Ahmed - saaiqa@student.unimelb.edu.au *1272488*

Project Overview

Language Processing

Using a rough (~1000 word) text from a person to use as a sort of “fingerprint” to create a word2vec (vector file of the text by what words are used and words that come before and after)
Use that “fingerprint” to compare with other texts to determine a % likelihood that the text is written by them.

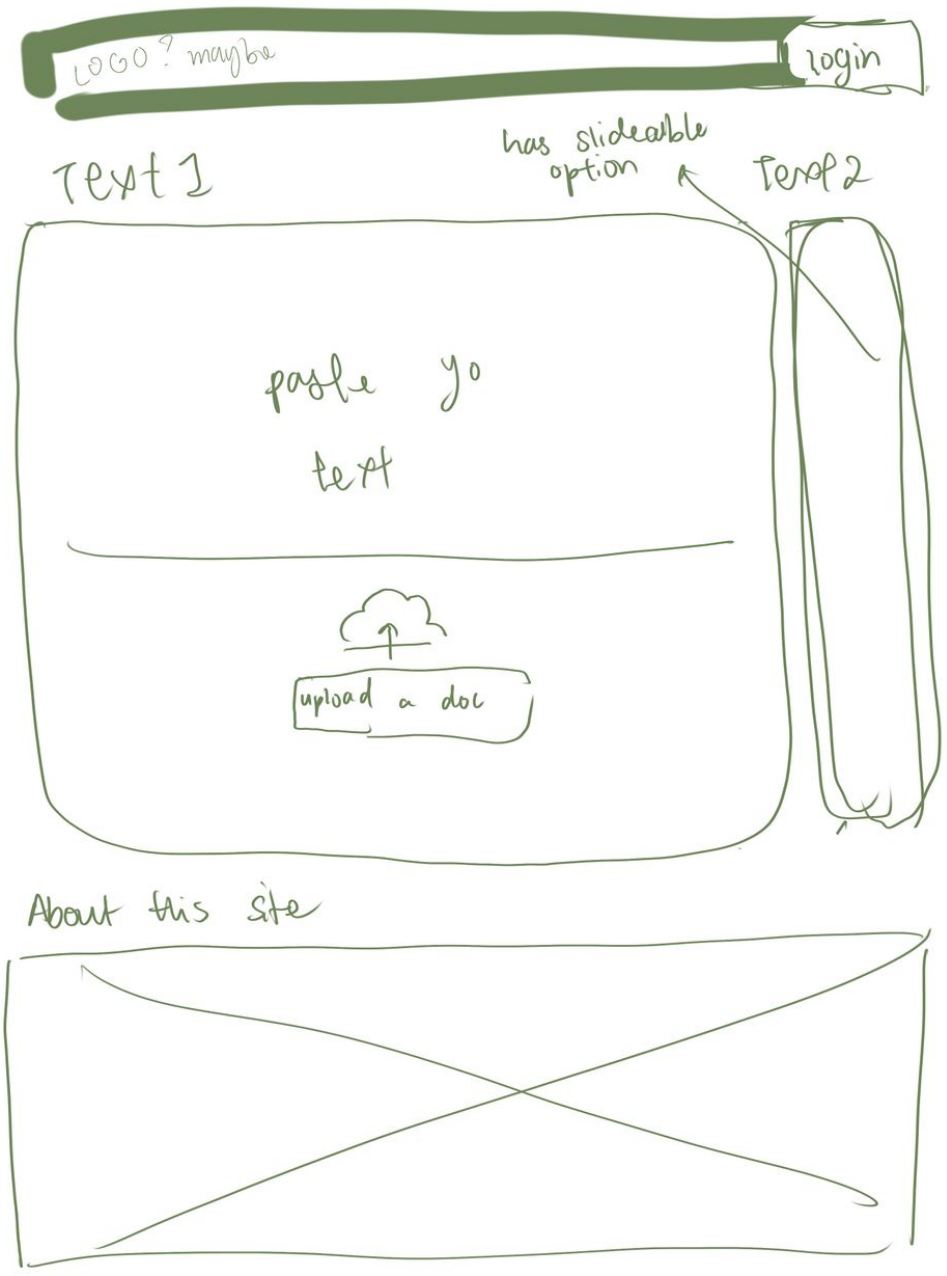
Build an interface

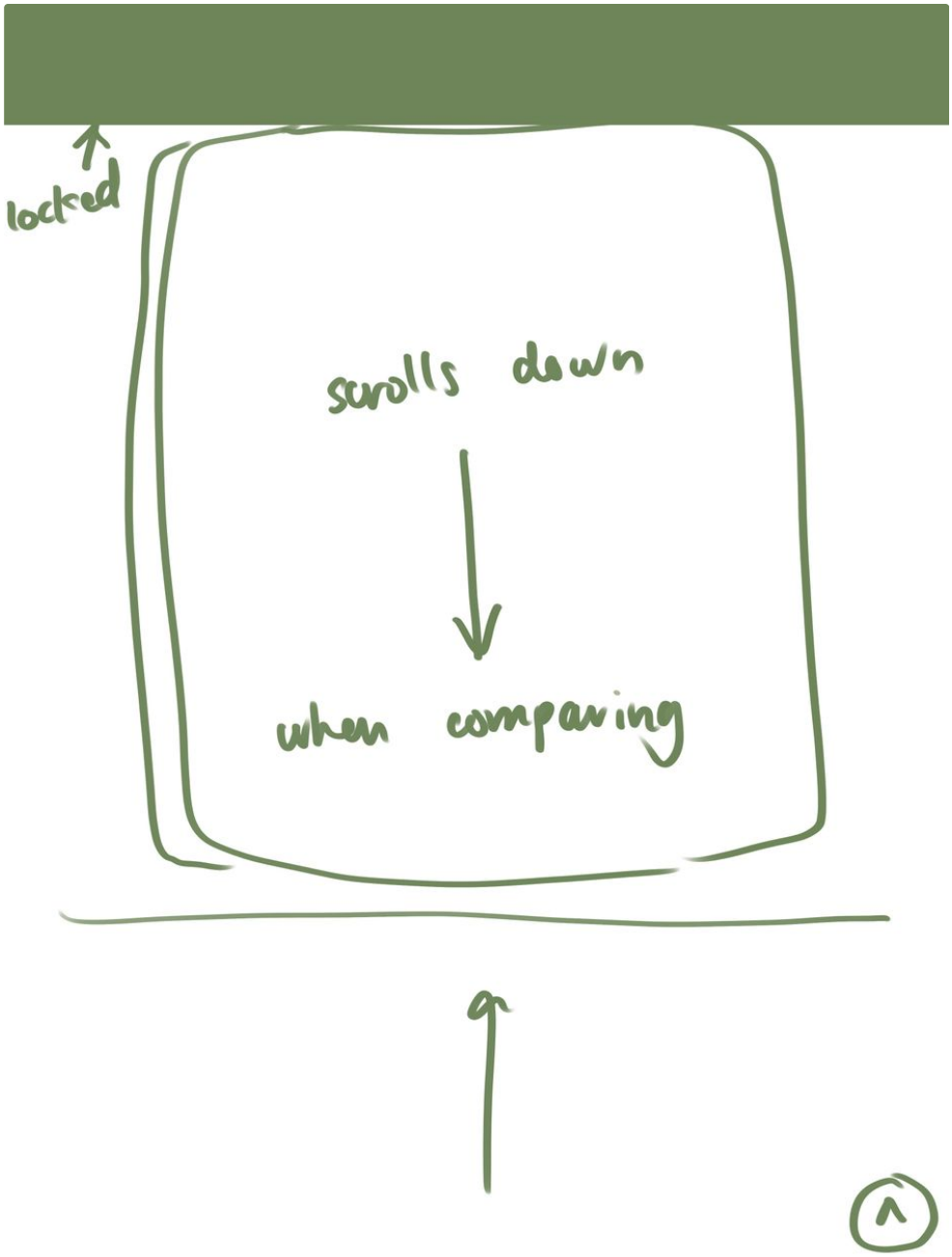
- Has a form to be used by anyone to input a text from someone and another input that’s potentially from them and compare similarity
- For similar/differences in documents, highlight the similarities and differences in the documents
- Potentially add feedback and comments on the analysis
- e.g. Usually use 1 rare word in main input but other input has 12 rare words
- Potentially incorporate multiple texts into someone’s profile
- To account for vocab/literary development

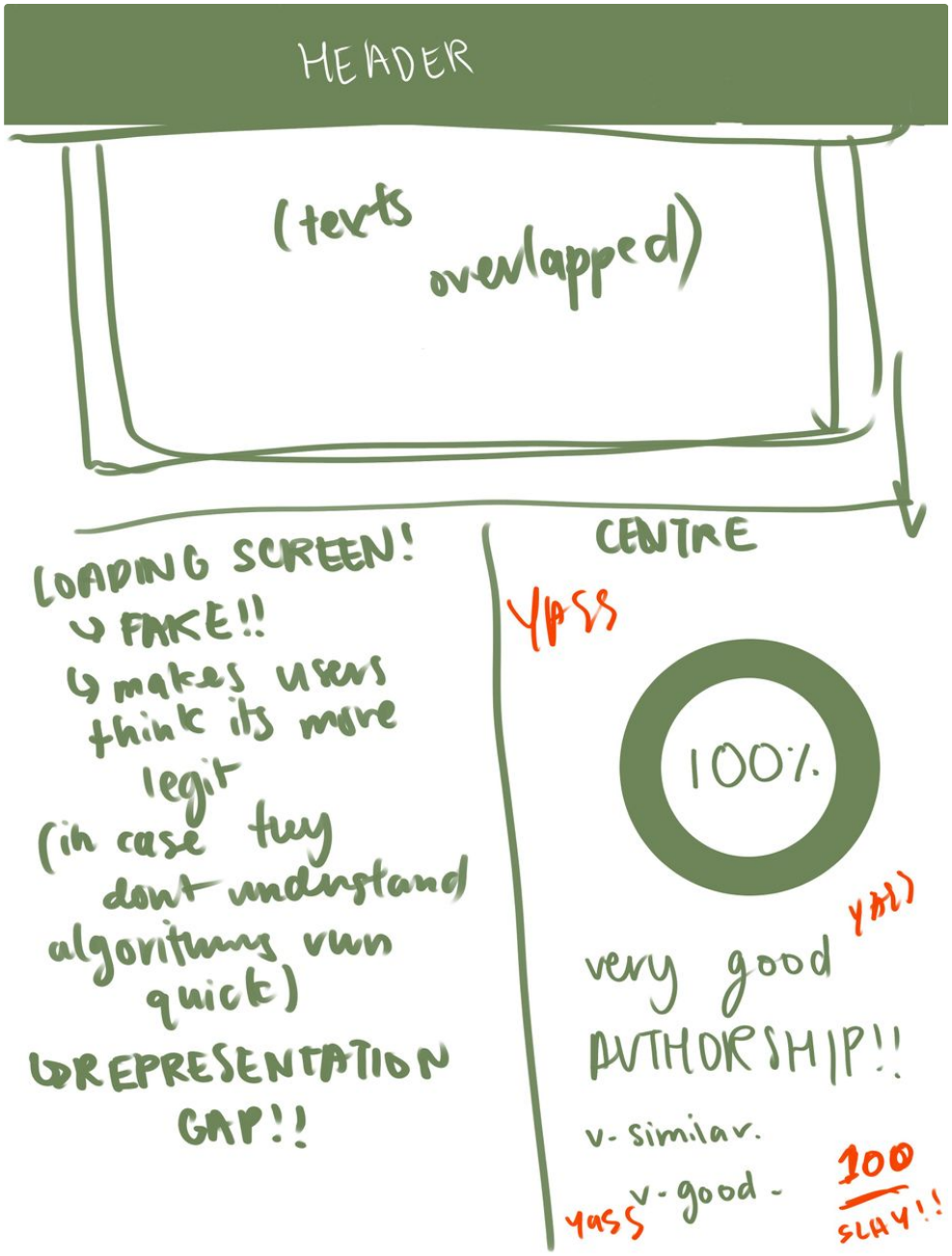
Prototypes

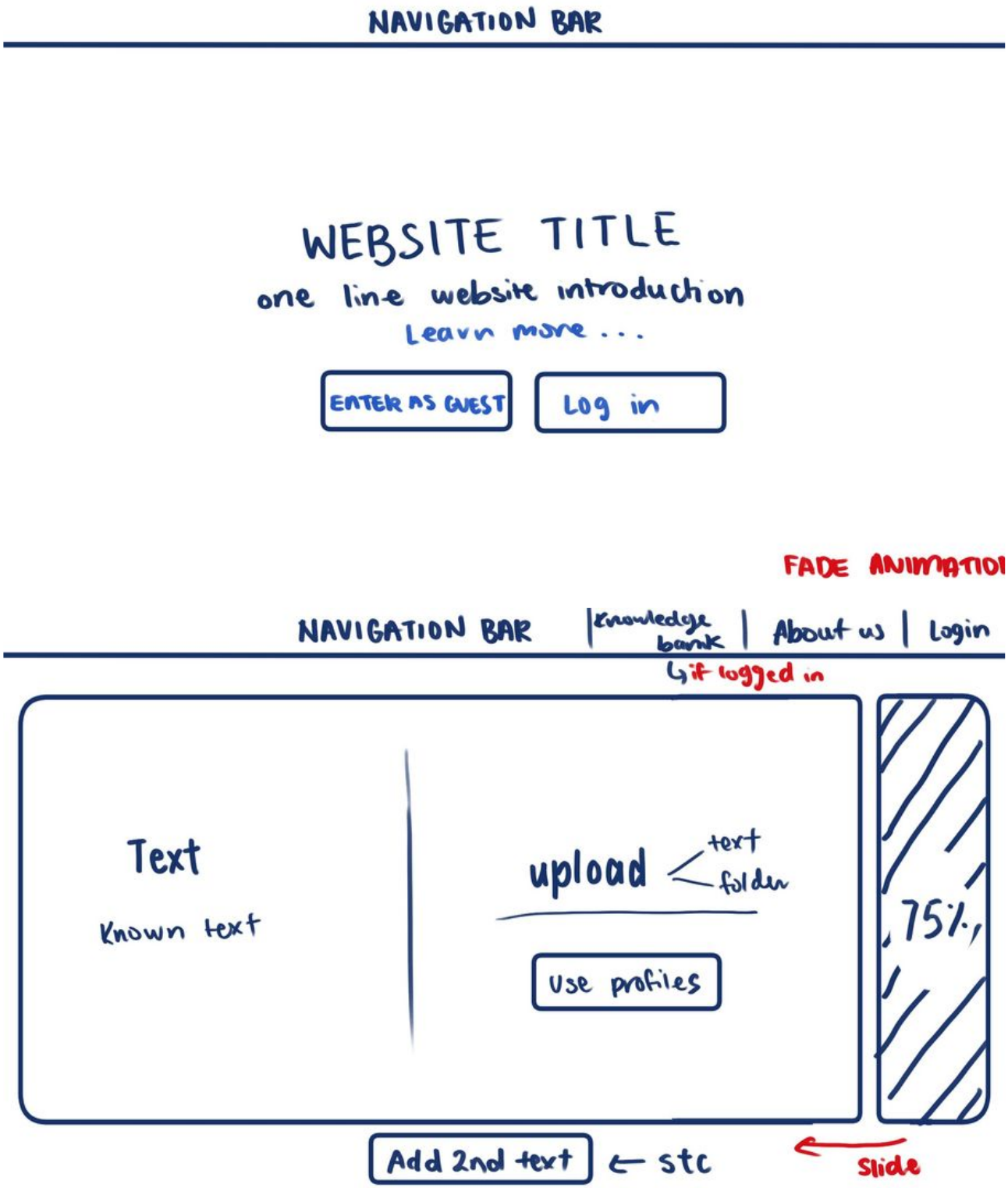
- Rough Prototyping
- Low-Fidelity Prototypes
- High-Fidelity Prototype

Rough Prototyping




locked
scrolls down
when comparing

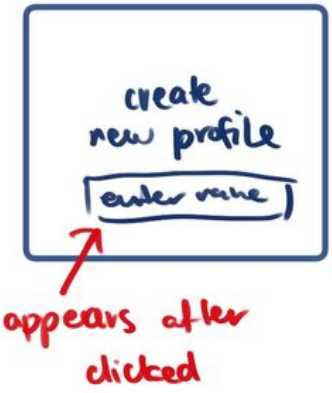

HEADER
(texts overlapped)
LOADING SCREEN!
↳ FAKE!!
↳ makes users think its more legit
(in case they dont understand algorithms run quick)
↳ REPRESENTATION GAP!!
CENTRE
YASS
100%
very good
AUTHORSHIP!!
v. similar.
yass v. good
100
SLAY!!


NAVIGATION BAR
WEBSITE TITLE
one line website introduction
Learn more...
ENTER AS GUEST
Log in
FADE ANIMATION
NAVIGATION BAR
knowledge bank
About us
Login
if logged in
Text
Known text
upload
text
folder
Use profiles
75%
Add 2nd text
← stc
Slide

if user isn't logged in and wants to use profiles

else:
open
Knowledge bank

KNOWLEDGE BANK



appears after
clicked

About

Text

KNOWLEDGE BANK - PROFILE NAME

Instructions

- ~~~
- ~~~
- ~~~
- ~~~

UPLOAD

- docx or pdf files only

- if already uploaded.

KNOWLEDGE BANK - PROFILE NAME

file name	date added	file type
hello_world.py	01/01/0101	doc

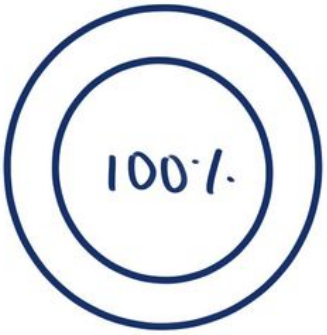
add more

+

Analysing Text...

→ Loading animation

RESULTS

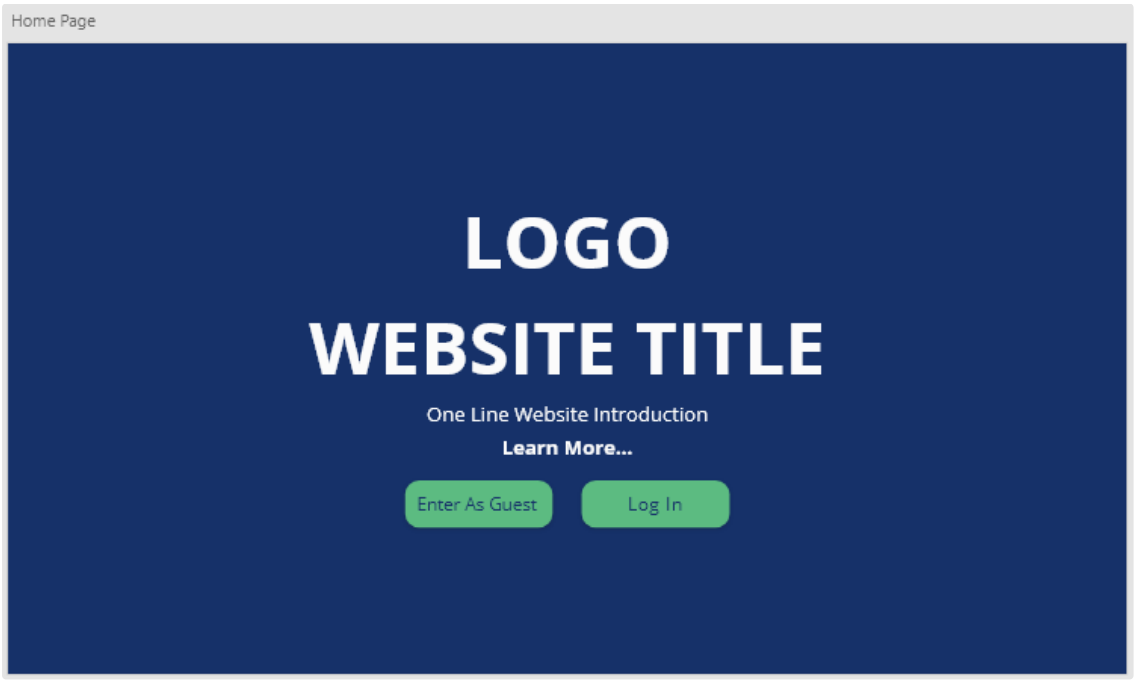


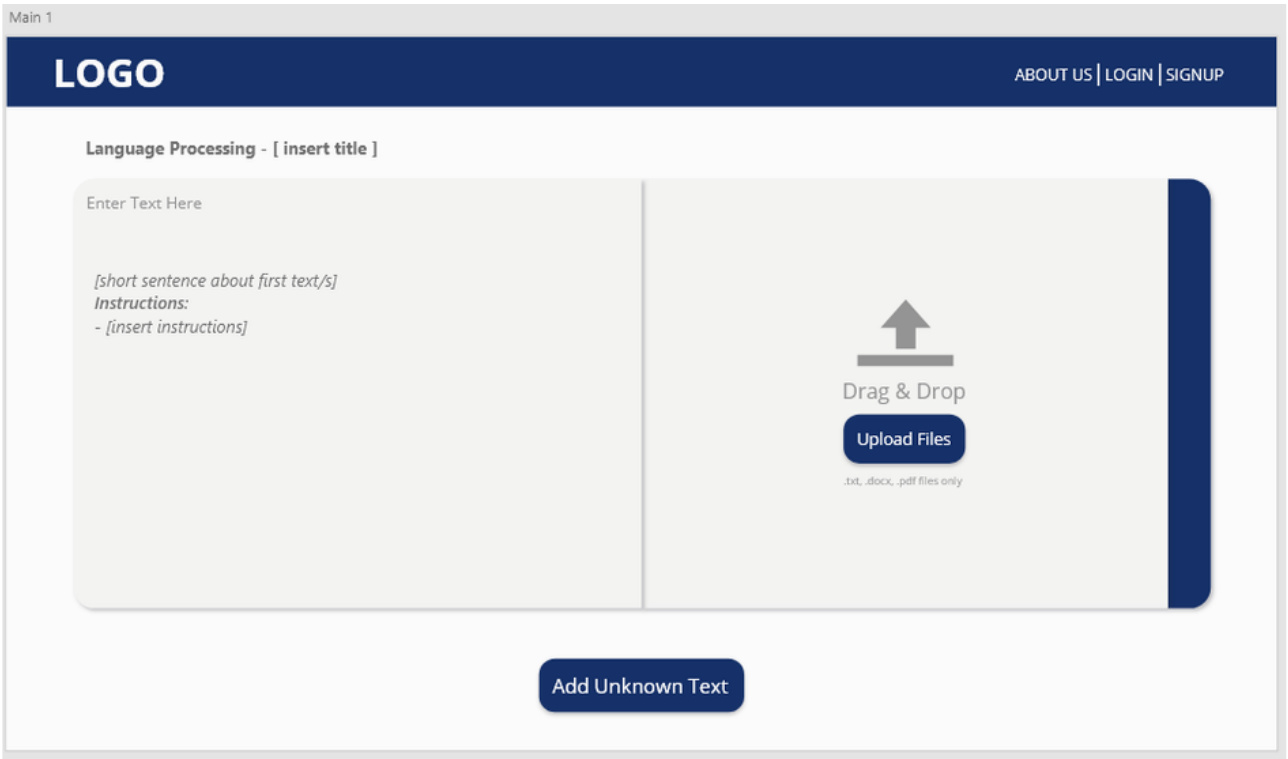
Basic Information

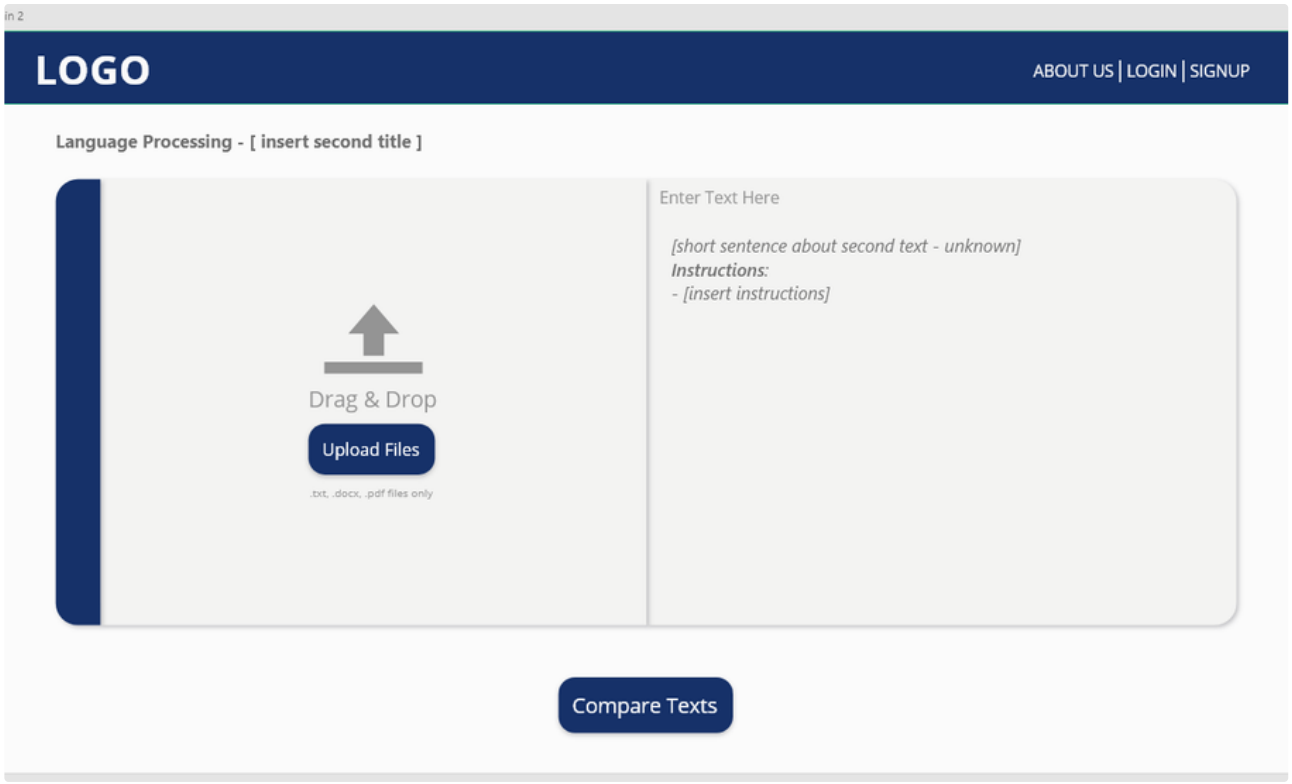
See detailed

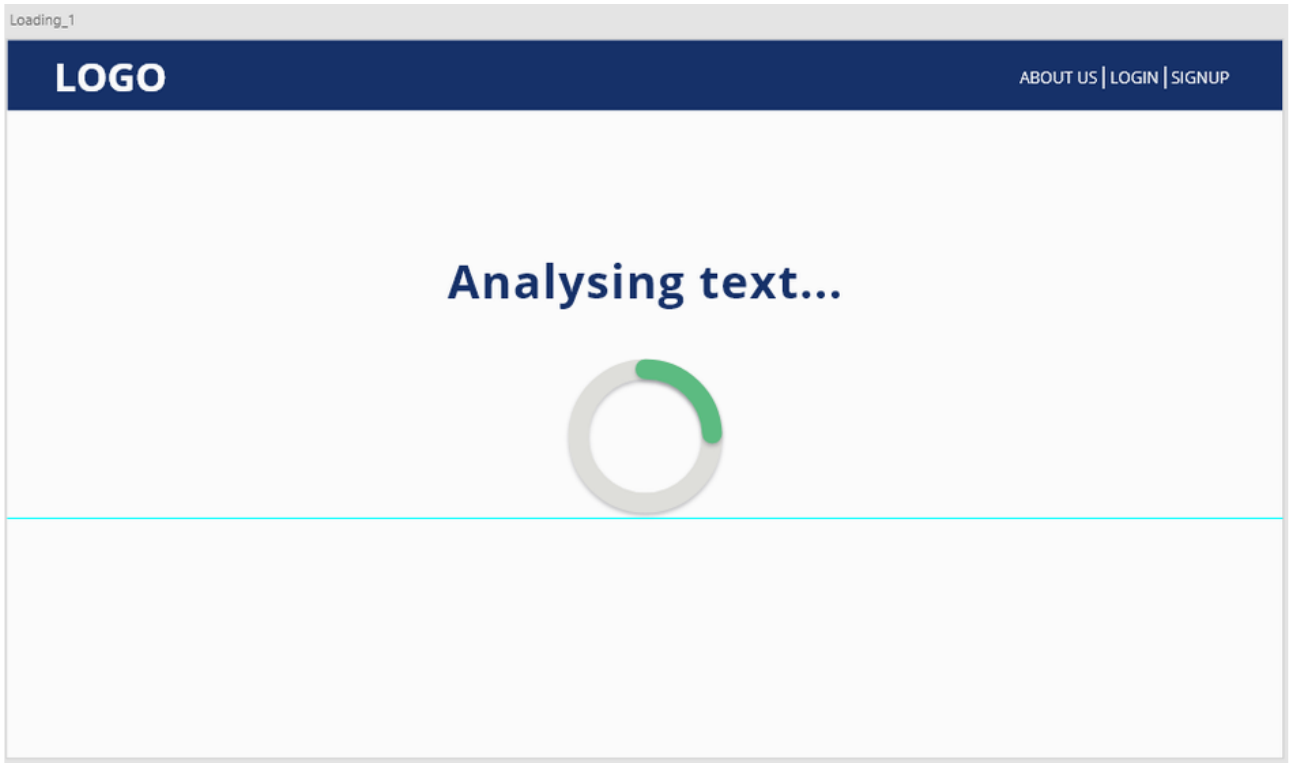
High-Fidelity Prototype

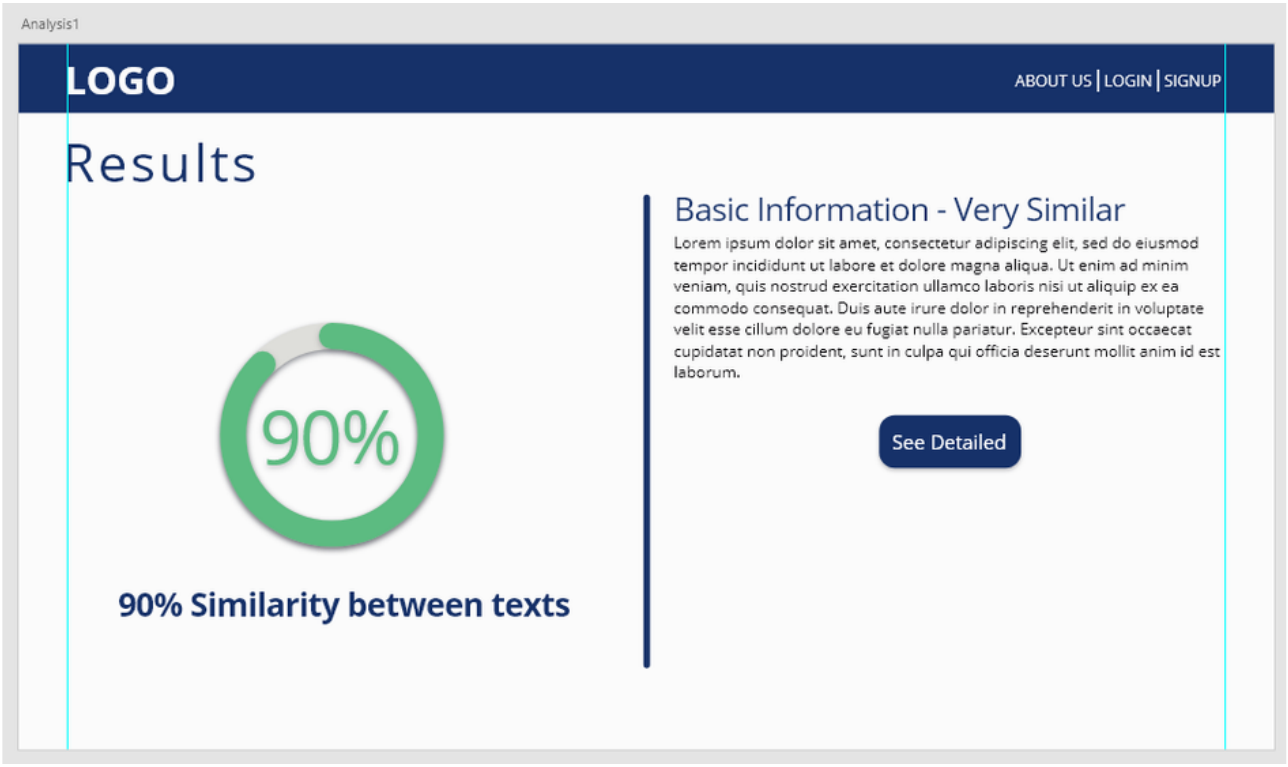
Changes from Eduardo’s feedback have been implemented



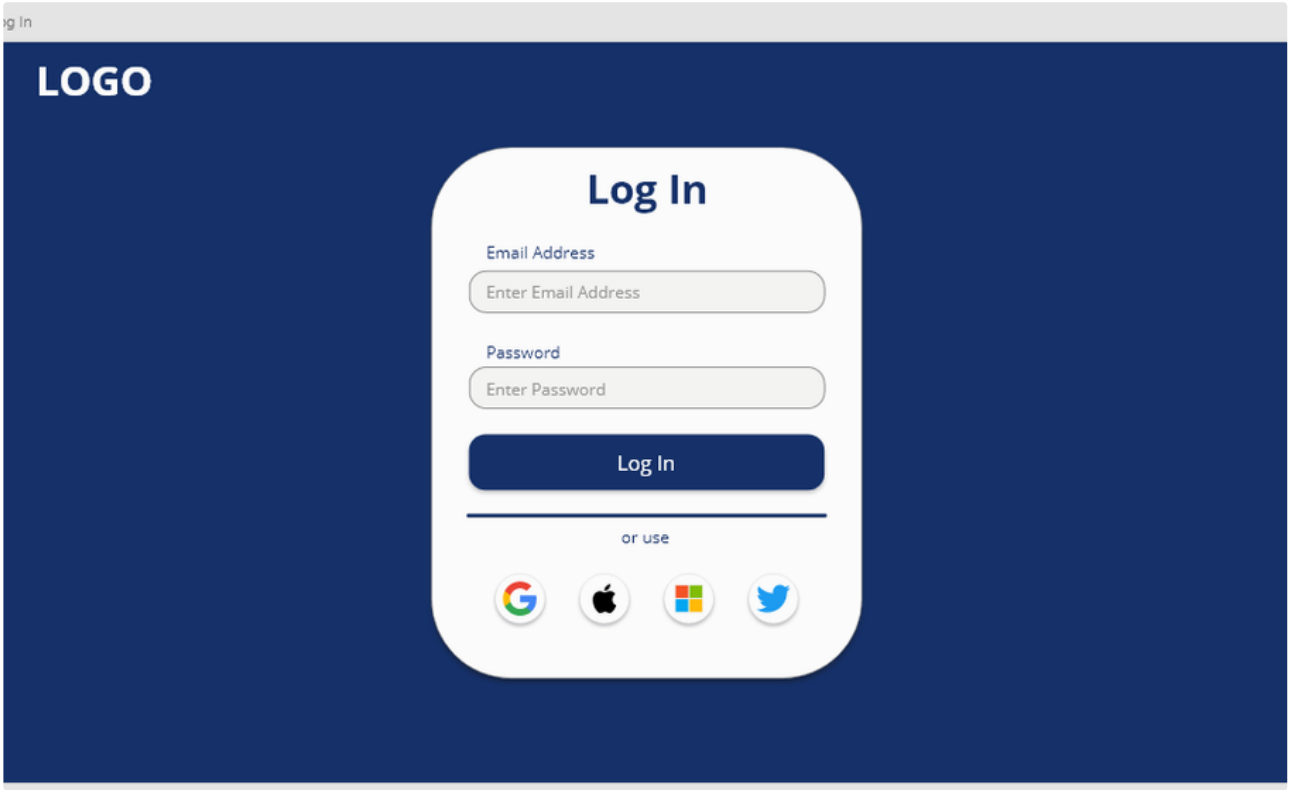



in 2
LOGO
ABOUT US | LOGIN | SIGNUP
Language Processing - [insert second title]
Drag & Drop
Upload Files
.txt, .docx, .pdf files only
Enter Text Here
[short sentence about second text - unknown]
Instructions:
- [insert instructions]
Compare Texts


Loading_1
LOGO
ABOUT US | LOGIN | SIGNUP
Analysing text...



Extensions for future version:



1 Sprint 1

Sprint **Planning**: Sprint 1 - Planning

Sprint **Review**: Sprint 1 - Review

Sprint **Retrospective**: Sprint 1 - Retrospective

Team **Structure**: Roles

Sprint Tasks:

- Visual Design Decisions
- Site Structure and Navigation
- User Stories
- Motivational Modelling
- User Persona
- Architectural Design

Visual Design Decisions

Theme: Smart/Elegant/Academic-core

Typography: Open Sans - Google Fonts

Imagery: Potentially create our own infographics - Lin

Features:

Theme - can toggle between light or dark

- update and add along the way

Colour Scheme: Coolors.co

Main colour - Navy Blue #163169

- inspiration from Unimelb

Accent colour - Green #5cbb81

Logo Designs

Site Structure and Navigation

Sitemap: Home (contains the function), Login, Profile,

Navigation Menu: Home, Login, Profile

Header: Title, Logo (on top left), User Profile (on top right)

Footer: About (what the website does), creditation

=> In the middle: big textbox (maybe 3/4 of the page) with options “paste your text” and “upload a doc” below it. The textboxes stack on top of each other like cards.

=> When you compare the texts, the site should automatically scroll down. Add a “back to top” button.

Website Decision

- Fake loading screen - depending on whether or not the model runs within a time limit or not
  - Representation gap between user and developers
    - A quicker load time can give off the impression for the less technologically inclined to believe that the website and model did not run clearly and decisively
  - In the case that the texts that are being analysed do require time to run, the loading screen will serve a purpose to let the user understand that its running
    - (!!) idea to have a message pop up if the texts are long, that the model is taking longer to run

👥 User Stories

AIM: To compare a set of texts that we know come from an individual and compare that with another given text to see the likelihood of the second text being written by the same individual that wrote the first set

---

EPIC: Compare the authorship of texts in a simple, elegant and intuitive manner.

As a **Teacher** User**:**

- I want to be able to compare my student's texts to see if they were written similarly
  - Medium
  - To be able to check for plagiarism between student works
- I want to be able to create a profile for my students to make it easier to identify their work/authorship
  - Medium
  - To ensure that their work was done by themselves
- *(Optional)* I want to be able to upload a folder/create a directory to create/manage my student's profiles

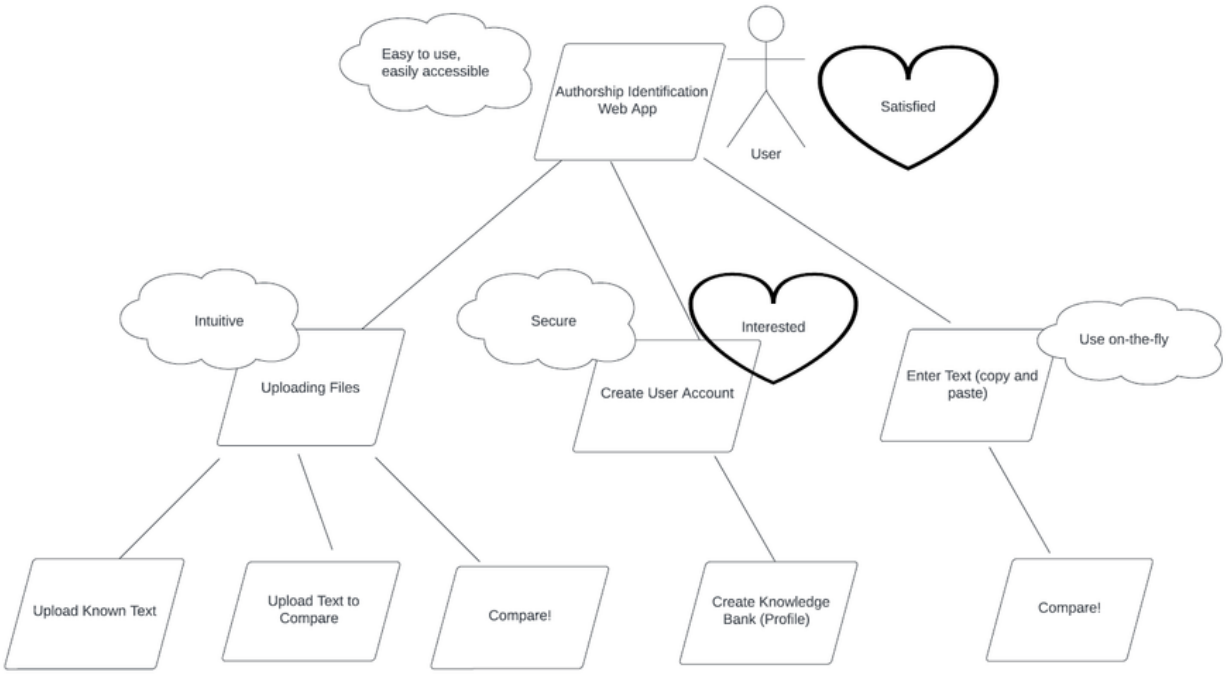
---

As an **Author/Student** User**:**

- I want to be able to compare one of my texts with another one of my texts to see if my authorship match
  - High
  - Must: main function of website, can help ensure consistency across works
- I want to be able to get a more accurate reading of my authorship similarity by adding more texts
  - Medium
  - Should: I should be able to get a more accurate score for a comparison by allowing the website too to look at more of my work
- (!!: Extensions)
  - I want to be able to see which parts of my texts have a higher similarity
    - (function: create a pdf and highlight similarities)
- I want to be able to add more of my original work to create my authorship profile
- I want to be able to see the history of what I have compared before
- I want to be able to see if my work has a high similarity to a text/author I have referenced from

---

Motivational Modelling



User Persona

Gerard Cain

"I need a tool that can identify potential instances of plagiarism without false positives."

Age: **45**
Work: **Professor of English Literature**
Family: **Married**
Location: **Melbourne, Victoria**

Personality

Introvert	Extrovert
Thinking	Feeling
Sensing	Intuition
Analytical	Creative

Organised | Hard Working | Empathetic

Goals

- Verify the originality of his students' assignments to prevent plagiarism
- Provide constructive feedback to students while maintaining trust.

Challenges

- Increasing instances of suspected plagiarism among his students.
- Balancing the need for strict plagiarism detection with fostering a positive learning environment.
- Problems with the currently available solutions.

Bio

Dr. Gerard Cain is a professor of english literature. He has a Ph.D in English Literature from the University of Melbourne. He has been teaching for 15 years and is well known for his expertise in Australian literature. Dr. Cain is dedicated to maintaining academic integrity in his classroom and ensuring that his students submit original work. He is proficient with common academic software tools and is open to using new technology to improve the educational experience for his students.

Motivation

Price
Convenience
Speed
Accuracy

Behavioural Insights

- Regularly assigns essays and research papers
- Values educational technology that assists in maintaining academic integrity
- Wants a user-friendly tool that can me integrated into his learning management system (LMS)

Emma Rodriguez

"Sometimes, I worry that my writing might appear inconsistent, especially when I'm exploring different topics "

Age: **21**
Work: **Student**
Location: **Melbourne, Victoria**
Gender: **Female**

Personality

Introvert	Extrovert
Thinking	Feeling
Sensing	Intuition
Analytical	Creative

Diligent | Motivated | Inquisitive

Goals

- Compare own texts to see if authorship matches
- Get accurate information about their own writing style
- Ensure consistency across works of writing

Challenges

- Struggles with maintaining a consistent writing style across assignments
- Concerned about accidental plagiarism when writing essays

Bio

Emma is a motivated student with a passion for literature. She is in her final year of her bachelor's degree studying English literature at the University of Melbourne. She often works on various writing projects such as essays and creative writing assignments as part of her studies. She also has a strong desire to improve her writing skills.

Motivation

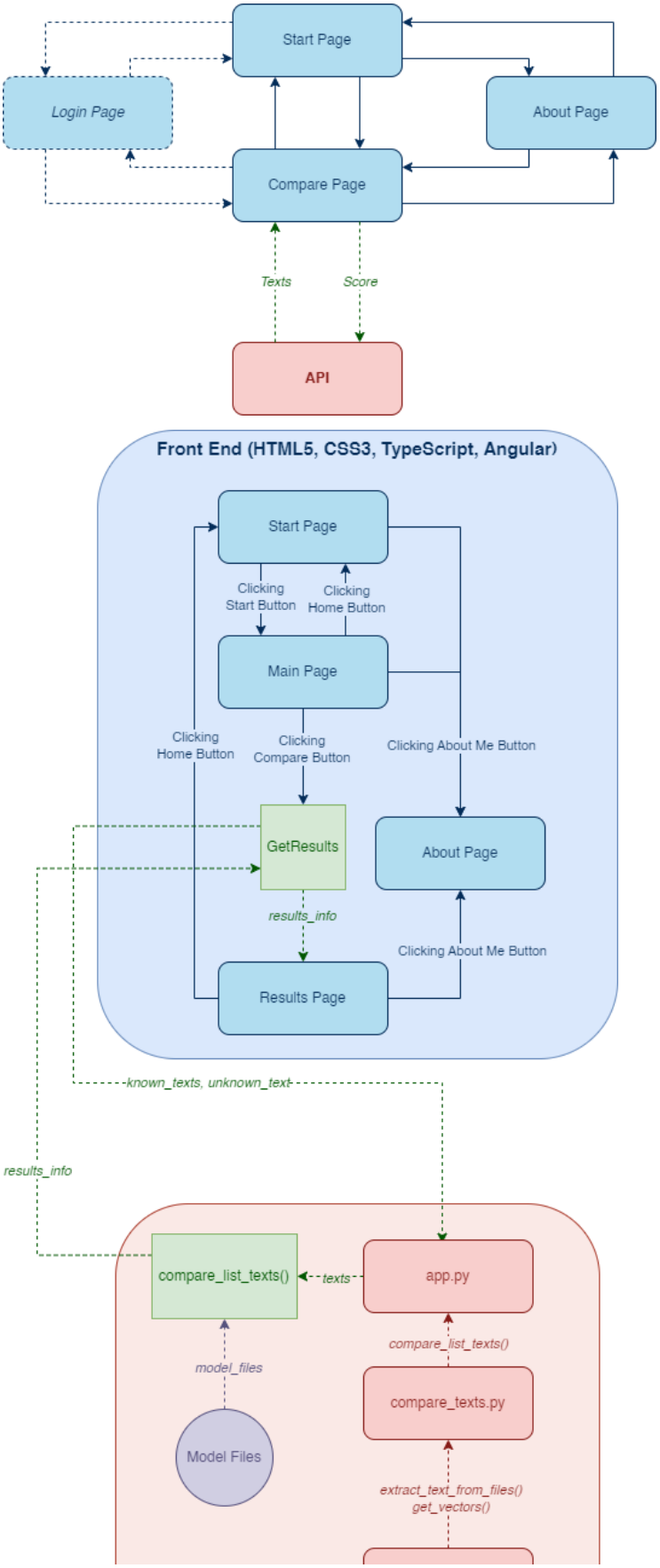
Price
Convenience
Speed
Accuracy

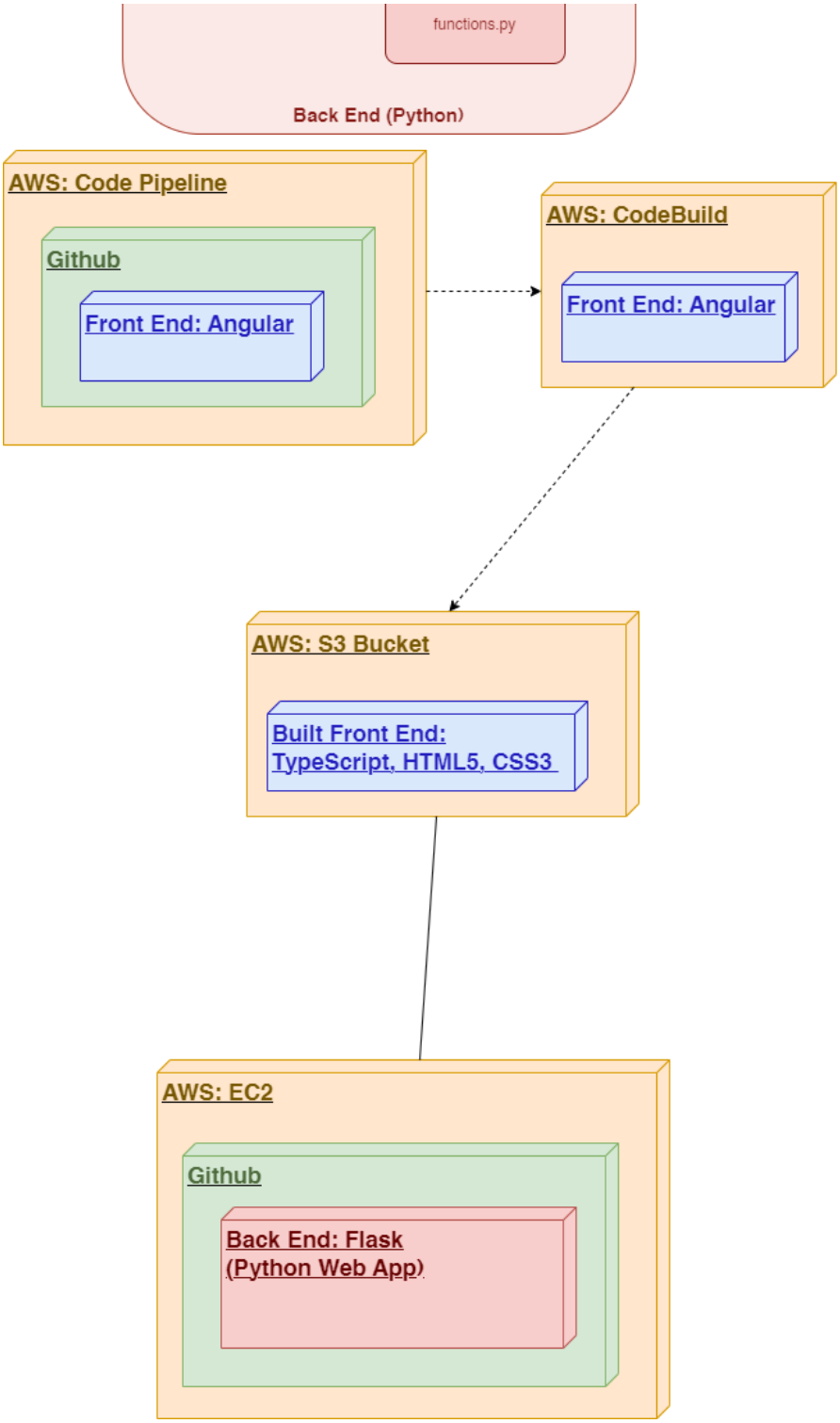
Behavioural Insights

- Regularly writes essays or creative pieces
- Values a user-friendly and reliable tool to help assess her own work
- Actively seeks out resources and tools that can help improve her writing skills

Architectural Design

High-Level Structure & System Behaviour





Where `compare_list_text()` takes in the texts inputted from the front end and the `model_files` from the algorithm and outputs `results_info` which is the authorship percentage.

Performance Optimization Considerations

- **Image optimization:** File formats, compression, resolution
- **Minification**: HTML, CSS, JavaScript files
- **Caching**: Browser caching, CDN usage
- Page load speed considerations

Database

Dependencies

- SQLAlchemy (ORM – object relational mapper) - used to access database

Working with Engines and Connections — SQLAlchemy 2.0 Documentation

- Database Driver/ Dialect

Dialects — SQLAlchemy 2.0 Documentation

Possible database

1. PostgreSQL
2. MySQL and MariaDB
3. SQLite
4. Oracle
5. Microsoft SQL Server

SQLAlchemy packages used

- create_engine, ForeignKey, Column, String, Integer, DateTime
- sqlalchemy.ext.declarative import declarative_base
- sqlalchemy.orm import sessionmaker

declarative_base = base class where all entities class inherited from

sessionmaker = allow us do stuff to database.

Helpful video - SQLAlchemy Turns Python Objects Into Database Entries

Architecture Resources

Resources for Architectural Plan

Part 1 - Set up backend, Database, frontend, flask http request

Using Python, Flask, and Angular to Build Modern Web Apps - Part 1

Part 2 - Set up Auth0 for both frontend and backend

Using Python, Flask, and Angular to Build Modern Web Apps - Part 2

Part 3 - Angular, set up roles in Auth0

Using Python, Flask, and Angular to Build Modern Web Apps - Part 3

pipenv documentation

pipenv

API research

Dependencies

- Flask
- Marshmallow

Flask package used

- Flask
- Jsonify
- request

Marshmallow package used

- Schema
- Fields

Algorithm Research

Algorithm Analysis

The algorithm has a `data` folder that's split into `train_data` and `test_data` where each folder is filled with folders of any name that contains `.txt` files where in which there are 1+ `knownX.txt` files and 1 `unknown.txt` file. *The* `test_data` *and* `train_data` *folders also contain* `contents.json` *and* `truth.json` *files which act as a sort of directory on what folders are to be used and what the answer is for each folder's authorship.*

All the `.txt` files are then preprocessed so as to be more space efficient and only contain the useful word data, this process as stated in the `preprocess_text()` function, "Preprocess a given text by tokenizing, removing punctuation and numbers, removing stop words, and lemmatizing".

Using a combination of all the training data set of preprocessed text, we train a **Word2Vector Model** which uses the trained dataset to learn relationships between words.

We then utilise the **W2V Model** alongside a **Style Vector** (Vector of punctuation, sentence lengths, words used and word count created using the function `calculate_style_vector()`) to create the final vector data which will be used by the model.

The main model itself is then built using a **SiameseNet** which sort of combines the `base_network` and `clf_network` (2 unique neural network models with different layers amounts and activation functions) so that both models can be trained together.

The model then validates the `clf_network` by using both a train and validation set and reducing loss, thus increasing the final model performance.

Finally the models are tested using `test_data` which has been transformed into word vectors, then transformed again using the `base_network` to create `siamese_vectors` that the `clf_network` model uses to get the final percentage score on the known texts vs unknown text authorship score.

Final Algorithm For The Project

The main 3 models that this entire process uses are the **W2V Model**, **Base Network Model** and **CLF Network Model**. So after using the `PAN14_data_demo.ipynb` file to train and validate the models, I saved the model files into a `model_files` folder so that the models can be reused without needing to be retrained, thus drastically speeding up the processing.

After the algorithm analysis I found the main process to get an authorship score for a folder of known and unknown texts was using the function `extract_text_from_files()` to get the processed texts and transform them into a `numpy` array word vector using the **W2V Model**. I then used the **Base Network Model** to obtain predictions on the known and unknown word vectors which were then transformed into representations by getting the `numpy` means. A combined array of the known and unknown representations was then used by the **CLF Network Model** to obtain the final score for the texts authorship.

A summary of the process would be:

1. Load models
2. Extract known and unknown texts from file
3. Transform texts into word vectors
4. Use **Base Network Model** to create unknown and known representations
5. Use the combined array of representations with the **CLF Network Model** to obtain the final authorship score

2 Sprint 2

Sprint **Planning**: Sprint 2 - Planning

Sprint **Review**: Sprint 2 - Review

Sprint **Retrospective**: Sprint 2 - Retrospective

Team **Structure**: Team Roles

Sprint Tasks:

- Coding Guidelines
- Documentation
- Sprint 2 - Team Decisions
- Team Roles
- Git Notes/Guidelines
- Prototype Changes
- Security
- Motivational Modelling [UPDATED]
- Current Website

Coding Guidelines

Naming Conventions

- Choose simple, meaningful and descriptive names for variables, functions, classes and files.

Code Organization

- Organize and separate code into respective sections and modules depending on its functionality
- Build and follow a file structure and directory hierarchy

Code Reusability

- Write reusable functions and modules if necessary
- Avoid duplicating code

Code Readibility

- Keep lines of code short and neat
- Make sure that the code is not overly complex or have a nested code structure

Comments and Documentation

- Write brief and simple comments to explain what the code is about
- Provide comments for complex functions, classes and modules describing the its purpose

Version Control

- Use github as the version control system
- Commit regularly and provide clear and concise commit messages

Dependency management

- Manage external dependencies required carefully using a package manager
- list all dependencies and packages used for the app in text file
- Keep dependencies up to date and secure

Testing and Quality Assurance

- Ensure that any code changes produce desirable result during testing before pushing it to the main branch

Error Handling

- use try-catch blocks if neccesary for proper error handling
- provide meaningful error messages

Maintaining Coding Guidelines

Through the use of GitHub Actions and code linting we are able to keep our coding guidelines maintained and thoroughly checked. We use the standard NodeJS linter and python linter (pylint using PEP8 conventions) as the standards used. Both of these linters adhere to the guideline standards we are looking for and as such were our chosen linters.

Documentation

To set up in Visual Studio Code:

1. Install Node.js here: Download | Node.js
2. Open up a terminal in VSC, and type: `npm install -g @angular/cli`
3. When done, navigate to the folder containing the pages of the website
4. You can run the website by typing: `ng serve`
5. You can view the website by opening up a browser and going to `localhost:4200`

Note: this only works if your terminal is `cmd`, not `powershell`.

Python needs to be installed for the compare page to work:

1. Do `Ctrl + Shift + P` and choose Python: Create Environment
2. Choose `Venv`
3. Choose the Python interpreter that you prefer
4. Select `requirements.txt`, then click "OK"

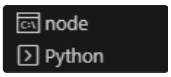
Note: if this is your first time running the website, you need to install `NLTK`. The steps are as follows:

1. Go to the `back-end` folder and select `functions.py`
2. Uncomment line 8, and save
3. You can discard the changes made to `functions.py` so it does not get pushed along with your commit

To run the algorithm along with the website:

1. Select `app.py` in the `back-end` folder
2. Click the “run” button located on the top-right of your screen (in VSC)
3. If this is your first time running it should install `NLTK`, then run the server

Note: you should have both of these under your terminal to use the website and algorithm together.



Sprint 2 - Team Decisions

Monday 28th August

- Decided on work delegation
- Decided on dates for meetings / task deadlines
- Created logo samples in preparation for logo decisions

Thursday 31st August

- Decided on backend Python packages
- Decided to use `input` for frontend
  - plan use of buttons linked to text area
- Flask and backend extensions
- User stories rehaul to fit version 1.0

Thursday 7th September

- Fixed utilizing `textarea` instead of `input` for a better visualization of inputting text into the website

Monday 11th September

- minor styling decisions for the front end of the website
- decided on the first iteration of upload (.txt focus first, other file types will come after everything is linked)

Wednesday 13th September

- upload files to include .docx and .pdf
- major switch on frontend side to shift to flex-box to ensure scaling of all components
- major restructuring of git files for readability and easy understanding

Monday 18th September

- continuous upload of of known files and ability to delete specific files from the already uploaded ones

Tuesday 19th September

- functionality and visual changes made under group decision
- final completed stamp on back end!

Thursday 21st September

- decided to add more security on API
- final wrap up decisions on sprint 2, largely just confirmation checks
  - confluence page alterations and documentation that we felt was needed

Team Roles

Development Team

Scrum Master - Lin

- Planning, organizing and managing of confluence, progress and meetings
- Cooperating with the front end team on the website design
- Cooperating with the back end team on making sure all intended features of the version are added correctly and efficiently

Frontend Developer - Jia Jie, Adrian

- Coding, executing and adjusting of all matters related to front-end
- Cooperating with the back end team on making sure the correct data and their corresponding data types are pipelined into the front-end

Backend Developer - Saaiq, Hanizam

- Coding, executing and adjusting of all matters related to back-end
- Cooperating with the front end team in making sure the connection between the front-end and back-end is seamless with errors from client and server side being handled correctly

Git Notes/Guidelines

- how we merge, what guidelines we have, wh en we have merge conflicts what do we do

Branching:

When branching, follow the general guidelines below:

Branch name:

> [name]/[task/item]

An example of this would be:

> Lin/website header

Commits

Commits within individual branches align to the following guidelines:

> [main work]: [short but descriptive line about commit]

Current main work tasks:

- Feat
  - Feature currently added
  - Code largely creates and implements feature being stated
- Fix
  - a bug fix or code fix
  - done after the commit of a feature

Merging Guidelines

Current:

- When merging to main, send a message to Slack to confirm with other team members, or
- Confirming with team members whilst in a meeting on Zoom

New:

- Implement branching
- Adhere to commit guidelines
- Continue to notify team members through Slack

Handling Merge Conflicts

- Contact specific team member/s to resolve merge conflict
- Organise a small meeting to handle merge conflict
- In the case of accidental overwrites, revert back to previous version/s, and notify team members

Prototype Changes

Changes made since last iteration of prototype:

- Animation of text input area has now been *removed*
  - Switched to two text input areas side by side for ease of use and switch between known and unknown text
- inclusion of an add text input area to file button in the known text area
  - to allow for more texts to be included via the text input area
- Removal of a loading screen while texts process
  - due to changes made to backend, text analysis was significantly sped up, reducing the need for a log in screen

Changes to be added

Analysis Low fidelity Prototype

- a breakdown of all available information provided by the model on the texts being analyzed

Security

API Security

Input Security

1. Using a `compare_model` which the API expects to make sure the correct data is sent and used by the function
    a. In this case the `compare_model` expects **form data** with the arguments `known_texts` , `unknown_text` , `known_files` , `unknown_file`
    b. Makes sure to include the specific data types of each argument, its requirement and a description for the docs for potential users to understand
2. Error checking the inputs sent with a `400: "No known texts provided"` and `401: "No unknown text provided"` post error that is sent through, which aborts the function as intended

Function Security

1. Error checking the function with a `500: "Error calculating score"` to make sure there is improper score being outputted
2. Removing duplicate and white space texts as well as ignoring files that couldn't be parsed to insure there is no API crashing

Output Security

1. Using a `score_model` which the API marshals to make sure only the required API data is sent through
    a. Is in the form of a JSON response with a `score` header as well as headers for other text statistics
    b. Includes the exact data typing's, requirements and description of each header in the model

Login Security

In the event that a sign up and login system is implemented

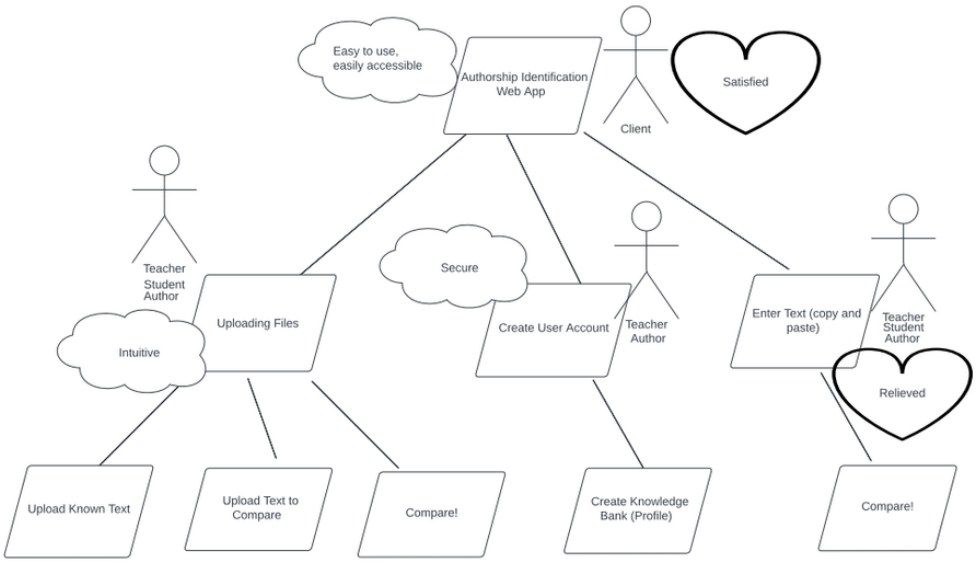
0Auth

- The use of the 0Auth service wouldn't just allow for a simplified implementation of an advanced login system but also would provide security in the handling of account data
- Would provide secure storage and accessing of sensitive password information

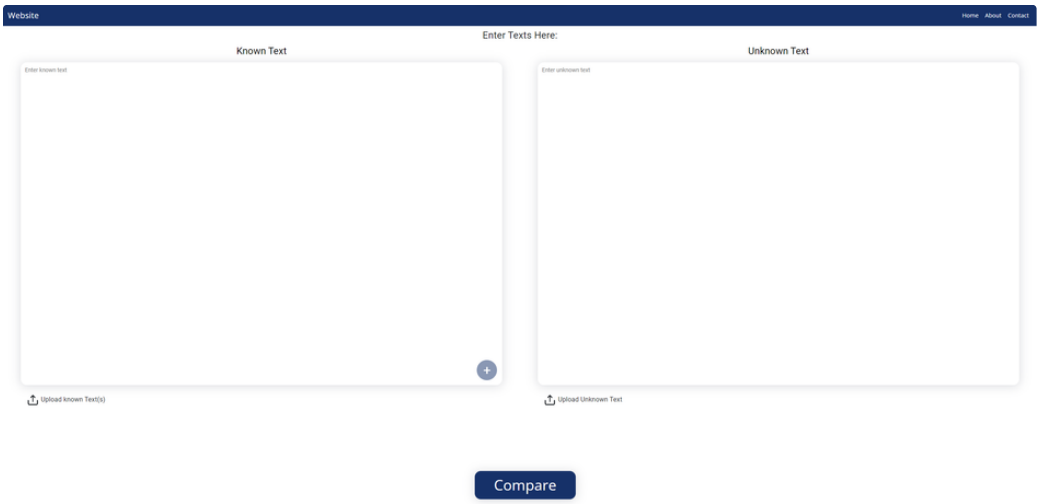
Profiles

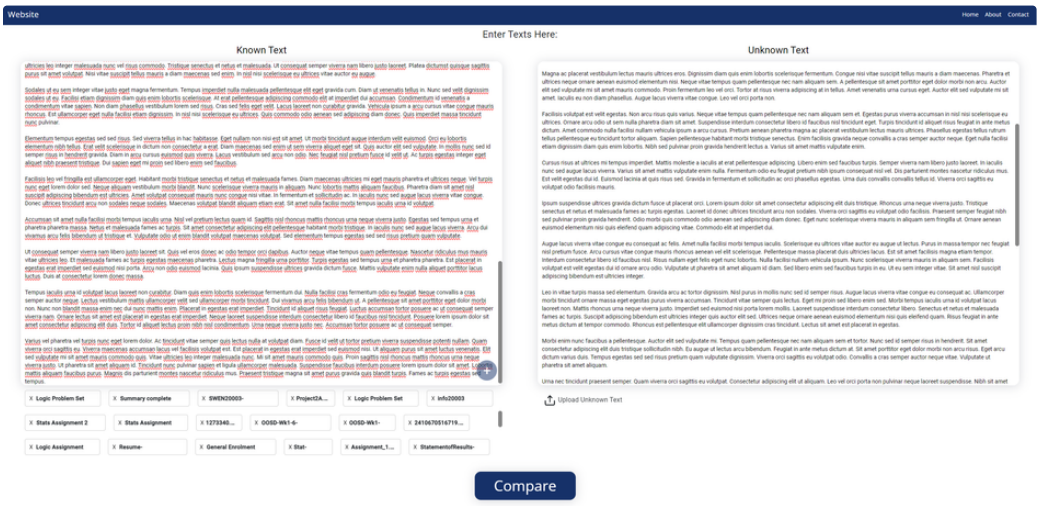
- Would need to implement authentication (potentially alongside the 0Auth authentication) with the API to access profile data
    - Help keep profile data private so only the correct user has access to view profiles

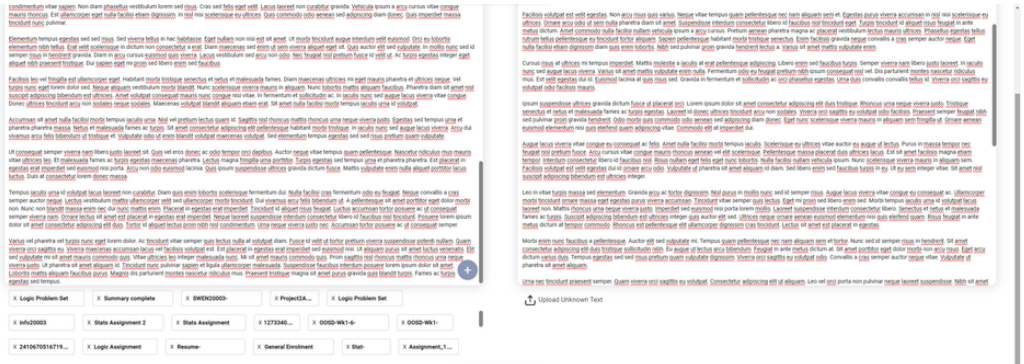
Motivational Modelling [UPDATED]



Current Website







Compare

Results

Similarity

100%

Explanation

Results

Similarity

100%

Explanation

Word Similarity

85.23

Punctuation Percentage

25.43 16.9

Average Sentence Length

14.3 82.05

Sentences Over Average Length

2.62 1.17

Sentences Under Average Length

4.38 1.22

Rare Word Percentage

16.38 38.07

Long Word Percentage

31.91 27.13

Words Over Average Length

49.76 39.75

Words Under Average Length

51.07 42.59

Type-Token Ratio

22.98 53.07

Word Count

10435 717.5

3 Sprint 3

- Updated User Stories
- Testing
- Deployment Process & CI/CD
- Website Updates
- Changes Made Since
- Model Performance Testing
- Tutorial Images

Updated User Stories

AIM: To compare a set of texts that we know come from an individual and compare that with another given text to see the likelihood of the second text being written by the same individual that wrote the first set

---

EPIC: Compare the authorship of texts in a simple, elegant and intuitive manner.

As a **Teacher** User**:**

- As a teacher, I want to be able to compare the works of my students to their own past work that I know of, and/or to potentially compare to another students work, and see the authorship similarity between them. This would allow me to better understand my student's current work, and better assist me in ensuring the authenticity of their work and make decisions related to it.
- I want to be able to get a more accurate reading of the authorship similarity by adding more texts that I know belong to the student based off past submitted work.
- I want to be able to upload documents of different file types for comparison as student submissions may vary in type.

Testing

- Acceptance Criteria
- Acceptance Test
- Test Cases
- Structural Testing
- Bug Testing
- Non-Functional Testing

Acceptance Criteria

Created based on updated user story.

ID	Feature	**Acceptance Criteria** *(Given/When/Then)*
1	Input Interface	Given the user is looking for a website to compare two different texts, when the user accesses the website, they then get greeted with a *user-friendly interface* where they can input a set of known texts and an unknown text for comparison
2	Security	Given the user is uploading documents that may be sensitive or personal, when the user uploads the documents to our website, our website processes the information then returns the result through the already pre-trained model without storing any of the documents
3	Multi-file type input	Given the user has documents of multiple types to upload, when the user uploads documents, they have the flexibility to upload .txt, .docx and .pdf, alongside textbox input. Then the user can start the comparison model
4	Results Explanation	Given that the user has uploaded and clicked compare, when the model has finished processing the texts, the results are then presented as a large visual circle with colour indicating similar percentage. The results is given alongside a set of data on the similarity information such as word similarity, punctuation percentage and more.
5	Instructions	Given that the user is not familiar with the website or has forgotten how to use it, when the user clicks at the bottom right of the website (the ? mark button), the instructions then pop up to help assist the user

Adequacy criterion = set of test obligations. E.g., cover all statements and all branches in the program under test

A test suite satisfies an adequacy criterion if

✔ all the tests succeed (pass)

✔ each test obligation is satisfied by >=1 test case(s)

Acceptance Criteria: set of predefined requirements that must be met in order to mark a user story complete.

- given [condition], when [something happens], then [result]

Acceptance Test

AC ID	AT ID	Acceptance Test	TC	Success?
1	1.1	User is able to see the home/welcome screen upon accessing website	**TC-5**	✓
	1.2	User is able to see the main features of website after successful load out of website	**TC-5**	✓
	1.3	User is able see all the navigation and buttons clearly	**TC-5**	✓
3	3.1	User is able to upload .pdf, .docx, .txt, or a combination of them	**TC-2**	✓
	3.2	User is able to use the text input box multiple times through converting them into files with the '+' button	**TC-3**	✓
4	4.1	User is able to successfully upload files/utilise the text boxes	**TC-2, TC-3**	✓
	4.2	User is able to successfully press the 'Compare' button	**TC-1**	✓
	4.3	User is able to see the animation and display of authorship comparison results.	**TC-1**	✓
	4.4	User is able to see the visual explanation bars on the details extracted from the comparison	**TC-1**	✓
5	5.1	User is able to click the '?' button at the bottom right of the website	**TC-4**	✓

AC - Acceptance Criteria, AT - Acceptance Test, TC - Test Case

Test Cases

Test Case [TC-1]

Test Type: Functional **Execution Type:** Manual

Objective: Test if results screen shows proper results after pressing compare button given correct input

Setup: Must have at least one known text and one unknown text

Pre-Conditions: Dynamic results bar shows resulting similarity percentage with colour depending on percentage

Notes:

1. Add known text(s) and unknown text through textbox or uploading files
2. Press compare button
   - results section should appear
   - results bar should have fill animation with colour based on percentage
   - explanation section should show evaluation metrics with numbers and similarity bars

Time constraint:

Minimum: 5 min

Maximum: 10 min

Test Case [TC-2]

Test Type: Functional **Execution Type:** Manual

Objective: Uploading files on compare page

Setup: Be on compare page with files ready to upload

Pre-Conditions: Any files uploaded to either the known or unknown texts sections should show up as file cards under the text boxes while can be removed by clicking the 'x' button

Notes: Types of files accepted: PDF, docx and txt

Time constraint:

Minimum: 5 min

Maximum: 10 min

Test Case [TC-3]

Test Type: Functional **Execution Type:** Manual

Objective: Utilize the '+' button in the text area to save text box text into a file - multiple inputs

Setup: Be on compare page with multiple texts ready to upload

Pre-Conditions: multiple texts of medium length separated for multiple input attempts

Notes: N/A

Time constraint:

Minimum: 5 min

Maximum: 10 min

Test Case [TC-4]

Test Type: Functional **Execution Type:** Manual

Objective: Seeing if tutorial pop-up button works

Setup: Be on compare page

Pre-Conditions: After clicking the '?' button in the bottom right of the page, a pop-up should appear with images containing instructions on how to use the website

Notes: Images should be easily readable and instructions should be clear for a new user

Time constraint:

Minimum: 5 min

Maximum: 10 min

Test Case [TC-5]

Test Type: Functional **Execution Type:** Manual

Objective: Fully functional load out of deployed product

Setup: Access to link - AuthoWrite

Pre-Conditions: Website is fully deployed, no crashes occurred and latests changes have been updated and passed through test cases

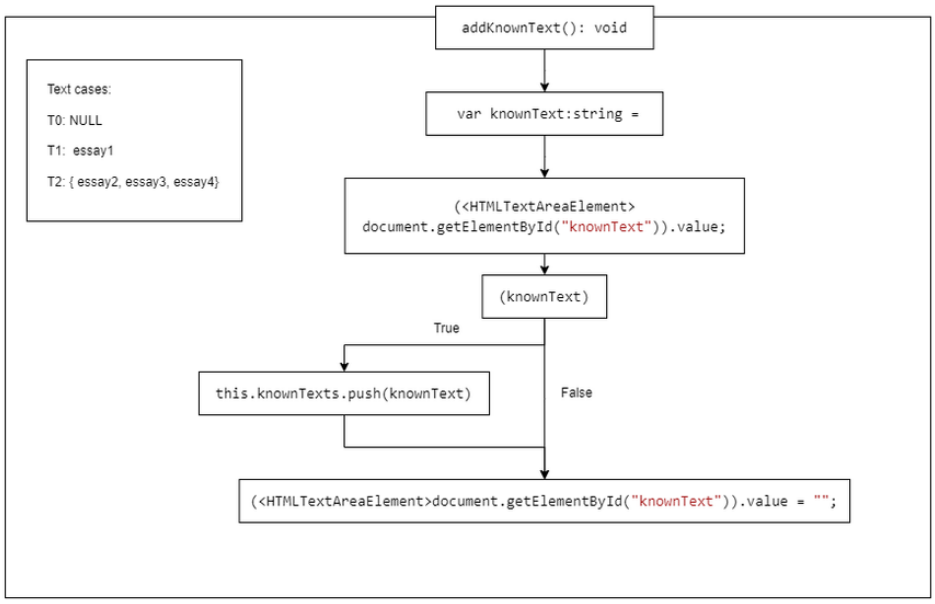
Notes: Check through that all things are loaded, buttons are all present

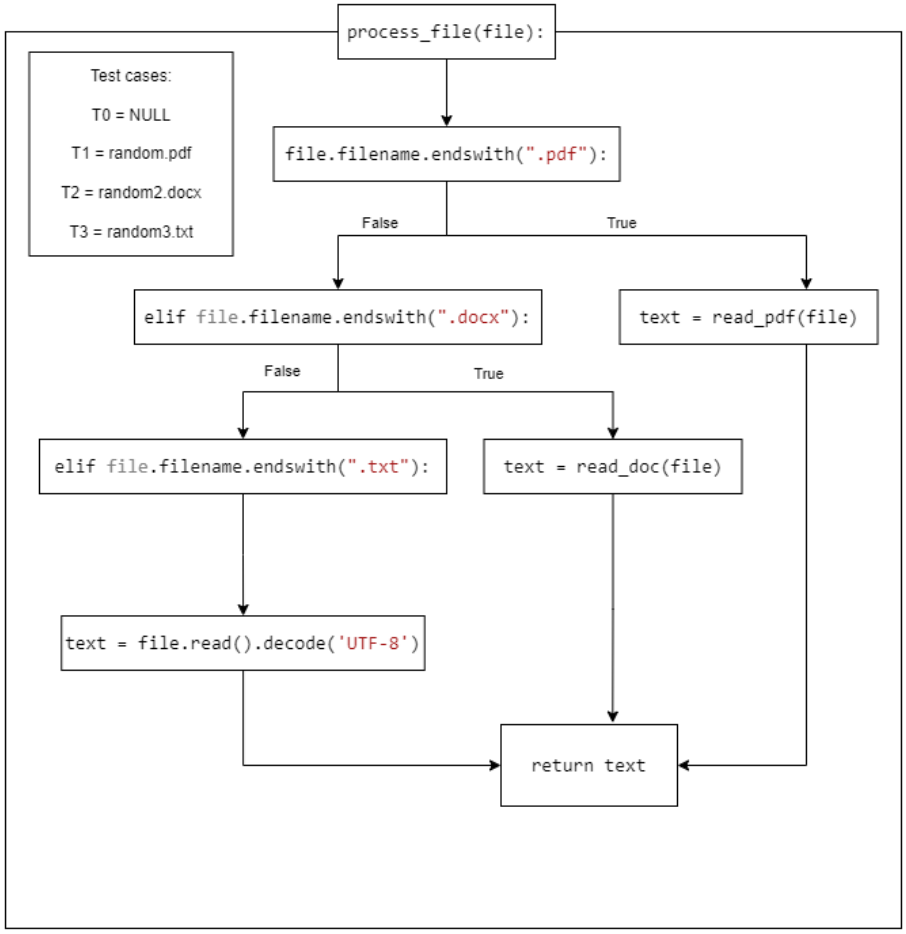
Time constraint:

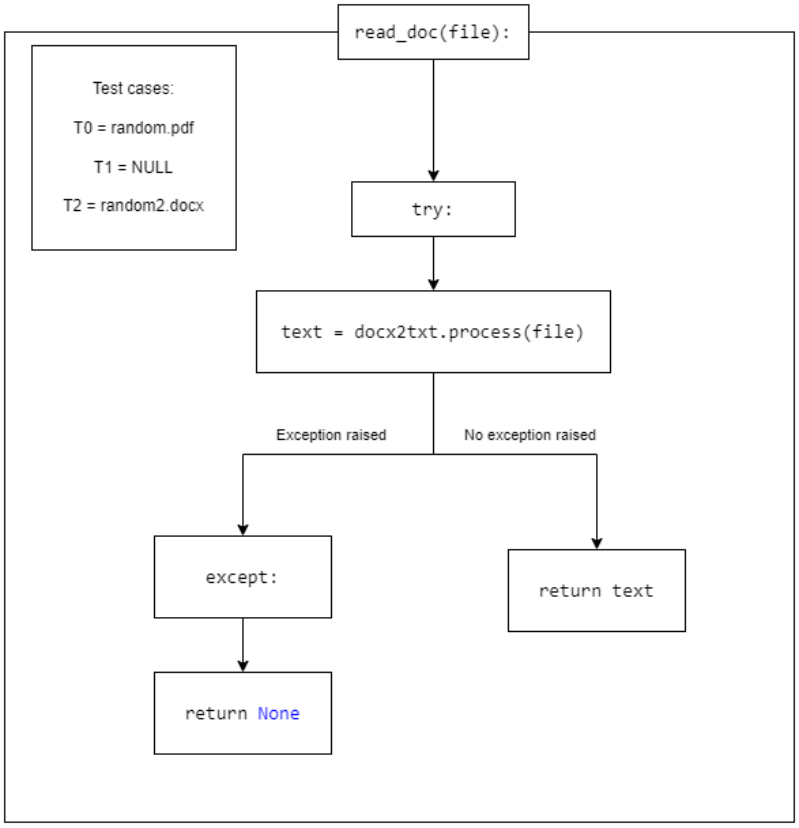
Minimum: 2 min

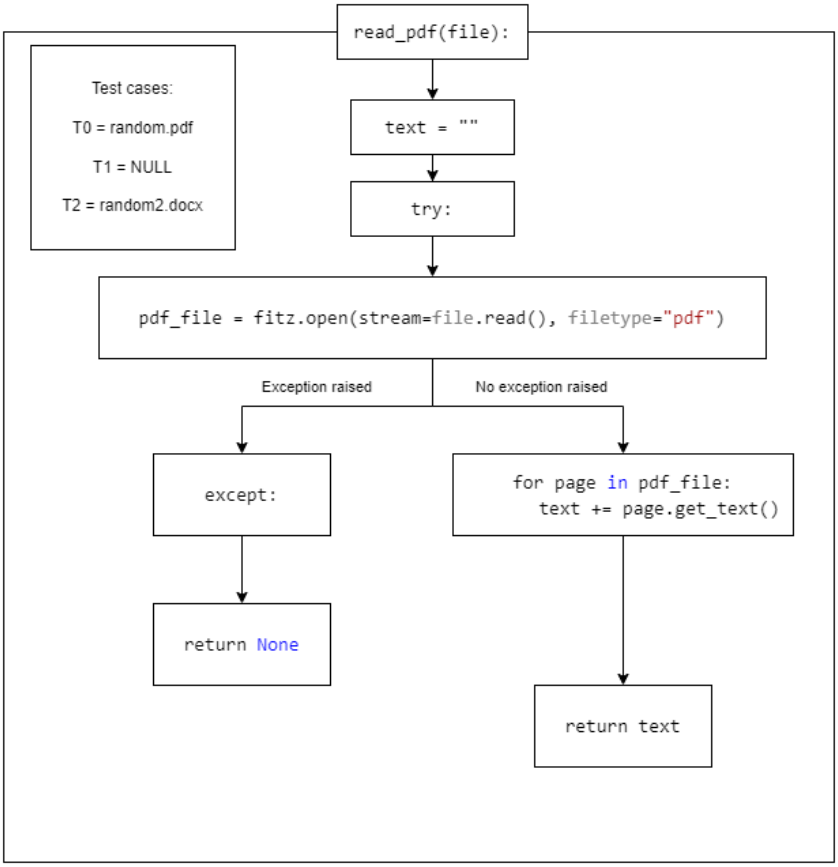
Maximum: 10 min

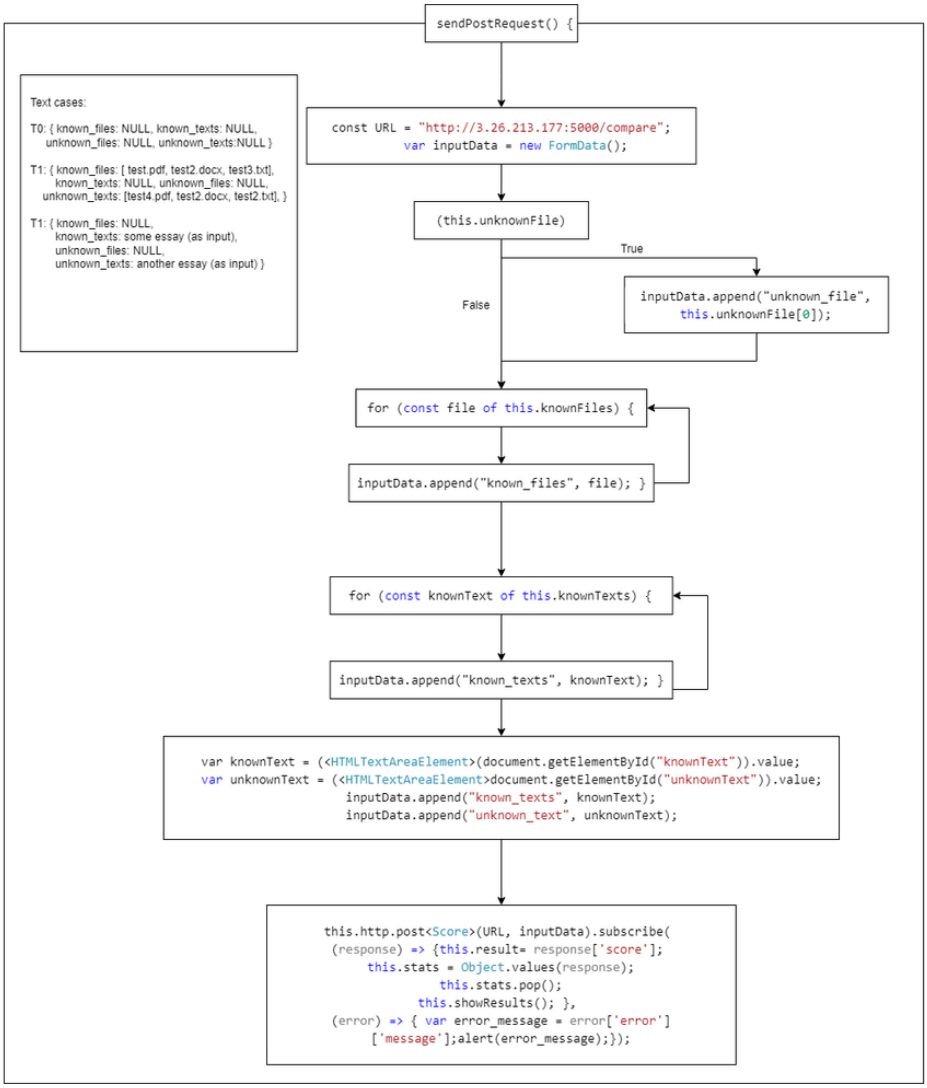
Structural Testing

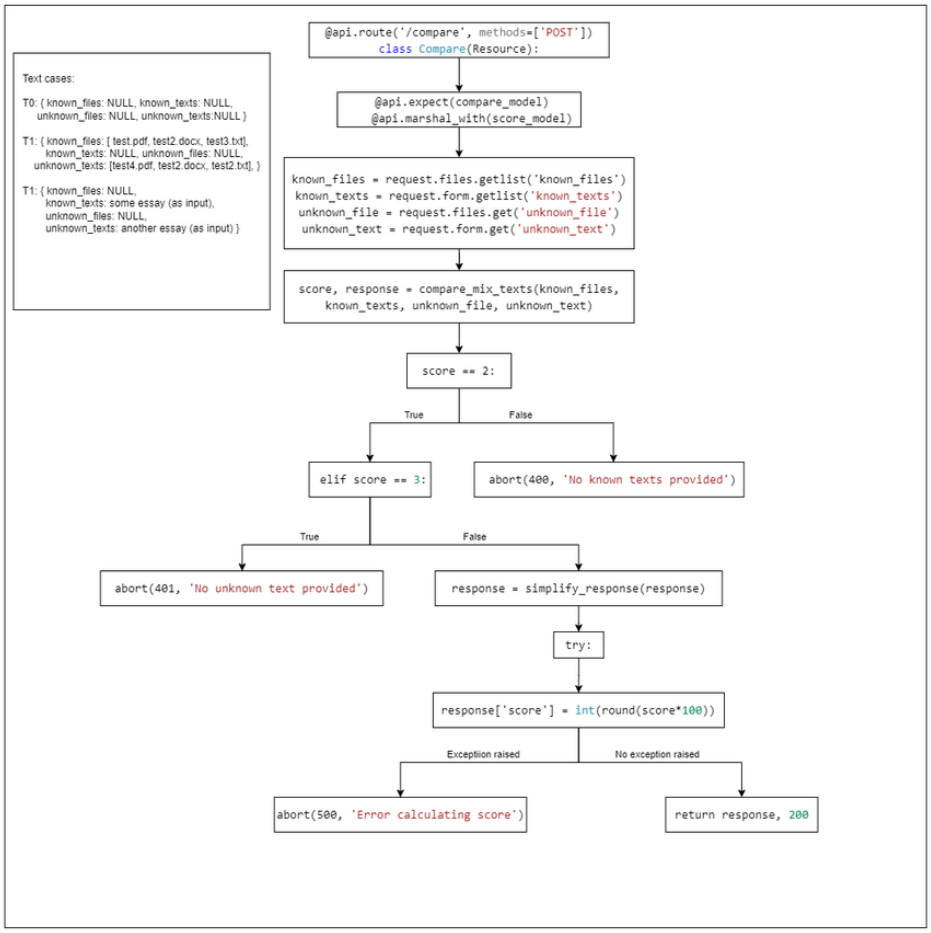



process_file(file):
Test cases:
T0 = NULL
T1 = random.pdf
T2 = random2.docx
T3 = random3.txt
file.filename.endswith(".pdf"):
False
True
elif file.filename.endswith(".docx"):
text = read_pdf(file)
False
True
elif file.filename.endswith(".txt"):
text = read_doc(file)
text = file.read().decode('UTF-8')
return text


read_doc(file):
Test cases:
T0 = random.pdf
T1 = NULL
T2 = random2.docx
try:
text = docx2txt.process(file)
Exception raised
No exception raised
except:
return text
return None


read_pdf(file):
Test cases:
T0 = random.pdf
T1 = NULL
T2 = random2.docx
text = ""
try:
pdf_file = fitz.open(stream=file.read(), filetype="pdf")
Exception raised
No exception raised
except:
for page in pdf_file:
text += page.get_text()
return None
return text


sendPostRequest() {
Text cases:
T0: { known_files: NULL, known_texts: NULL, unknown_files: NULL, unknown_texts:NULL }
T1: { known_files: [test.pdf, test2.docx, test3.txt], known_texts: NULL, unknown_files: NULL, unknown_texts: [test4.pdf, test2.docx, test2.txt], }
T1: { known_files: NULL, known_texts: some essay (as input), unknown_files: NULL, unknown_texts: another essay (as input) }
const URL = "http://3.26.213.177:5000/compare";
var inputData = new FormData();
(this.unknownFile)
True
False
inputData.append("unknown_file", this.unknownFile[0]);
for (const file of this.knownFiles) {
inputData.append("known_files", file); }
for (const knownText of this.knownTexts) {
inputData.append("known_texts", knownText); }
var knownText = (<HTMLTextAreaElement>(document.getElementById("knownText")).value;
var unknownText = (<HTMLTextAreaElement>document.getElementById("unknownText")).value;
inputData.append("known_texts", knownText);
inputData.append("unknown_text", unknownText);
this.http.post<Score>(URL, inputData).subscribe(
(response) => {this.result= response['score'];
this.stats = Object.values(response);
this.stats.pop();
this.showResults(); },
(error) => { var error_message = error['error']
['message'];alert(error_message);});


Text cases:
T0: { known_files: NULL, known_texts: NULL,
unknown_files: NULL, unknown_texts:NULL }
T1: { known_files: [test.pdf, test2.docx, test3.txt],
known_texts: NULL, unknown_files: NULL,
unknown_texts: [test4.pdf, test2.docx, test2.txt], }
T1: { known_files: NULL,
known_texts: some essay (as input),
unknown_files: NULL,
unknown_texts: another essay (as input) }
@api.route('/compare', methods=['POST'])
class Compare(Resource):
@api.expect(compare_model)
@api.marshal_with(score_model)
known_files = request.files.getlist('known_files')
known_texts = request.form.getlist('known_texts')
unknown_file = request.files.get('unknown_file')
unknown_text = request.form.get('unknown_text')
score, response = compare_mix_texts(known_files,
known_texts, unknown_file, unknown_text)
score == 2:
True
False
elif score == 3:
abort(400, 'No known texts provided')
True
False
abort(401, 'No unknown text provided')
response = simplify_response(response)
try:
response['score'] = int(round(score*100))
Exceptiion raised
No exception raised
abort(500, 'Error calculating score')
return response, 200

Bug Testing

Functional testing:

- best for missing logic faults
- A common problem: Some program logic was simply forgotten
- applies at all levels
  - unit - form module interface spec
  - integration - from API or subsystem spec
  - system - from system requirements spec

- tested corrupted files
- tested scroll bars - still need to change
  - include edge
- tested files which do not contain any text (eg. only images)

Mobile Testing

- Found an initial issue of a white background when zooming out of the page
  - Solution: needs to be changed to blue
- New issue after deployment of a failure in the responsive design
  - Solution: fix issues with flex and flex-grid

Non-Functional Testing

- **Visibility**
  - When we hit “compare”, the page automatically scrolls down to the results section. We implemented this feature because it was not intuitive as to where the results were.
  - We added a tutorial page that shows clearly how to use the website.
  - We have file icons whenever a user adds a file, and users can also press the “X” to delete them. We implemented this feature because using only alert boxes to notify the addition of a file was not very clear, and also this allows users to remove files easily.
- **Match**
  - Also for adding files, we use the universal “+” symbol to show users that they can add files. Furthermore, “X” is easily recognisable as removing files.
  - The tutorial symbol is “?” which is also a commonly used symbol for help.
  - We have basic instructions as placeholders in the textboxes to guide users.
- **Control**
  - The adding and removing of files allows users to freely add and remove files, so the users do not need to refresh the page to clear unwanted files.
- **Consistency**
  - We clearly label our textboxes with “known” and “unknown” to indicate to our users what exactly they should put into each textbox and we also show results consistently in this way.
- **Error Prevention**
  - If there are no files added or no text inputted, there will be an error message notifying users of this.
- **Recognition**
  - Most symbols such as “+”, “X”, “?” are easily recognisable, the compare button is a button with the “compare” text on it.
- **Flexibility**
  - Flexibility between adding text input or adding files. Can also mix and match between text / files. Furthermore, .txt, .docx and .pdf file types are all supported.
- **Minimalism**
  - Not much text or elements on screen before users click the “compare” button, only relevant information is needed. Tutorial is only shown if user decides to click “?”.
- **Recover**
  - Our website does not have many errors, the only one being the error message for no file/text input, written in plain English.
- **Help and Documentation**
  - The error message for no file/text input is written in plain language.

UX: how people feel about a product and their impression and how it feels to them and how they feel afterwards

Heuristic Evaluation

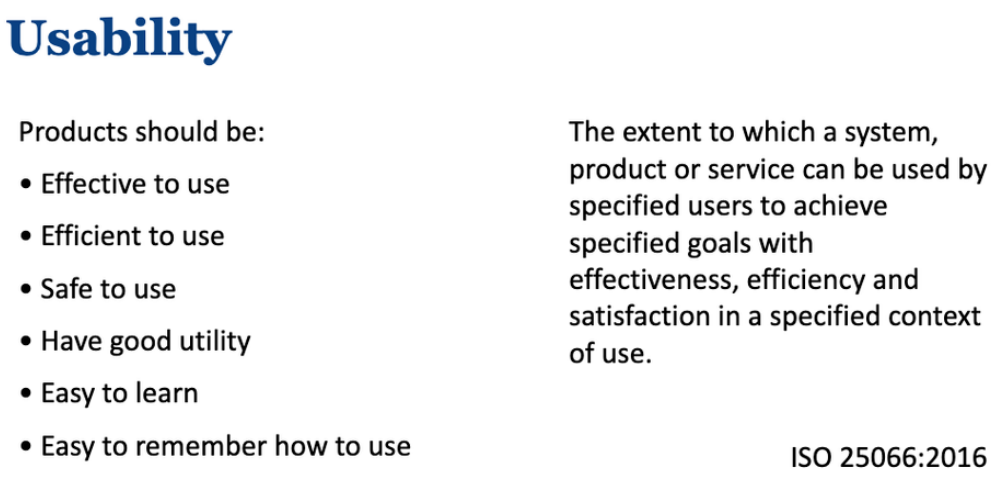
- **Visibility** - should always keep users informed about what is going on (ie appropriate feedback)
  - we should probably add a popup if their analysis runs over a certain amount of time
- **Match** - between system and real world (speak users' language, simple common language, follow real world conventions)
- **Control** - user control and freedom, support undo and redo

- Users often choose system functions by mistake and will need a clearly marked “emergency exit” to leave the unwanted state without having to go through an extended dialogue.

- **Consistency**
  - Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.
- Error Prevention
  - Even better than good error messages is a careful design which prevents a problem from occurring in the first place.
- Recognition - Make objects, actions, and options visible.
  - (!!) add an ℹ️ button to pull up instructions again when users need it?
  - “Instructions for use of the system should be visible or easily retrievable whenever appropriate.”
- Flexibility - Accelerators - unseen by the novice user - may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users.
- Minimalism - Dialogues should not contain information which is irrelevant or rarely needed
- Recover - Help Users Recognise, Diagnose, and Recover from Errors
  - error messages should include plain language expression, precisely indicate the problem, construct/suggest a solution
- Help and Documentation
  - Troubleshooting: Make sure your system indicates errors and messages in plain language

In line with user stories, ask questions:

- Will the user know what to do?
- Will the user see how to do it?
- Will the user understand from the feedback whether the action was correct or not?



UX: how people feel about a product and their impression and how it feels to them and how they feel afterwards

Deployment Process & CI/CD

The entire deployment of both the front end and back end was done on AWS (Amazon Web Service).

To ensure a successful deployment, the repository needed to be split into 2 repositories for the front and back end separately as they are built under different architectures and have different requirements for running.

To also ensure a successful CI/CD of maintaining both our code structure and to unit test our code, we implemented a linting and test case check for both repositories using **GitHub Actions** to validate our repositories before pushes or merges to the main branch.

Front End Deployment

The front end deployment was done through using **CodePipeline** to take the github repository, download the repository and then build the Angular app using **CodeBuild** which is then sent through to the **S3 Bucket** that is publicly shown.

This ensures a successful CI/CD as any changes to the main branch is automatically detected by CodePipeline and automatically pulls the new version and rebuilds the program and updates all the files in the S3 bucket. Any issues caused with an unsuccessful pull or build will just not update the current website and still have the old working version running so no issues are noticed for the user.

For the code structure checking, we are using the standard NodeJS linter that is included with NodeJS and is easily implementable with GitHub Actions.

For the code testing, thanks to our use of Angular, we are able to use **Angular**'s already supplied unit tester which can be done locally using the command `ng test`. We can simply include testing with our GitHub Actions pipeline by creating a testing build and running the `ng test` command to unit test and check the functionality of all components in the project.

Back End Deployment

Since the back-end deployment is a **Flask** web app that utilises the python package **tensorflow**, The usual method of *AWS ElasticBeanstalk* cannot be used for free as there isn't sufficient memory requirements for the app to be functional. As such the alternative method was to run the application on an **EC2 instance**. This allowed for more control at the expense of ease of use.

The process then was to manually set up the application at first by connecting to the EC2 instance and installing both git and python to the server and cloning the private github repository using the SSH cloning method. Afterwards the commands to setup running the server were:

```
git pull  # pull latest github repository

pip install -r requirements.txt  # installing dependencies

python -c 'import nltk; nltk. download(["punkt", "stopwords","wordnet"])'  # install nltk packages (run once)

export FLASK_APP=application.py  # export flask web app

nohup flask run --host=0.0.0.0
```

Which allowed for continuous running of the server.

To make sure the application would always automatically update with our latest repository, another timing script was created to check for any github repository changes every 10min and in the event of a change, for the commands to be rerun.

For the code structure we used pythons standard linter `pylint` to keep check of our coding.

For the process of unit testing, the built in `unittest` package was used as it integrates seamlessly with Flask allowing for local checking without needing to completely run the web app. This allowed for us to do a large number of tests on every potential file type making sure the correct response was outputted. This unittest process was then implemented in the GitHub Actions pipeline so that all changes or merges to the repository are checked by the linter and tester.

Website Updates

Website link:

AuthoWrite

New Github Repositories:

Back-end: https://github.com/Re-Roll/AuthoWrite-back-end Connect your Github account

Front-end: https://github.com/Re-Roll/AuthoWrite-front-end Connect your Github account

Changes Made Since

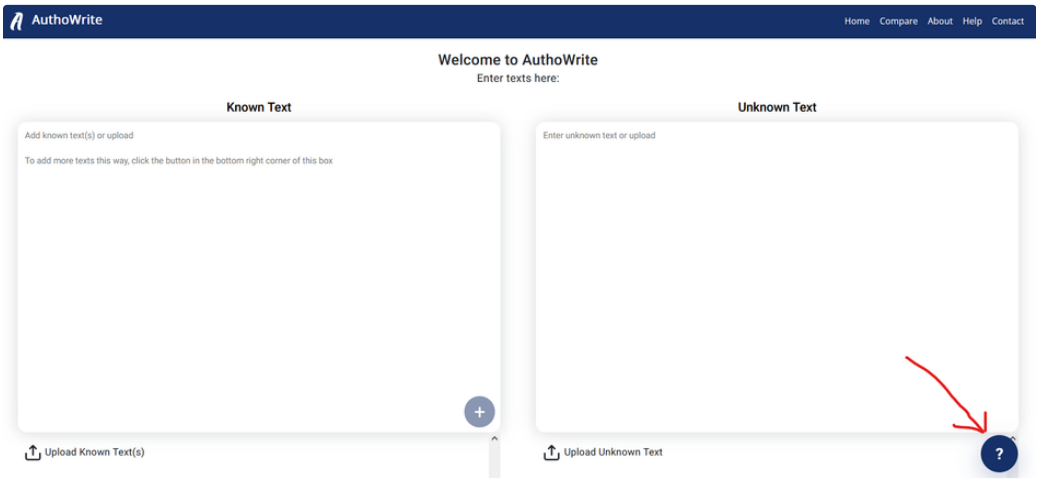
As we're nearing the end of the last sprint, here are some of the changes that have been made

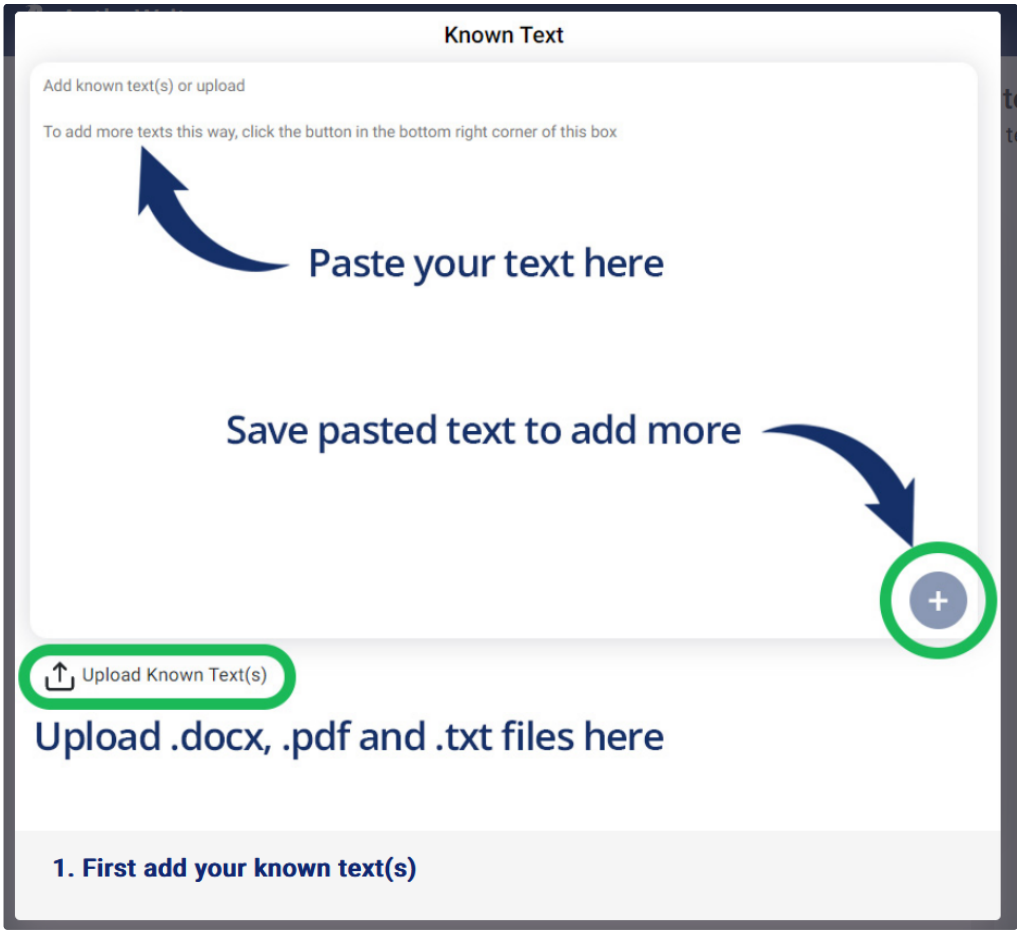
- **Features we have moved to future extension possibility/potential**
  - These features include the signup/login feature to create profiles. This is to maintain our focus on the functionality and improve upon the model that was given to us in order to meet the clients requirements
- **What has changed since sprint 2**
  - Additions:
    - Restyled website homepage with interactable background

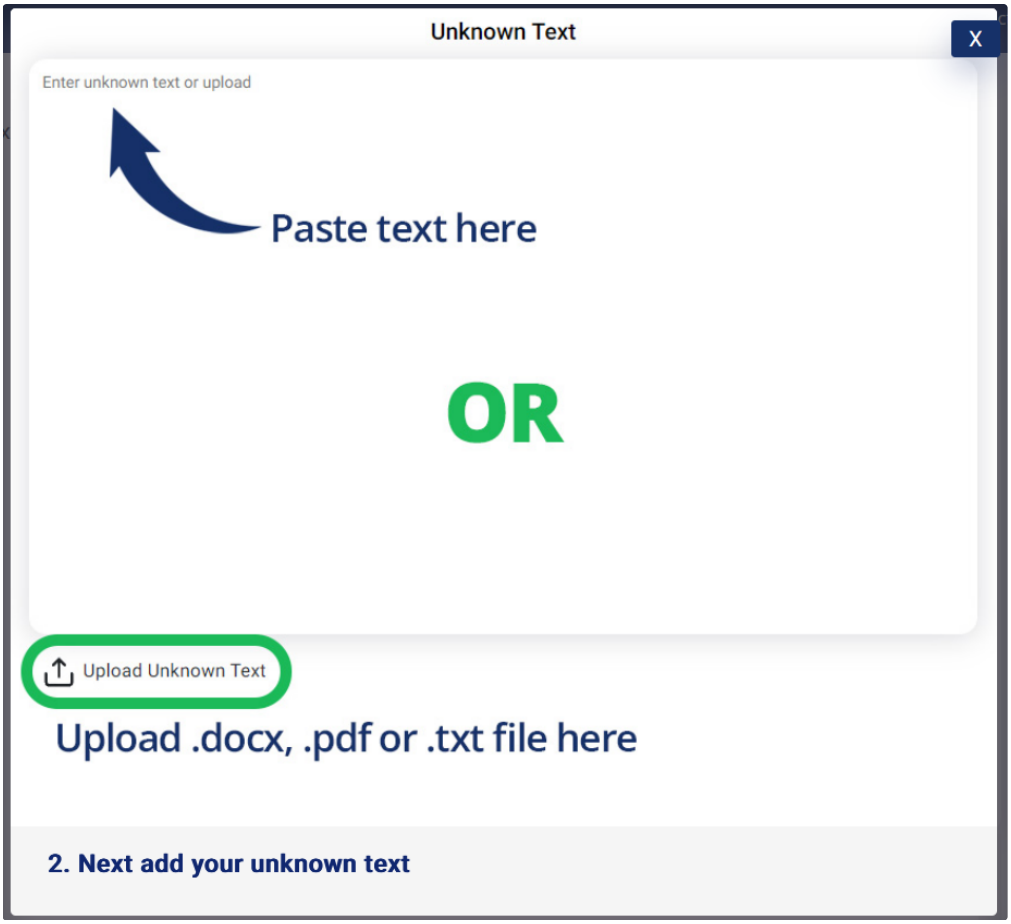
- Created new git repositories for deployment (separated into front-end and back-end)
- Website deployment has now been complete and deployed with AWS
- A tutorial button has now been added which leads to a popup with visual instructions for aid
- Testing changes are now being implemented for the website

Model Performance Testing

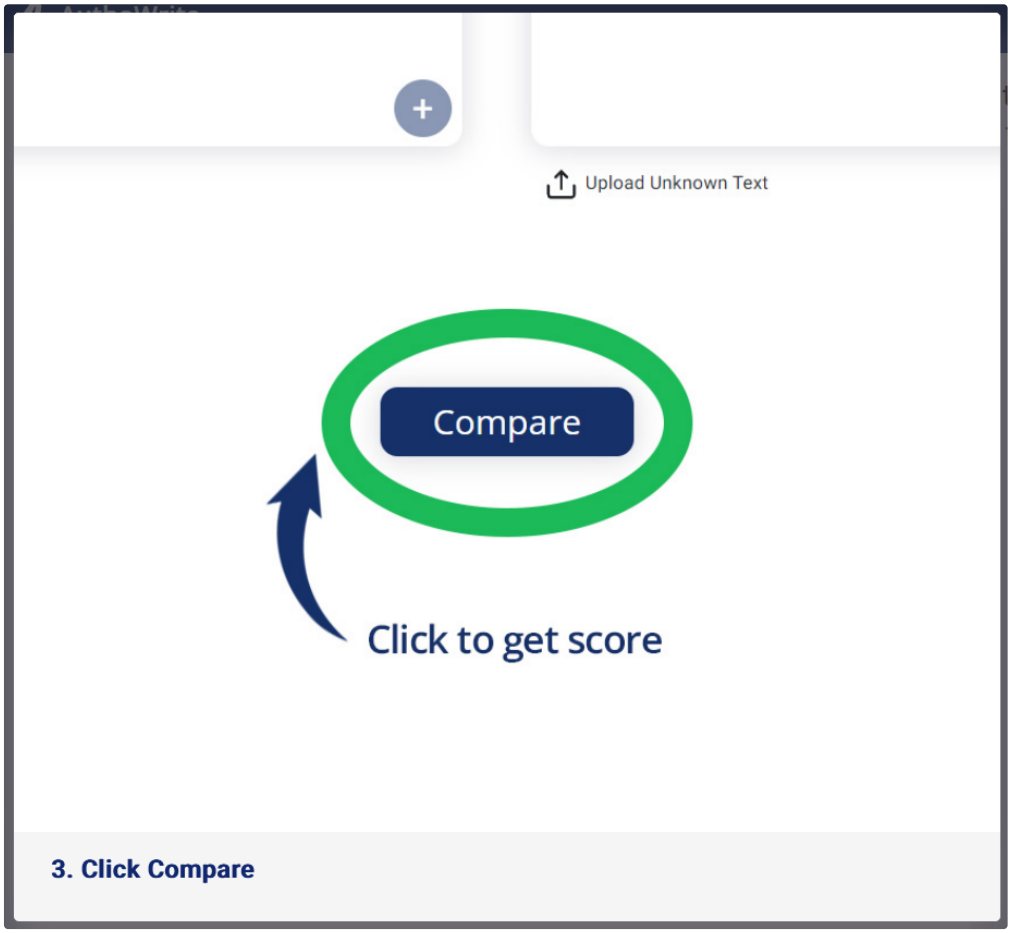
Tutorial Images



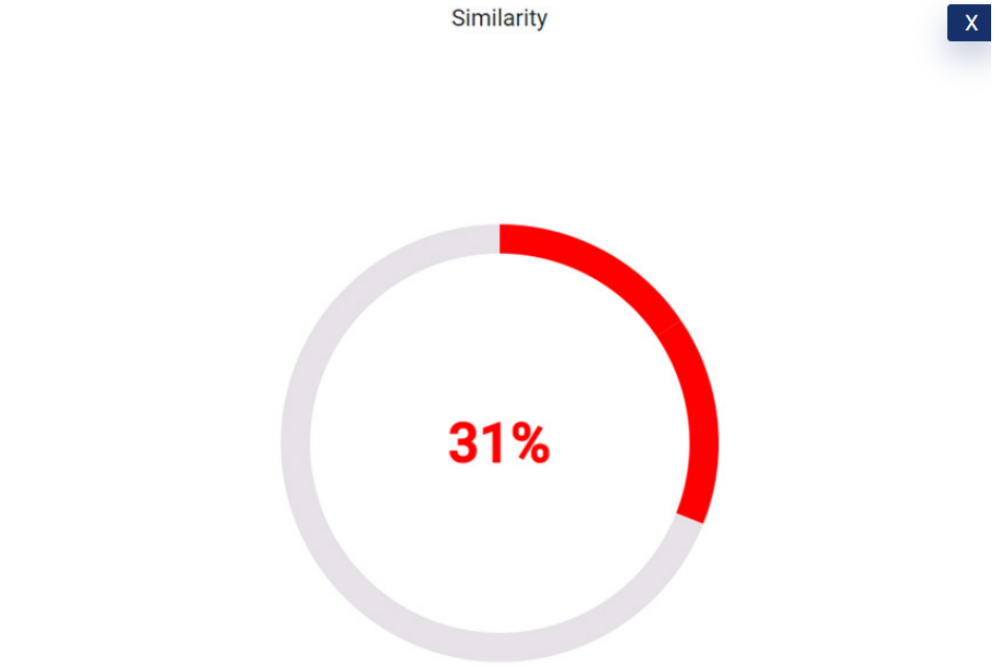




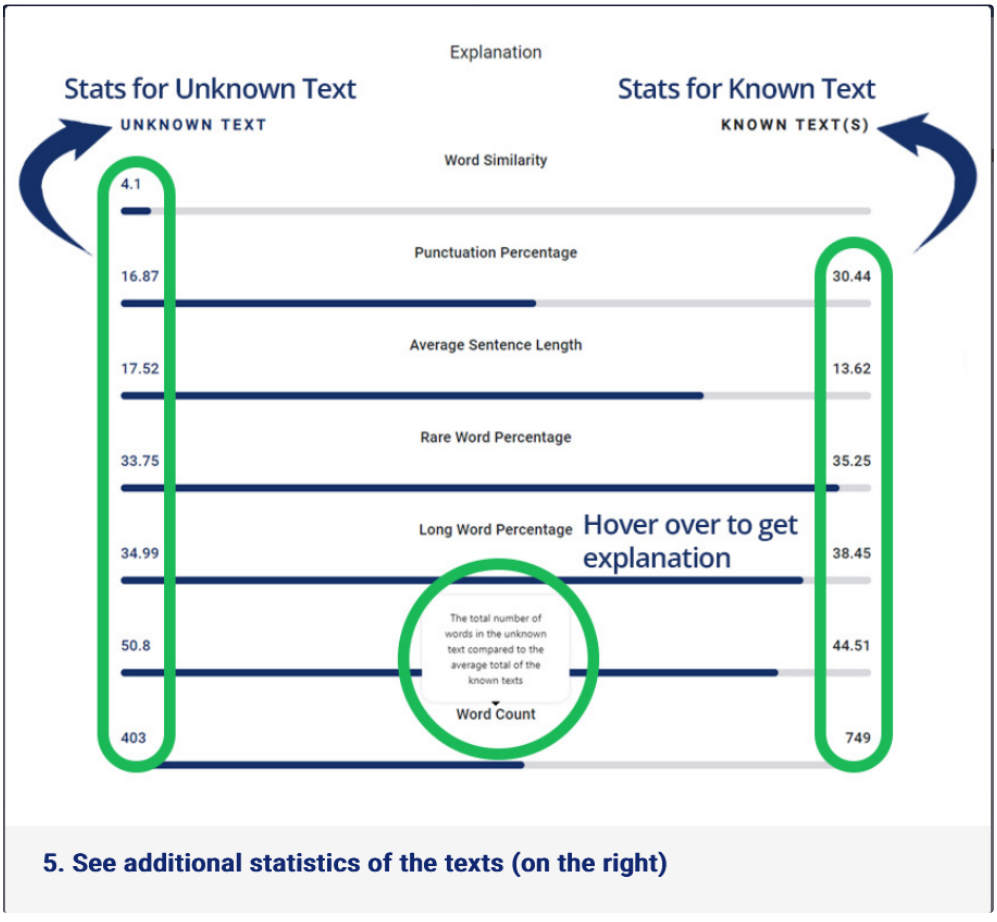
2. Next add your unknown text



3. Click Compare



4. See your result (on the left)



5. See additional statistics of the texts (on the right)

Minutes

- Sprint 1 Meetings
- Sprint 2 Meetings
- Sprint 3 Meetings

Sprint 1 Meetings

Friday 11 Aug

All participants present

Discussion

- Establishment of the beginning of the project
  - hashing out of broad tasks and subtasks as a goal
- Decide upon what platforms we will be using as a group, different options are considered and run through given the recommendations from workshop and lectures
  - (!!) vital: ensure that all decided upon platforms are all up and running
- Confluence set up of pages of hurdles and tasks to do
- Jira set up tasks and and subtask
- Established first meeting time based on availabilities using when2meet
- General housekeeping and role delegation was decided upon this meeting alongside communication of roles (Scrum master - Lin)
  - meetings even if someone isn't available but keep notes and maintain updates throughout for missing members, and receive updates through Jira
- Rough brainstorming of items that maybe included into final product - created subpages
- Mentioned tools to look into:
  - Angular
  - Electron

Summary

Goals for the coming week

- Ensure everything is set up and running
- Have a scrum meeting
- Sprint planning
- Update/keep track of decisions and Jira

Decisions

Platforms:

- Planning of tasks - Jira
  - also has timeline mode for visual representation of tasks
- Recording of all progress - Confluence
  - multifunctional
- Main communication - Slack
  - to familiarize ourself with a more professional discussion/communication app
- Meetings - Zoom
  - potentially set up a running weekly meeting for planning and standups

Actions:

- Research into Angular and Electron
- Finish setting up

Monday 14 Aug

All participants present

Discussion

- Discussions into what we want our theme and website to represent. Debate and delegate tasks and subtasks to begin tasks. Read more under decisions.
- Being sprint planning and mapping out of tasks that we want and need to be delegated out. Decided upon our responsibilities and commitment times we have available in the week
  - Currently all are available in the week to follow through with tasks
- Role mapping out of what we want main roles to look like:
  - 1-2x Website designer
  - 2x Website creator
  - 2x Backend to front end linking
  - Team's main focus: **teamwork** - helping wherever possible. Specifically, after we have completed our tasks or if help is requested
- We talked through what our team knows currently on the things that may be required in the project: Came through on shared knowledge from members - more on in decisions
- Final closing of meeting: reiterating what our tasks and subtasks decided upon was and what we want to be doing

Summary

Goals for the coming week

- Finish through with delegated tasks and receive feedback from all team members on work done

Decisions

- To discuss through the main visual and thematic designs together to ensure cohesive communication
- Architecture and technologies:
  - Framework: Angular
  - Adobe XD for design
  - Flask for python

Actions:

- Get clarification on whether or not a high Authorship score means good or bad comparison results from model
- Create user stories - Adrian
- Base rough sketches of prototype, need not be clear but to help map out decisions for upcoming week - Lin
- Colour themes, font picks and theme of website - Hanizam, Saaiq
- Draft motivational modelling - Thomas

Workshop meeting - Friday

Notes

Required update and restructure to confluence. Needed emergency restructure and update of Jira. Entire team has decided to spend more time understanding project and updating tasks and notes to relevant platforms (jira and confluence) to have a uniform presentation.

- Sort out standup presentation format

(!!) Is priority

Wednesday 23 Aug

All participants present

Discussion

- Planning and updating of subtasks
- Standup meeting to see where we all are at:
  - Motivational model - Thomas
  - Angular tools and update - Thomas
  - personas/user stories - Adrian
  - documentation - Lin
  - API - Hanizam
  - prototype update - Saaiq
    - followup update from Lin
- Confirmation, discussion and changes made according to recommendations and constructive advice for low-fidelity prototype
  - This includes:
    - Welcome screen - Something to welcome in users both first-time and recurring users
    - base use screen - main screen where model is used
    - log in/signup screen - profile related, for users to reuse their uploaded text
    - Knowledge bank - where docs are stored for users who have created an account
      - how it will look with no current profiles
      - how it will look when a new profile created
      - when a knowledge bank already exists, and the type of format we want to go with it
    - results screen
- including updates from individual communication between members
- high-fidelity prototype notes according to advice from low-fidelity model
- a refresh of the Jira board was required to plan for further plans before the end of the week and started planning briefly for sprint 2

Summary

Goals for the coming week

- Finish up all prototypes both low and high-fidelity
- Ensure that all documentation of current work is updated in confluence
- Finish up with API, software and architecture research
- Begin looking into potential tasks - angular, auth0

Decisions

- The visual designs of the prototypes and the functionality decisions
  - All visual designs and functionality got approval from all team members, relevant feedback was actioned
- User stories update and a persona review to ensure track

Actions:

- Angular
- Jira board updates
- prototype uploads, progress updates needed

Sprint 2 Meetings

Monday 28 August

All participants present

Discussions:

- Standup meeting
  - we had done part planning during workshop previously, thus this was a small update
  - notable mentions:
    - saaiq - function, takes in known and unknown text to spit out authorship percentage
- Sprint 2 Planning Update
  - Created a breakdown of tasks in Jira in preparation for sprint 2 based.
  - General task broken into:
    - angular related front end tasks
      - creation of the different portions of the front end website
    - logo creation
    - confluence update
    - backend code and functions and flask work
  - tasks that have been created were delegated to relevant group members and we discussed about workload + assistance from members
  - look ahead, planning dates and trajectory
    - we set out some expectations for when we expect this to be done

Summary

Goals for the coming week

- Attempt/complete tasks logged in Jira
  - ensure consistent updates along the way
- Ensure that all documentation of current work is updated in confluence
- take time out to understand all the programs that will be used but begin where possible

Decisions

- Create logo samples in preparation for decision

Actions:

- Angular front-end begin
- Jira board update
- confluence wrap up of sprint 1

Thursday 31 August

Saaiq & Hanizam:

- consult each other about technical and functional issue of the backend.
- Finalizing on the flask app and the algorithms.
  - Got the algorithm functioning on other devices without issues
  - Connected the algorithm function with the flask app
  - Got the basic front end and back end connection with inputting into the algorithm and getting an output to the front end all functioning
- discussing further on the backend extension
- Got some very rough prototype designs for a logo to be reviewed

Thomas & Adrian:

- Got google fonts to work for angular
- Created textbox that gives console warnings every time user inputs text. This is how it is currently. What we plan to do is to **have it store the first text on a button press, clear it (to allow for input of the second text), store the second text on a button press, then clear it**.
- Decided to use Text area instead of input as it suits our purposes better
- Planning to use Friday's meeting to get advice from other members to work out how to store input value, as well as storing value on button press

Update from **Friday 1st September** meeting:

- EDIT 1: we should make a post request for our app.py to work (this connects frontend to backend in an easy way), so we do not need to do all the storage mentioned above.
- EDIT 2: we are going to have two text boxes on screen (one for known text and the other for unknown text) and a "compare" button. When the button is pressed, the contents of both text boxes will be saved into known.txt and unknown.txt respectively, and a HTTP POST request will be made to connect to the Python app.

Lin:

- updated meeting minutes
- wrap up for sprint 1
- confluence largely up to date with progress and updated alongside group members
- updated user stories in accordance with basic version 1.0 of language processing website
- update of prototypes in response to Eduardos feedback

Overall:

- lookahead and plan for trajectory:
  - aim is to finish basic website by next week with fully linked back-end and front end

Due to large completion rate from backend investigation alongside flask connection and confluence is up to date, all members will be updated on the progress of angular and split front end tasks to make it more manageable

Current concerns:

- Connection between front and backend for passing of texts to model

    - revisit after more of front-end has been developed through the now largely handled backend, meetings will be had between front and back end for consistency
- Hold meeting in future for Logo vote

Thursday 7th September

All participants present

Hanizam & Thomas

- Worked on connecting the back end flask API to the front end
- Created a function to post data to the API to then display on the front end
  - Had issues with errors of posting the data
  - Had issues on the data being sent not being parsed by the back end correctly

Adrian

- Fixed utilising `textarea` instead of `input` for a better visualisation of inputting text into the website
- Added final report page with a score and template to add info on authorship

Lin

- Researched implementing animations for the front end
  - i.e. known to unknown text(s) input transition
  - Loading bar animations
  - Authorship Score visualisation animation
- Researched potentially using React for the front end in a worst-case scenario where back end cannot connect with Angular front end

Saaiq

- Fixed the back-end API to properly parse and take in JSON data
- Fixed the front-end to take in and read inputted text
- Fixed front-end function to post data in the correct JSON format

Summary

Goals for the coming week:

- Finish the front end by the end of next week
- Look into uploading files
- Ensure similarity to prototypes and update progress alongside

Monday 11th September

All participants present

Discussion

- More branches and merging being done, ensuring that there were no merge conflicts and deletions
- Saaiq, Thomas, Adrian and Lin:
  - largely front-end progress into ensuring it looks like what we intend of it to be
  - Text box area formatting and scroll look - Lin
- Hanizam
  - Research and investigation on uploading files - *.txt files first.
- Saaiq
  - Updated API structure
    - Added documentation of using the API (with flask_restx)
    - Added JSON models in which the API methods need to follow
      - Makes sure the API sends and receives the correct type of data
      - Allows code to be more scalable
  - Updated front end to send and receive data of correct JSON model type

Summary

Goals for the coming week

- Finish front-end and backend for version 1.0
- Get upload files working
- Aesthetics if we have time

Decisions

- No major decisions, minor decisions include styling placements

Actions:

- Upload files - single, multiple and different file types
- Research into log in'
- Welcome screen

Wednesday 13th September

All participants present

Discussion

- Main file formatting and restructuring
- Upload files is complete (Hanizam) - both front end and back end
  - More under summary
- Welcome page is complete - Adrian
  - has been moved into main page as the first thing users see
- Main focus on website styling by the end of the week
-

Goals for the coming week

- Converting website pages into flexbox for scalability
- investigation into mozilla and different platforms
- update Jira to follow along

Decisions

- Upload files to include docx and pdf files

Actions:

- Upload files - single, multiple and different file types
- Research into log in'
- Welcome screen
- Animations

Tuesday 19th September

All participants present

Discussion

- Making buttons/titles scale with window by adjusting flexbox options
- Find proper fonts (font weight, colour, etc) for headings
- Make results section look better - find a way to display explanations (can use graph visuals), the backend part is already completed
- Issue with Python algorithm - talk to Eduardo about it on Friday (Saaiq)
- Make similarity circle change colour dynamically
- Add functionality - “add known text” button to add the textbox content to the list of known texts
- Improve “upload” buttons (with outline, icon, etc)
- Increase the margin between textboxes slightly
- Fix scaling issue with navbar - right now the links disappear too early when the window size gets reduced
- Fix random white bar on top of navbar
- Think about Snapshot testing

Thursday 21 September

All participants present

- Updated Confluence in preparation for sprint 2 wrap up
  - (!!) Pivot and enforcing commit/branching guidelines for future commits, branching and merging
  - updated coding, merge and git guidelines
  - updated motivational model and personas to match user stories
    - merged author with student users
  - Added API security - input security, function security and output security
  - mentioned changes between sprint 1 prototype
- Sprint 2 retrospective complete
- Separated Team decisions into another page

Goals for the coming week

- Begin sprint planning for sprint 3
- Wrap up final version 1.1

22nd September - Meeting with Yue

Details of the meeting

- Feedback we received was largely functionality-based and on user experience:
  - Due to the long length of our textbox, users would have to scroll more than usual to see the documents that they have uploaded (which is present underneath the text input area). A shorter text area would benefit user experience
  - The button to add the current text area's text as a document was unlabeled and thus for a user to understand the buttons use, it would be beneficial to include a description.
  - Upload button's visibility increase to users

Things we realized during the meeting that will need to be addressed:

- Mozilla update for scroll bar
  - Due to different web browsers, updates will need to be made to ensure that out website presents the same way to all users.
- Document changes made to the model
  - Our group has made changes to the model to increase its efficiency and function, however these changes will need to be documented and presented to Client for further assurance in direction.
- Polish main feature page!
- More tool tips for new users
  - Directions for first-time users to help with their experience with our website

Questions that were answered:

- testing - test to cover 80-90% of the code to ensure full coverage
- deployment - CI/CD pipeline for testing or deployment
  - document it for deployment

Meeting with Eduardo

Ask Eduardo about button names, upload file selection (big database, one-by-one, folder, etc)

Header - locked?

website name - Authorights (STC)

25th August

Prototype website:

Changes to add:

- second one - “file”, first one - “files”
  - text in regards to the two different boxes for model input
- enter “what: text? - student text?
  - instructions for webapp user to understand what goes into the input boxes and how to use it the way it is intended
- “90%” circle - what does it mean? similarity? do not put any accusations, just say “the texts are similar”
  - texts to add either below or next to final comparison result page
- could have profile details on the right side (basic information), code might enable this
- responsive header (auto-resizing), locked
- extension:
  - profile with history
  - document the reasoning behind colours
  - saving model for profiles? put it in a database?

1st September

- updated user stories to present to Eduardo, where we had an extra two user stories to the intended single Teacher user
- but the additional two user stories were well accepted
- had no additional questions regarding progress and website at this time

Sprint 3 Meetings

Thursday 28th September

All participants present

Discussions:

- Sprint 3 Planning Update
  - Created a breakdown of tasks in Jira in preparation for sprint 3 based.
  - General task broken into:
    - completing last extension for website
      - identifying labels
      - styling explainations
      - homepage update
      - update and complete all pages of the website
    - Different types of testing
    - tasks that have been created were delegated to relevant group members and we discussed about workload + assistance from members

Summary

Goals for the coming week

- Attempt/complete tasks logged in Jira
  - ensure consistent updates along the way
- Complete as much of front end as early as possible

Monday 2nd October

All participants present

Discussions:

- Home page/help page/about page changes
- Textboxes on home page
- Popups for tutorials

Goals for the next meeting

- Making decisions with respect to:
  - Popups for tutorial or a separate help page?
  - What content to put inside tutorial?

Wednesday 4th October

All participants present

Discussions:

- Frontend deployment
- Shifted to a new repository (for frontend)
- Write “about” section
  - Write what Eduardo wants?
  - About us - the developers
  - About the website - what it does?
  - About the algorithm
- Finalised the tutorial images to be used for popups

Goals for the coming week

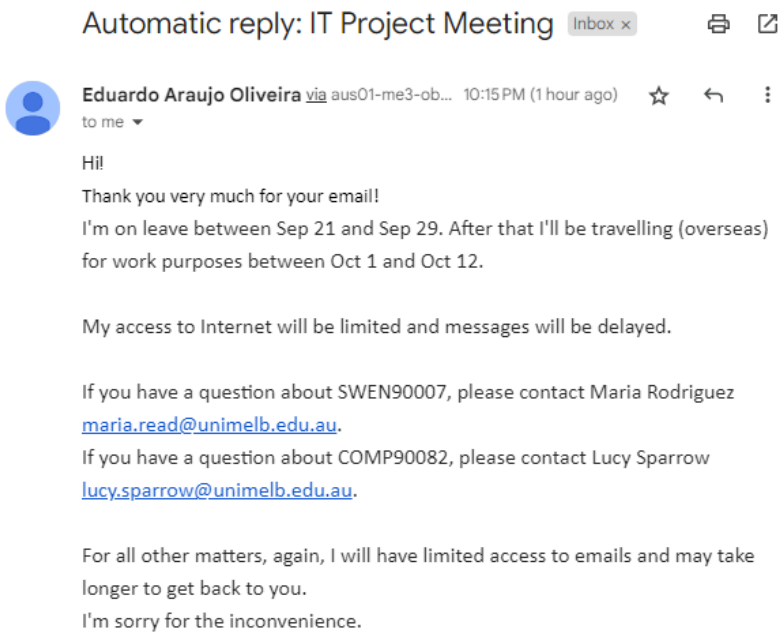
- Plan to commence testing in the next meeting

Monday 9th October

All participants present

Discussions:

- Evaluated website to see what last changes/fixes we still need to make:
  - Updated tutorial popup
  - Change file upload limit (was causing deployment issues)
- Looked into testing:
  - Acceptance Criteria
  - Functional Testing
  - Acceptance tests
  - Test cases
  - Structural Testing
  - Non-functional testing
  - Load testing
- Updated user stories to fit with final product
- Deployment Notes:
  - Changes to asset directories to work with AWS
  - Changes in Angular build requirements for successful build
- Updated Jira Board Tasks
  - Marking off completed tasks
  - Adding new tasks
- Attempted communication with client (unsuccessful) :(



Automatic reply: IT Project Meeting Inbox ×

Eduardo Araujo Oliveira via aus01-me3-ob... 10:15 PM (1 hour ago)
to me

Hi!
Thank you very much for your email!
I'm on leave between Sep 21 and Sep 29. After that I'll be travelling (overseas) for work purposes between Oct 1 and Oct 12.

My access to Internet will be limited and messages will be delayed.

If you have a question about SWEN90007, please contact Maria Rodriguez maria.read@unimelb.edu.au.
If you have a question about COMP90082, please contact Lucy Sparrow lucy.sparrow@unimelb.edu.au.

For all other matters, again, I will have limited access to emails and may take longer to get back to you.
I'm sorry for the inconvenience.

Goals for the next meeting

- Planning ahead what we need to get done

- Ethics and security report

Thursday 12th October

All participants present

Discussions:

- Mostly updates from team members since Monday, alongside completing the remaining artifacts for sprint 3.
- Website updates:
  - Creating the automated test for CI/CD pipeline
- No major decisions made today, largely cleaning/wrapping up

Goals for the next meeting

- Prepare notes for meeting with Eduardo (during class)

Meeting with Eduardo 13th October

All participants present

Discussions:

Feedback from Eduardo:

- Clarify result statistics - cannot tell if it is for known text or unknown text
- Explanation for result statistics so they make more sense:
  - Popup when hovering over it
  - OR expand button?

Summary

Goals for the coming week

- Eduardo will send document for about and contact page so we can update those sections on the website
- Handover document
  - Document features and their progress
  - What have we done overall

Monday 16th October

All participants present

Discussions:

- Planning for final presentation:
  - Reading over requirements (for what needs to be in the presentation)
  - Thinking about who should take up which parts of the presentation
- Finalising results section:
  - Hovering over statistics will show explanations

Summary

Goals for the next meeting

- Write the script
- Create the presentation slides

Wednesday 18th October

All participants present

Discussions:

- Drafted script for presentation
- Delegated parts of the script to each team member
- Created the slides for the presentation

Summary

Goals for the next meeting

- Finish the script
- Practice presenting
- Finalise slides

Thursday 19th October

All participants present

Discussions:

- Practice presentation
- Updated parts of the script to better suit the time
- Finalised slides to have images and better formatting
- Chose texts to compare for the demo

Potential Extensions

Given the time:

- Profile pictures
- Animation loading
- Users + Profile
- Profile Sharing

Sprint 3 Planning Notes:

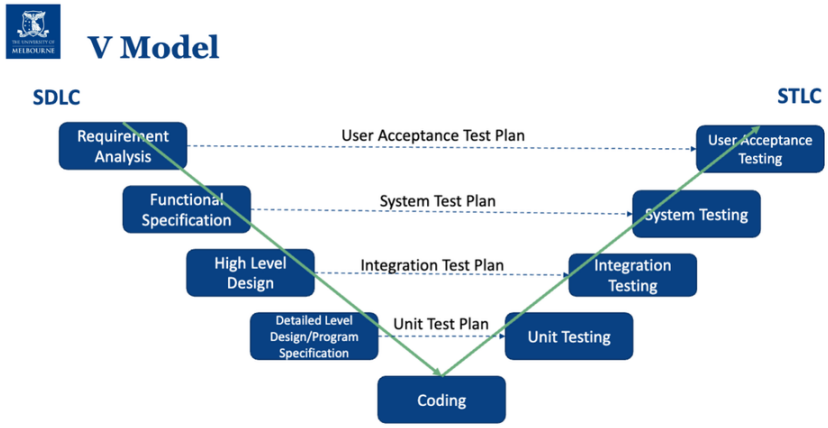
Testing (W7):

Software testing - identify correctness, completeness and quality of developed computer software

- conducted with intent of finding errors in software so it can be corrected with release

Seven Principles:

- Exhaustive testing is not possible
- Defect clustering - a small number of modules contain most of the defects detected.
- Pesticide paradox
  - The test cases need to be regularly reviewed & revised, adding new & different test cases to help find more defects.
- Testing shows a presence of defects.
- Absence of error is a fallacy.
- Early testing.
- Testing is context dependent



Terminology

- Test suite: a set of test cases.
- Test case: a set of inputs, execution conditions, and a pass/fail criterion.
- Test or test execution: the activity of executing test cases and evaluating their results.
- Adequacy criterion: a predicate that is true (satisfied) or false (not satisfied) of a <program, test suite> pair

Adequacy criterion = set of test obligations. E.g., cover all statements and all branches in the program under test

A test suite satisfies an adequacy criterion if

✓ all the tests succeed (pass)

✓ each test obligation is satisfied by >=1 test case(s)

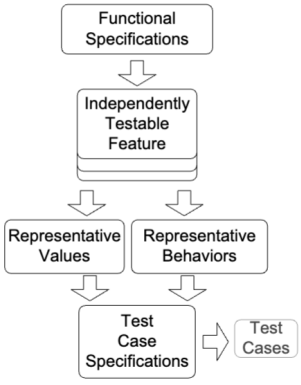
Acceptance Criteria: set of predefined requirements that must be met in order to mark a user story complete.

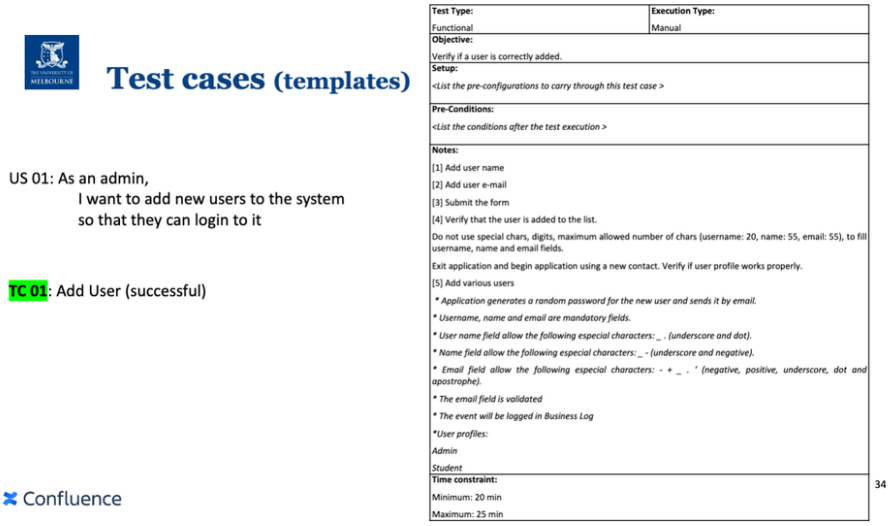
- given [condition], when [something happens], then [result]

User Story ID	User Story	Given	When	Then
1	As an athlete user, I can see the latest update for my sessions in my dashboard's calendar so that I can track my current program's completion in a more informative way.	I have been doing exercises everyday	I check sessions in my dashboard's calendar	• I can see completion of my current and previous programs
2	As an athlete user, I can receive some badges for my achievements after I overcome some challenges during my sessions so that I can be motivated to stay on track for my "Building".	**Many challenges related to my personal program listed in my session**	**I complete challenges in my session's program**	• **I receive badges for corresponding challenges**
		I have received badge when I complete the challenge	I click on the social media button next to the badge section	• It shares my badges on the social platform
3	As an athlete user, I can see route map tracking in my session description & feedback so that I can get an overview of session feedback visually.	I am doing some outdoor exercises	I click session in the dashboard	• It show route tracking map • feedback

Test cases:

- A documented set of preconditions (prerequisites), procedures (inputs / actions) and postconditions (expected results) that a tester uses to determine whether a system under test satisfies requirements or works correctly.
- **When:**
  - supposed to be alongside development (i.e: writing test cases for each task in the sprint.)
  - but since this is not what we did due to our learning processes, we'll need to spend a chunk of time going through tasks that have been done in order to prepare test cases
  - given that we have time for user sign ups and knowledge bases, we'll create tests alongside





Test cases (templates)

US 01: As an admin,
I want to add new users to the system
so that they can login to it

TC 02: Add User (unsuccessful)

Test Type: Functional	**Execution Type:** Manual
Objective: Verify application behavior when user could not be added.	
Setup: *<List the pre-configurations to carry through this test case >* •	
Pre-Conditions: *<List the conditions after the test execution >* •	
Notes: [1] Start filling the form for a new user and cancel operation. User list is not modified. [2] Try to add user with an existent username. [3] Try to add user when username, name and/or email fields are empty. [4] Try to add user with Invalid content type: [4.1] Using more than maximum allowed number of chars (username: 20, name: 55, email: 55). [4.2] Use special chars, accents, empty, spaces, only spaces to filled username, name and email fields. [4.3] Try to add user when email format is not valid. ** User name field allow the following special characters: _ . (underscore and dot).* ** Name field allow the following special characters: _ - (underscore and negative).* ** Email field allow the following special characters: - + _ . ' (negative, positive, underscore, dot and apostrophe).* ** The email field is validated*	
Time constraint: Minimum: 20 min Maximum: 25 min	

Confluence

36

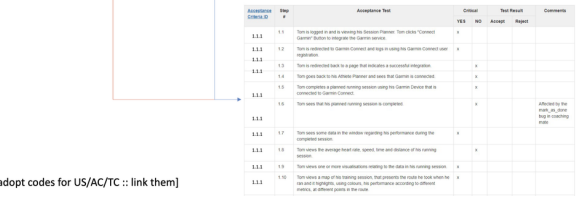
- looking at these templates, we most likely will be going back to update our user stories

US01 ⟶ AC01, AC02

AC01 ⟶ TC01, TC02, TCn

Linking user stories and test cases in the sprint

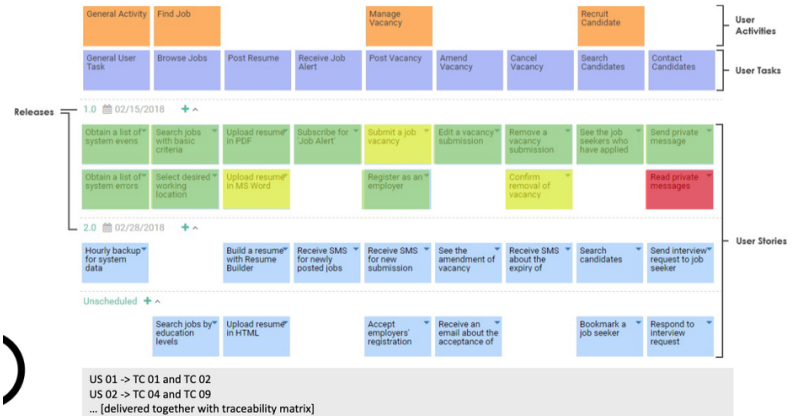
Epic 1: Data Visualization



adopt codes for US/AC/TC :: link them]

REQUIREMENT ID	REQUIREMENT DESCRIPTIONS	TC 001	TC 002	TC 003
SR-1.1	User should be able to do this.	x		
SR-1.2	User should be able to do that.	x		
SR-1.3	On clicking this, the following message should appear.		x	
SR-1.4			x	
SR-1.5		x		x
SR-1.6				x
SR-1.7			x	

Requirements vs. Tests



All above is functional testing: linking user stories, acceptance criteria and test cases

- Requirements based, black-box (not looking at code)

Structural testing

- white box/glass box/code-based testing

Steps:

- Create functional test suite first
- Measure structural coverage to see what is missing
- Interpret unexecuted elements

Statement/block coverage

Adequacy criterion:
"Each statement/block executed at least once"

Coverage:

$$\frac{\#\text{ executed statements/blocks}}{\#\text{ statements/blocks}}$$

Blocks consists of multiple statements

Rationale: a fault in a statement/block can only be revealed by executing the faulty statement/block

Adequacy criterion: each branch (edge in the CFG) must be executed at least once

Coverage:

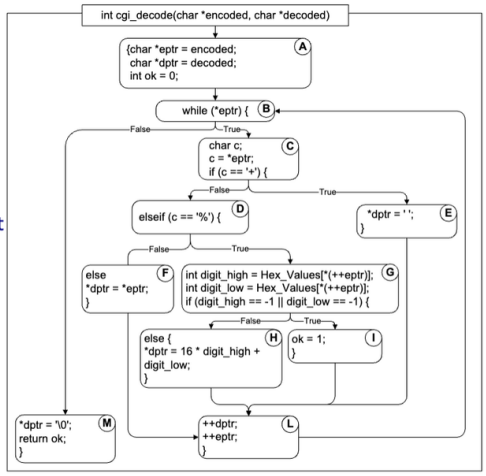
- executed branches
- branches

Example

T_0 =
{"", "test",
"test+case%1Dadequacy"}
17/18 = 94% Stmt Cov.

T_1 =
{"adequate+test%0Dexecut
ion%7U"}
18/18 = 100% Stmt Cov.

T_2 =
{"%3D", "%A", "a+b",
"test"}
18/18 = 100% Stmt Cov.



Coverage is not size

Coverage does not depend on # test cases

- $T_0, T_1 : T_1 >_{coverage} T_0$ $\quad T_1 <_{cardinality} T_0$
- $T_1, T_2 : T_2 =_{coverage} T_1$ $\quad T_2 >_{cardinality} T_1$

Minimizing test suite size is seldom the goal

- small test cases make failure diagnosis easier
- a failing test in T_2 gives more info than one in T_1

Condition testing

Branch coverage

- considers how a computation is broken into cases
- but only roughly:
  - groups cases with the same branch outcome

Condition coverage

- performs detailed case analysis
- considers *individual conditions* in a compound expression
  - e.g., both parts of digit_high == 1 || digit_low == -1

Basic condition testing

Adequacy criterion: each basic condition must be executed at least once

Coverage:

- truth values taken by all basic conditions
- 2 x # basic conditions

Functional testing:

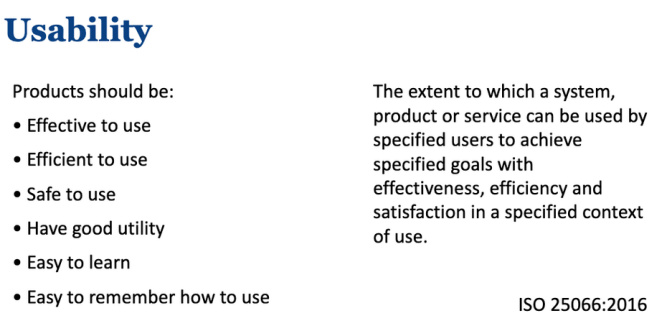
- best for missing logic faults
- A common problem: Some program logic was simply forgotten
- applies at all levels
  - unit - form module interface spec
  - integration - from API or subsystem spec

- system - from system requirements spec

Structural (code-based) testing

- will never focus on code that isn't there!
- applies to relatively small parts
  - unit
  - integration

Non-functional Testing



UX: how people feel about a product and their impression and how it feels to them and how they feel afterwards

Heuristic Evaluation

- Visibility - should always keep users informed about what is going on (ie appropriate feedback)
  - we should probably add a popup if their analysis runs over a certain amount of time
- Match - between system and real world (speak users' language, simple common language, follow real world conventions)
- Control - user control and freedom, support undo and redo
  - Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue.
- Consistency
  - Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.
- Error Prevention
  - Even better than good error messages is a careful design which prevents a problem from occurring in the first place.
- Recognition - Make objects, actions, and options visible.
  - (!!) add an ℹ button to pull up instructions again when users need it?
  - "Instructions for use of the system should be visible or easily retrievable whenever appropriate."
- Flexibiity - Accelerators - unseen by the novice user - may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users.
- Minimalism - Dialogues should not contain information which is irrelevant or rarely needed
- Recover - Help Users Recognise, Diagnose, and Recover from Errors
  - error messages should include plain language expression, precisely indicate the problem, construct/suggest a solution
- Help and Documentation
  - Troubleshooting: Make sure your system indicates errors and messages in plain language

In line with user stories, ask questions:

- Will the user know what to do?
- Will the user see how to do it?
- Will the user understand from the feedback whether the action was correct or not?

Load Testing:

- How well your program can cope with processing a large amount of data or multiple of users using your server at the same time?

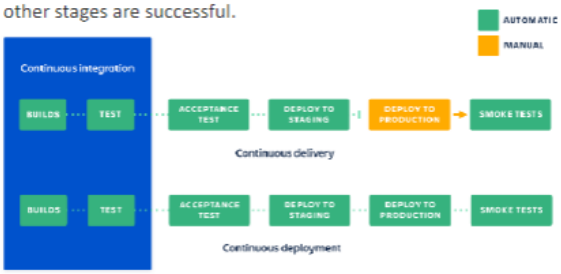
Notes

- (!!) Do we have error popup functioning? check when im home

Deployment

CI/CD

- Continuous Integration: Integrating your code changes back to your main branch. Changes are checked by automatically building, testing code.
- Continuous Delivery: Extension of CI. Automates the release process so that you can deploy the application at any time.
- Continuous Deployment: Extension of continuous delivery. Automate deployment to production environment if all other stages are successful.

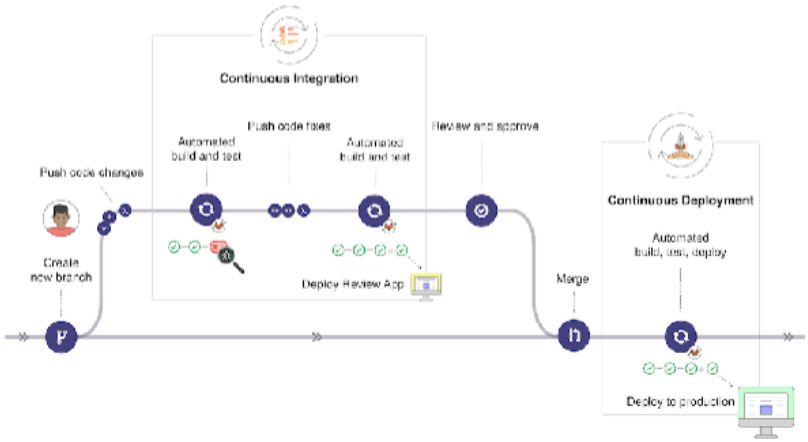


DevOps Pipeline

A set of steps executed when a condition is met (e.g. a pull request is made)

Need to ensure checks are performed before code is deployed

Typical workflow:



Deployment

- Set of steps required to make your application accessible to the public
- Three main models:

SaaS	PaaS	IaaS
Software as a service.	Platform as a service.	Infrastructure as a service.
The vendor manages everything.	Vendor manages architecture, but you develop the application and manage data.	You manage the infrastructure, data, and application.

- Examples:
  - SaaS: Dropbox, Salesforce, Cisco WebEx, GoToMeeting
  - PaaS: AWS Elastic Beanstalk, Microsoft Azure, Heroku, Google App Engine
  - IaaS: DigitalOcean, Amazon Web Services (AWS), Cisco Metapod, Microsoft Azure, Google Compute Engine (GCE)
- PaaS is likely easiest for this subject

Handover (checklist to be used as a reference)

Updated README file (i.e. guidelines for system's configuration and installation, list of files included, changelog, detected bugs, test cases, traceability matrix)

- **Documents** (e.g. user stories, personas, motivational model) to be exported as PDF and added to code repository (docs/). Test cases to be exported and added to code repository (tests/)
- **Description of key algorithms** (e.g. calculating the service price or any algorithms that are unusual) + mechanism of importing data from 3rd party systems,
- **Description of key classes and the application's layers** (this simply makes it easier for the new team to find the place that requires changes),
- **Description of the database structure** (additionally, in case of a relational database, a description of their correlations should also be included; this can be presented on ERP - Entity Relationship Diagram),
- **Deployment guidelines** (i.e. how to deploy the source code, database and run the project on a new server, any necessary administrator/test customer login credentials, access to code repositories, databases and servers as well
- Any **licensing agreements** that need to be considered

17

☐ Create a 'Handover' page and link it to your code repository

Handover – Table of contents example

Table of Contents

Example of table of contents

[example on github]

Table of contents

[example for this subject]

Ethics & Security Report

Ethics Report

There are a couple ethical issues that can occur throughout this project. The main ethical concerns would come with the use of machine learning in the process of creating the authorship detection model. The other, relatively more minor issues that are addressed by our group is the ethics of our code transparency and group member interactions.

Group Member Interaction Ethics

As a group, 4 of our members (Adrian, Lin, Saaiq and Thomas) already knew each other for a while. We had gone through other group projects and such together and by the starting of this subject, already knew how we all work and how we work with each other. This meant all the regular steps of ensuring everyone is accountable for their work and actions was already the standard which we came into this subject with.

The addition of our 5th group member Hanizam, we believe went quite well. There were no issues communicating the standards and expectations we had for each other that had to be held.

The basic idea was built under the philosophy of openness. This meant both being open and understanding of everyone's situations as well as making sure there is open communication between all group members. If any issues arised for any group member, we would all make sure to come into dealing with the problems with understanding of that group member's problems instead of one of punishment. This understanding would also further help enforce making sure there is open and early communication on any issues which a group member knows will occur, that way the rest of the group can be prepared and accommodate for it.

Code Transparency

The issue of code transparency is more a mixed issue with it being a topic of the ethics of the security of the code. In our case as we are running a web application, the front end possesses no issues with code transparency as every page on the website is meant to be publicly accessible and all the code that needs to be secured is within the Flask Web API.

The Flask Web API keeps its code transparency by using multiple files to help keep the scope of the functions separate and protected. All the required API functions are kept within 1 file where it can only access the required function to compare the texts and the functions to process word and pdf files. All external processes are kept in a separate `docu_functions.py` file, while the functions to compare texts created by us are kept in the `compare_texts.py` file and the remaining functions aggregated from the client are all kept separately in `functions.py`.

Machine Learning Ethics

With our project being on the topic of using trained deep learning models to estimate the likelihood a text was written by an author. 2 main issues result from pursuing such a project, that being the use of data in the training of these models as well as the implications of the model results.

A big problem that's on the rise is the issue of the ethics with data usage, in our context, for the use of training machine learning models. Data that's being used to train our models need to come from ethical sources where all the texts being used are from users that knew their texts would be used in the potential training of machine learning models. For this project the data and training of the model came directly from the client, and as such the data being used for training was the responsibility of our client.

There is also the problem on the data that user's of our website upload themselves. There is the potential where if a user had approved of an unknown text as being written by that author, we could then store their data as a new instance for the retraining of our model. In our case since we did not implement a login feature, we did not store user data on texts so there would be no accurate way to implement adding user data into training the model. But if we did want to do so we would make sure to implement a check box that user's can choose to tick to

allow their texts to be used to help improve the model. As such our model only processes the texts and outputs the statistics and author ship score without having any user data be permanently stored.

The other issue of the implication of our model results is the high likelihood that the model would eventually output a false negative or positive. We make sure that this issue is addressed through a simple and clear note below each score stating that the results may not be accurate and to not use this website's results as a sure fire answer to if a text was written by someone.

Security Report

Security Issues

1. **DDoS (Distributed Denial of Service) Attacks**
    a. An attack performed where a malicious party attempts to overload the server with an information overload to overwhelm the server's network traffic and result in a potential complete shut down of the application.
    b. **Solution:** The use of AWS (Amazon Web Service) to deploy both the front and back end allows for the automatic use of their **AWS Shield** service which comes with automatic protection against DDoS attacks as well as other potential malicious network attacks
2. **Improper Text and/or File's being Uploaded**
    a. This would be when a malicious user attempts to send over corrupted or potentially malicious files with the intent to obtain information that shouldn't be possible or cause a crash to the back end API.
    b. **Solution:** The structure of the back-end API was built such that all data that came in had to follow the correct format and all the data which would be outputted had to follow the **Score Model** which can be seen within the API documentation. The front end is also created such that only the specified file types are sent through and that only the correct number of files and texts of each type (known and unknown) are supplied.
3. **Attacks on The Back End**
    a. This would be the process of the user bypassing using the front end to utilize the model through just the back end API. Essentially ignoring all the security measures set up in the front end.
    b. **Solution:** Is to setup protection at the back end level. As the data coming in is a multipart form where any file can be sent through, the files are processed in the back end to make sure to only parse the correct file types while the rest are ignored. Everything is parsed using UTF-8 encoding so there would be no chance of any malicious files causing disruptions at least in the text-processing process.
    c. **Potential Solution:** Another measure that could be added for protection would be setting up an authentication to use the API so that only those with the right token's have the ability to utilise the API requests. The issue with that is, it would be an absolute must to implement if we had a login system with privately stored data to be requested but since there is none implemented for now and everything is public with nothing stored, adding authentication would be essentially useless. You can pretty much assume that anyone with the ability to attempt to harm the API would also have the ability to just use the site and copy the token generated to then utilise the back end. As such, we have decided that adding authentication wasn't necessary for the current project scope.

Security Evaluation

We believe that for the final scope of our project we have an overall strong security of our product.

Our front end is secured such that files being uploaded can only be of specifically supported file types (`.txt` `.pdf` `.docx`). And the inputted text is of a string type which handles all characters that JavaScript strings are capable of. This all helps make sure the input to the back end is secure from the front end. To keep the security intact on the data outputted from the back end to the front end, a score model is also created from the front end's side which make's sure that only the headers which match the score model are processed in the code.

Since we aren't implementing a login feature, there is no need for a database to store login details and profiles. As such there is no potential threat of a malicious party attempting to steal that data.

This leaves us with the back end API which is kept protected by using the methods `.expect()` and `.marshal_with()` to make sure the API function only takes in the expected type of values as well as the model's output being the expected model type.

The use of AWS for the deployment of both the front and back end helps bring additional network security as outlined above in the 1st security issue. AWS also brings additional benefits on making sure there is still security even in the event of the web app's being stress

tested. This is especially the case with the back-end where potentially gigabytes of files could be uploaded at once to be processed, where the program would safely crash in an overload event. Even though that is the case, that is an area for improvement where setting limits on the number of and size of files allowed to be uploaded as well as in the back end the size of files processed would help keep the servers from being overloaded by any user.

Handover Documentation

Project Title: AuthoWrite **Date of Handover:** 20/10/23

1. Project Overview

Provide a brief description of the website, its purpose, target audience, and key features.

Authowrite is a website specializes in checking Authority similarity. This website is targeting multiple range of academic audience such as teachers and students. It allows a user to upload known texts and an unknown text and compare the writing styles to determine if the texts are written by the same author.

2. Project Team

- **Scrum Master/Developer:** Lin XIn Xie
- **Backend Developer :** Saaiq Ahmed
- **Backend Developer:** Hanizam Zaharyudin
- **Frontend Developer:** Jia Jie Zheng
- **Frontend Developer:** Adrian Podiono

3. Project Scope

- Project is a website which provides an interface for a user to use the given machine learning model which analyses authorship similarity
- Handover includes two git repositories (front-end and back-end) for the website.

4. Technical Details

- **Website URL:** AuthoWrite
- **Collaboration Tools:** Slack, Jira, Confluence, Zoom
- **Programming Languages:** HTML5, CSS3, TypeScript, Python 3
- **Frameworks/Libraries:** Angular 2, Bootstrap, Flask, Tensorflow
- **Version Control:** Github
- **Deployment:** Amazon Web Service (AWS),
  - **Frontend:** CodePipeline, CodeBuild, S3 Bucket
  - **Backend:** E2 Instance

5. Website Features and Functionalities

- Upload files or copy in text
- Compare writing styles of known texts to unknown text
- Determine authorship similarity between unknown text and known texts as a percentage
- Gives more detailed statistics about differences in writing styles between texts.

6. Design and User Interface

- UI is design to be user-friendly and easy to understand/use
- Colour scheme and design in Visual Design Decisions

7. Testing and Quality Assurance

- Performed functional and non-functional testing
- Test cases for functional testing: Test Cases
- Non-functional testing: Non-Functional Testing